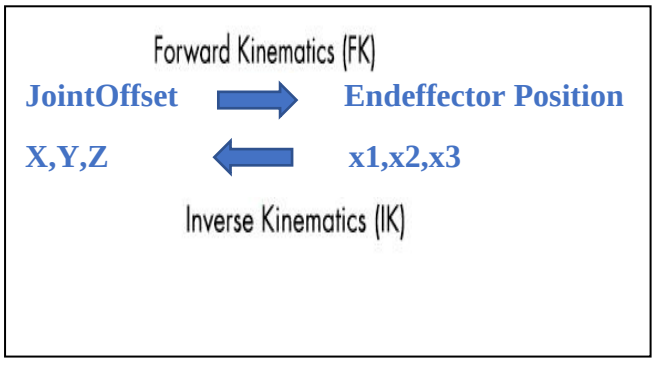
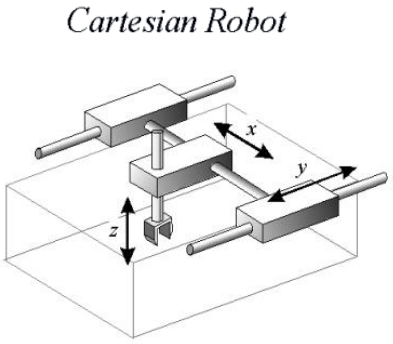
Project : Kinematic Analysis of GANTRY Robot.

Abstract:

An **industrial robot** is comprised of a **robot manipulator**, power supply, and controllers. Robotic manipulators can be divided into two sections, each with a different function: **Robot Arm** and Body. Robot manipulators are created from a sequence of **link** and **joint** combinations. The links are the rigid members connecting the joints or axes. There are 2 types of joints: **Linear joint** and **Rotary joint**. The rotary joint consists of the **Revolute joint (R)** and a **Twisting joint (T)**. The linear joint consists of a **Prismatic joint (P)** and a **Sliding joint (S).** Each manipulator has its **DOF** (Degree of freedom), the number of degrees of freedom is equal to the **total number of independent movements** that an object can perform. Pick and place task is one of the most important tasks in the industrial field handled by the "**GANTRY ROBOT**". A gantry robot consists of a manipulator mounted onto an overhead system that allows movement across a horizontal plane. Belt-driven linear actuators are widely used in gantry systems from electronic assembly and electronic manufacturing to vision systems and industrial automation. Gantry robots are also called Cartesian or linear robots. They are usually large systems built with linear guide rails that perform pick-and-place applications, but they can also be used in welding and other applications. In this project, we are using a **GANTRY ROBOT** whose **DOF** is **3** and consists of **3 Linear joints**. We use the link and joint parameters to carry out the manipulator's kinematic analysis (Kinematics). We obtain the **position and orientation** information of the end-effector of a manipulator of known geometry for a given set of joint angles in **forward kinematics**. On the other hand, in **inverse kinematics**, we **determine the joint offset values** of a manipulator of known geometry for a given position and the orientation information of its end-effector. The kinematic equations between the working space and the joint space are deduced by the homogeneous transformation principle. In the present project, kinematic analysis is done using **ROBOANALYZER** software to find out the position of the tool or end effector for different and joint variables and we use **CPRog** Software for programming the Gantry Robot Kinematics.



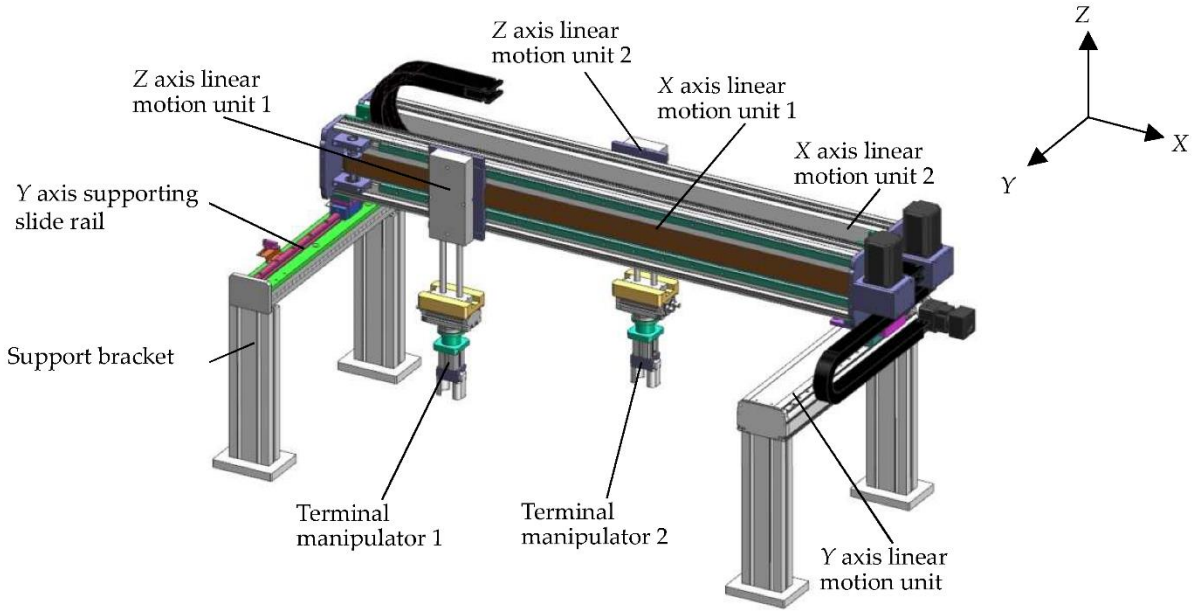


Introduction to GANTRY Robot:

A **GANTRY robot** consists of a manipulator mounted onto an overhead system that allows movement across a horizontal plane. Gantry robots are also called **Cartesian or linear robots**. They are usually large systems that perform **pick and place applications**, but they can also be used in **welding** and other applications like for observing the behaviour of patients in hospital wards. The fast motion of a gantry robot is usually associated with undesirable induced oscillations of the suspended object. Because of these oscillations, gantry robot manoeuvres are performed slowly contributing to low site efficiency and high transportation costs. These undesirable oscillations can be minimized at reasonably high travel speeds by designing effective controllers. To solve this problem precautions should also be considered so as not to use an unbalanced structure and also the most important was the mechanism of the robot. The robot structure requires high stability so that when the camera housing is positioned in the required locations it will not disturb and ensure the safety of the patient.

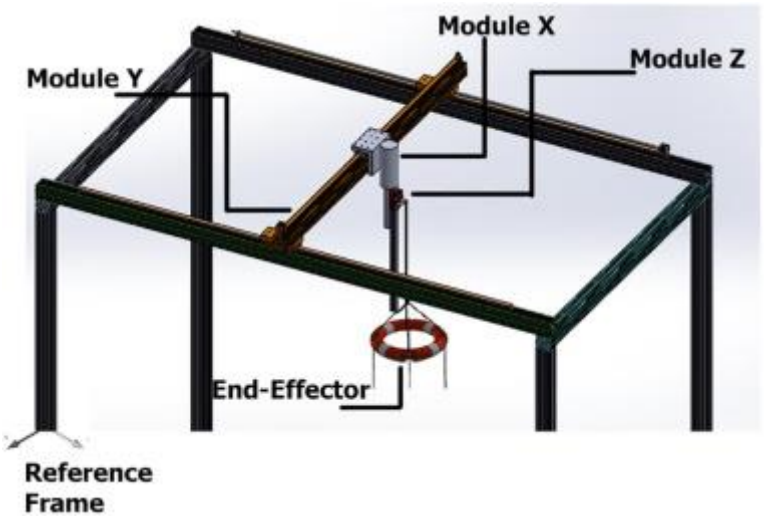
The **GANTRY rob**ot is most used for **pick-and-place** where high accuracy is required. Generally, a GANTRY robot can operate at a higher speed and with optional cleanroom specifications.

Industrial robots are defined as 'multi-functional manipulators designed to move parts through various programmed motions. As such, robots provide **consistently reliable performance,** and **repetitive accuracy** and are able to handle heavy workloads perform in harsh environments. Additionally, robots can be **quickly reprogrammed** to reflect changes in production needs and cycles.

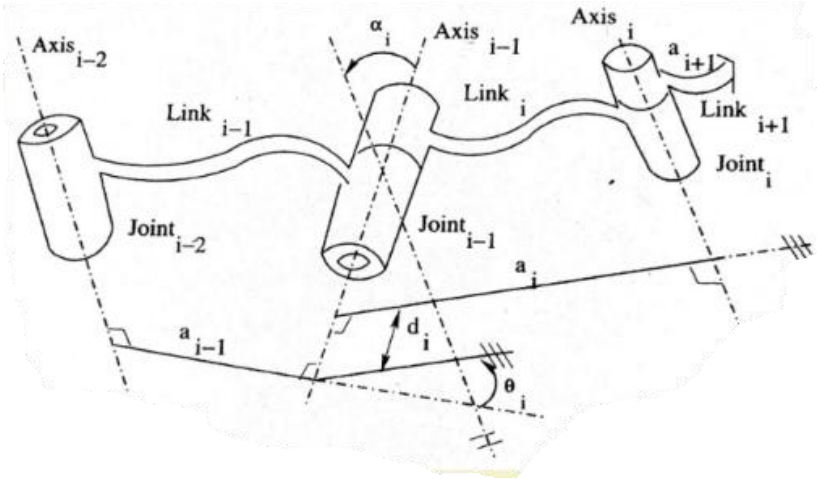


Robot Kinematics:

To specify the geometry of the planar 3R robot, we require **link length, joint angle, and twist angle parameters.** In the figure, the three **joint offsets** are labeled **X, Y, and Z**. These are obviously **variable**.



The precise definitions for the link lengths and joint angles are as follows.



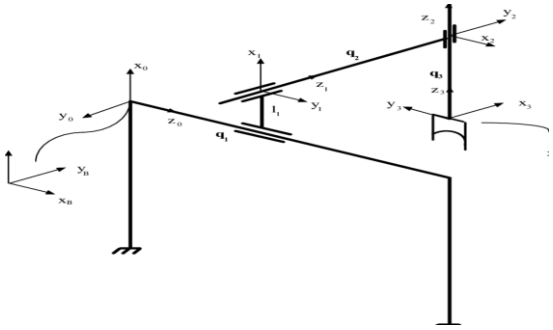
Length of link i (ai): It is the mutual perpendicular distance between Axisi-1 and Axisi

Angle of twist of link i (*αi*): It is defined as the angle between Axisi-1 and Axisi

Offset of link i (di): It is the distance measured from a point where ai-1 intersects the Axisi-1 to the point where ai intersects the Axisi-1 measured along the said axis

Joint Angle (θi): It is defined as the angle between the extension of ai-1 and ai measured about the Axis Notes: i-1

Another set of variables that is useful to define is the set of coordinates for the end effector. These coordinates define the position and orientation of the end effector. With a convenient choice of a reference point on the end effector, we can describe the position of the end effector using the coordinates of the reference point (x, y) and the orientation using the angle f. The three end effector coordinates (x, y, f) completely specify the position and orientation of the end effector.



Homogenous Transformation Matrix

To keep representation matrices square, if we represent both orientation and position in the same matrix, we will add the scale factors to the matrix to make it a 4 X 4 matrix. If we represent the orientation alone, we may either drop the scale factors and use 3x 3 matrices or add a 4th column with zeros for a position to keep the matrix square. Matrices of this form are called homogeneous matrices, and we write them as follows:

F is a 4x4 matrix that can describe a translation, rotation, or both in one matrix

$$F = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Could be rotation around the z-axis, x-axis, y-axis, or a combination of the three.

$$R_z = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} \qquad R_y = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix}$$

$$\mathrm{Trans}_{z_{n-1}}(d_n) = \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_n \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

$$\mathrm{Rot}_{z_{n-1}}(\theta_n) = \left[\begin{array}{cccc} \cos\theta_n & -\sin\theta_n & 0 & 0 \\ \sin\theta_n & \cos\theta_n & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

$$\mathrm{Trans}_{x_n}(r_n) = \left[\begin{array}{cccc} 1 & 0 & 0 & r_n \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

$$\mathrm{Rot}_{x_n}(\alpha_n) = \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_n & -\sin\alpha_n & 0 \\ 0 & \sin\alpha_n & \cos\alpha_n & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

$^{n-1}T_n = Screw_Z \text{ X } Screw_X$

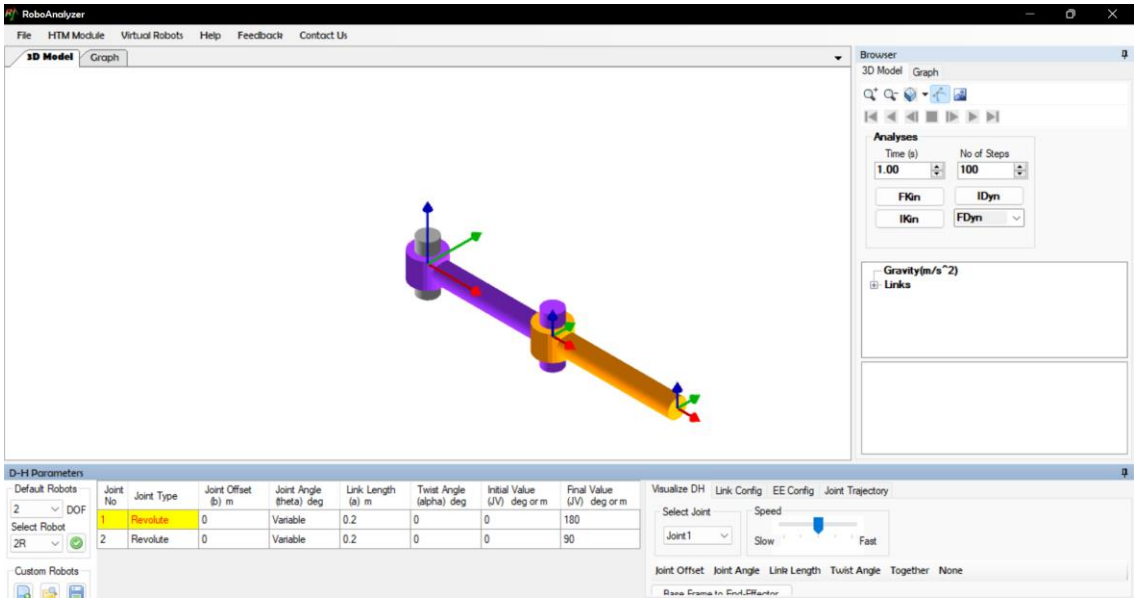
$$^{n-1}T_n = \left[\begin{array}{ccc|c} \cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\sin\alpha_n & r_n\cos\theta_n \\ \sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & r_n\sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} & R & & T \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

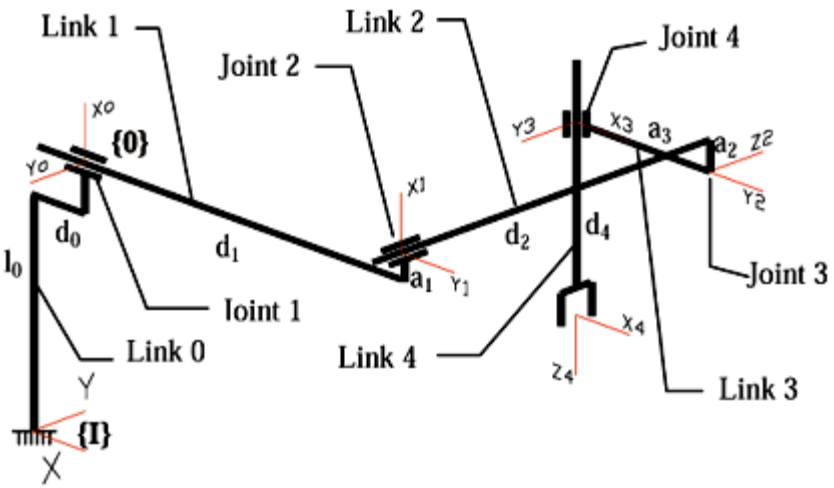
where *R* is the 3×3 submatrix describing rotation and *T* is the 3×1 submatrix describing translation.

Robo Analyzer:

RoboAnalyzer® is a 3D model-based software that can be used to teach and learn Robotics concepts.
It is an evolving product developed in the Mechatronics Lab, Department of Mechanical Engineering at IIT Delhi, New Delhi, India. Its development started under the guidance of Prof. S.K.Saha in order to support the learning/teaching of the topics covered in his book "Introduction to Robotics" published by Tata McGraw Hill, New Delhi.

Interface of Robo Analyzer Software





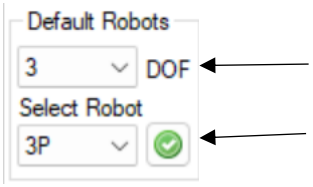
Joint Parameters

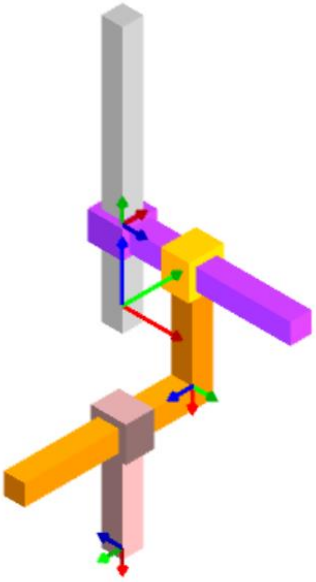
Joint(i)	Joint Type	Joint offset(b) meters	Joint angle(θ) degrees	Link length(a) meters	Link twist(α) degrees
1	Prismatic	0.1 to 0.3	90	0	90
2	Prismatic	0.1 to 0.2	=90	0.15	90
3	Prismatic	0.1 to 0.2	0	0.15	90

KINEMATIC ANALYSIS OF GANTRY ROBOT using robo analyzer

3D Model of 3P robot

First, we must import the GANTRY robot model into the scene so, we must change the Default option of Robot on the left side bottom-most bar. We must change the **DOF** of the robot to **3** and then we must select the **3P** then tap on the green tick mark to import our model.



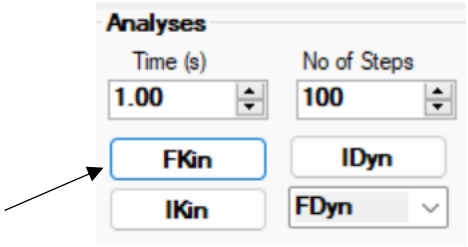


D-H Parameters

After importing the robot, we must change the joint parameters of our own values. Here I have measured the link lengths of the basic gantry robot from the **Model in CRProg** and selected the parameters as follows.

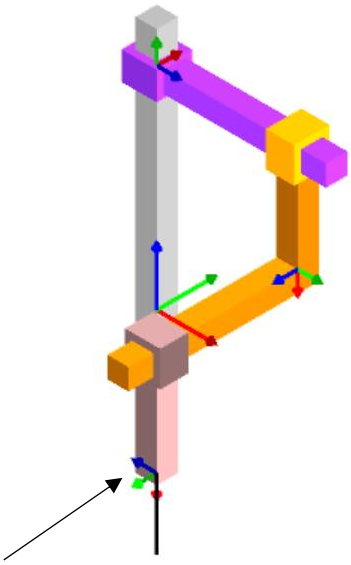
Joint No	Joint Type	Joint Offset (b) m	Joint Angle (theta) deg	Link Length (a) m	Twist Angle (alpha) deg	Initial Value (JV) deg or m	Final Value (JV) deg or m
1	Prismatic	Variable	90	0	90	0.1	0.3
2	Prismatic	Variable	-90	0.15	90	0.1	0.2
3	Prismatic	Variable	0	0.15	90	0.1	0.2

Analysis

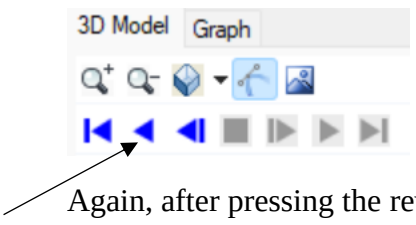


In the right-side bottom-most bar, we can see the analysis bar, we must tap on the **FKin (Forward Kinematics)** option to complete the analysis of Forward Kinematics.

After that, we must press the play button to analyze the Forward Kinematics of the robot and the robot will start moving and we can see in the below image that the end effector stops at the desired position of the given joint parameters data.

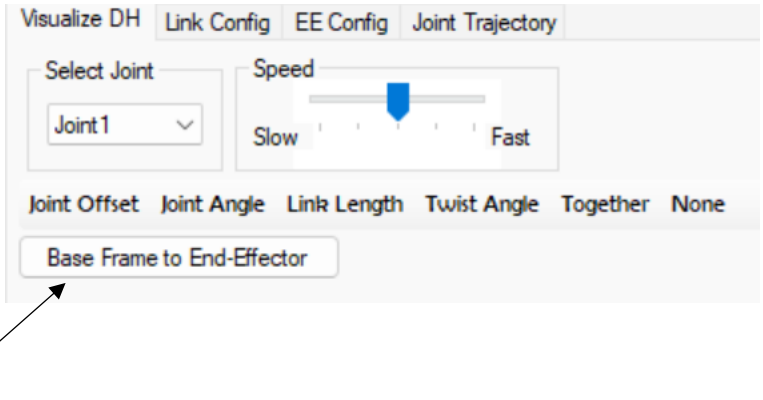


Isometric View of Robot (FKin)



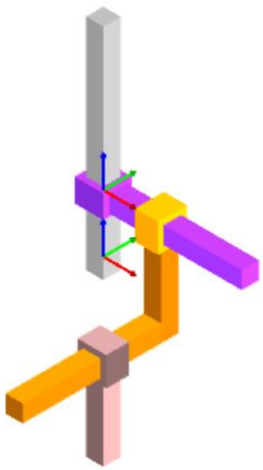
Again, after pressing the rewind button, the robot comes to the initial position.

Frame Transformation

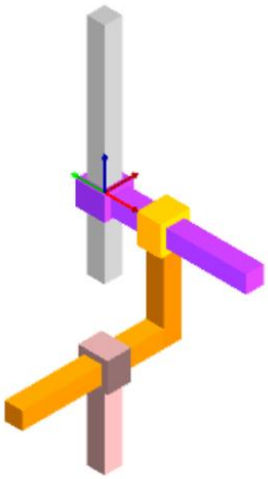


- **Joint 1**

Joint offset = 0.1 to 0.3(variable), so the frame of the axis move in Z-direction

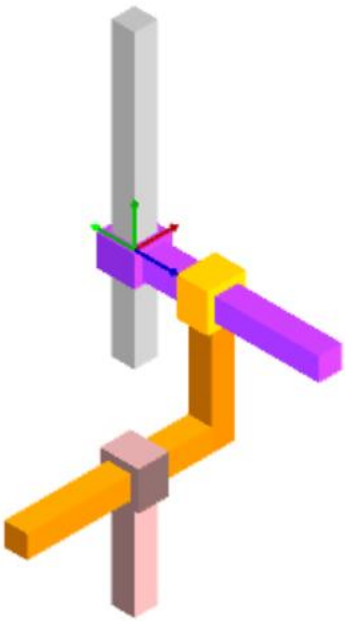


Joint angle = 90



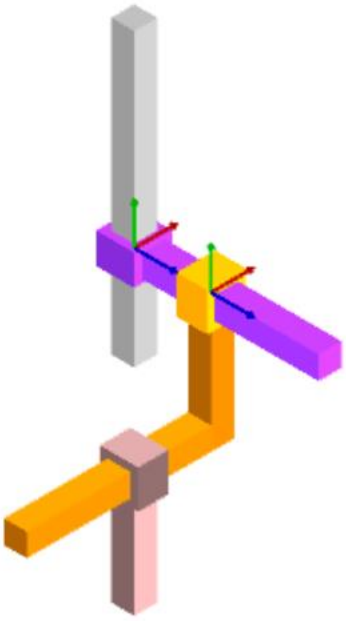
Link length = 0m so the axis translates 0m along X-direction

Link twist = 90, so the frame doesn’t make an angle between the (Z)Axisi-1 and (Z)Axisi

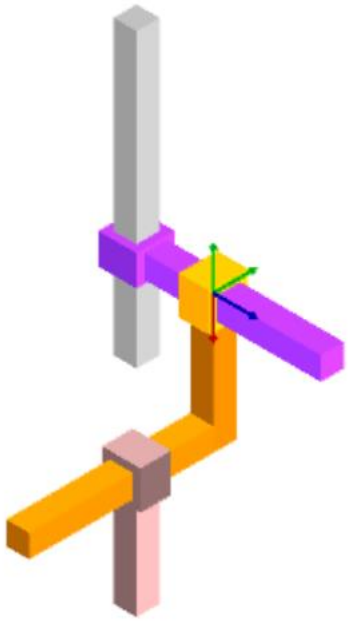


- **Joint 2**

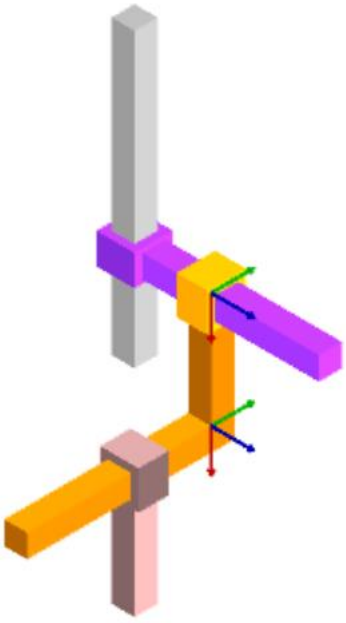
Joint offset = 0.1 to 0.2m(variable), so the frame of the axis move in Z-direction



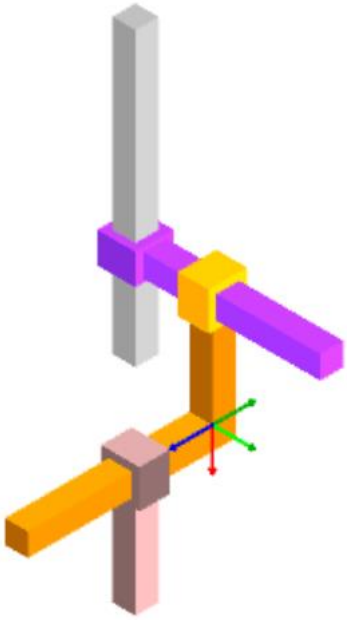
Joint angle = -90



Link length = 0.15m so the axis translates 0.1m along X-direction

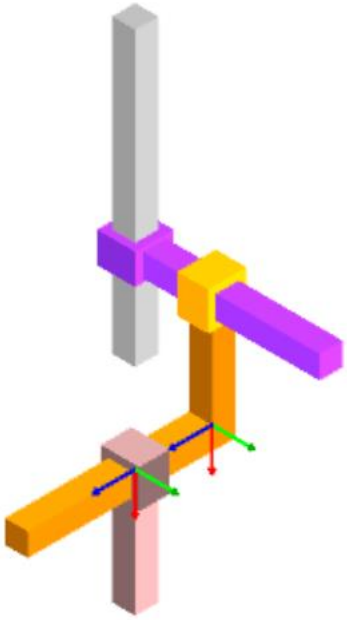


Link twist = 90, so the frame make an angle between the (Z)Axisi-1 and (Z)Axisi



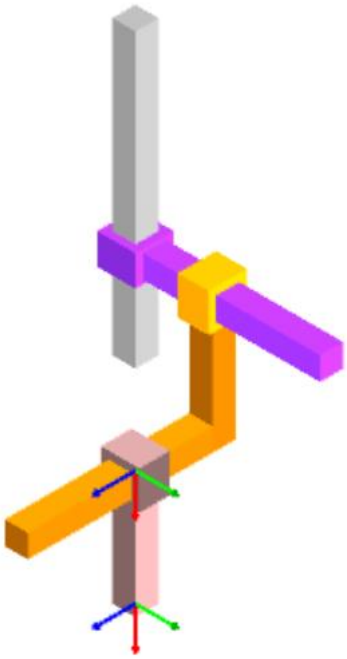
- **Joint 3**

Joint offset = 0.1 to 0.2 (variable), so the frame of the axis moves in Z-direction

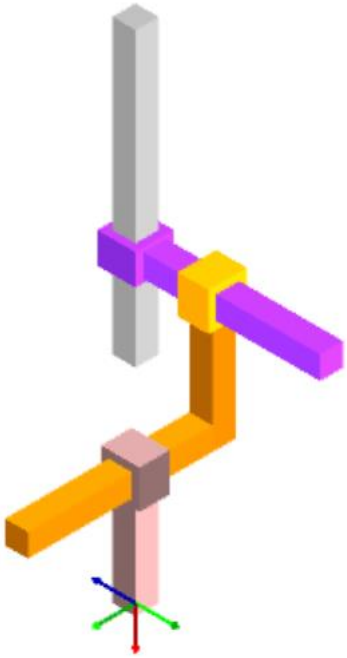


Joint angle = 0

Link length = 0m so the axis doesn’t translates along the X-direction



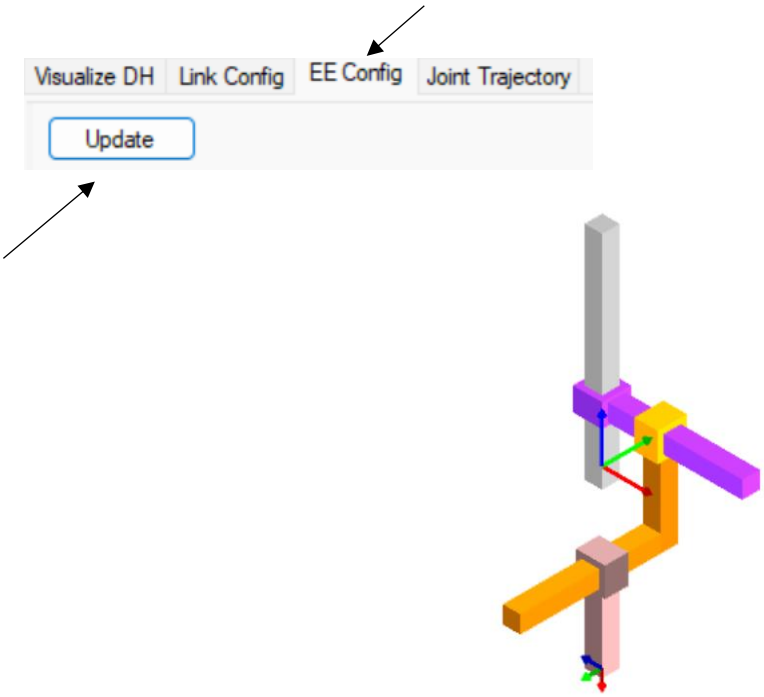
Link twist = 0, so the frame doesn’t make an angle between the (Z)Axisi-1 and (Z)Axisi



End Effector Configuration

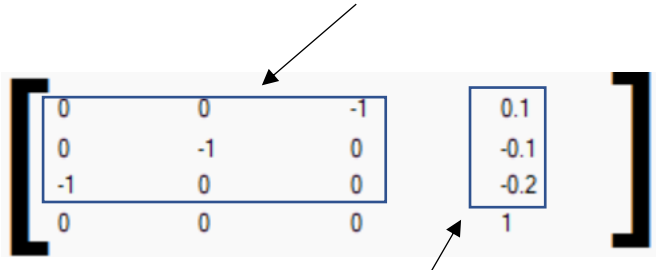
- **Initial Configuration**

To obtain the end effector configuration we need to tap on **EEConfig** which is in the bottom of the right side and then we need to tap on update.



After tapping on update we will get the initial end effector configuration or transformation matrix

Rotation Matrix (There is a rotation of the end effector)

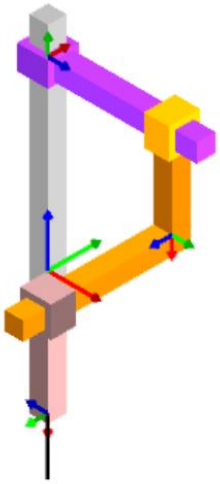


Translation Matrix (End effector is (0.1m,-0.1m,-0.2m) distance away from the base frame at initial configuration)

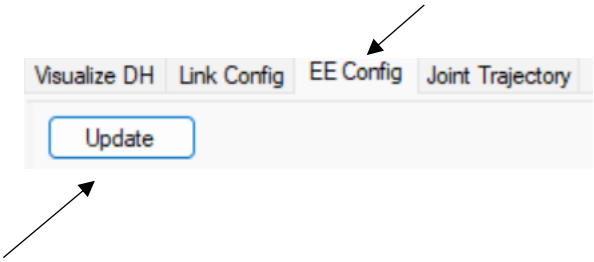
- **Final Configuration**

Now to obtain the final end effector configuration we need to move the robot's end effector to the final position by applying the forward kinematics of the given join parameters data.

After applying the Forward Kinematics, the robot comes into the desired end position.

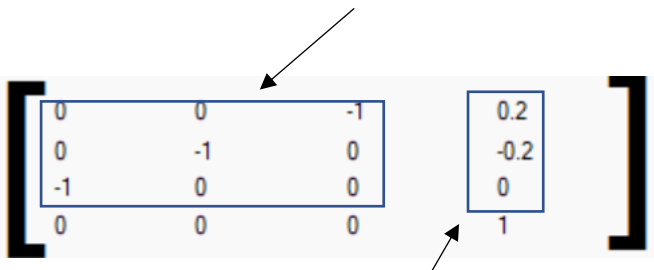


To obtain the final end effector configuration we need to tap on **EEConfig** which is in the bottom of the right side and then we need to tap on update.



After tapping on the update, we will get the final end effector configuration or transformation matrix

Rotation Matrix (The final frame is oriented about the Z axis w.r.t base frame)



Translation Matrix (End effector is (0.2m,-0.2m,0m) distance away from the base frame at final configuration)

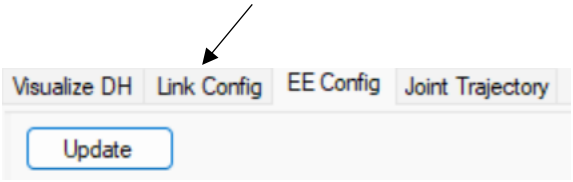
Link Configuration

We know that

- ⇨ transformation matrix of link 3 w.r.t base frame = transformation matrix of link 1 w.r.t base frame X transformation matrix of link 2 w.r.t 1 X transformation matrix of link 3 w.r.t 2

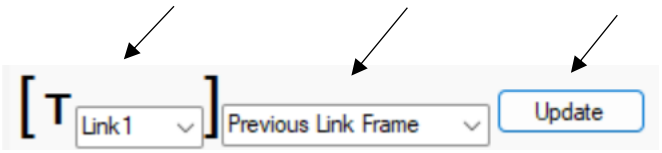
- ⇨ ${}^{0}T_{3} = {}^{0}T_{1} x {}^{1}T_{2} x {}^{2}T_{3}$

To operate the link configuration, we need to tap on **Link Config**



After that, we must select the main frame and reference frame accordingly, and then we need to update them to get the Transformation matrix.

Main frame Reference frame Gives the Transformation matrix



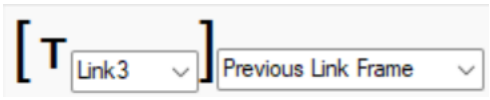
- **Link 1 w.r.t Base frame**

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0.3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Link 2 w.r.t 1**

$$\begin{bmatrix} 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & -0.15 \\ 0 & 1 & 0 & 0.2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Link 3 w.r.t 2**



$$\begin{bmatrix} 1 & 0 & 0 & 0.15 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0.2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Link 3 w.r.t base frame**

$$\begin{bmatrix} 0 & 0 & -1 & 0.2 \\ 0 & -1 & 0 & -0.2 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Hence,

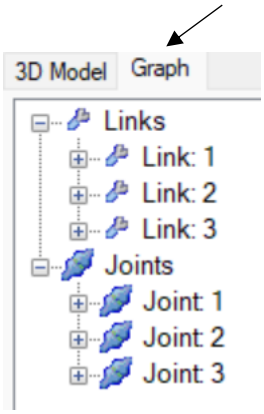
$$\begin{bmatrix} 0 & 0 & -1 & 0.2 \\ 0 & -1 & 0 & -0.2 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ({}^{0}T_{3})$$

$$({}^{0}T_{1}) \qquad ({}^{1}T_{2}) \qquad ({}^{2}T_{3})$$

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0.3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & -0.15 \\ 0 & 1 & 0 & 0.2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0.15 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0.2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Graphs

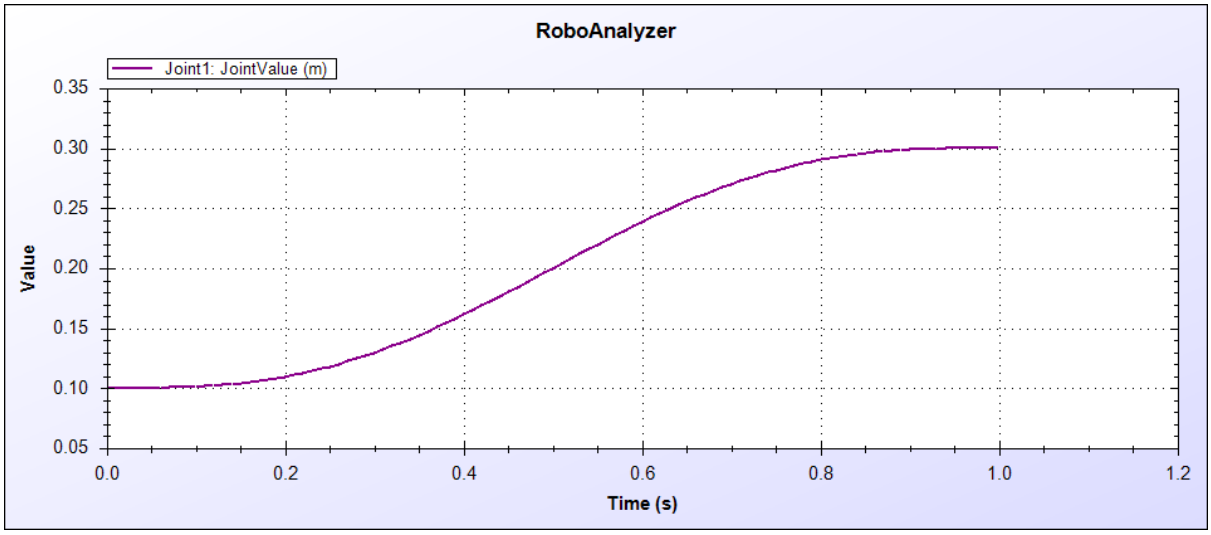
To get the different graphs of joints (joint angle, joint velocity, etc..) and links in robo analyzer 1st we must tap on the **FKin** option, and then we need to tap on the **Graph** option which is on the top right side. Then we need to select the different options which we see in the below image to access different graphs



Joint Angle Graphs

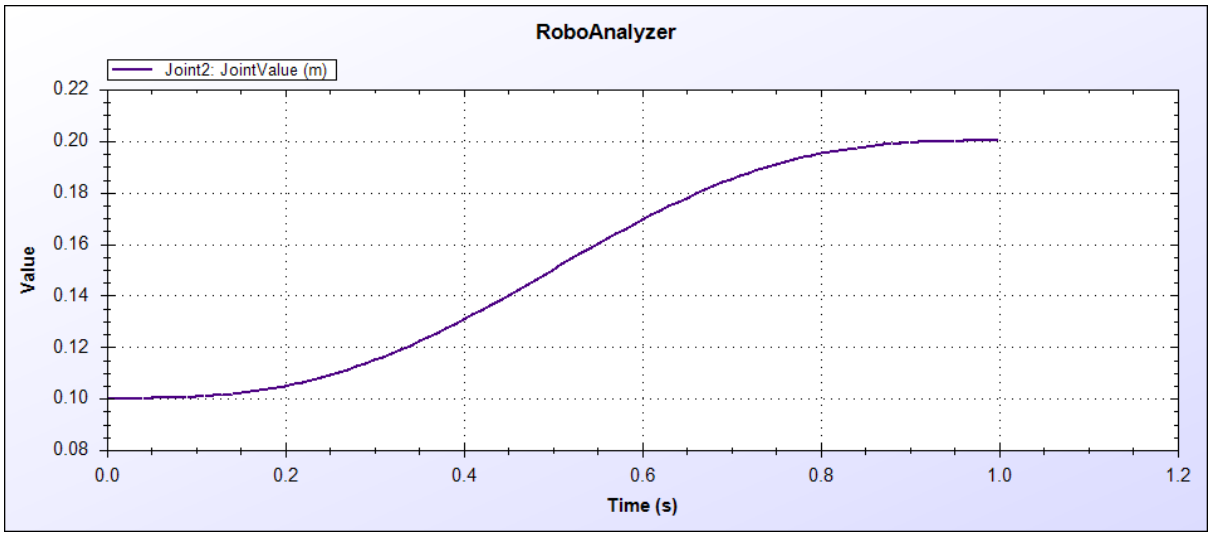
- **Joint 1**

  The joint offset of joint 1 is variable as time increases the movement of the joint starts from 0.1m and ends at 0.3m within 1sec and the graph will be a nonlinear curve.



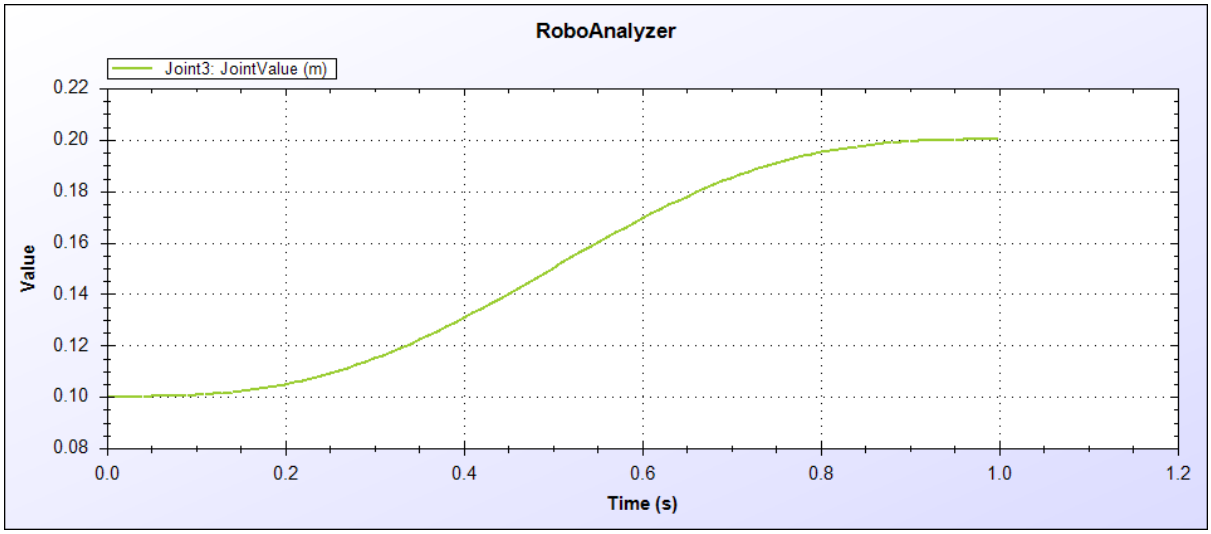
- **Joint 2**

  The joint offset of joint 2 is variable as time increases the movement of the joint starts from 0.1m and ends at 0.2m within 1sec and the graph will be a nonlinear curve.



- **Joint 3**

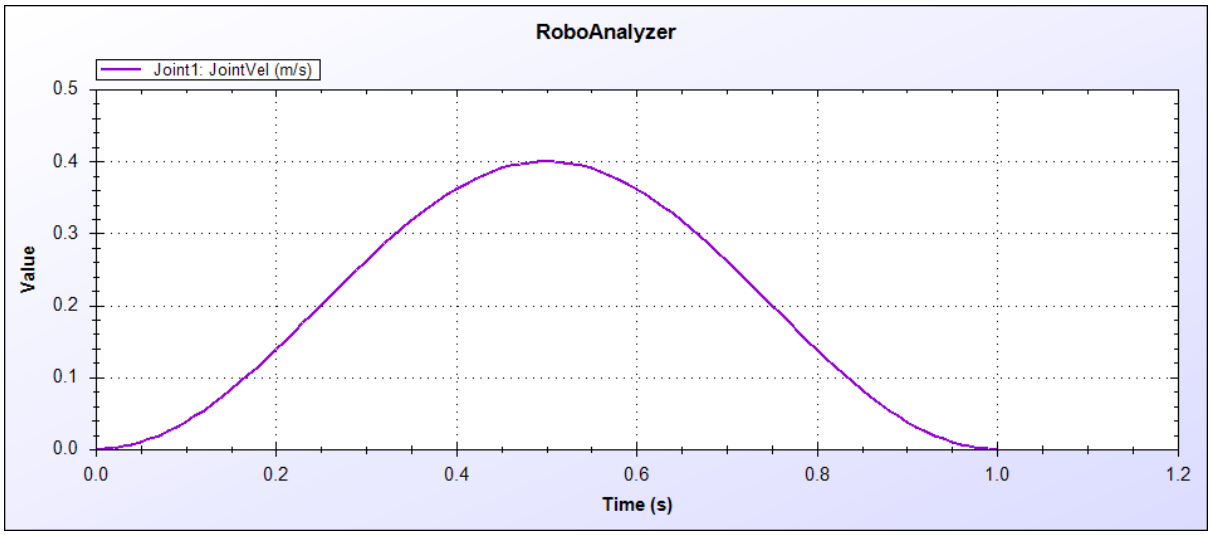
  The joint offset of joint 3 is variable as time increases the movement of the joint starts from 0.1m and ends at 0.2m within 1sec and the graph will be a nonlinear curve.



Joint Velocity Graphs

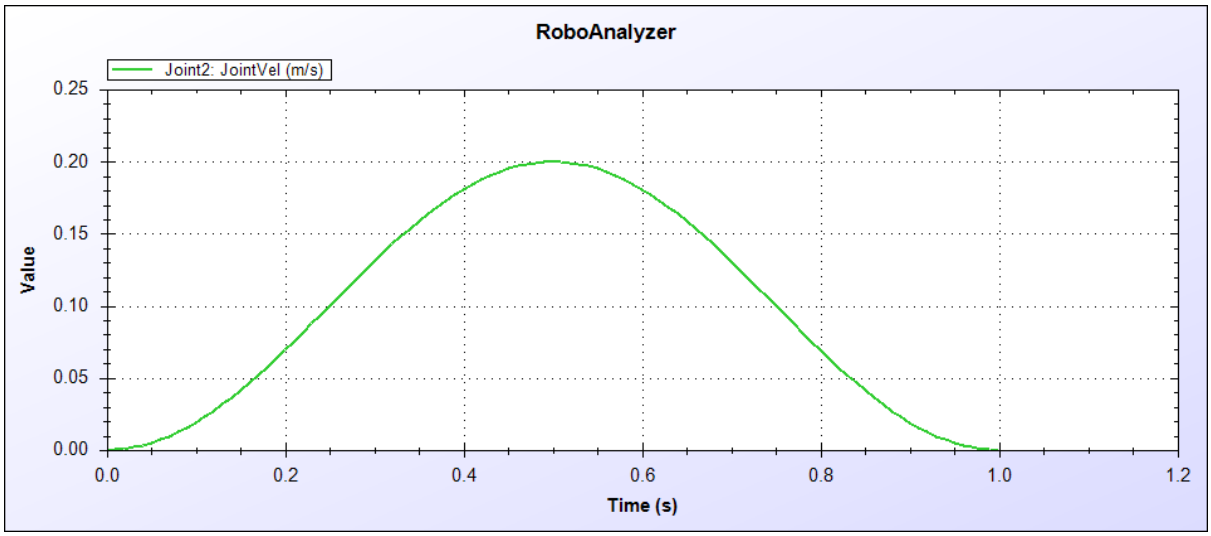
- **Joint 1**

  The joint velocity of joint 1 is increasing as time increases up to half of the total period of settling time and then the velocity of the joint starts decreasing gradually within 1sec, and the graph will be an exponential curve.



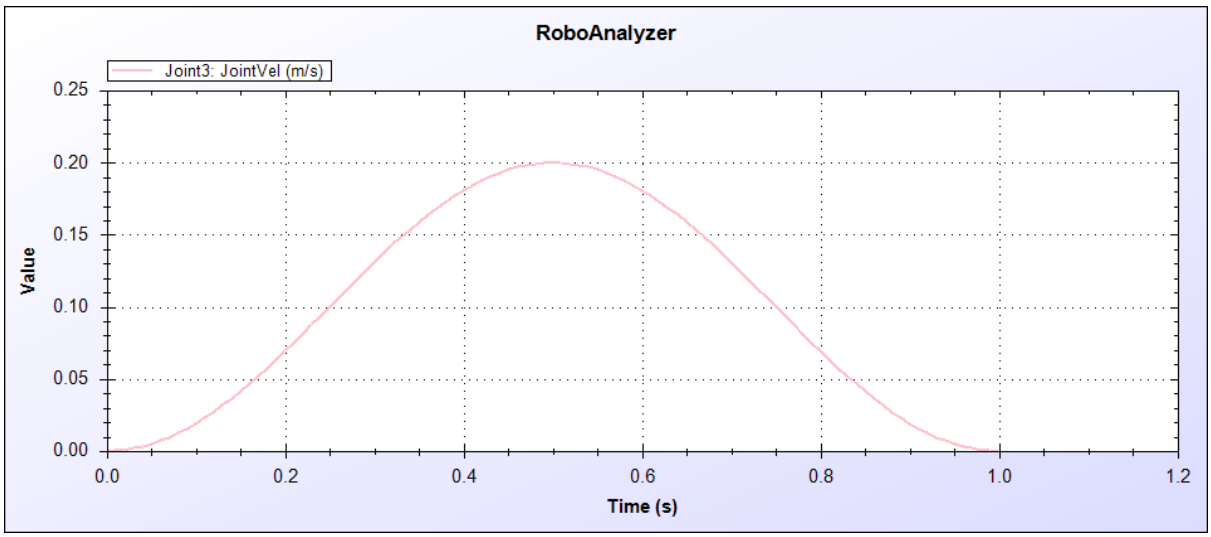
- **Joint 2**

  The joint velocity of joint 2 is increasing as time increases up to half of the total period of settling time and then the velocity of the joint starts decreasing gradually within 1sec, and the graph will be an exponential curve.



- **Joint 3**

  The joint velocity of joint 3 is increasing as time increases up to half of the total period of settling time and then the velocity of the joint starts decreasing gradually within 1sec, and the graph will be an exponential curve.



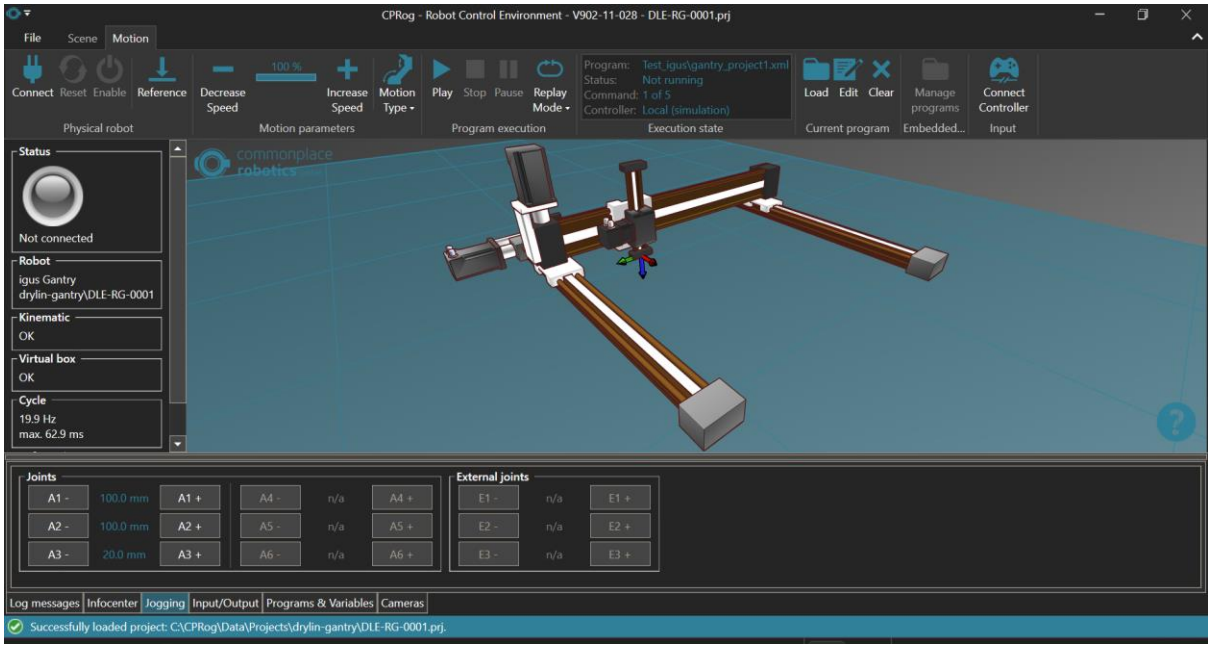
Commonplace robotics programming (CPRog)

The simulation and programming environment **CPRog** offers the possibility to program robots with the help of an extensive 3D visualization.

This software can be used as a complete controller in the training and maker area.

For industrial use, the **Linux-based real-time extension CPRog** Core must be interposed, to which the created programs are loaded. The big advantage of CPRog Core is the **increased reliability in data transfer** and significantly minimized latencies. Together CPRog and CPRog Core are **optimum to control your robot**.

Interface of CPRog Software

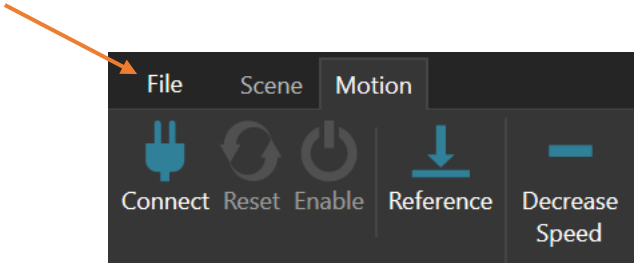


Kinematic analysis and Programming using CPRog

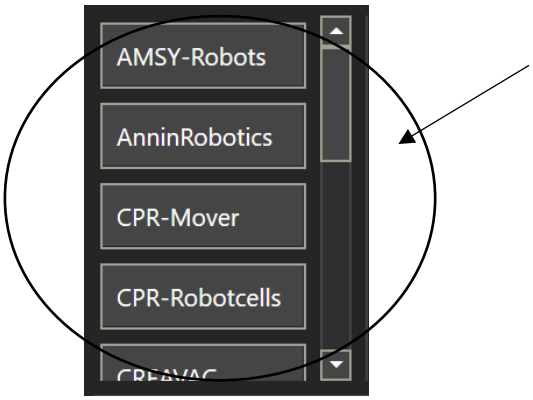
Industrial automation deals primarily with the automation of manufacturing, quality control, and material handling processes. The Gantry system is one of the pioneer platforms in industrial material handling systems.

A range of software tools and methods were found to be useful and necessary for efficient engineering and integration. In the present work The kinematic analysis of gantry
The robot is done using a Robo analyzer and C-prog software.

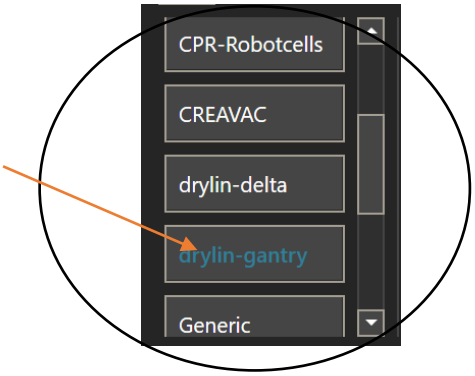
To begin Kinematic programming, we need to tap on the **FILE** option which is on the topmost left side of the interface.



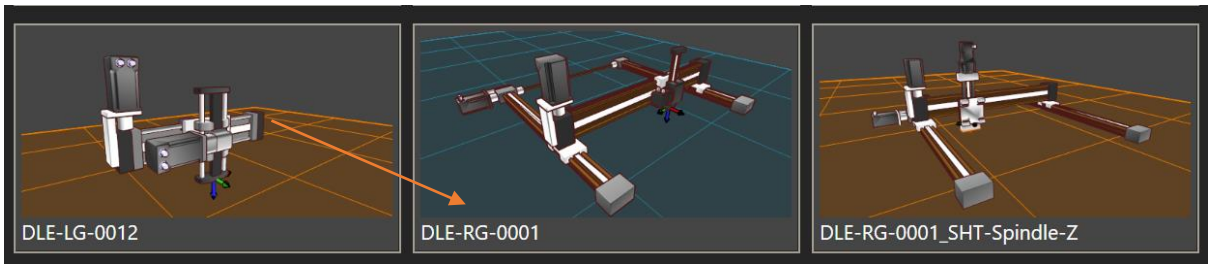
Then we will see these options on the screen



First, we need to add the **ROBOT** in the scene so, we need to tap on the

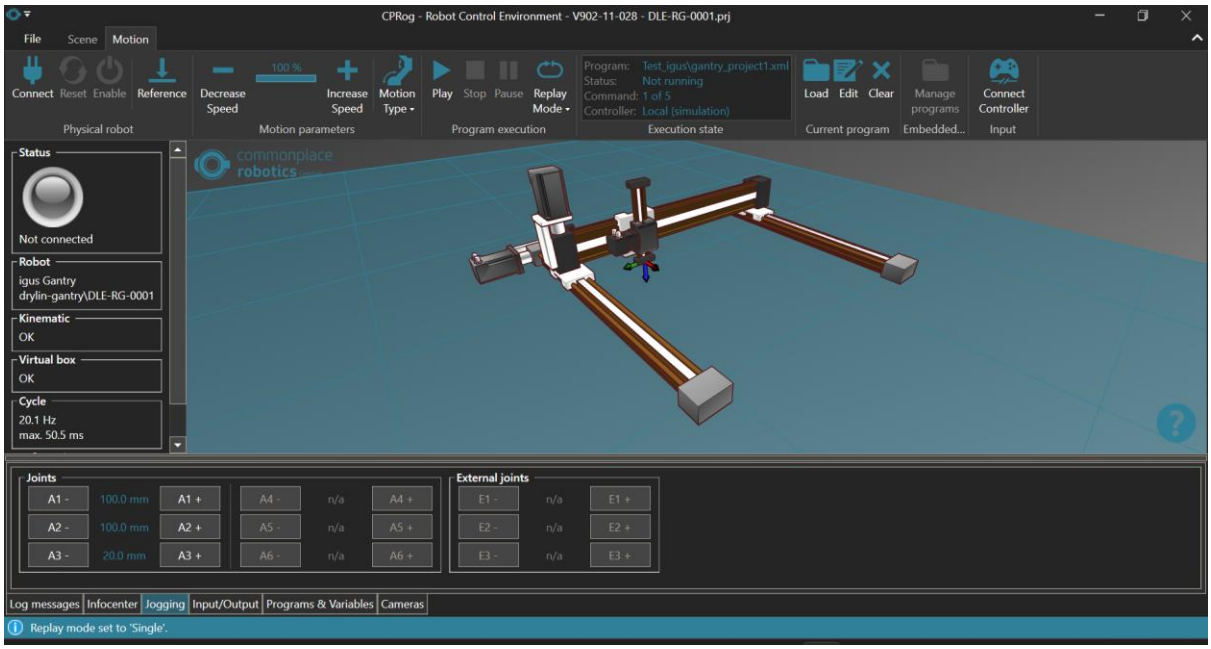


drylin-gantry option. Now we need to tap on the DLE-RG-001 Robot.

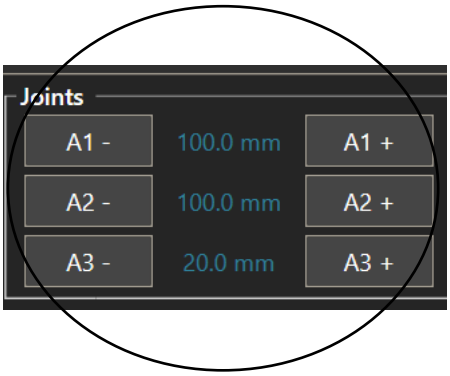


Now we will see this page, where the required robot got loaded into the scene.

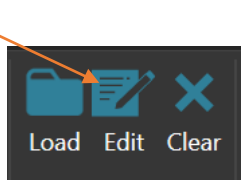
Axis representation → RED – X-axis GREEN – Y-axis, BLUE – Z-axis



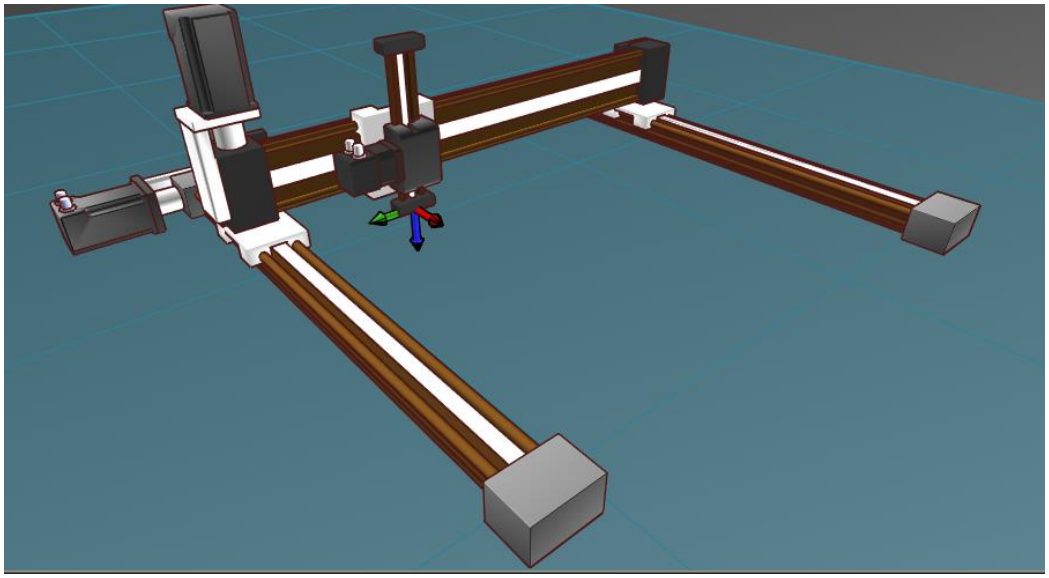
On the bottommost left side, we can see the joint offset variable slider which is used to set a particular value of joint offset values in code or we can directly change the values in parameters.



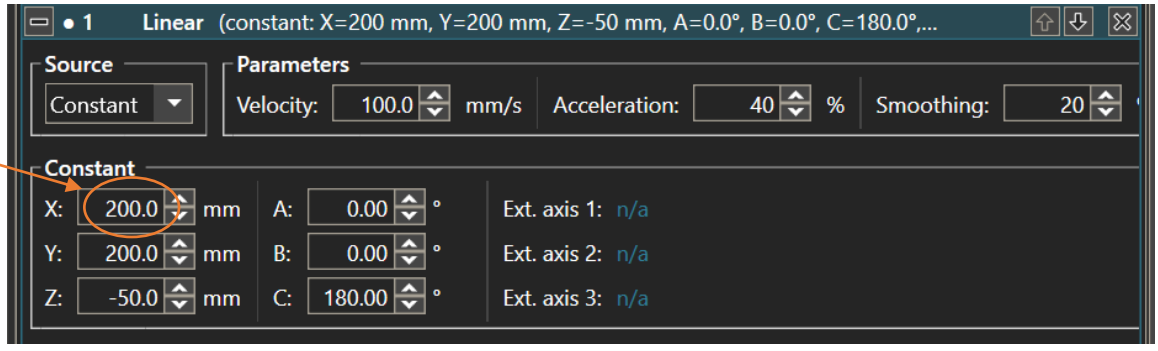
Now **clear** the program by using the clear option and we must select the **Edit** option which is top right, to edit the code of the gantry program.

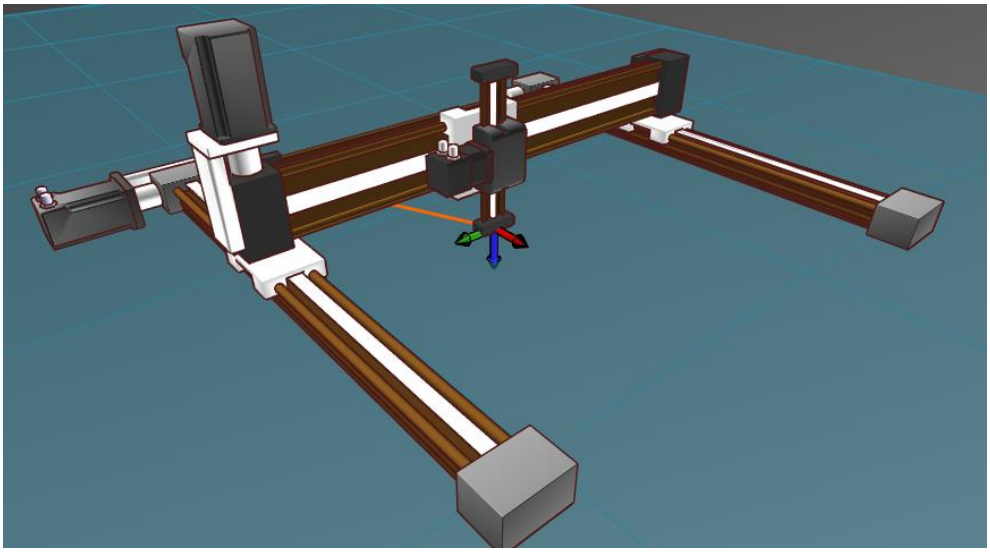


Now choose the **file** and **select new** we will see the following tab.



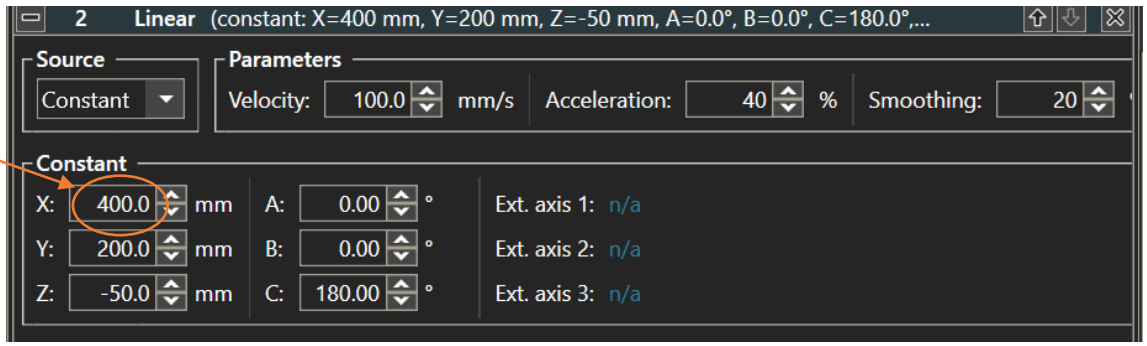
Now, tap on the action and select Linear motion and edit the X coordinate to 200mm.

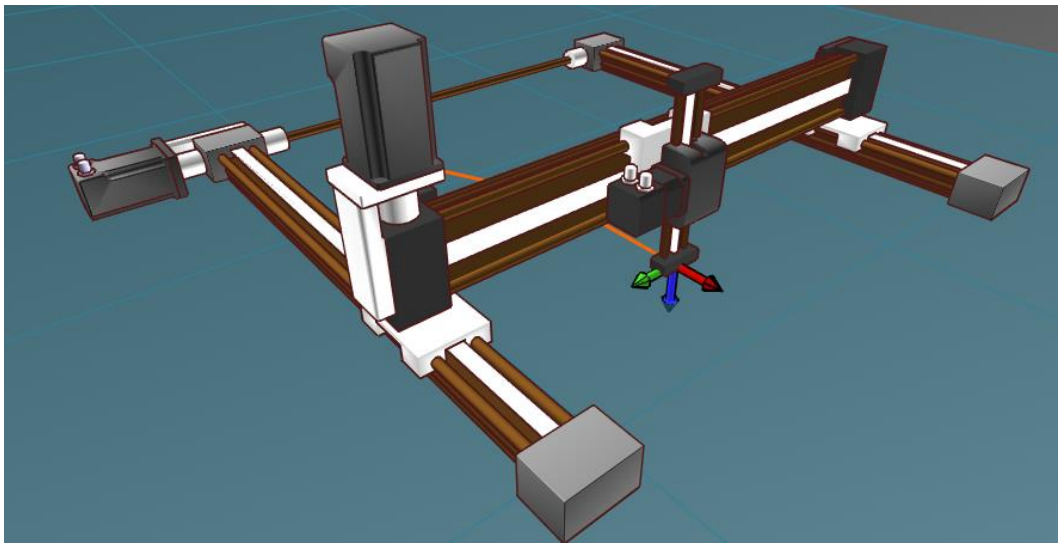




Now, tap on the action, again select Linear motion and edit the X coordinate to 400mm.

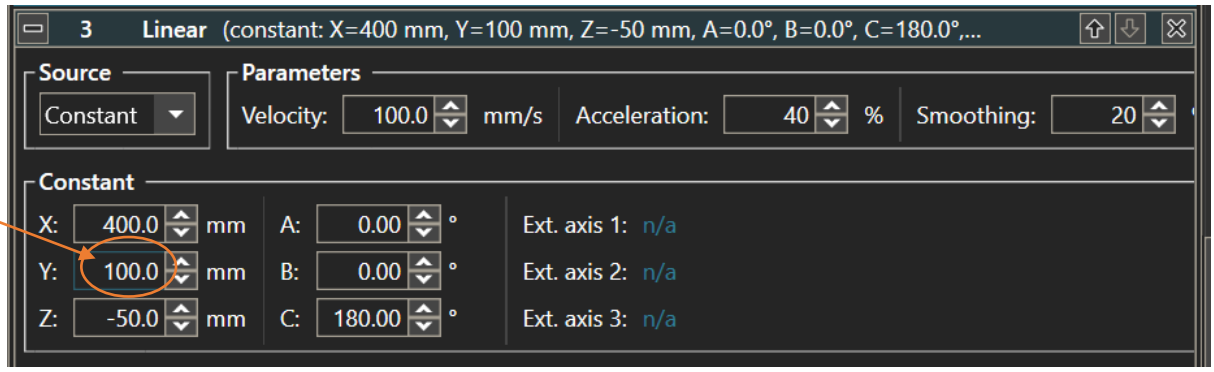
This makes the gantry robot move STRAIGHT in x direction from 200mm to 400mm

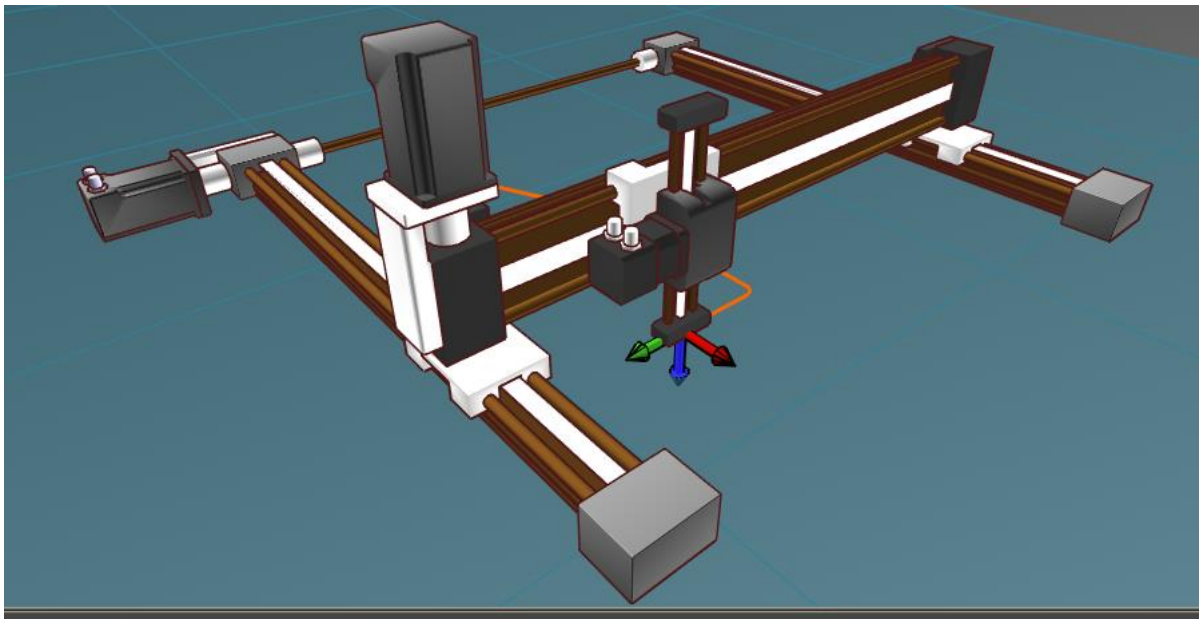




Now, tap on the action, again select Linear motion and edit the Y coordinate to 100mm.

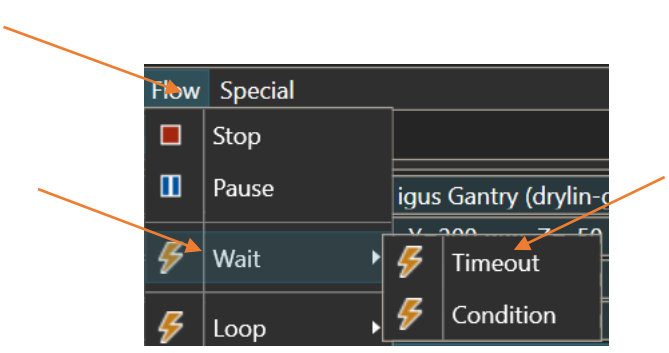
This makes the gantry robot move LEFT in the y direction from 200mm to 100mm

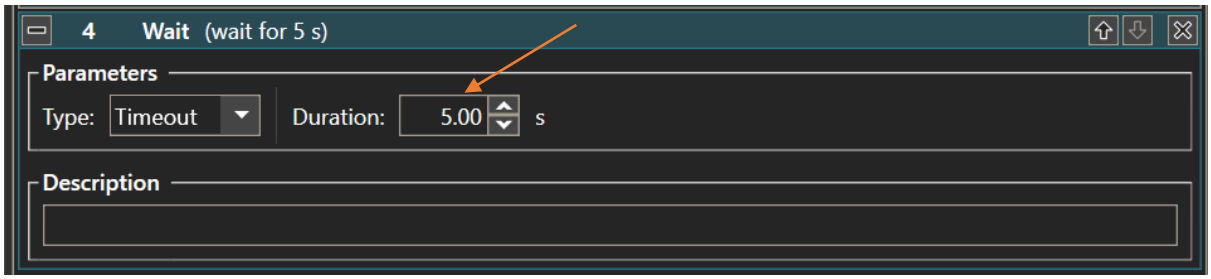




Now, tap on the flow, select wait, and timeout. Edit the timeout to 5sec.

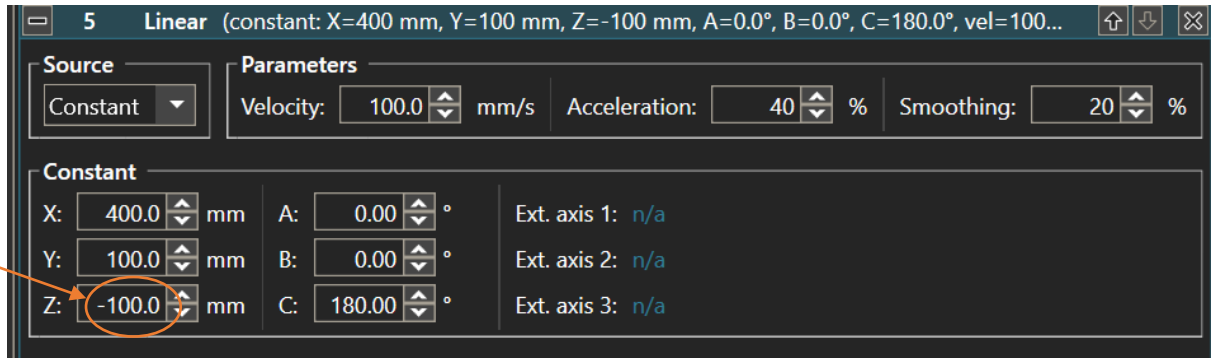
This makes the gantry stop for 5sec (Let us assume that object will come to that coordinate under 5sec). And then it will continue its action.

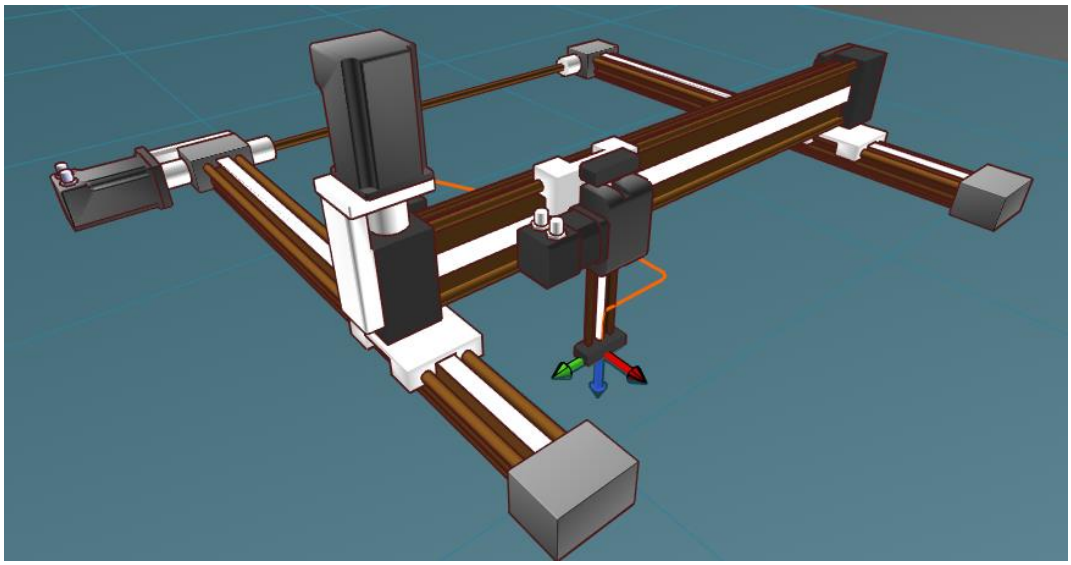




Now, tap on the action, again select Linear motion, and edit the Z coordinate to -100mm.

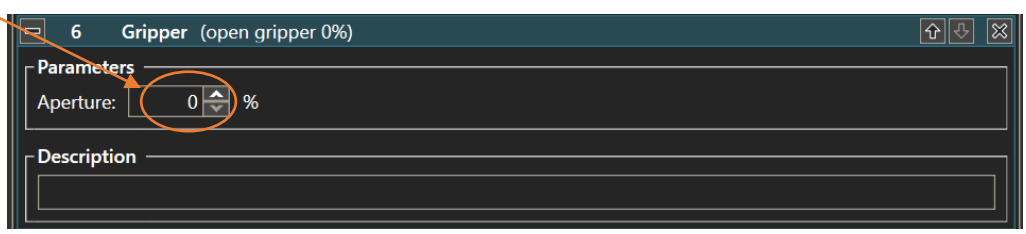
This makes the gantry robot move DOWN in a z-direction from -50mm to -100mm





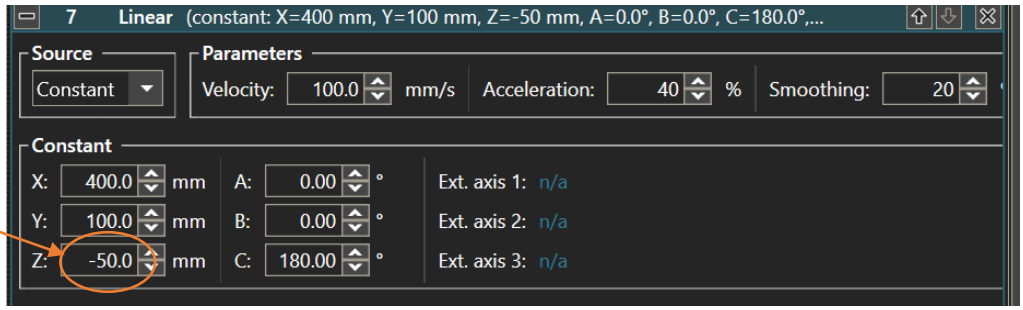
Now, tap on the action, select gripper, and edit the Aperture to 0%.

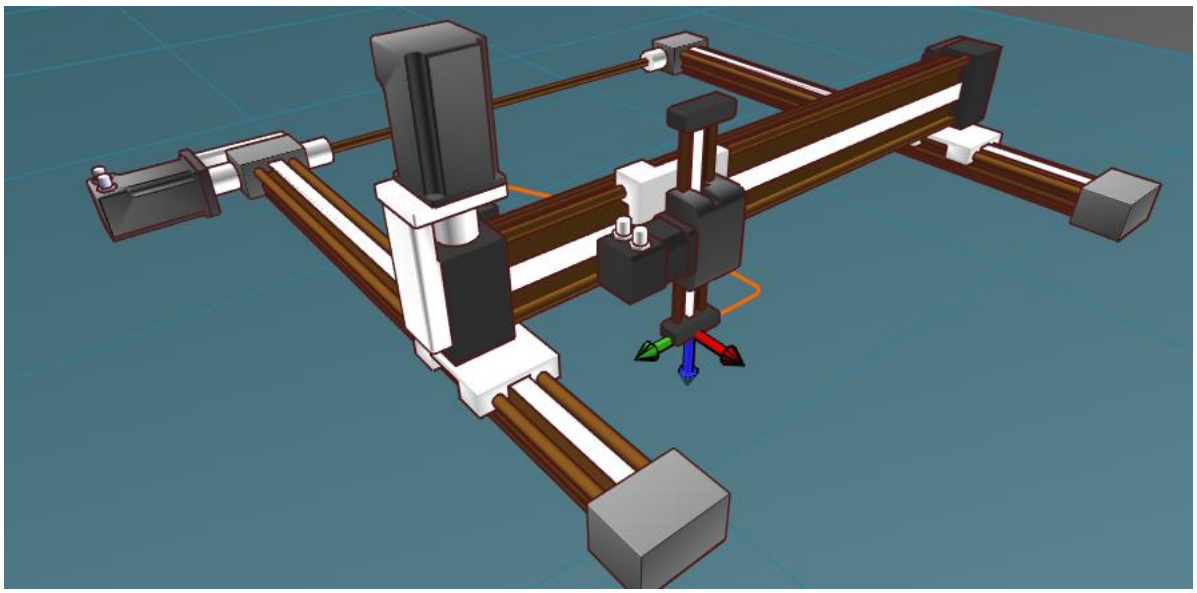
This makes the gantry robot pick the object or closes the gripper.



Now, tap on the action, again select Linear motion, and edit the Z coordinate to -50mm.

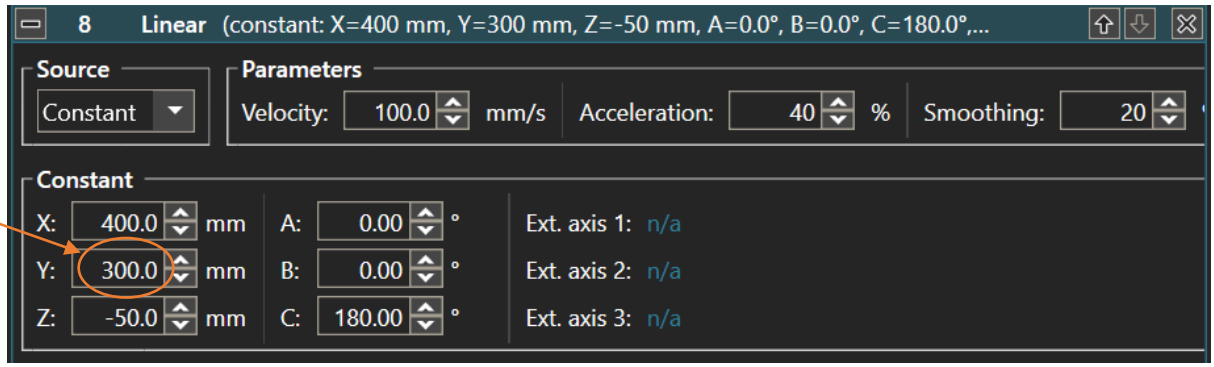
This makes the gantry robot move UP in a z-direction from -100mm to -50mm

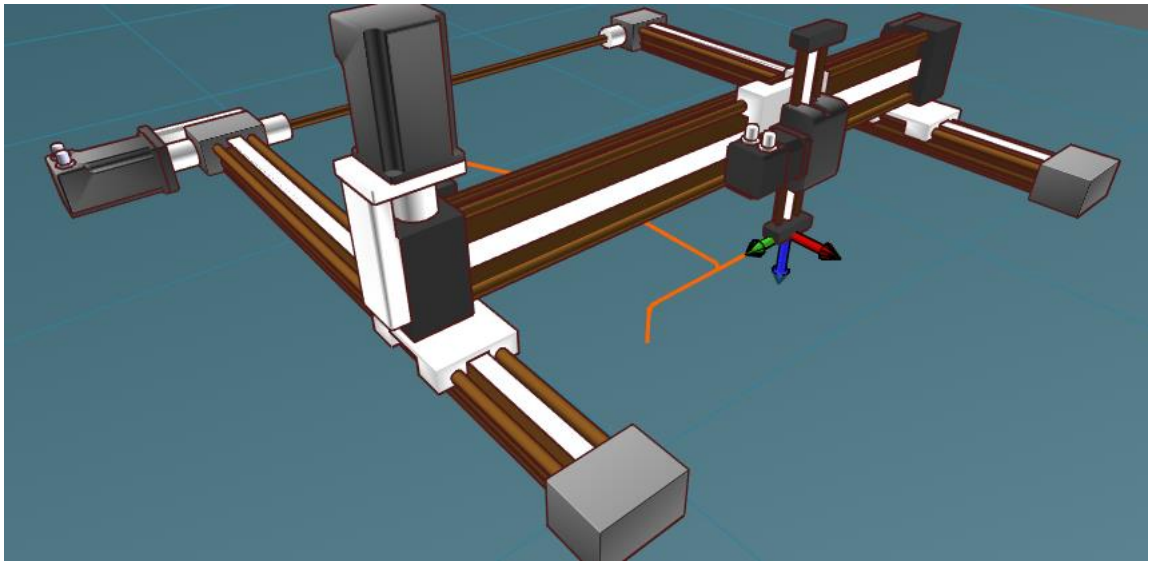




Now, tap on the action, again select Linear motion and edit the Y coordinate to 300mm.

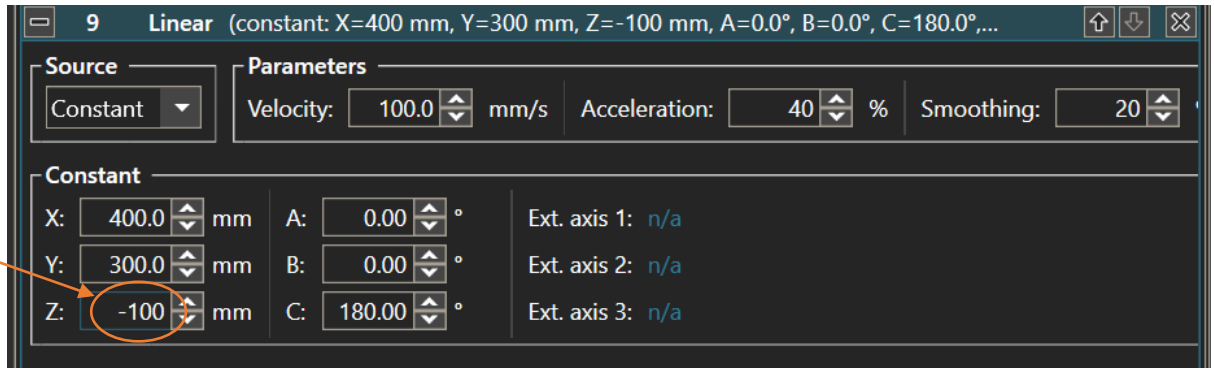
This makes the gantry robot move RIGHT in the y direction from 100mm to 300mm

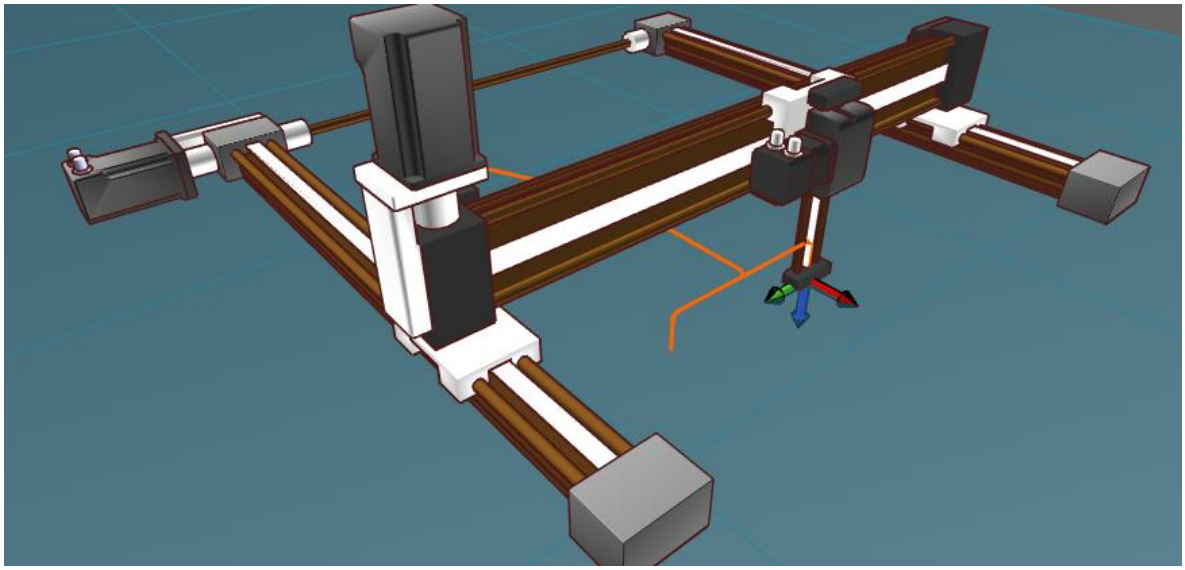




Now, tap on the action, again select Linear motion, and edit the Z coordinate to -100mm.

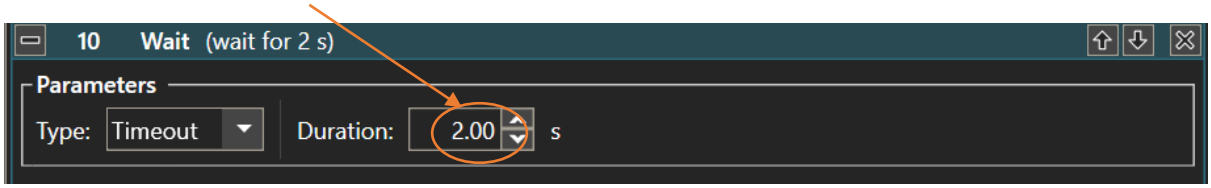
This makes the gantry robot move DOWN in a z-direction from -50mm to -100mm





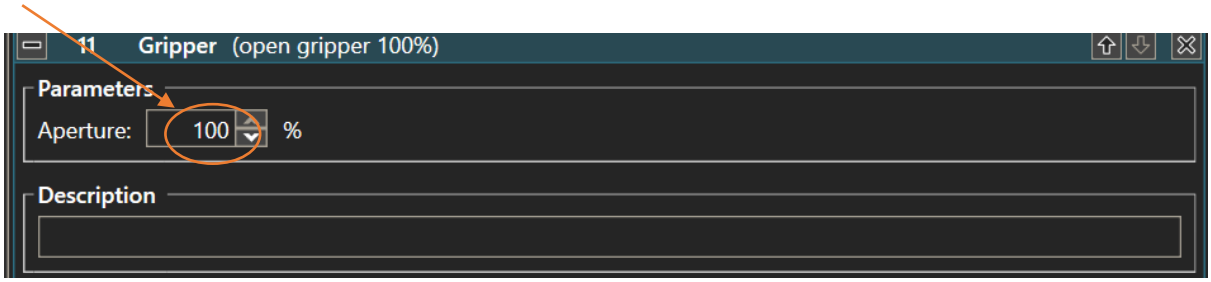
Now, tap on the flow, select wait, and timeout. Edit the timeout to 2sec.

This makes the gantry stop for 2sec. And then it will continue its action.



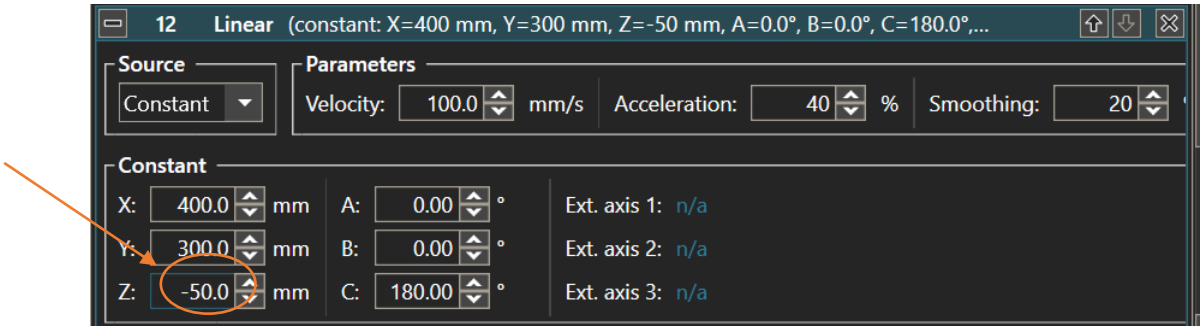
Now, tap on the action, select gripper, and edit the Aperture to 100%.

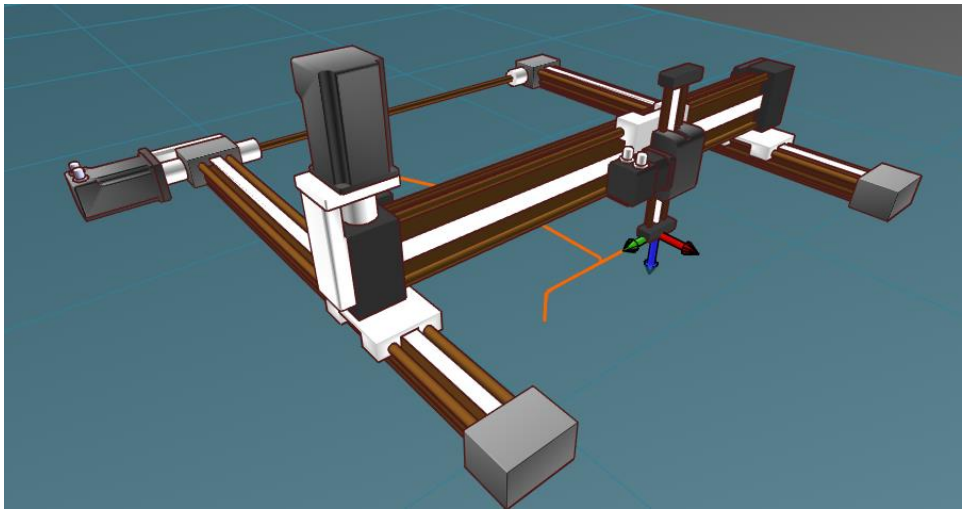
This makes the gantry robot drop the object or opens the gripper.



Now, tap on the action, again select Linear motion, and edit the Z coordinate to -50mm.

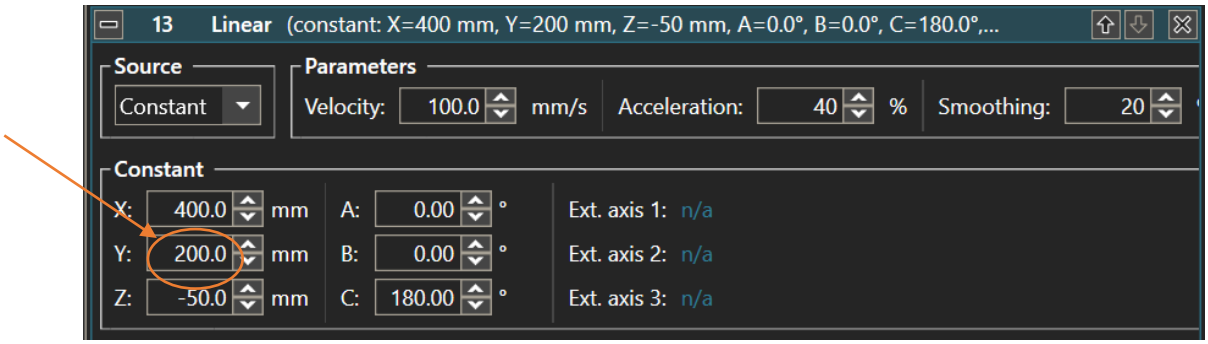
This makes the gantry robot move UP in a z-direction from -100mm to -50mm

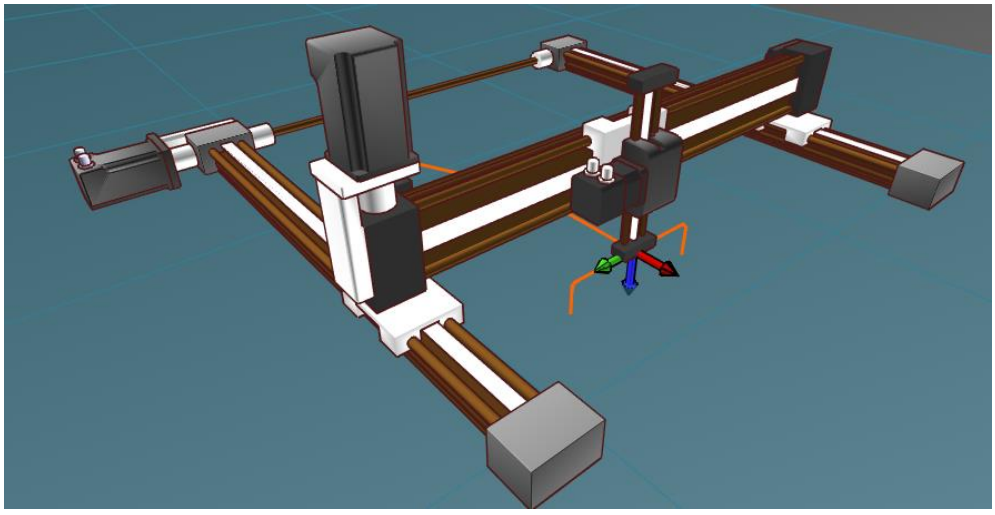




Now, tap on the action, again select Linear motion and edit the Y coordinate to 200mm.

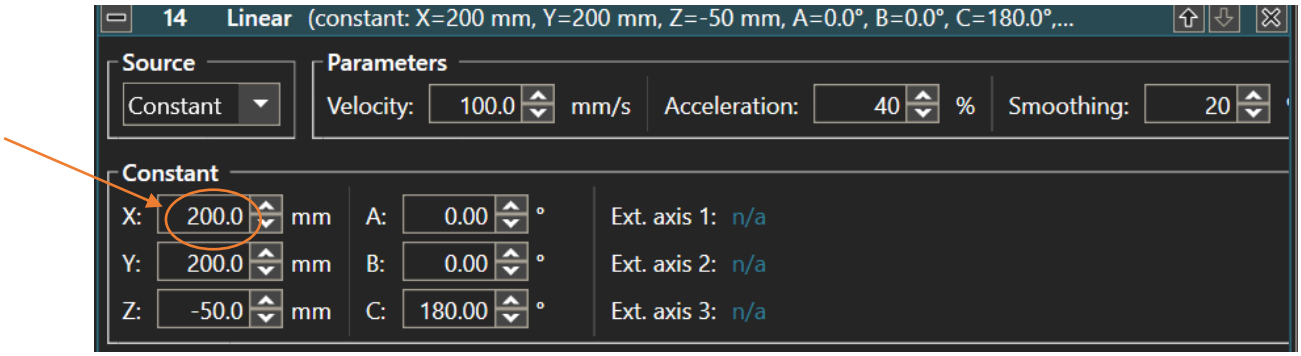
This makes the gantry robot move LEFT in y direction from 300mm to 200mm

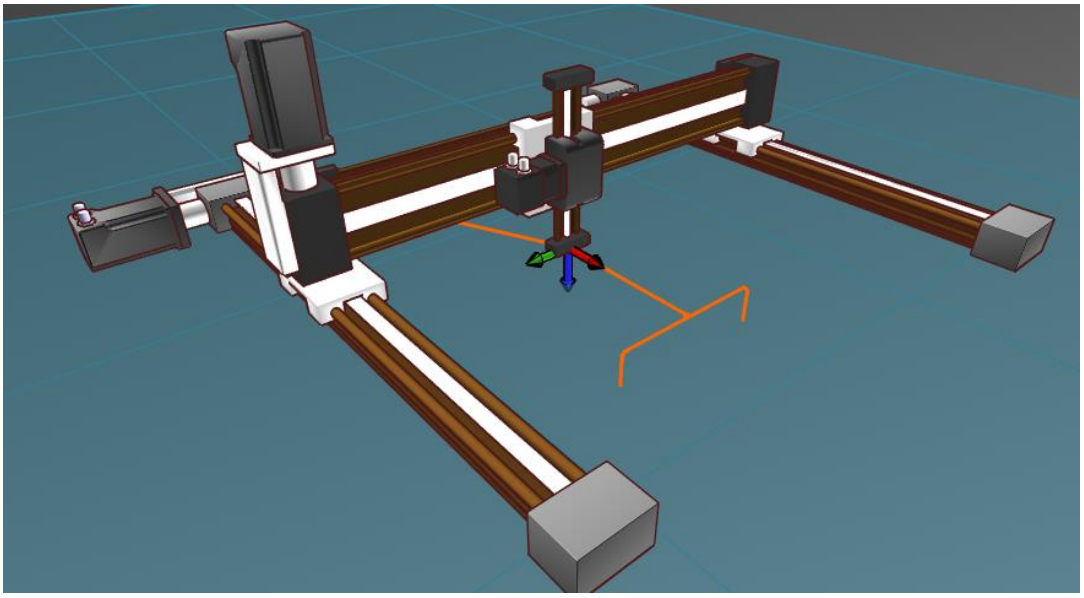




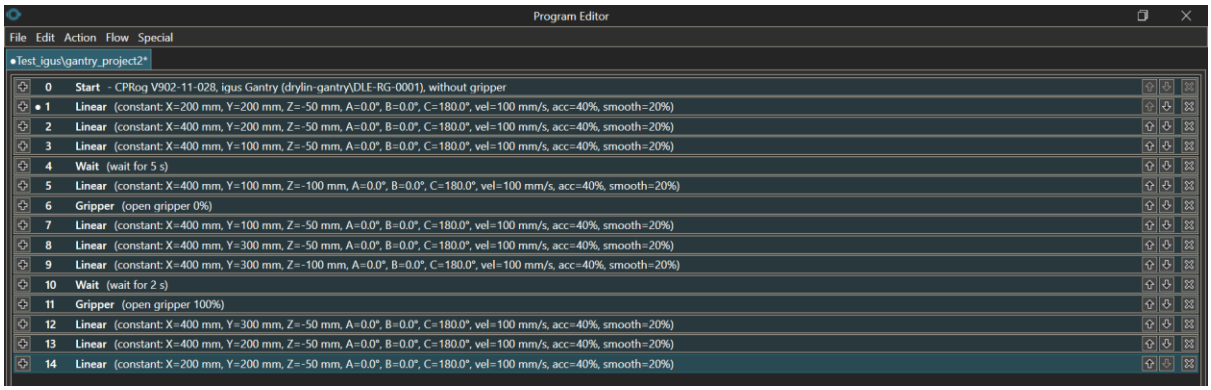
Now, tap on the action, again select Linear motion and edit the X coordinate to 200mm.

This makes the gantry robot move BACK in x direction from 400mm to 200mm

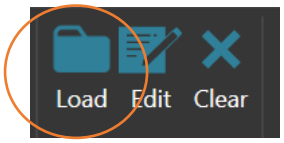




PROGRAM FOR COMPLETE PATHWAY OF GANTRY



Now save the .xml code file and load the file and run the program and gantry tracks its path using code.



PATHWAY OF GANTRY

https://drive.google.com/file/d/1MfcJsx3U3C2HrvxFrdgmQ-PzWvbQkaJP/view?usp=sharing

Conclusion:

- A complete analytical solution to the forward kinematics of the gantry Robot is derived in this project. Kinematic analysis of the 3DOF GANTRY robot was simulated using Robo Analyzer software.
- At the end of this project, we came to a clear understanding of Robo Analyzer Software.
- By using Robo Analyzer we obtained forward kinematics of the end effector of the robot for the given joint parameters.
- By using graphs in Robo analyzer software we analyzed the change in joint offset with respect to time and the change in joint velocities with respect to time.

The forward kinematic analysis of the gantry robot is investigated. The Gantry model is prepared and solved for positioning of the end-effectors using a Robo analyzer based on the D.H Method and now it is become easy to study the kinematic of the gantry robot and changing the parameters to see the behavior of the system for getting the best design and configuration of the robot. Furthermore, programming for the IGUS Gantry robot is done using C-prog which simplifies Gantry Robot Kinematic Analysis.

----------xxxxxxxxx----------