Introduction to Robotics
ME 5711
Fall 2018
Tuesday and Thursday 4:00-5:15 PM Rm 9-241

COURSE OUTLINE:

A system engineering approach to robotics, science and technology. To familiarize students with the mechanical design and control of the manipulators (Robots). Topics including kinematics, trajectory planning, control, programming of manipulators and simulation, along with introduction to artificial intelligence and computer vision. The students are required to design and do theoretical analysis of an innovative project in robotics, along with some hands-on-experience with the Universal (UR-3).

INSTRUCTOR:

Dr. Behnam Bahr, Professor
Mechanical Engineering Department
California State Polytechnic University, Pomona
Office 9-127A
Phone (909) 869-2440
e-mail: bbahr@cpp.edu

PREREQIUSITE

Graduate Student in Engineering, knowledge of linear –Algebra, and programming languages such as Matlab. And control system.

OFFICE HOURS:

3:00pm- 4:00pm Th,, 5:00pm- 6:00pm, Th, and by appointment

REQUIRED TEXT:

John J. Craig, "Introduction to Robotics Mechanics and Control," Addison-Wesley, 4th Edition.

COURSE GRADING: The grade is based on the curve system.

Homework	15
Special Assignment using UR-3 Robot off line and in a lab	15
Midterm Exam	25
One Comprehensive Exam	15
Final Project Report	30
Total	100

* The Project can be experimental or theoretical, and should be a publishable quality work. Students are encourage to publish their work in the conference proceedings. Therefore, you need to start your project as soon as possible.

HOMEWORK:

Are assigned from lecture and reading assignments and are collected on a weekly basis. Assigned Problems and computer programs are also part of the homework.

EXAMS:

The exams will be comprehensive, open book, notes and handouts

Final Project:

The final project report is accepted electronically.

Introduction to Robotics
ME 5711
Fall 2018
Tuesday and Thursday 4:00-5:15 PM Rm 9-241

Tentative will be revised as soon as a room is assigned for experiments: **topics include some of the following and more as the course progresses.**

The students are expected to read the handout and watch the website for the use of the UR-3 robot before coming to class. The first time we demo the robot, but the goal is for you to learn the laboratories before coming to class and conduct experiments.

DATE	TENTATIVE TOPIC(S)	READING
8/23/18	Introduction to Robotics	Syllabus
8/28/18	Introduction to Robotics and Group Assignment	Ch 1
8/30/18	Object manipulation through space	Ch 2
9/4/18	Object manipulation through space	Ch 2
9/6/18	Joint geometry and notations	Ch 2
9/11/18	Joint geometry and notations	Ch 2
9/13/18	Kinematics & Inverse kinematics eqs. and their solutions	Ch 2
9/18/18	Kinematics & Inverse kinematics eqs. and their solutions	Ch 3
9/20/18	Kinematics & Inverse kinematics eqs. and their solutions	Ch 3
9/25/18	Kinematics & Inverse kinematics eqs. and their solutions	Ch 3
9/27/18	Kinematics & Inverse kinematics eqs. and their solutions	Ch 3
10/4/18	UR3 – Experiment 1 & 2	Demo by Dr. Bahr
10/9/18	UR3 – Experiment 1 & 2	Website and handouts
10/11/18	UR3 – Experiment 3	Website and handouts
10/11/18	UR3 – Experiment 3	Website and handouts
10/16/18	UR3 – Experiment 4	Website and handouts
10/18/18	UR3 – Experiment 4	Website and handouts
10/23/18	EXAM I	
10/25/18	UR3 – Experiment 5	Website and handouts
10/30/18	UR3 – Experiment 5	Website and handouts

Introduction to Robotics
ME 5711
Fall 2018
Tuesday and Thursday 4:00-5:15 PM Rm 9-241

11/1/18	Trajectory Generation	Ch 4
11/6/18	Trajectory Generation	Ch 4
11/8/18	Cartesian and joint Interpolation	Ch 5
11/13/18	Force Analysis	Ch 7
11/20/18	Force Analysis	Ch 7
11/27/18	Force Analysis	Ch 7
11/29/18	Computer Vision	Handout
12/4/18	Computer Vision	Handout
12/6/18	Student Project	
12/11/18	Student project	
12/7/17	**Final Project Report Due**	
	Final Exam	

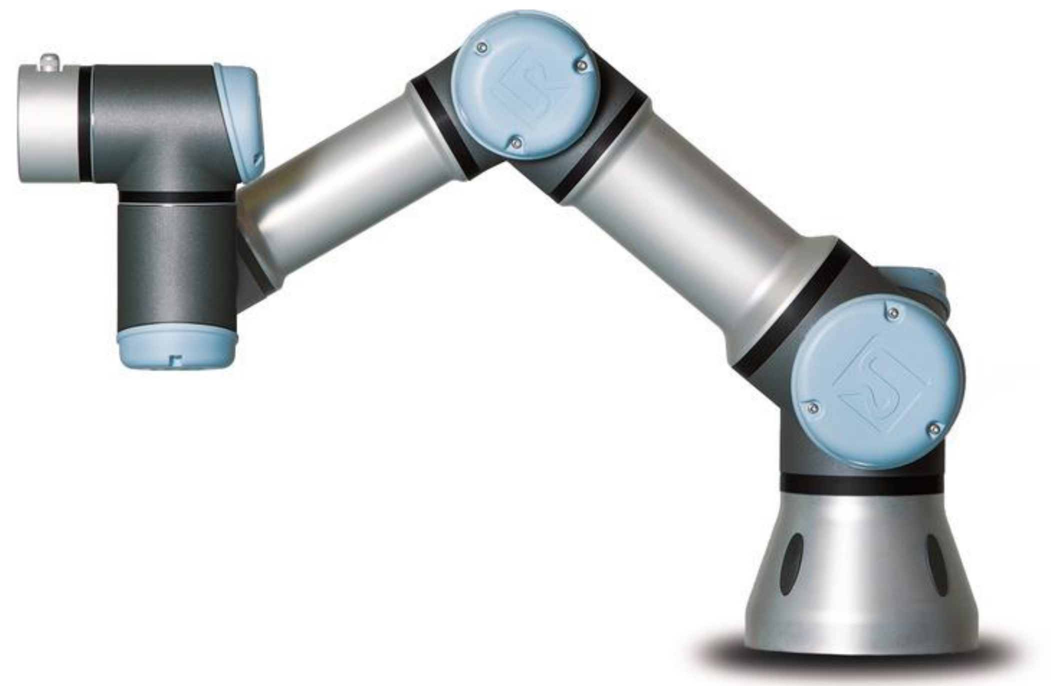
SAFETY:

Adhere to safe practices while doing any experiment. Always wear safety glasses
When using shop equipment. Never work on live electronic circuitry, and always work in a group.

Operation:

No late homework can be accepted for grading.
No make-up Exam will be given without prior consent of the instruct

Lab 1: Introduction to UR3 Robotic Arm



California State Polytechnic University, Pomona

Mechanical Engineering Department

2018

Table of Content

Objective:

Establish an understanding of the UR3 robotic arm features, capability and terminology. Learn the basic functionality of the arm through creating and running a process program.

Equipment:

- 6 DOF robotic arm
- Control box
- Teach pendant

Theory:

The UR3 is a 6 degree of freedom robotic arm with +/- 360 degrees of rotation on all joints (other than wrist 3, which has infinite rotation). The maximum payload of the arm is 3kg. The arm is capable of detecting collisions which will engage a protective stop.

The teach pendant is the human to machine interface on which the robot can be programmed and controlled. PolyScope is the software used by the teach pendant to control the robot. Once a program is opened, the left pane of the display shows the program tree, which lists sequence of commands that will be executed.

The control box houses the processor and power hardware for the robot. Peripherals such as sensors or tooling can interface with the Input/Output channels within the box. Additionally, ports for USB and ethernet are available from the bottom of the control box.

For every program, the controller needs information on the tool being used: The center of gravity, tool center point, and tool mass. The TCP of the tool is where the tool will interact with its environment (eg: contact points of a gripper, end of a nozzle, ect.). The relative distance from the TCP to the tool mounting flange needs to be entered into the program.

The robot has three types of motion between waypoints: MoveL, MoveJ, and MoveP

> <u>MoveL:</u> Linear motion between waypoints. Use when the path of the TCP is important, or in confined space to avoid collision.
>
> <u>MoveJ:</u> Joint move between waypoints. This is a nonlinear path which provides the fastest motion type. Use when path of TCP is not important, or in free space.
>
> <u>MoveP:</u> Process move between waypoints. This moves the TCP linearly between waypoints while maintaining the TCP at a constant velocity. Use in applications such as welding or gluing. This move rounds each vertex with a blend radius, which can be adjusted in the program. This motion type also includes

CircleMove, which creates a three point arc. The waypoints to be defined are the via point, which is any point on the arc, and the arc endpoint.

The work envelope is the range that the robotic arm can perform in. Each degree of freedom contributes to the total envelope. While moving within this envelope, there are trajectories in which the commanded acceleration of a joint will approach infinity. These are known as singularities, and typically occur when arm joints align with each other or when approaching the extremes of the work envelope. Since the physical robot cannot match these commands, the arm will typically oscillate excessively as it approaches the singularity, then the emergency stop will halt the program.

Though the robot can detect physical collisions when they occur, it cannot preemptively determine that it will collide with the outside world, or itself. Care should be taken that the trajectories avoid these situations.

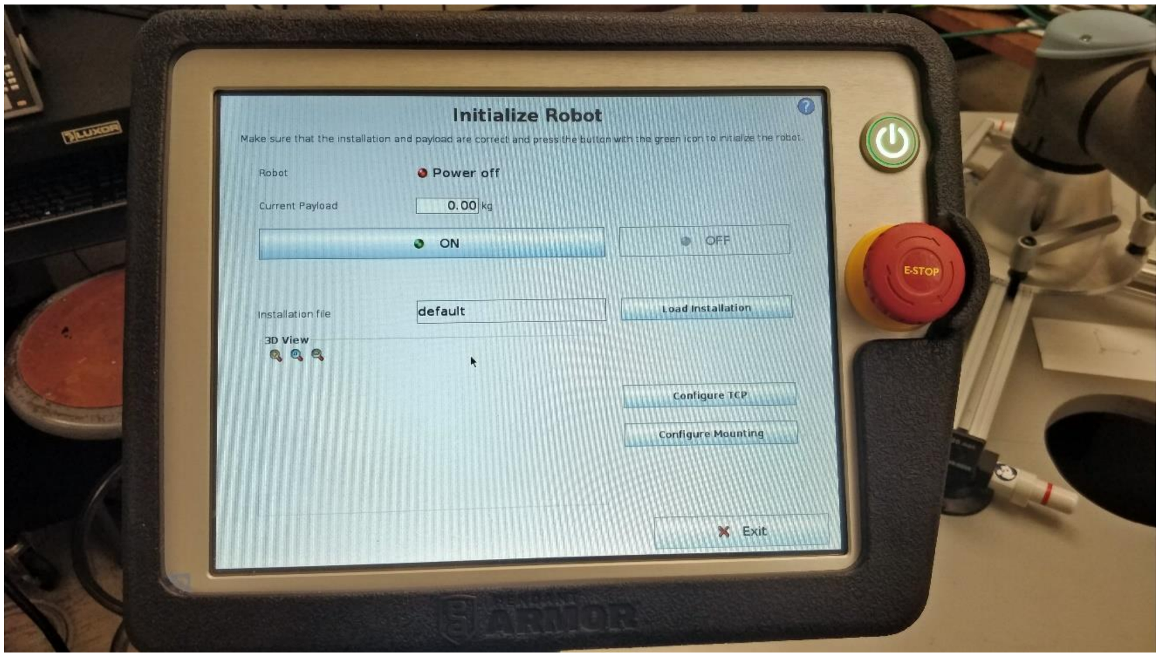
Results:

1) Start up the UR3
2) Configure the TCP
3) Create programs for the Pattern or Bearing Module Processes
4) Save the program internally or externally to a USB drive

Procedure:

Startup:

- Click the power button on the teach pendant and allow the robot to power up (~1 min.)
- Ensure the E-stop button is disengaged (rotate clockwise until in up position)
- Click “Go to initialization screen” then click the “ON” button to enable power to the joints



Initialize Robot

Make sure that the installation and payload are correct and press the button with the green icon to initialize the robot.

Robot: Power off

Current Payload: 0.00 kg

Installation file: default

Figure 1: Initialization Screen

- Click “START” to release the mechanical brakes, then click “OK”

Tool Setup:

- From the “Program Robot” page, click the “Installation” tab at the top left of the screen

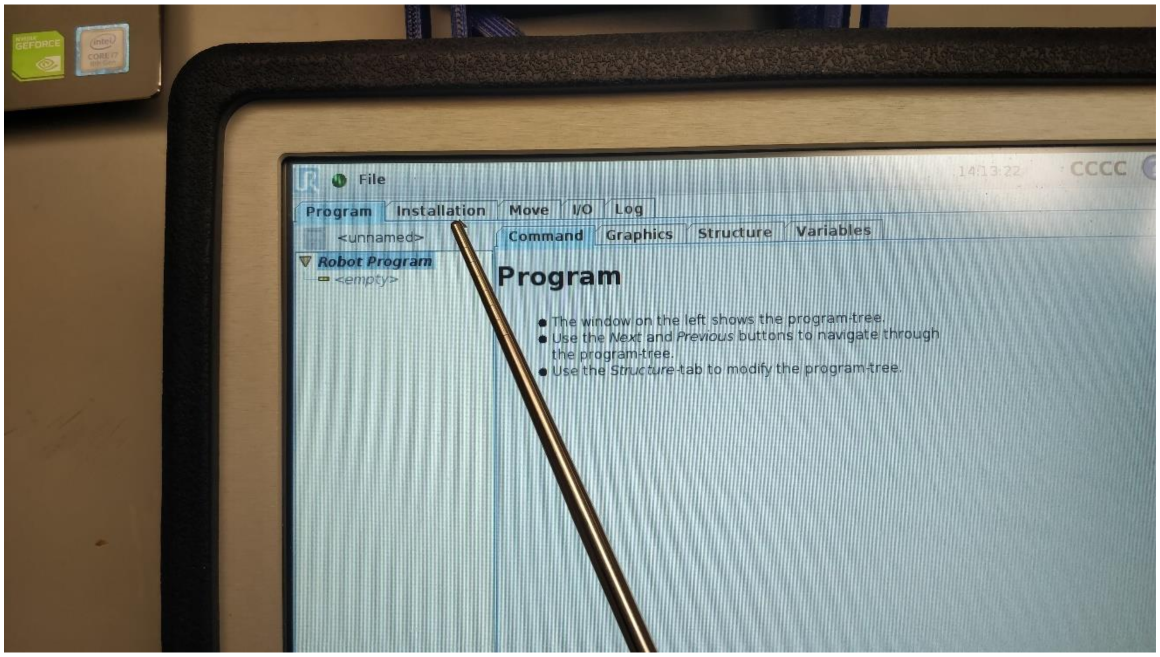
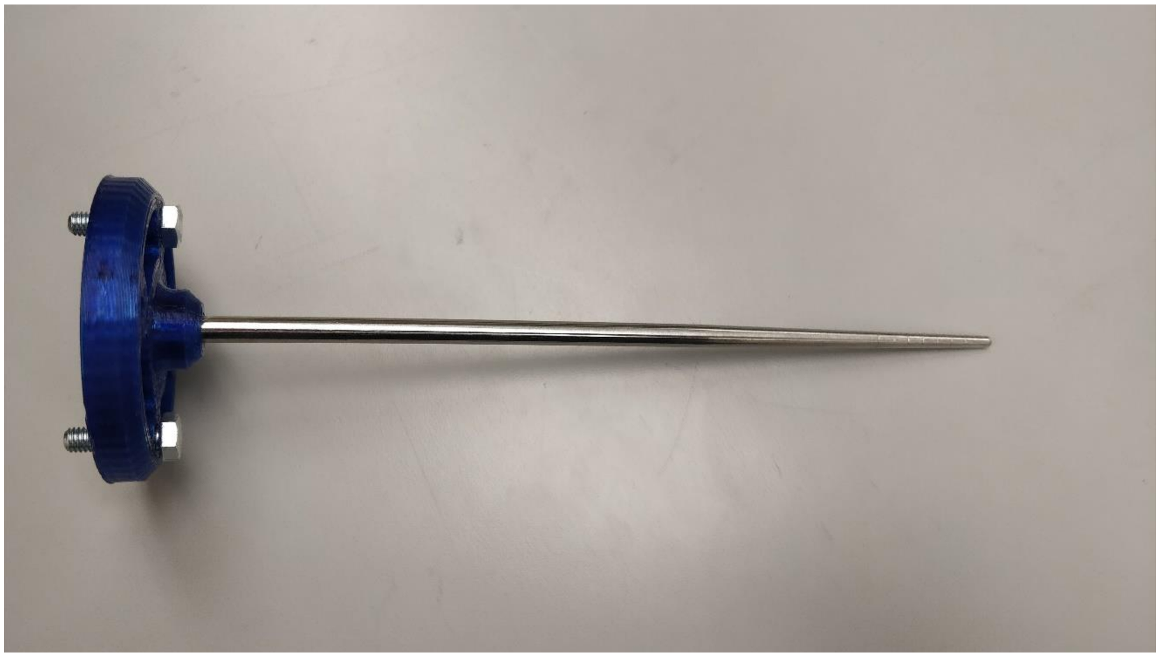

File
Program
Installation
Move
I/O
Log
<unnamed>
Command
Graphics
Structure
Variables
Robot Program
<empty>
Program
The window on the left shows the program-tree.
Use the Next and Previous buttons to navigate through the program-tree.
Use the Structure-tab to modify the program-tree.

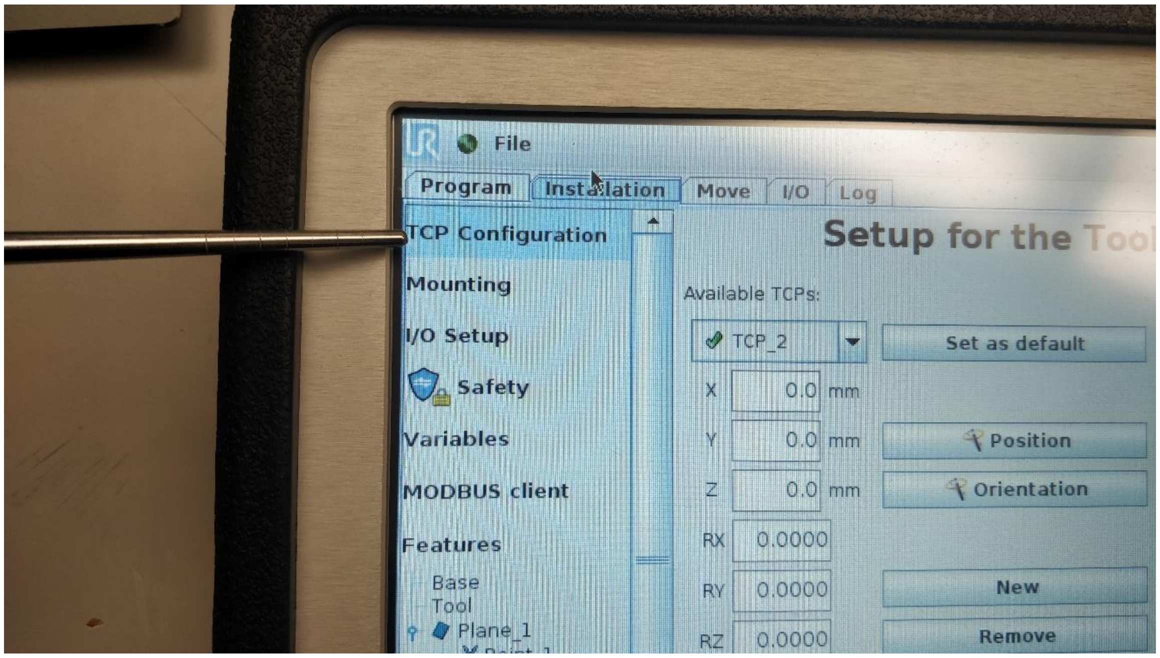
Figure 2: Program Robot Screen

- Select “TCP Configuration” and enter the distances from the mounting flange to the TCP in the X,Y, and Z fields. If the tool is rotated relative to the tool flange, input the rotational orientation into the RX, RY, and RZ fields.



- *Figure 3: Metal Rod Attachment*

- For the metal rod attachment, X = 0mm, Y = 0mm, Z = 200mm



File

Program | Installation | Move | I/O | Log

TCP Configuration
Mounting
I/O Setup
Safety
Variables
MODBUS client
Features
Base
Tool
Plane_1

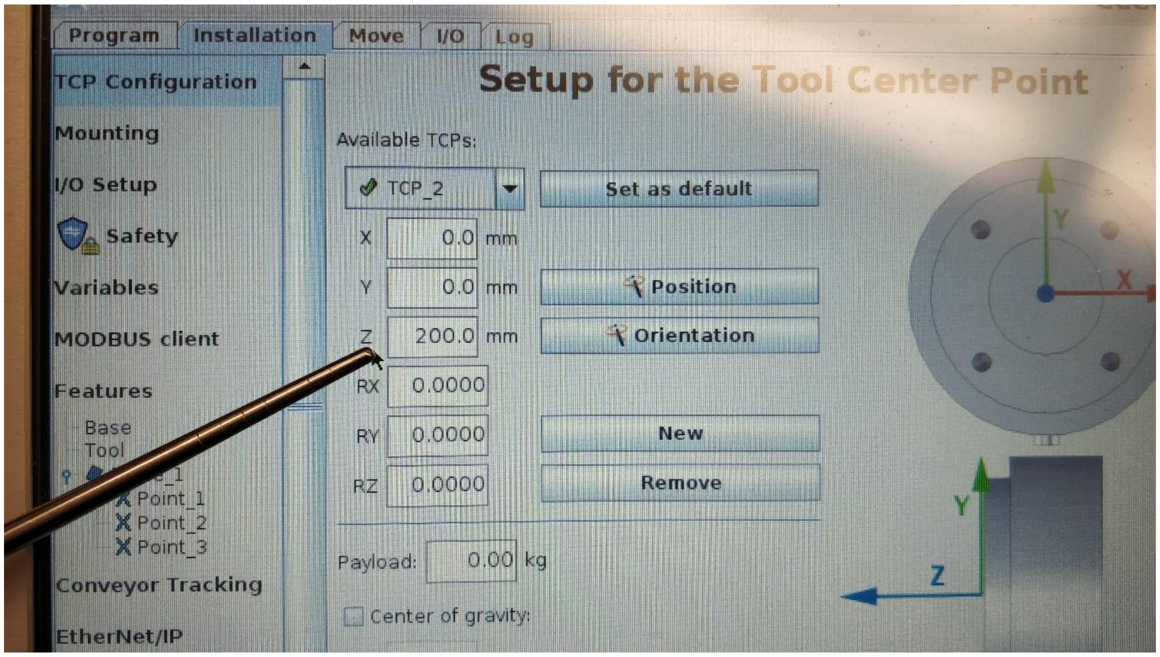
Setup for the Tool

Available TCPs:

TCP_2 | Set as default

X 0.0 mm
Y 0.0 mm | Position
Z 0.0 mm | Orientation
RX 0.0000
RY 0.0000 | New
RZ 0.0000 | Remove

Figure 4: TCP Configuration



Program | Installation | Move | I/O | Log

TCP Configuration
Mounting
I/O Setup
Safety
Variables
MODBUS client
Features
Base
Tool
Point_1
Point_2
Point_3
Conveyor Tracking
EtherNet/IP

Setup for the Tool Center Point

Available TCPs:

TCP_2 | Set as default

X 0.0 mm
Y 0.0 mm | Position
Z 200.0 mm | Orientation
RX 0.0000
RY 0.0000 | New
RZ 0.0000 | Remove

Payload: 0.00 kg

Center of gravity:

Figure 5: TCP Coordinate Inputs

- Input the tool weight in the "Payload" field
  - For the metal rod attachment, Payload = 0.05kg

- If needed, input the distance from the tool flange to the center of gravity in the CX, CY, and CZ fields
  - Not necessary for using the metal rod attachment
- Click the "Set as default" to save the TCP configuration

Programming:

- From the main page of the teach pendant, click “Program Robot” then select “Empty Program”

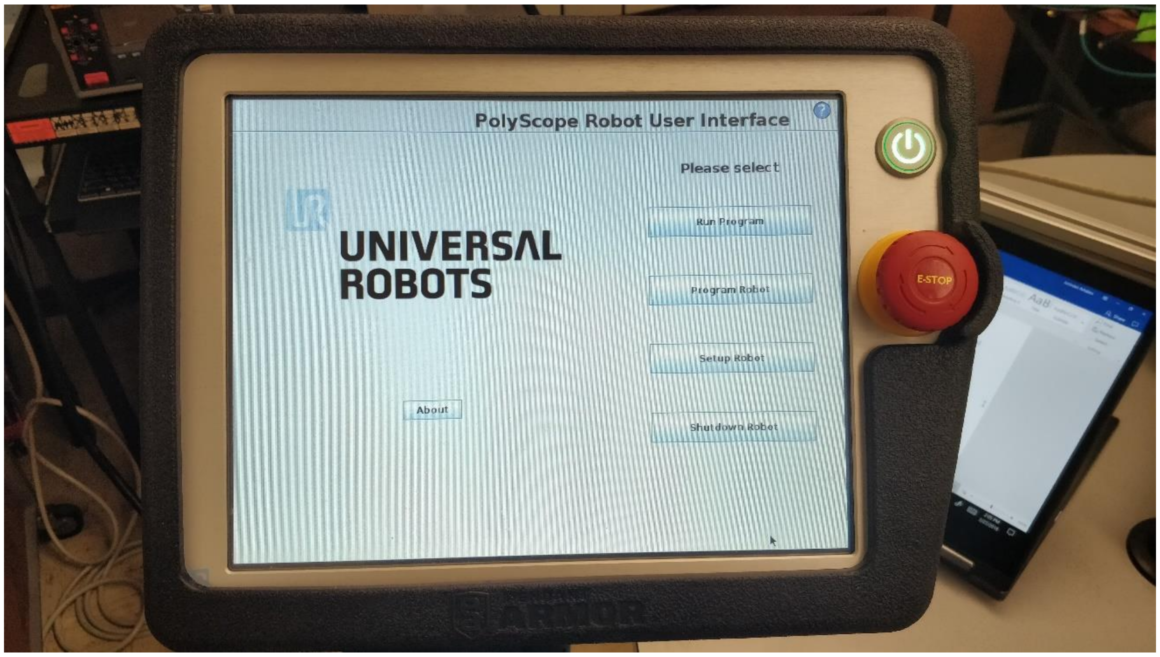


Figure 6: Main Page

- *If an End Effector is attached to the robot, Configure the TCP using the **Tool Setup** Instructions above
- On the top of the Program pane, select the “Structure” tab

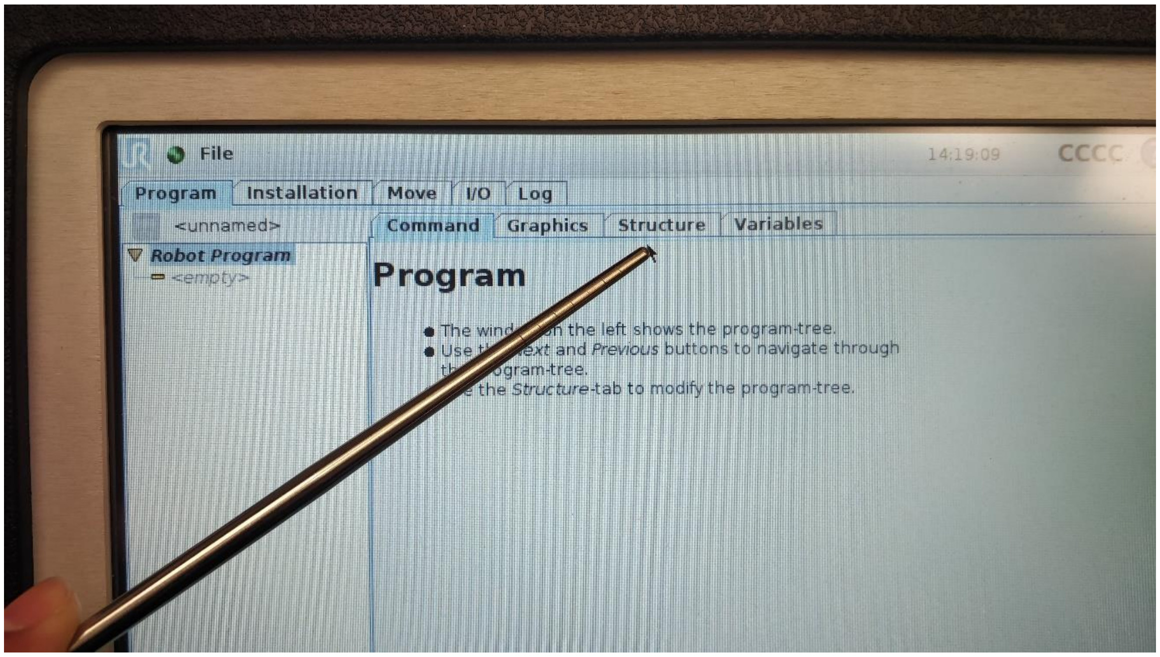


Figure 7: Program Robot Screen

- On the Program Structure Editor page, select “Move”. A motion *MoveJ* and waypoint *Waypoint_1* are added to the program tree

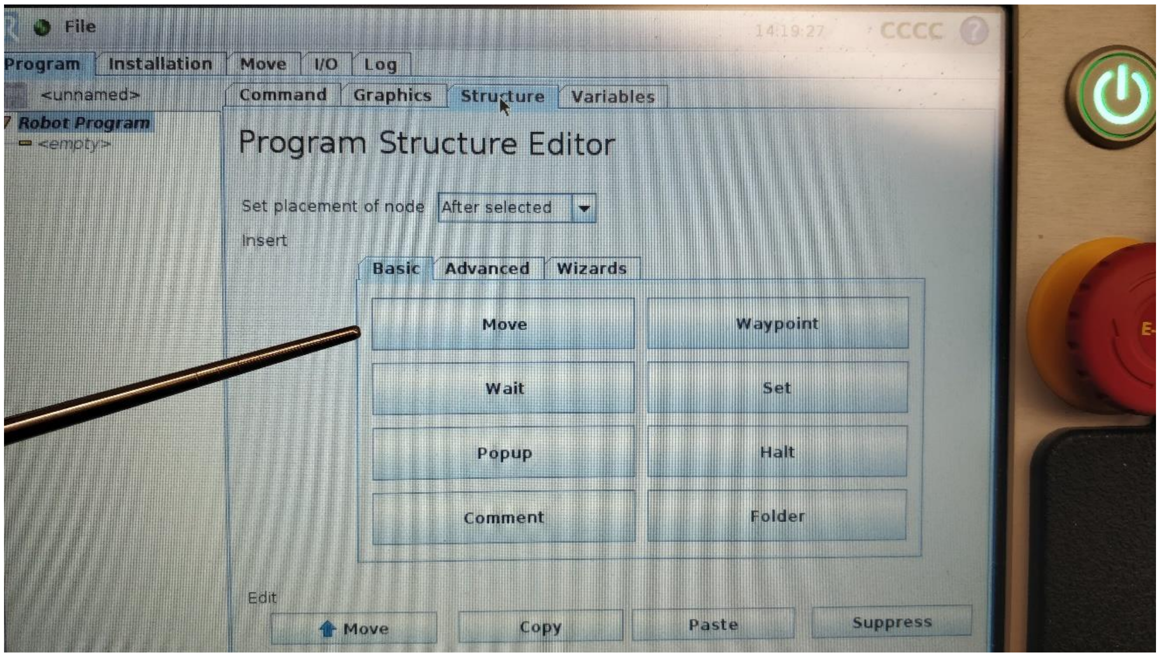


Figure 8: Program Structure Editor

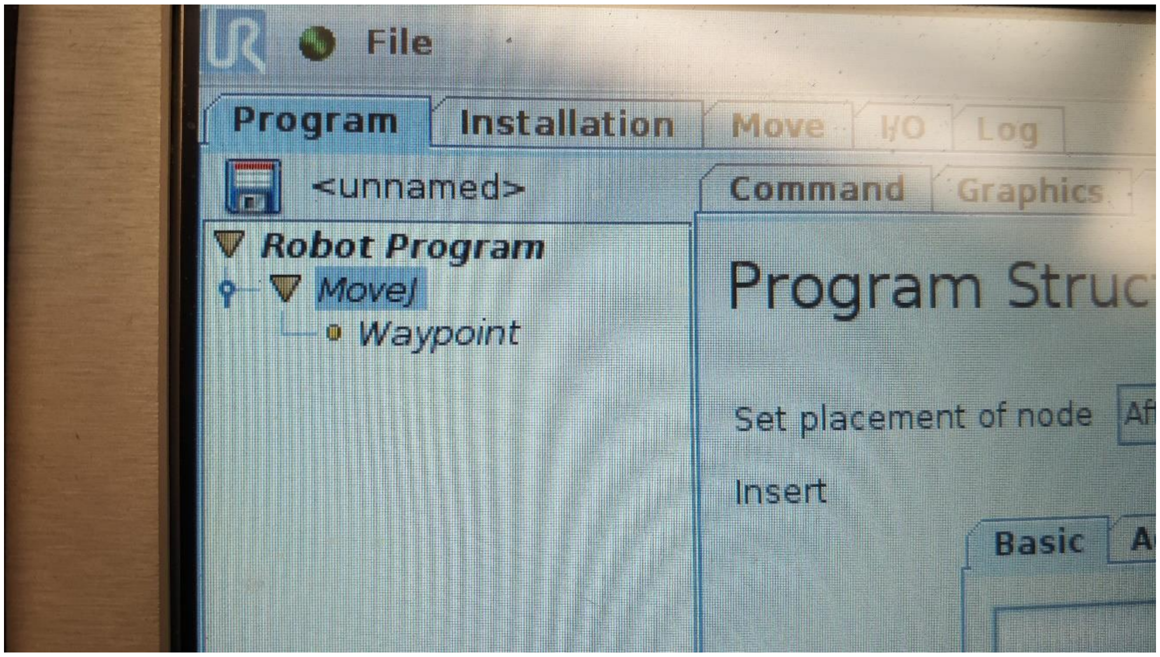
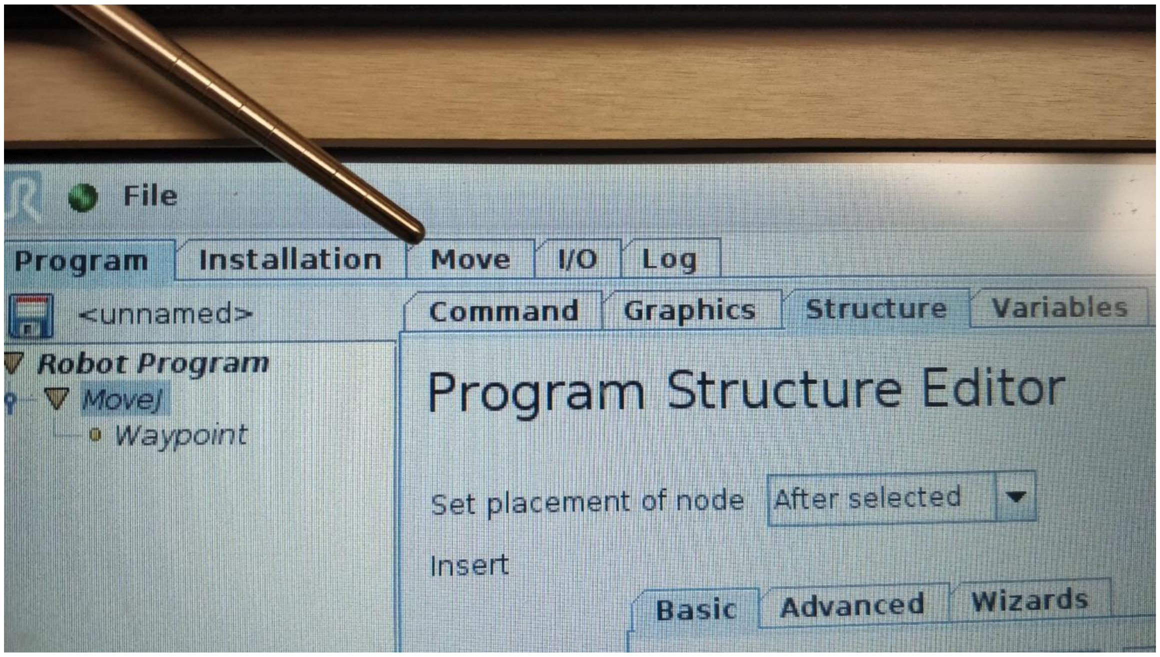


Figure 9: Program Tree

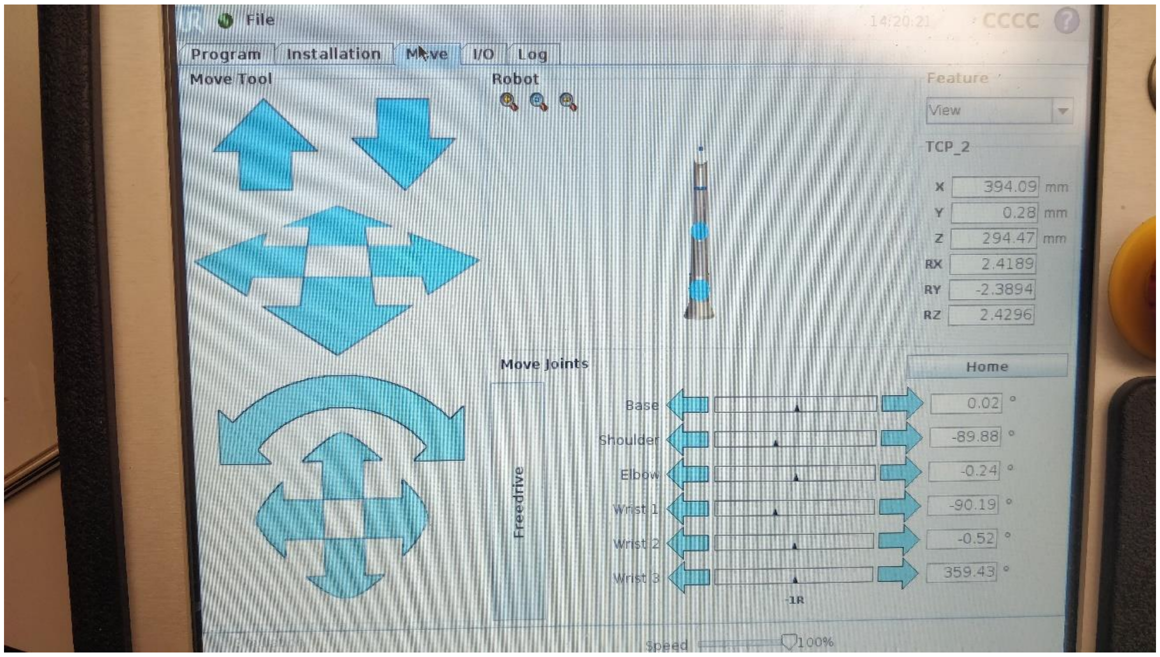
- Select the “Command” tab, then “Set this waypoint” and adjust the position of the arm using one of the methods:
  - The Move Tool arrows move the TCP in 3D Space
  - The Move Joints arrows increment the angle of the selected joints
  - The Joint angles can be edited to move to specific joint angels

- The TCP position can input directly and the arm will move the TCP to the new location
- The Freedrive button (one on the screen, one on the back of the teach pendant can be held down and the arm can be physically moved to a new location



Set placement of node: After selected

Figure 10: Move Tab Location



TCP_2

X	394.09	mm
Y	0.28	mm
Z	294.47	mm
RX	2.4189	
RY	-2.3894	
RZ	2.4296	

Base	0.02	°
Shoulder	-89.88	°
Elbow	-0.24	°
Wrist 1	-90.19	°
Wrist 2	-0.52	°
Wrist 3	359.43	°

Figure 11: Move Tool Control Screen

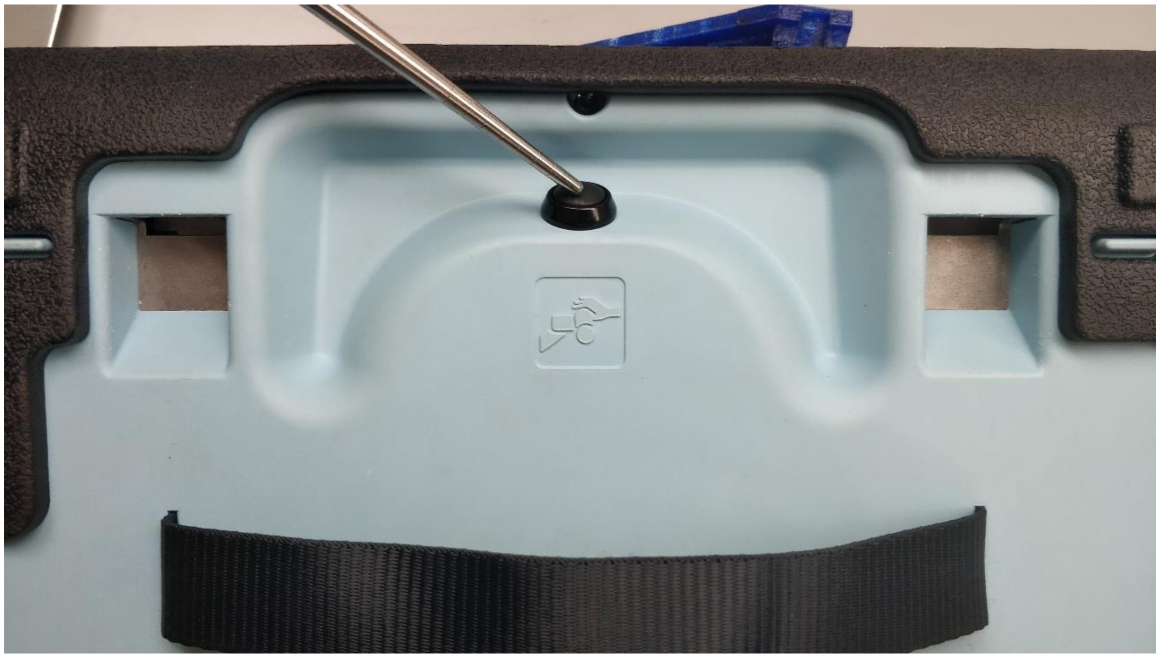


Figure 12: FreeDrive Button

- Once the arm is in the desired location and orientation, select "OK" at the bottom right hand corner of the screen to save the waypoint

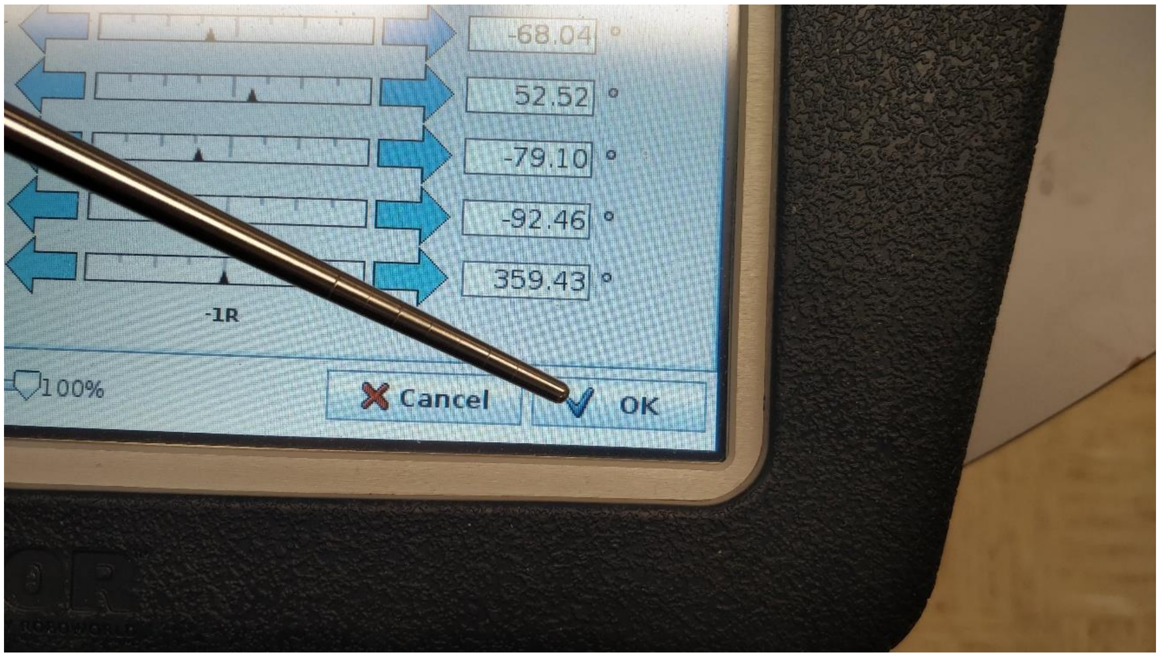


Figure 13: Confirm Waypoint

- Click "Add waypoint after" to add another waypoint to the program tree. Use the instructions above to set the next waypoint

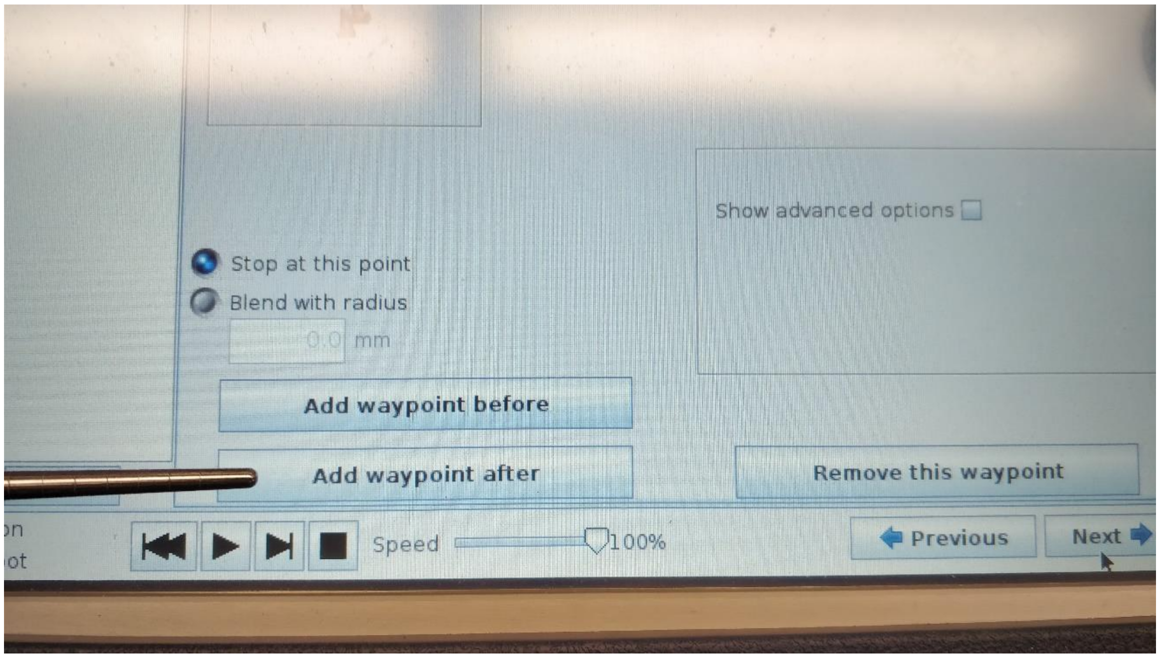


Figure 14: Add Waypoint After Location

- Repeat these instructions until all the needed waypoints are set.

- When the program is played, the arm will move through the waypoint in order from top to bottom of the Robot Program tree

    - By default, the robot will use MoveJ to move efficiently between waypoints. To change the move type being used, reference the **Edit Motion Type** instructions below

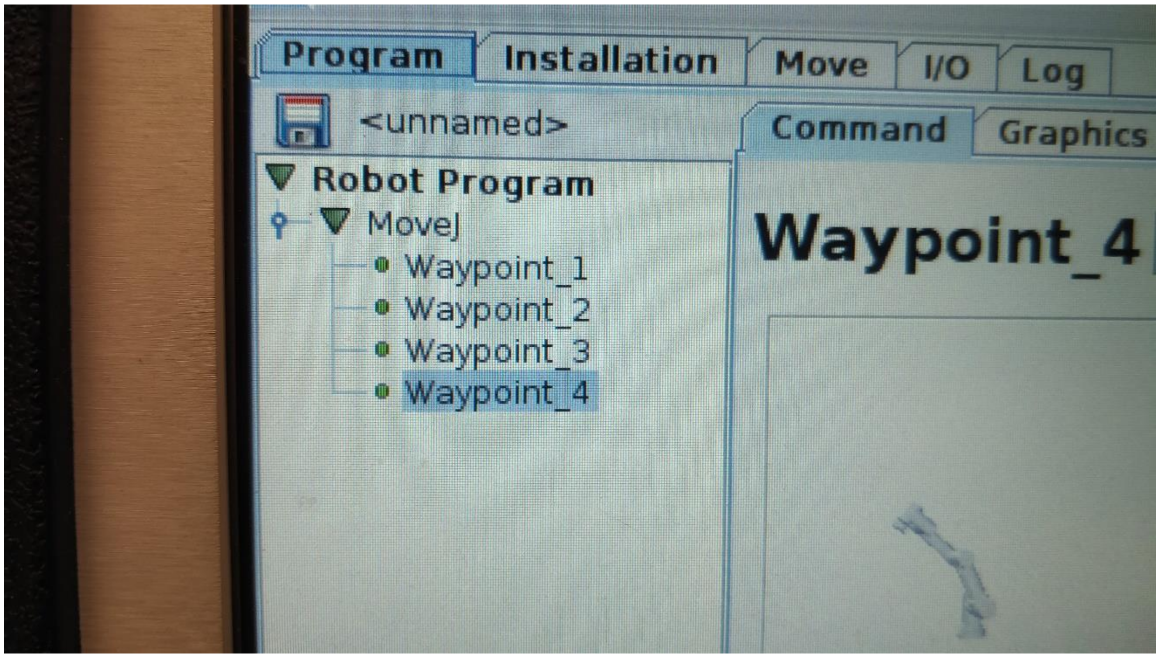


Figure 15: Waypoints in Program Tree

Edit Motion Type:

- In the Robot Program Tree, select the motion to be edited.
- Select the “Command” tab, then click the pulldown in the top right corner
- Select the new move type
  - Details on the various move types and on which to select can be found in **Theory**
    - The Robot Program Tree will automatically update the current move type

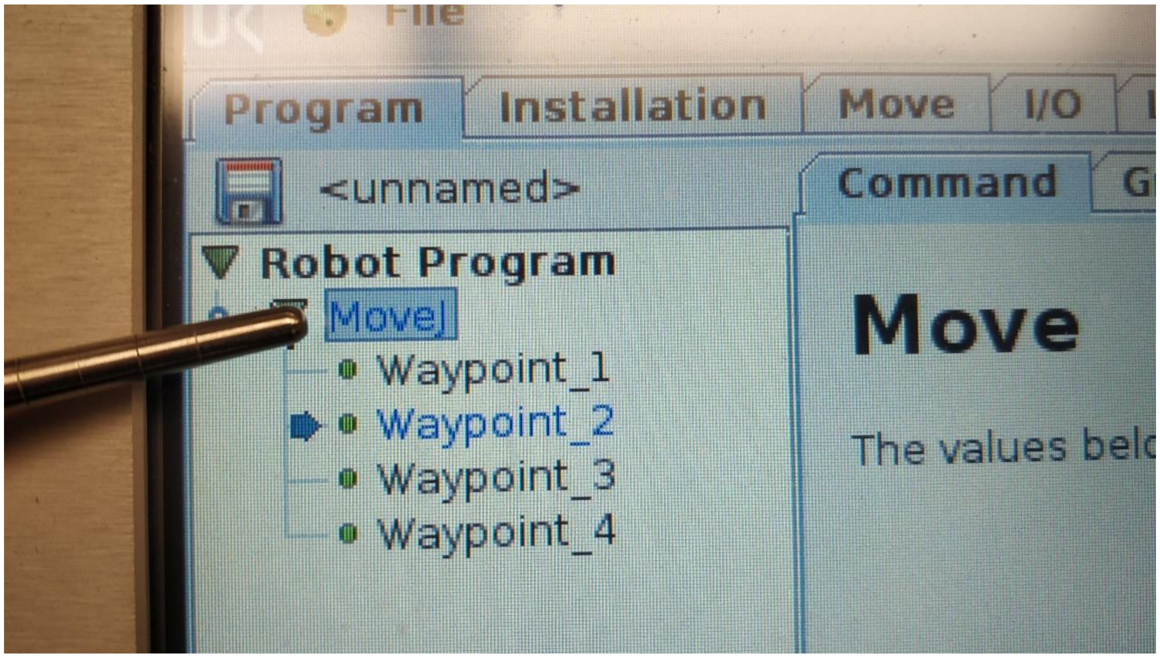


Figure 16: Move Type Indication

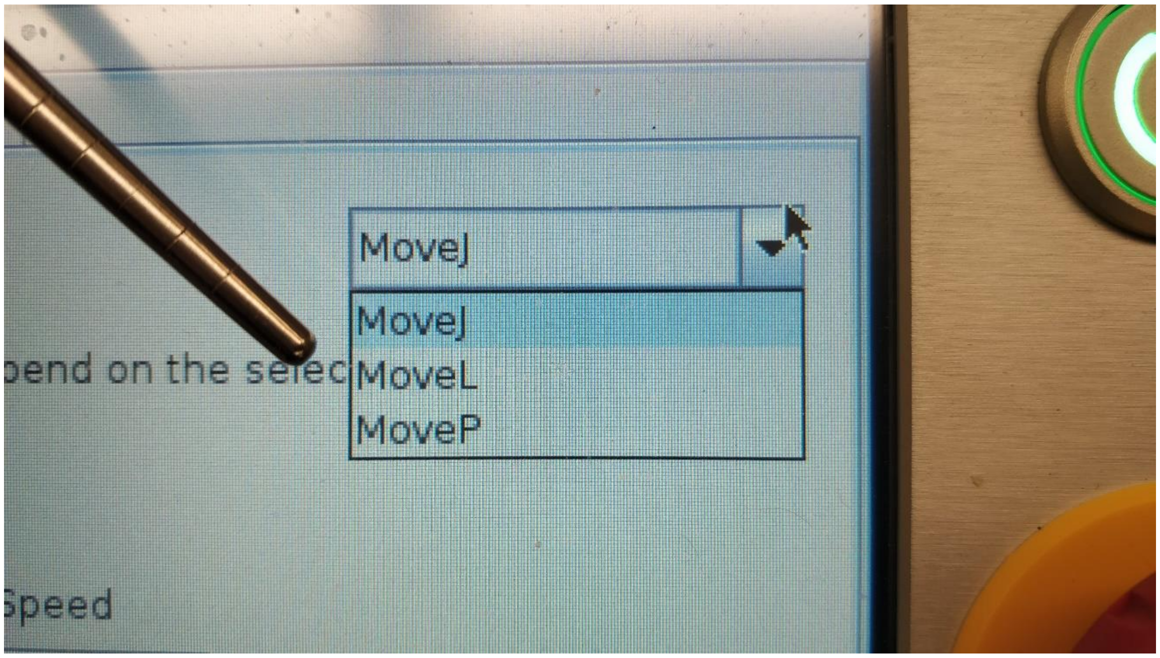


Figure 17: Move Type Selection

Running a Program:

- Navigate to the “Program” tab. The controls for running a program are located at the bottom of the screen.
- Select a speed for the arm to run at. Speeds are chosen as a percentage of fastest TCP velocity (150mm/s by default)
- Select the start icon at the bottom of the screen
- Press and hold “AUTO” until the arm reaches the first waypoint. Press “OK”
- Press the play icon again. The program will begin to run.

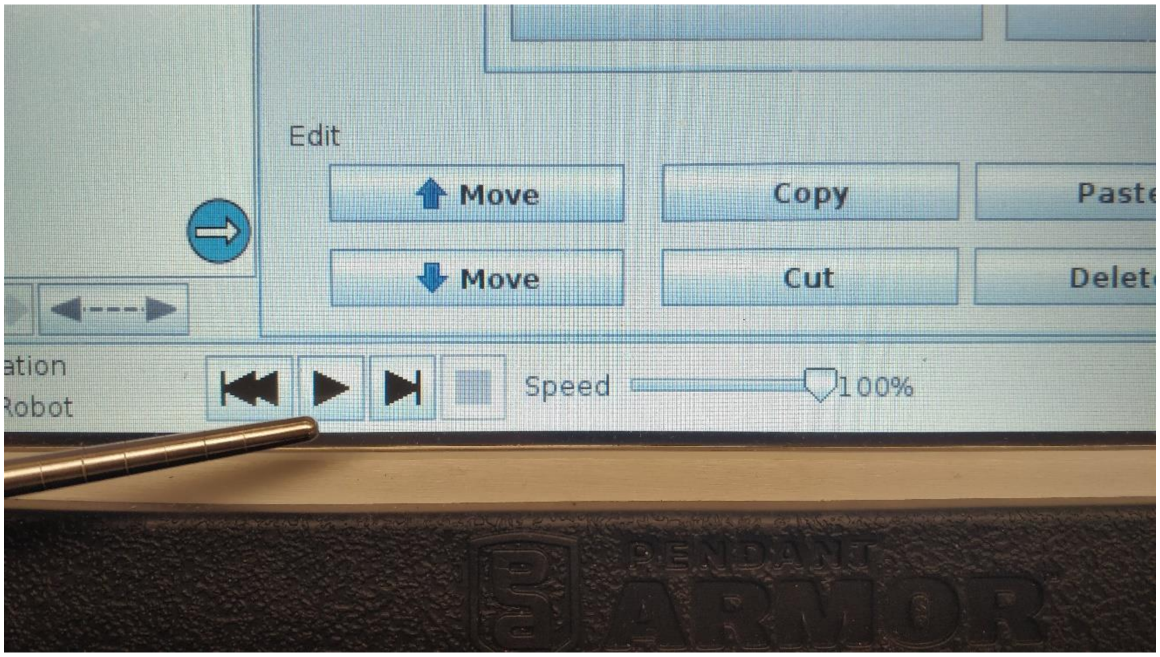


Figure 18: Program Control Interface

- Once the robot has run through all the waypoints on the Robot Program Tree, the arm will MoveJ to the first waypoint and iterate the loop.
- To stop the robot:
  - The pause button will temporarily stop the arm, when play is pressed again, the arm will continue from its location on the Robot Program Tree
  - The stop button will stop the arm, when play is pressed again, the arm will have to be reset to the first waypoint at the top of the Robot Program Tree

Saving a Program:

- Once a robot program has been completed, it can be saved internally or externally.
  - To save internally (on the robot PC):
    - Click “File” then “Save As”

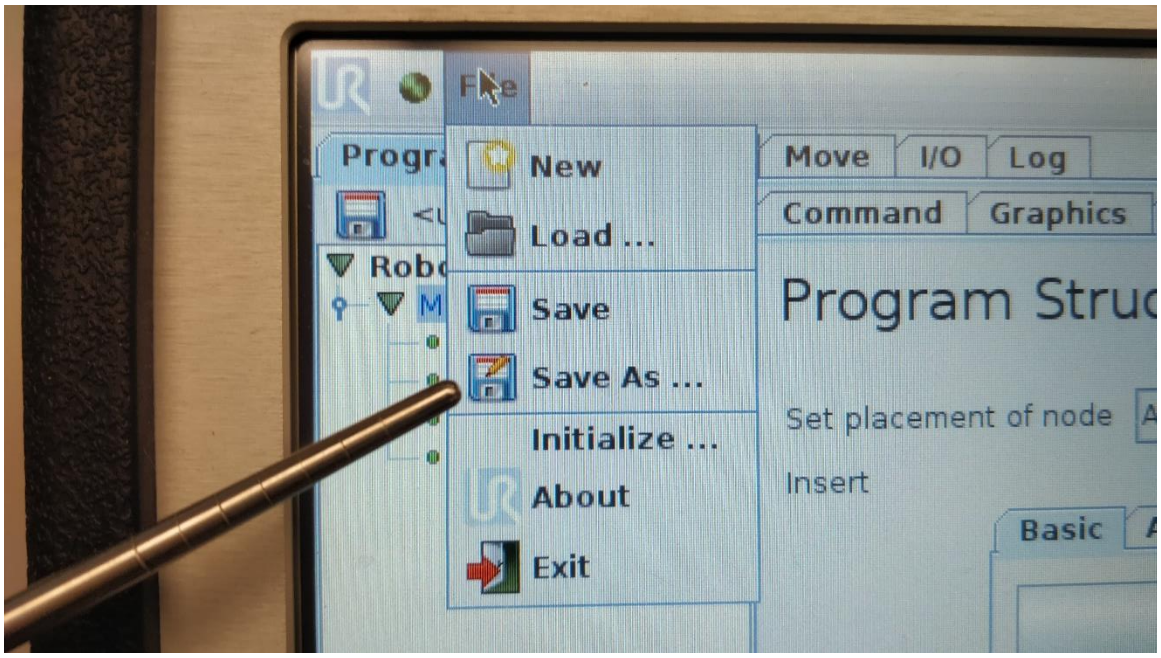


Figure 19: Program “Save As” Location

- Enter the name to save the file as, and “.urp” as the file type, click “Save”

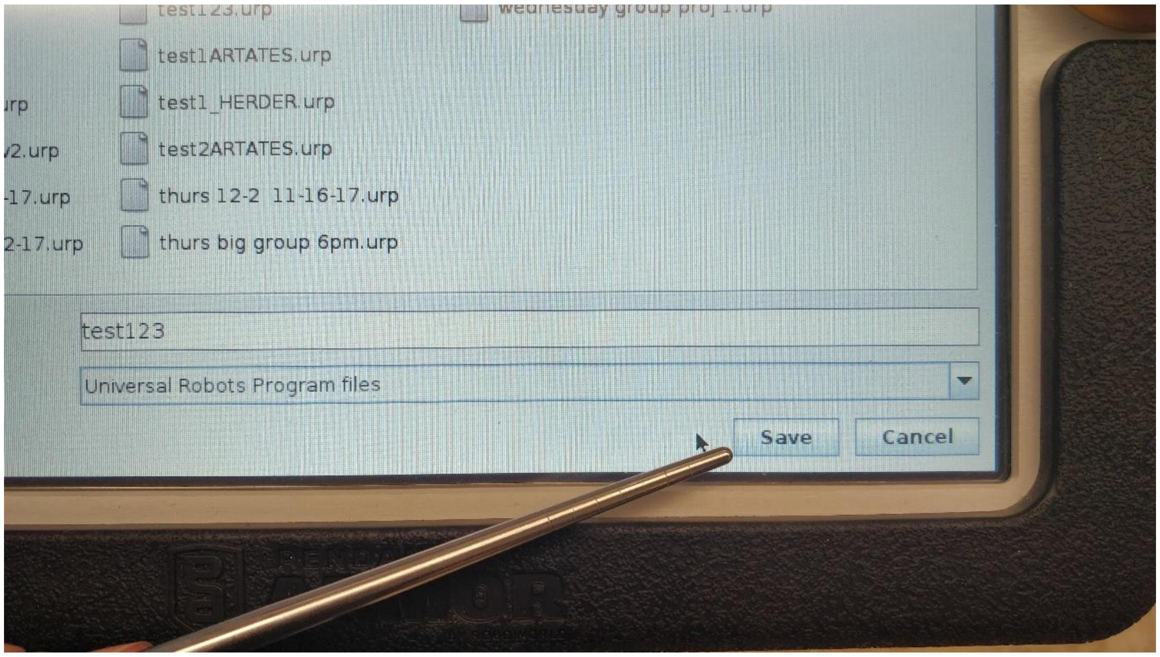


Figure 20: Save Icon

- To save externally (to a USB drive):
    - Insert a USB drive into the port on the right side of the teach pendant

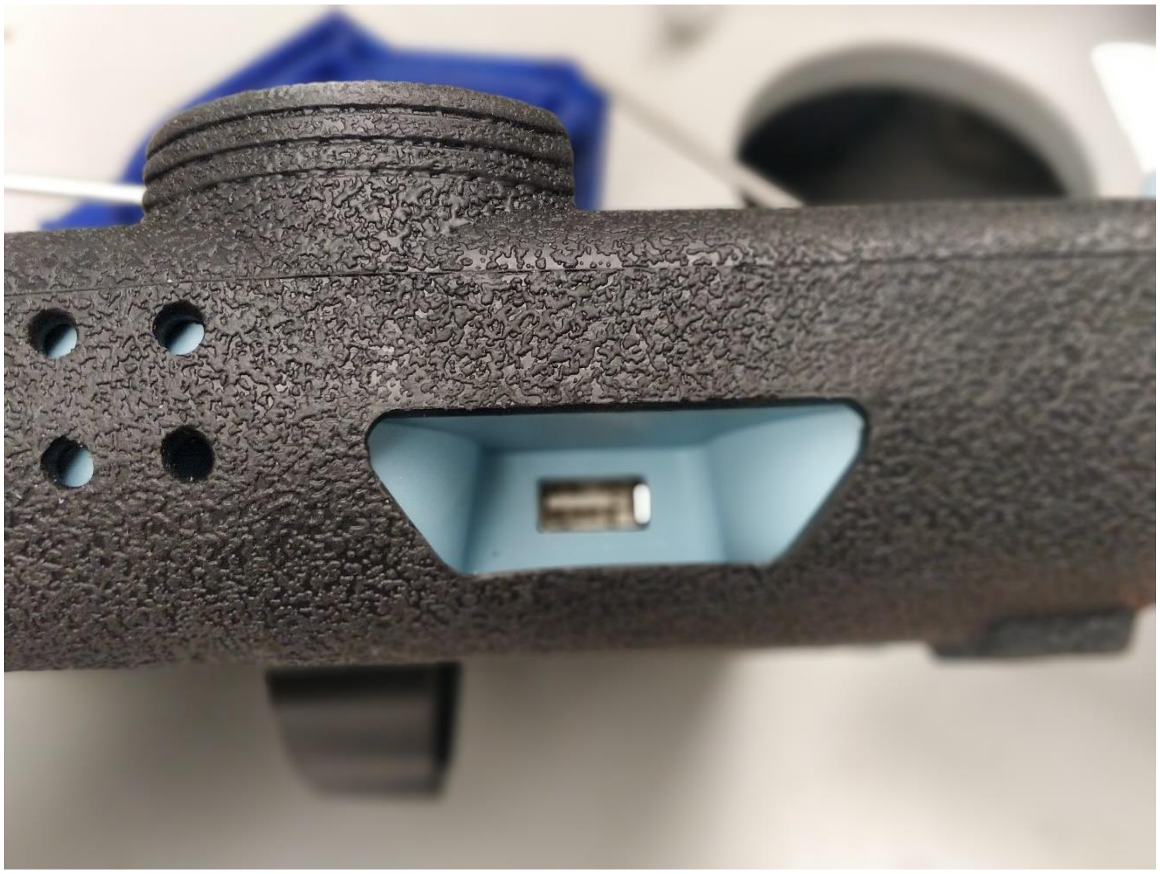


Figure 21: USB Drive Location on Teach Pendant

    - Click “File” then “Save As”

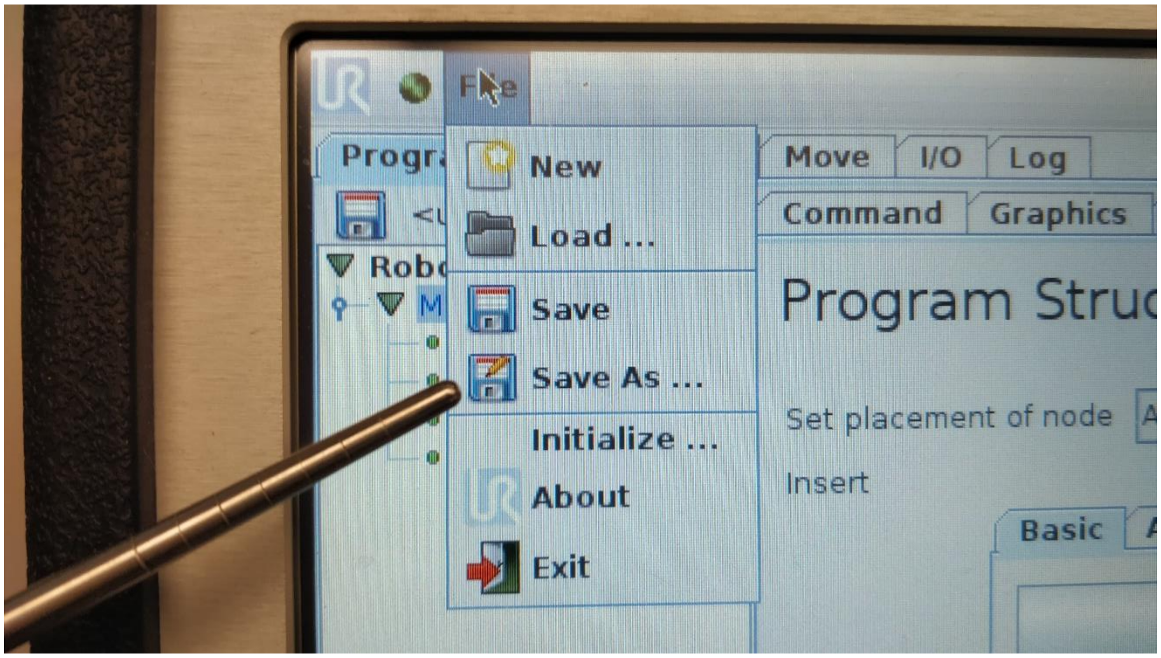


Figure 22: "Save As" Location

- Select the "Current Directory" pull down, and navigate to the USB Drive location
- Enter the name to save the file as, and ".urp" as the file type
- Click "Save"

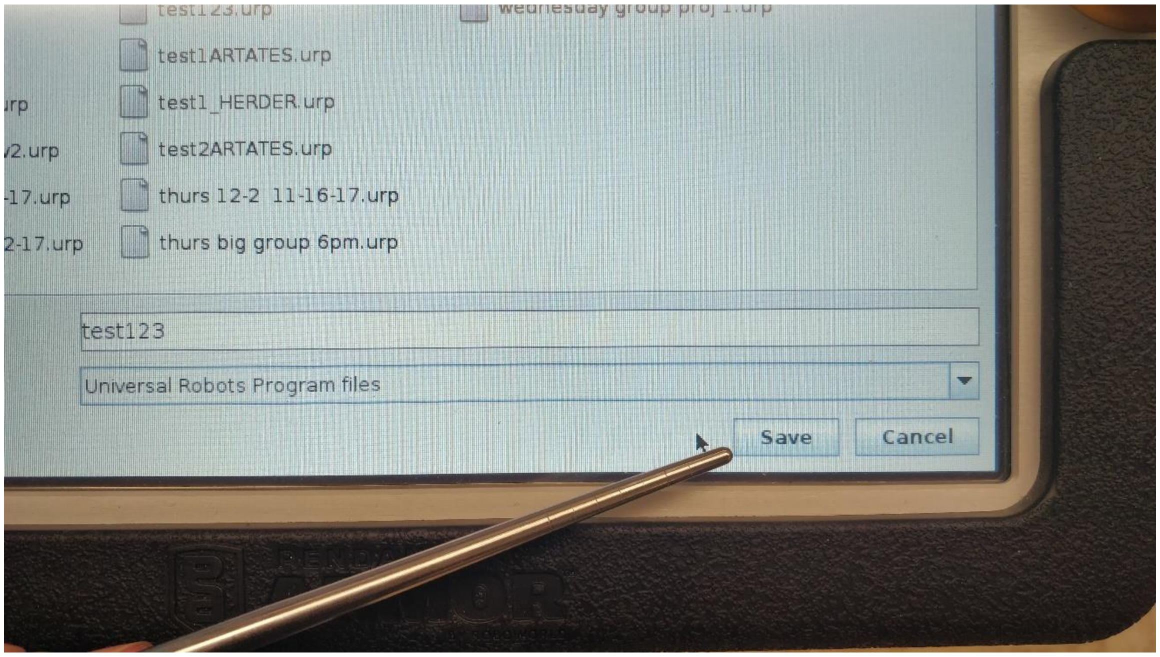
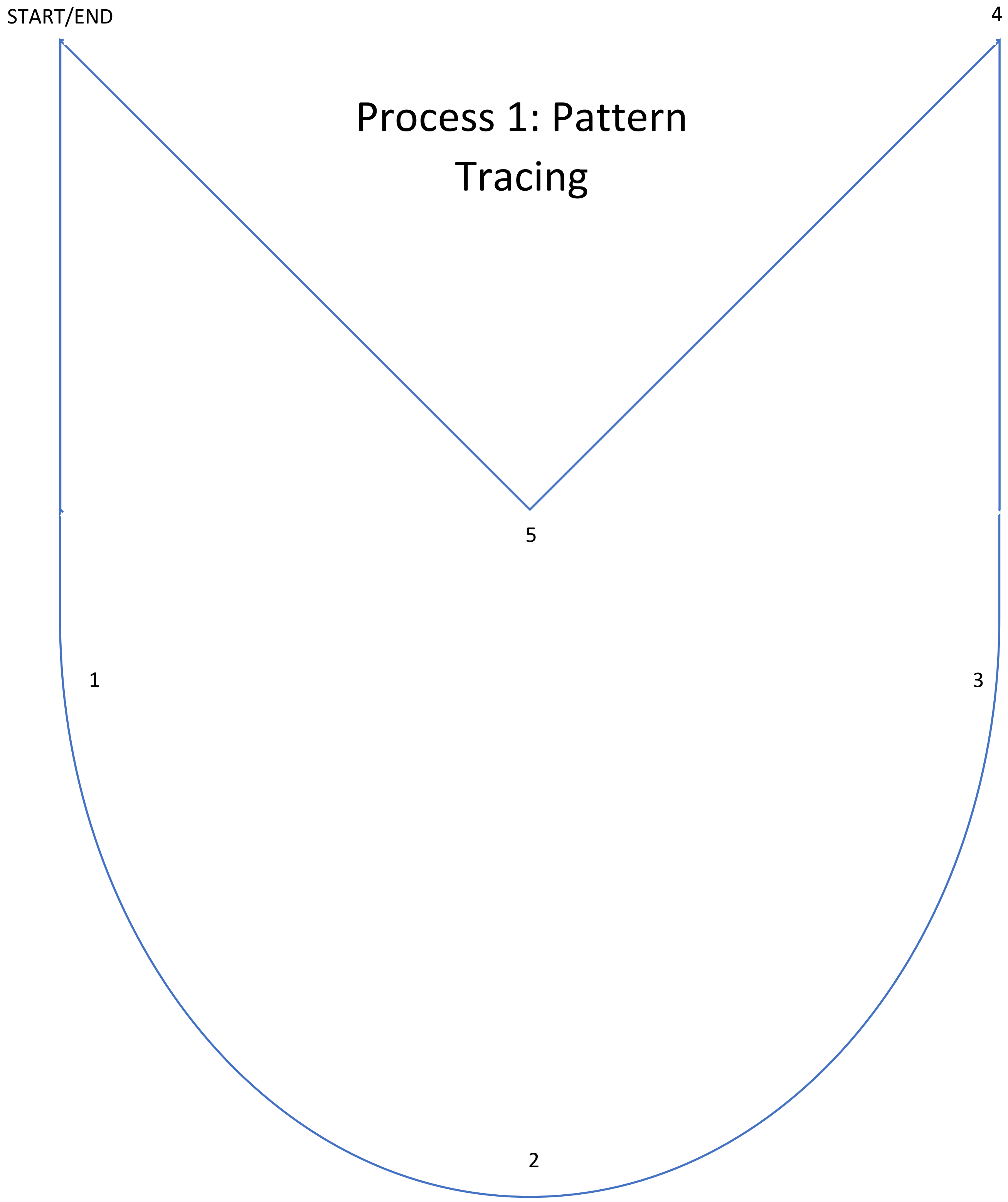
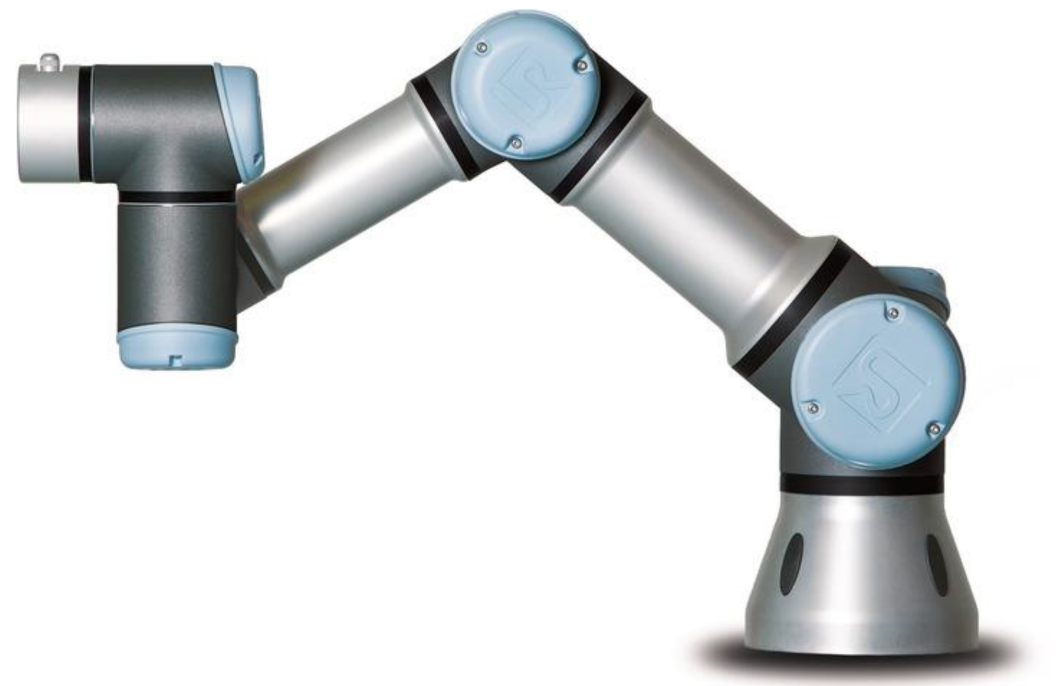


Figure 23: Save Icon

Process 1: Pattern Tracing



UR3 Robotic Arm: Bearing Module



California State Polytechnic University, Pomona

Mechanical Engineering Department

2018

Objective:

The Bearing Module represents a 3-dimensional process which requires consideration of both TCP location and orientation. The objective of this lab is to create a looping process which unloads a bearing from the hopper, places and moves the bearing through the module, and reloads the bearing to the top of the hopper. If executed properly, the program will be able to run indefinitely until the program is stopped.

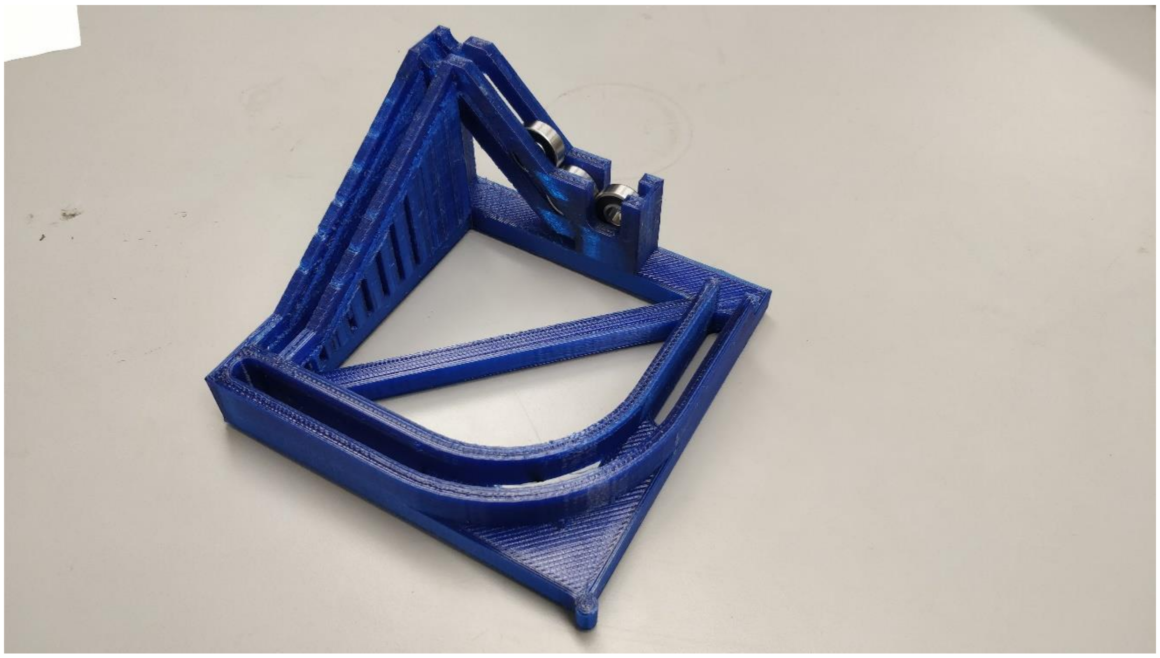


Figure 1: Bearing Module Process

Equipment:

- 1x Metal Rod attachment
- 1x Bearing Module
- 4x 608 bearings

Procedure:

Using the instructions from **Introduction to UR3 Robotic Arm**, create a program using the teach pendant.

- Choose an initial waypoint in space in which the tool is away from all physical objects, and the arm is in a configuration which all joints are not approaching singularities
    - After the program completes, the arm will MoveJ back to this waypoint
- The next important position requires the tool to be aligned concentric with the bearing to be picked up

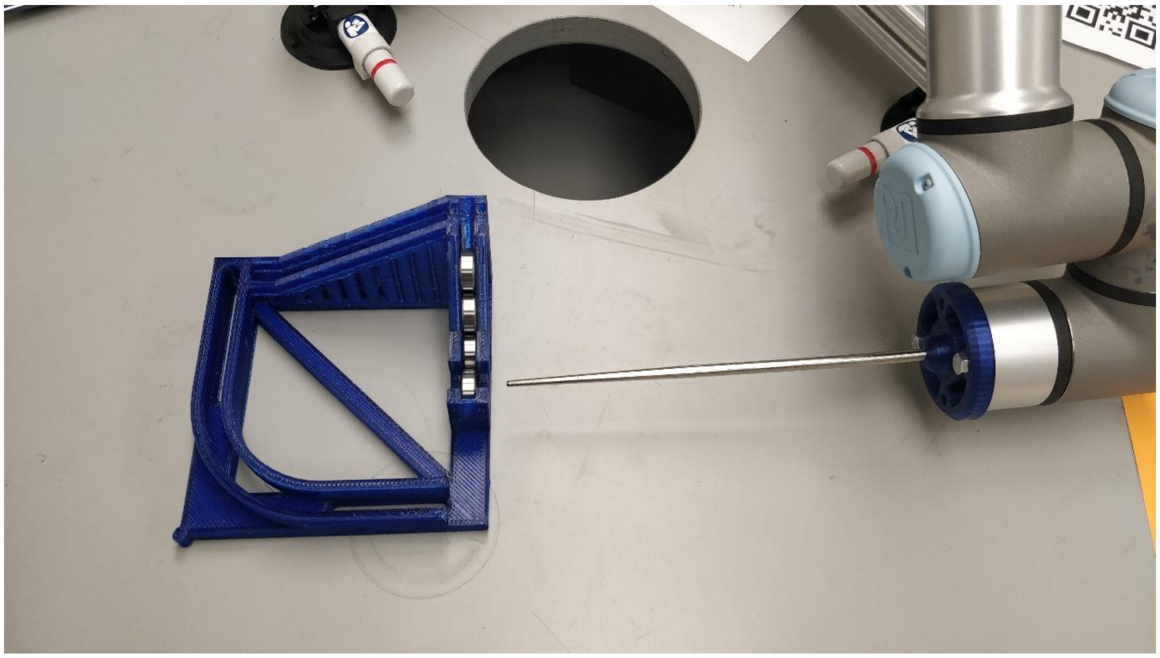


Figure 2: Tool Alignment with Bearing

- The tool can then be linearly moved into the center of the bearing as shown:

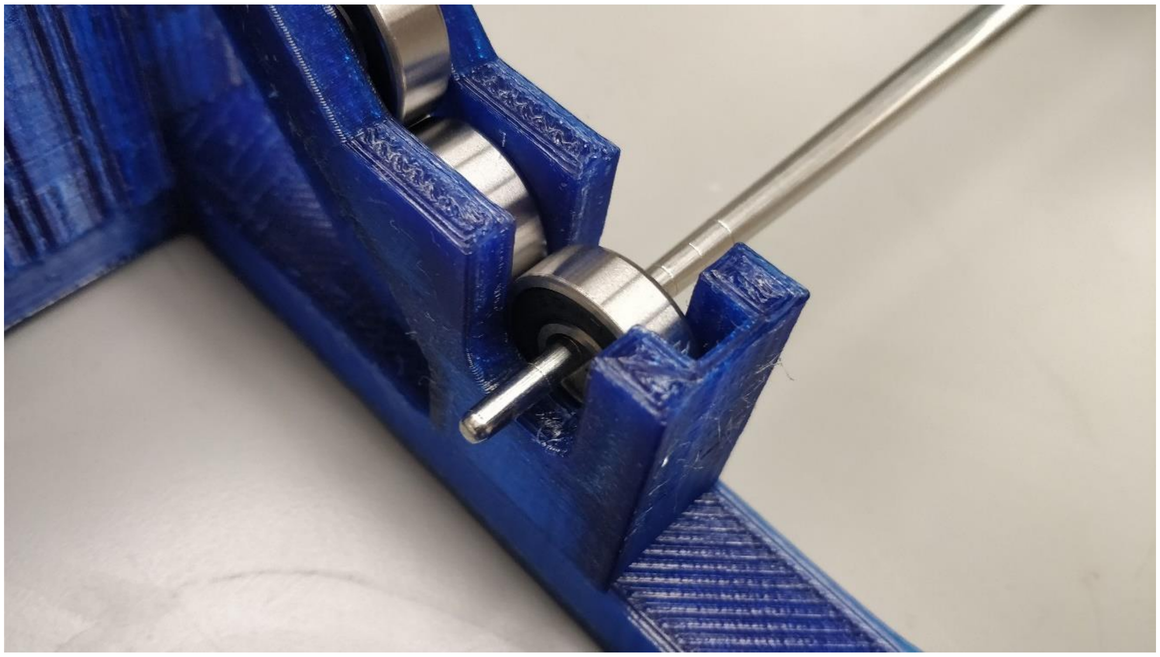


Figure 3: Tool Position to unload bearing from hopper

- Move the tool to retrieve the bearing out of the hopper. This can be accomplished using either a linear move or a rotation of wrist 2 if the plane of the joint is in a vertical orientation

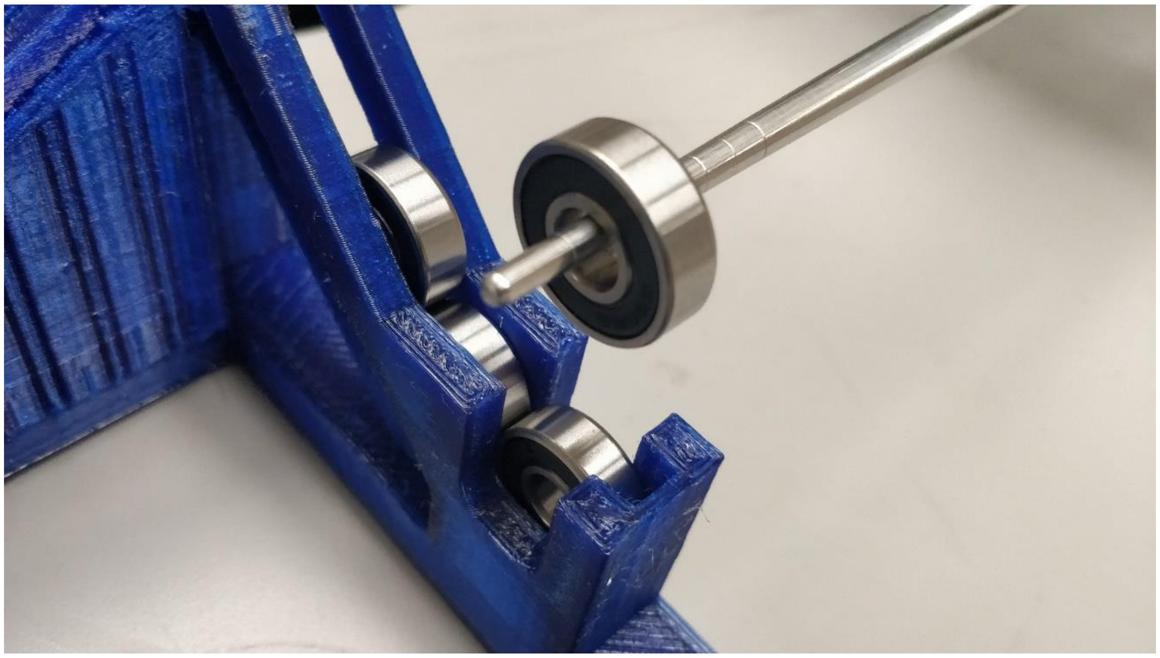


Figure 4: Successful Unloading of bearing from hopper

- The bearing is then placed on the lower track, and oriented to that the bearing is horizontal

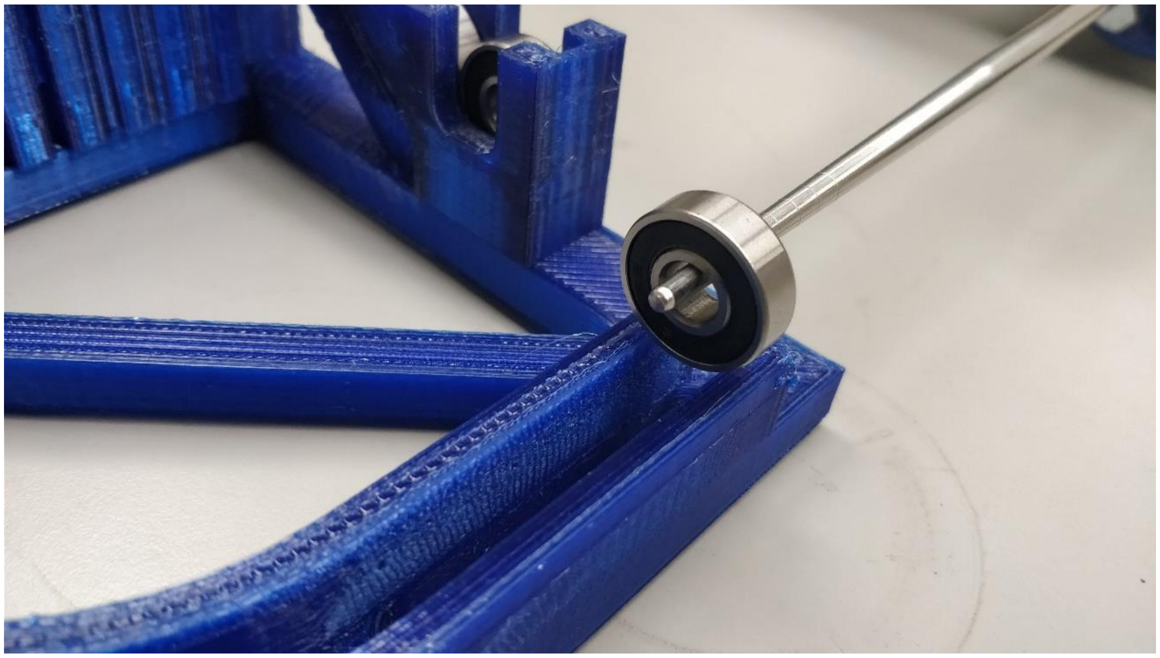


Figure 5: Placement of Bearing on lower track

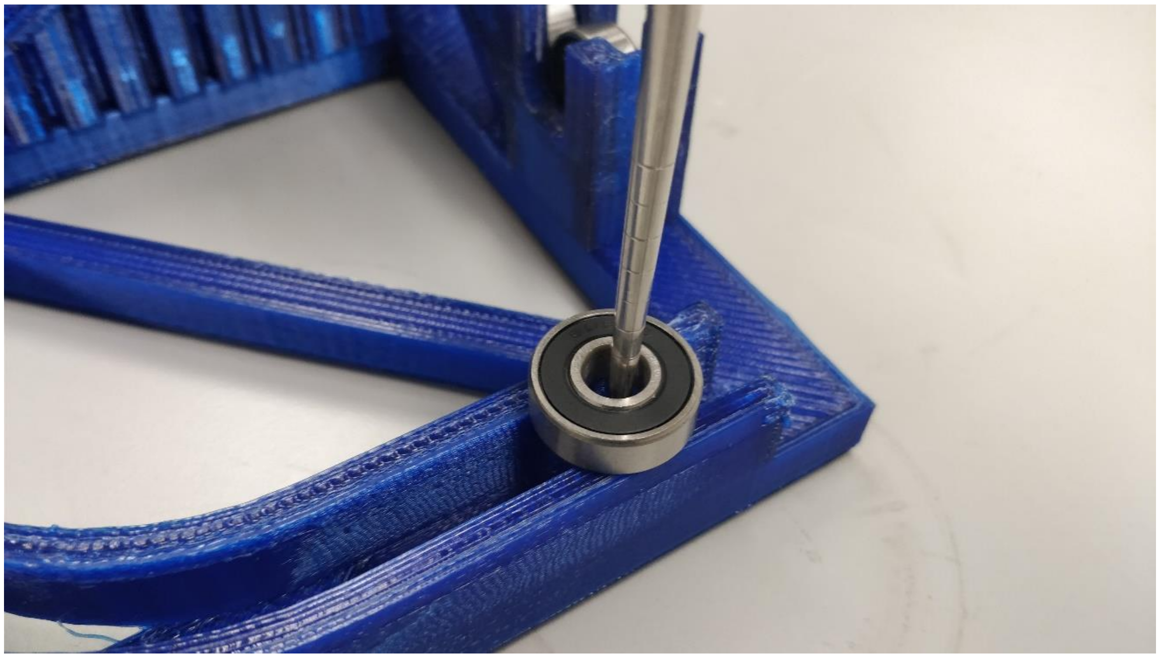


Figure 6: Corrected Orientation of bearing on lower track

- Once the bearing has been placed on the track, it needs to be moved to the opposite corner of the module. Use a process move with a blend radius to keep the bearing on track through the curve

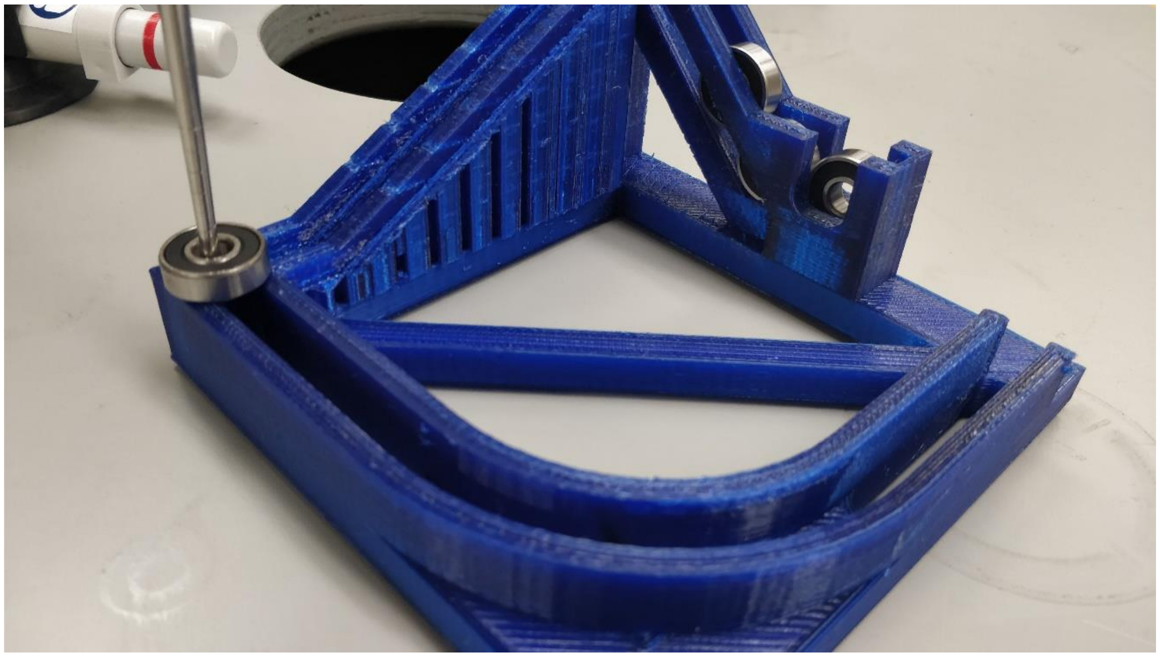


Figure 7: Movement of bearing along lower track

- The next step is to move the bearing up the ramp with a linear move

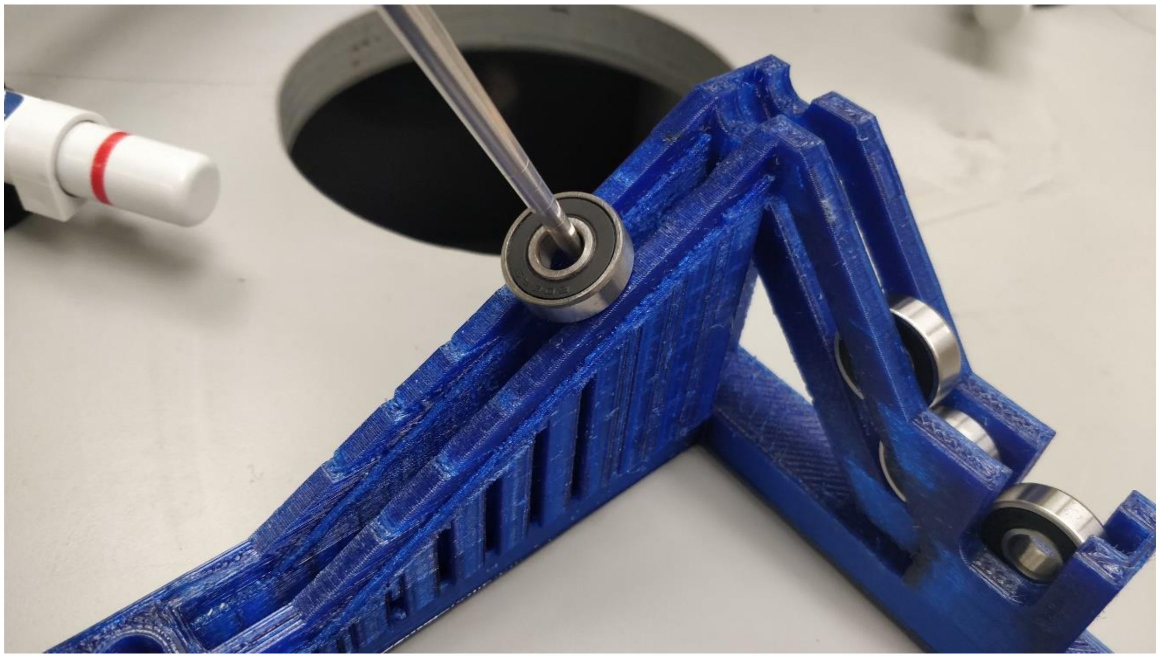


Figure 8: Movement of bearing along upper track

- Finally, once the bearing is situated at the top of the ramp, rotate the arm about the TCP to orient the bearing back in to the hopper. After the last waypoint, the arm will move back to the initial waypoint and the program will start again. This completes the bearing module process

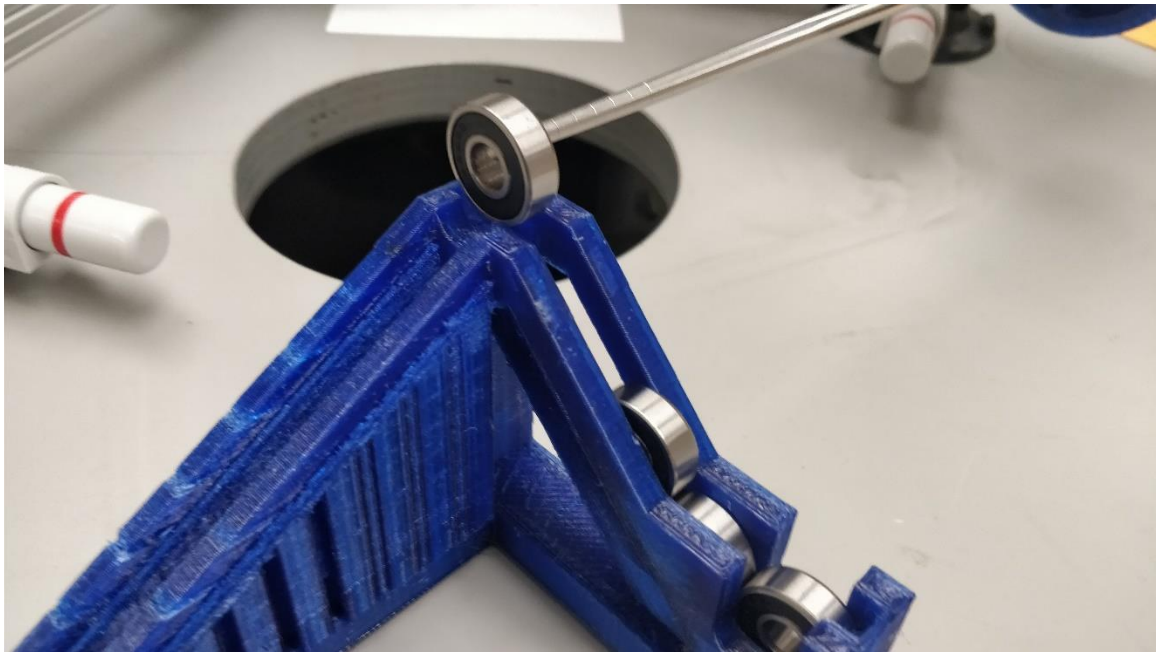


Figure 9: Reloading bearing into hopper

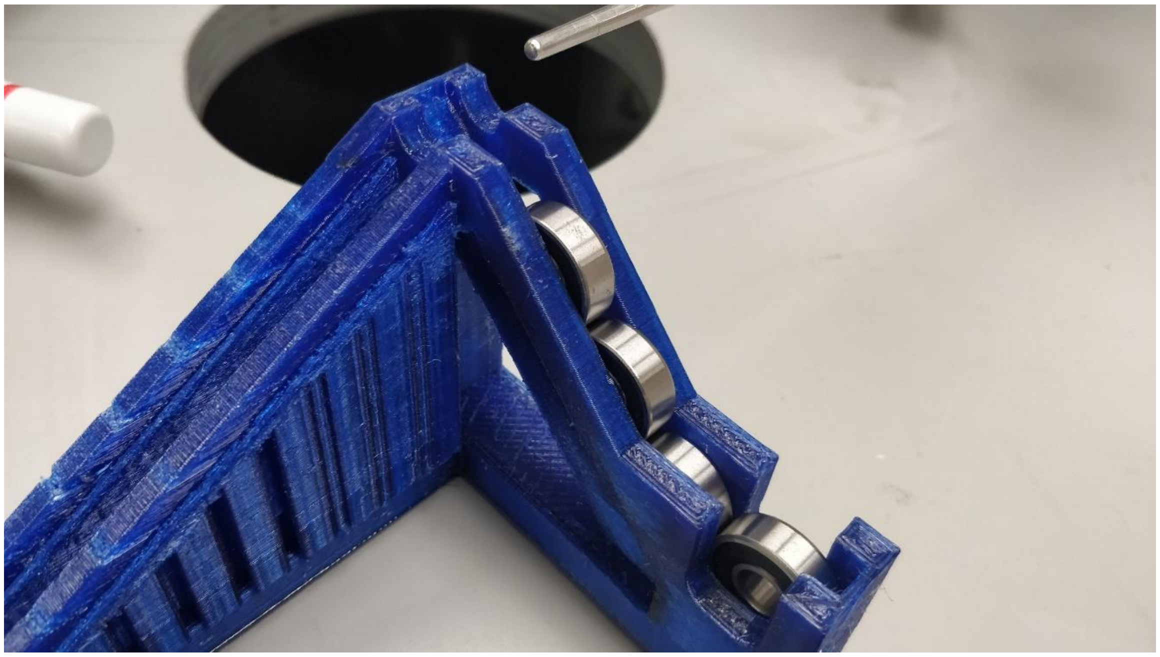
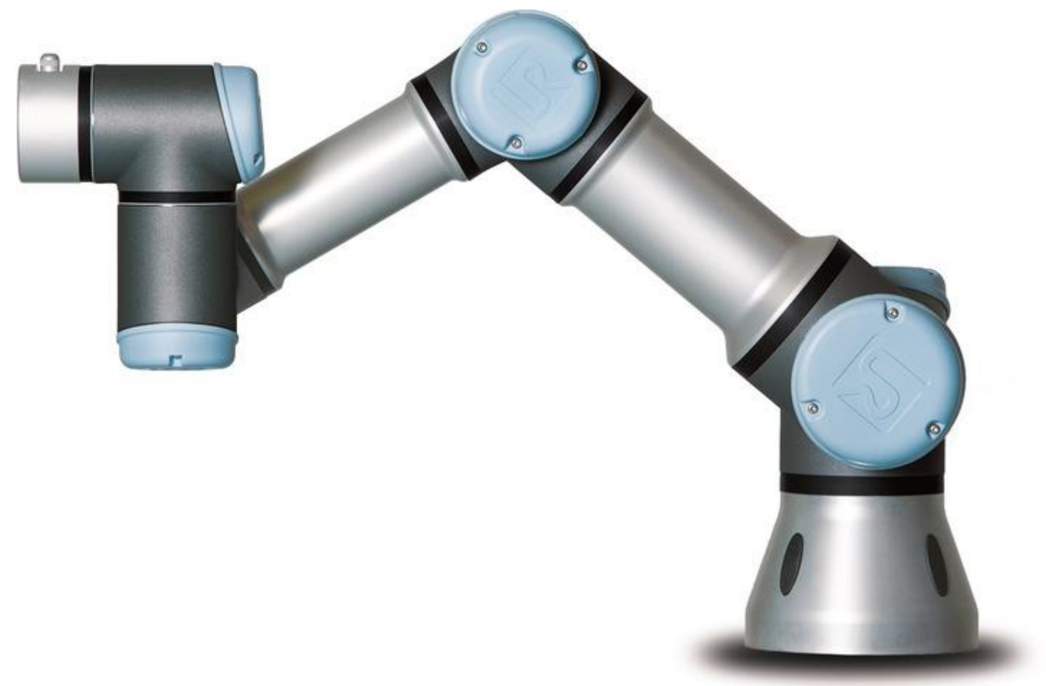


Figure 10: Completion of bearing module program

UR3 Robotic Arm Lab - GPIO



California State Polytechnic University, Pomona

Mechanical Engineering Department

2018

Table of Content

Objective:

In this lab, you will interface with the General Purpose Input/Output (GPIO) of the UR3 Robotic Arm through hardware and programming using the teach pendant. A switch will provide an input to the UR3 to start a program, and an output will turn on a LED signifying the completion of the program.

Equipment:

- UR3 robotic arm
- Control box
- Teach pendant
- Microswitch
- 24v Led

Theory:

GPIOs are useful for enabling the UR3 to interact with other hardware and sensors during operation. For example, break-beam sensors can provide an input signal to the arm to denote when a part is ready to be picked up, and an output signal can be used to control the position of a gripper tool mounted to the end of the arm. The GPIO is accessible inside the control box of the UR3 arm, and consist of Configurable, Digital, and Analog Inputs/Outputs. For the UR3 arm, the Input pin logic is considered LOW between 0-5v, HIGH between 11-24v, and take a maximum input current of 15mA. Output pins are available to sink to ground, which can carry a maximum current of 1A. For this lab, we will be interfacing with the Digital Inputs/Outputs, denoted by the grey plugs:

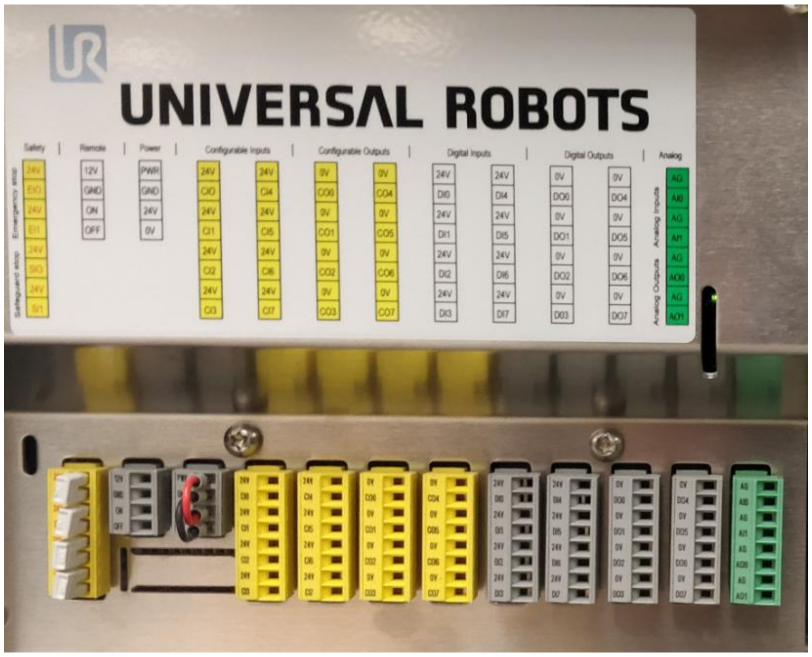


Figure 1: GPIO within UR3 Control Box

Within the software on the teach pendant, the I/O tab provides a real time display of the current states of the GPIO, with a blue circle indicating HIGH, and a grey circle indicating LOW:

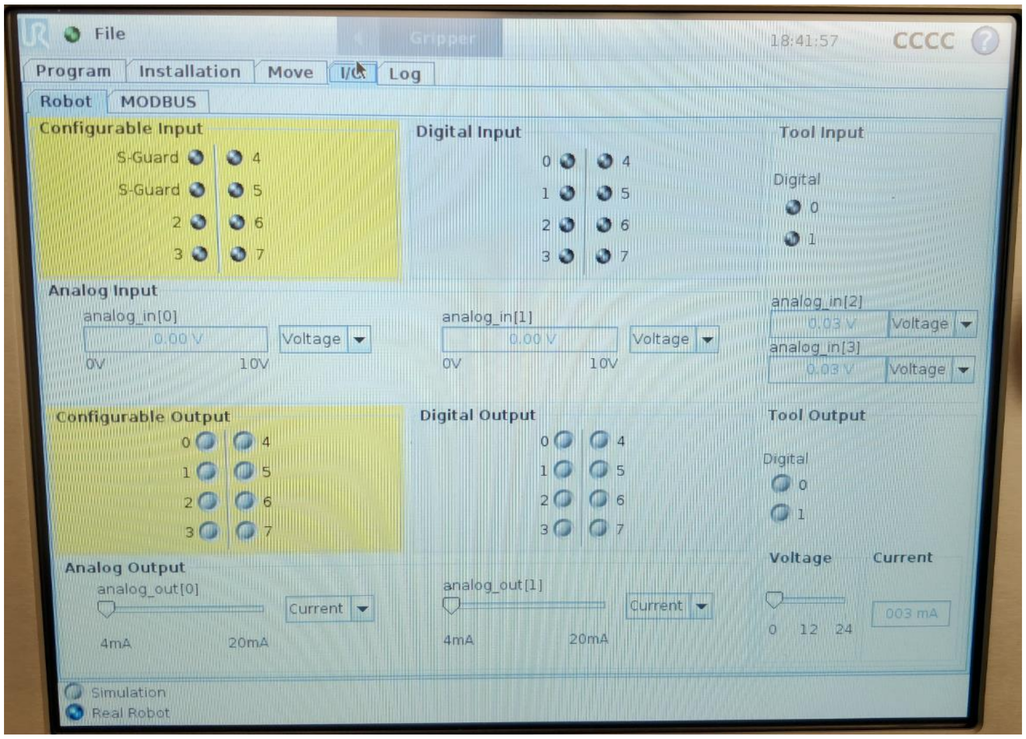


Figure 2: I/O Display on Teach Pendant

Results:

Create a program which starts once a switch is pressed. After the program completes, a LED should turn on. The program should once again wait for the input signal from the switch before running again.

Procedure:

1) Wire a switch between 24v and DI1 on the grey digital input plug as shown:

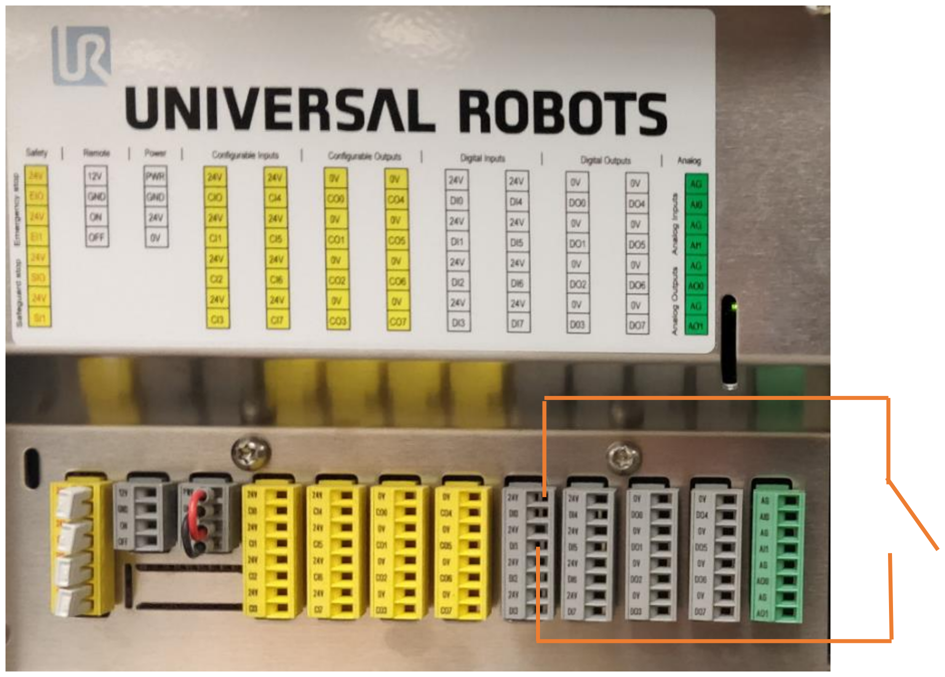


Figure 3: Wiring schematic of input switch

2) Wire a 24v tolerant LED between 0v and DO1 on the grey digital output plug as shown:

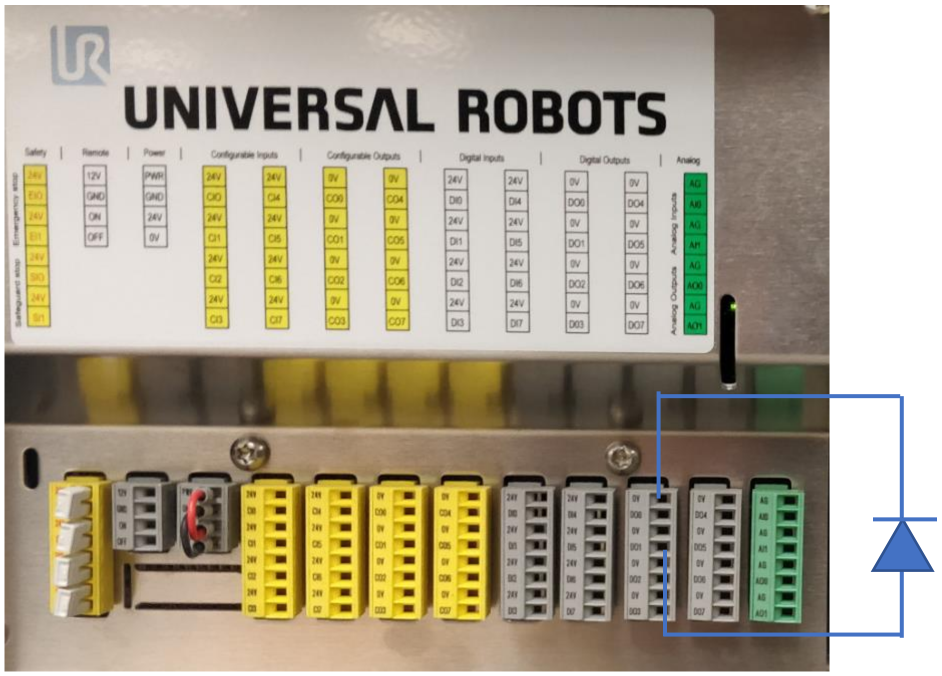
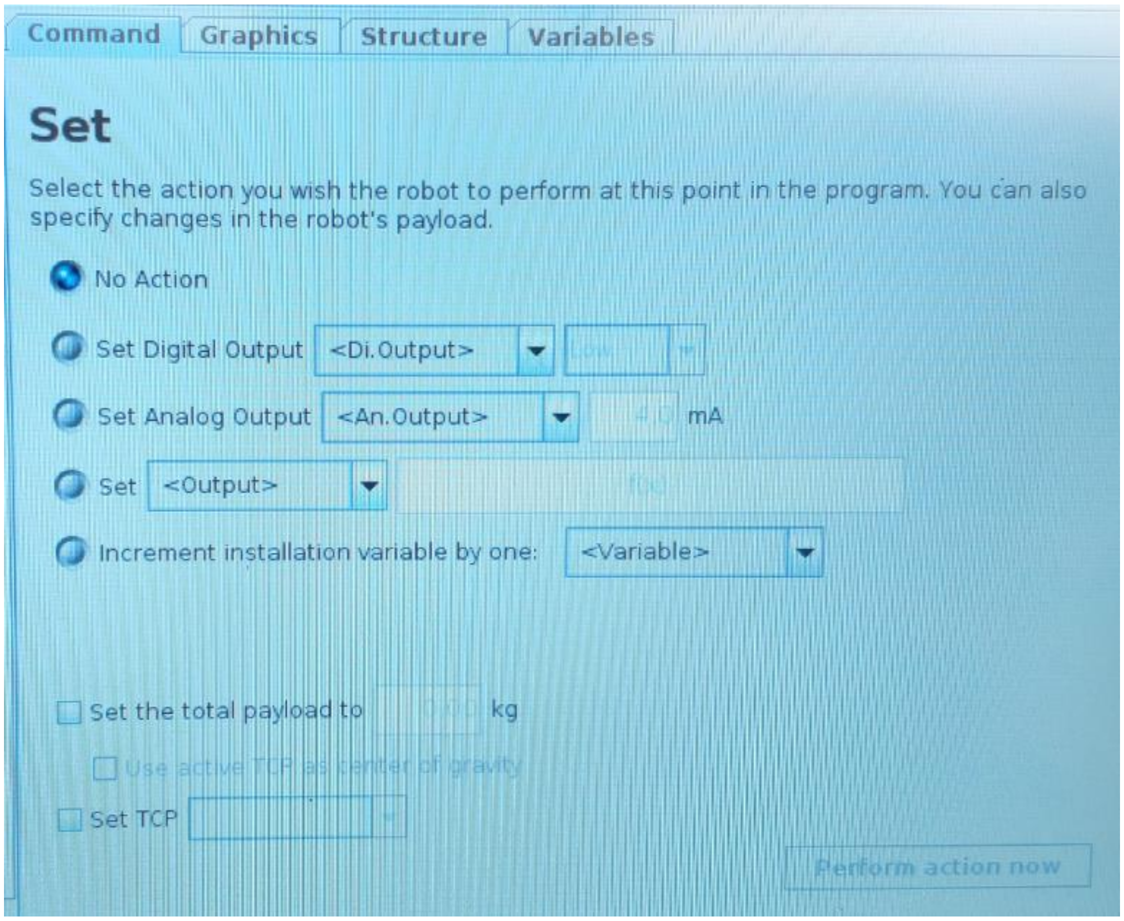


Figure 4: Wiring schematic of output LED

3) Start up and calibrate the UR3 arm using procedures from the **Introduction to the UR3 Robotic Arm** lab.
4) Generate a program with at least 3 waypoints. First and last waypoint should be the same location and orientation. The remaining waypoints can be arbitrary. These will be the program that will run between our input command and output signal.
5) Once a program has been created, select the first **MOVE** in the program structure tree, and from the structure tab add a **SET** command underneath the waypoint. Select the command tab to edit the action:


Command | Graphics | Structure | Variables

Set

Select the action you wish the robot to perform at this point in the program. You can also specify changes in the robot's payload.

[✓ No Action]

Set Digital Output <Di.Output> Low

Set Analog Output <An.Output> mA

Set <Output>

Increment installation variable by one: <Variable>

Set the total payload to kg

Use active TCP as center of gravity

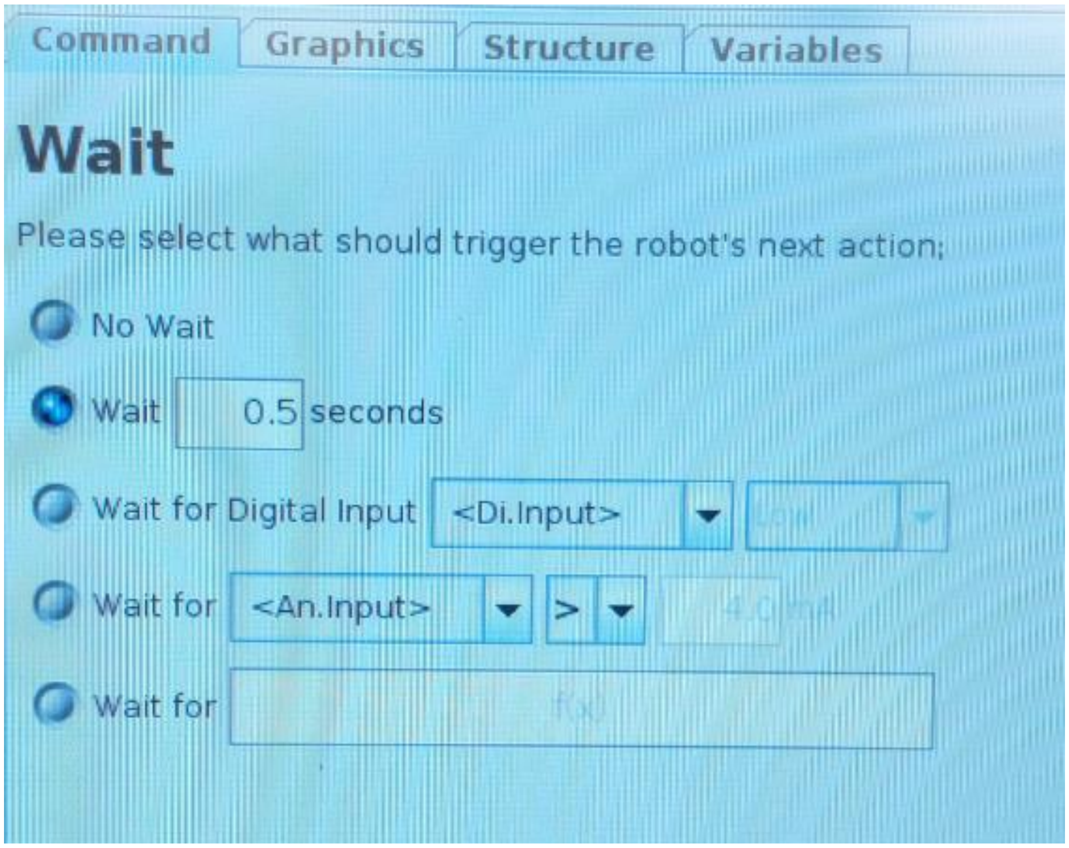
Set TCP

Perform action now

Figure 5: The Set command page on the teach pendant

Set the Digital Output DO1 to LOW. This ensures the LED is turned off before the program begins.

6) Next, from the structure tab, add a **WAIT** command directly under the **SET** command that was just created. Under the command tab, you can adjust the functionality:



Wait

Please select what should trigger the robot's next action:

No Wait

[✓ Wait] 0.5 seconds

Wait for Digital Input <Di.Input>

Wait for <An.Input> >

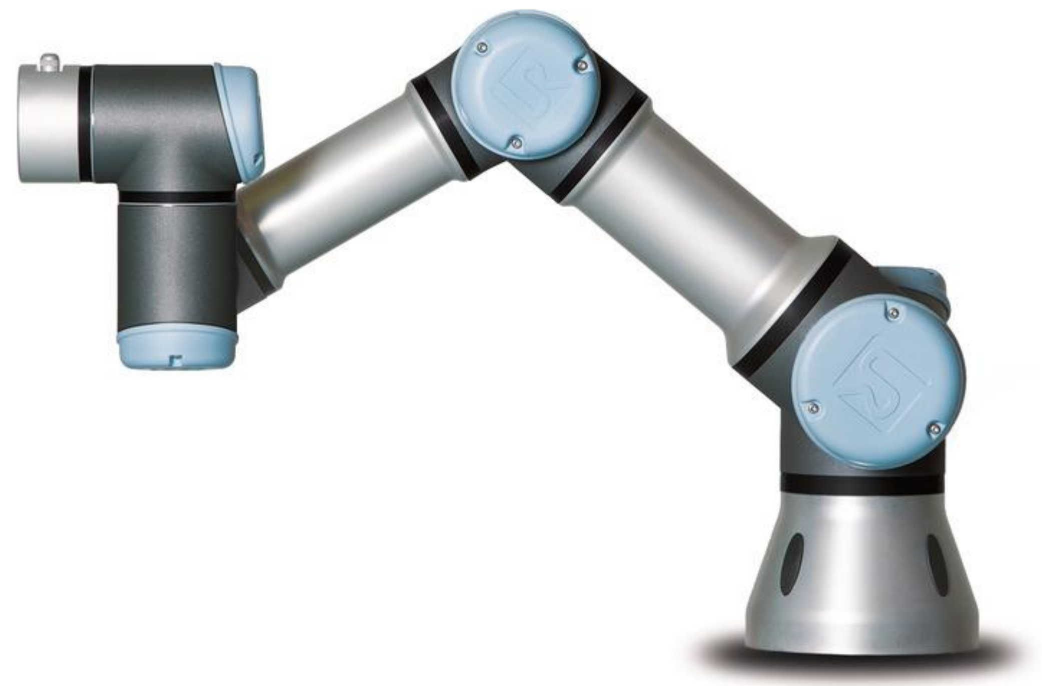
Wait for

Figure 6: Wait command parameters

Select Wait for Digital Input of DI1 to be HIGH in this tab. This waits for the input signal of the switch wired earlier.

7) Finally, select the last move in your program structure tree. Navigate to the Structure tab and add a **SET** command underneath the waypoint. Select the command tab, and set DO1 to HIGH. This turns on the LED, signifying the completion of the program.

Palletizing with UR3 Robotic Arm



California State Polytechnic University, Pomona

Mechanical Engineering Department

2018

Objective:

Create a palletizing program for the UR3 robot utilizing the teach pendant and OE software.

Equipment:

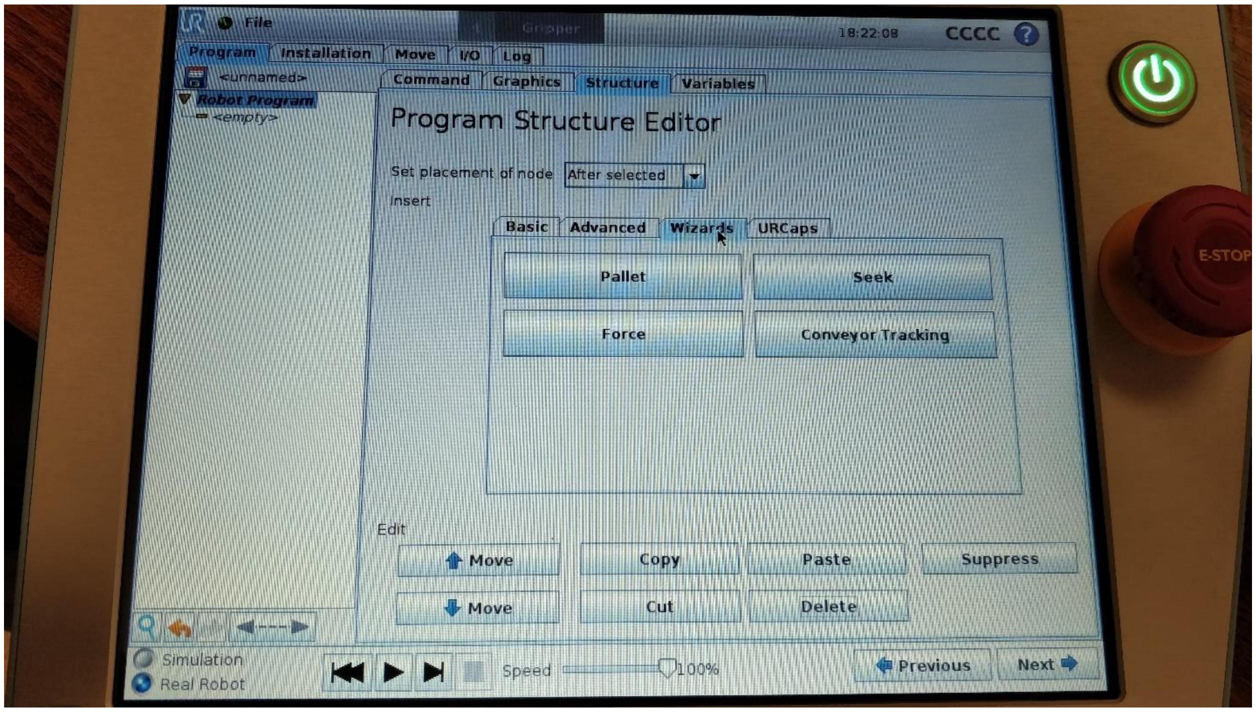
- 6 DOF robotic arm
- Control box
- Teach pendant

Theory:

The pallet wizard allows the UR3 robot arm repetitively execute movements in a predefined array. This is known as palletizing, and is useful for applications such as part unloading and packaging.

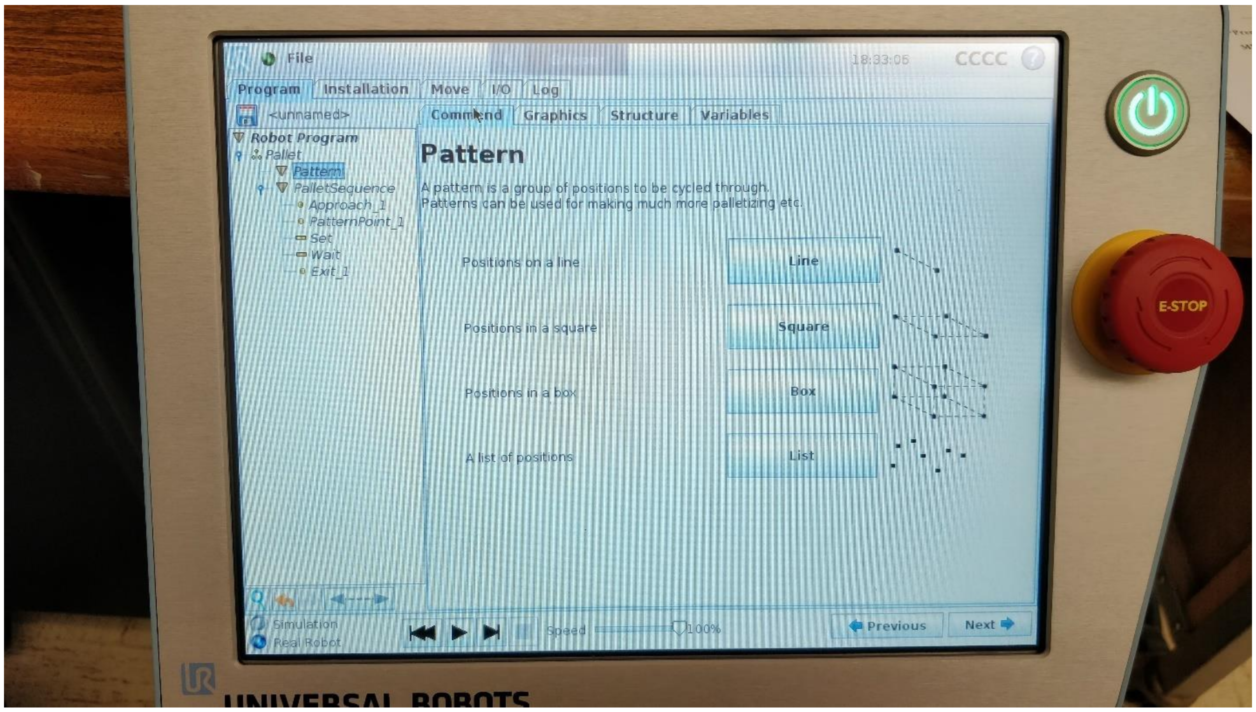
Procedure:

- Startup the UR3 arm, initialize and start a new program as described in Lab 1.
- Set the TCP parameters for the end effector being used.
- Select the “Program” tab, then the “Structure” tab, then the “Wizard” tab, and finally the “Pallet” button.



A Pallet structure is inserted in the program tree. In the Pattern folder, the type of pattern and number of positions in the pallet is set. In the Pallet Sequence folder, the sequence of movements that occur for each position is set. This includes how to approach a workpiece, how to place a workpiece, and how to exit agin from each position once a workpiece has been placed.

- Click the “Pattern” folder in the program tree, then the “Command” tab.



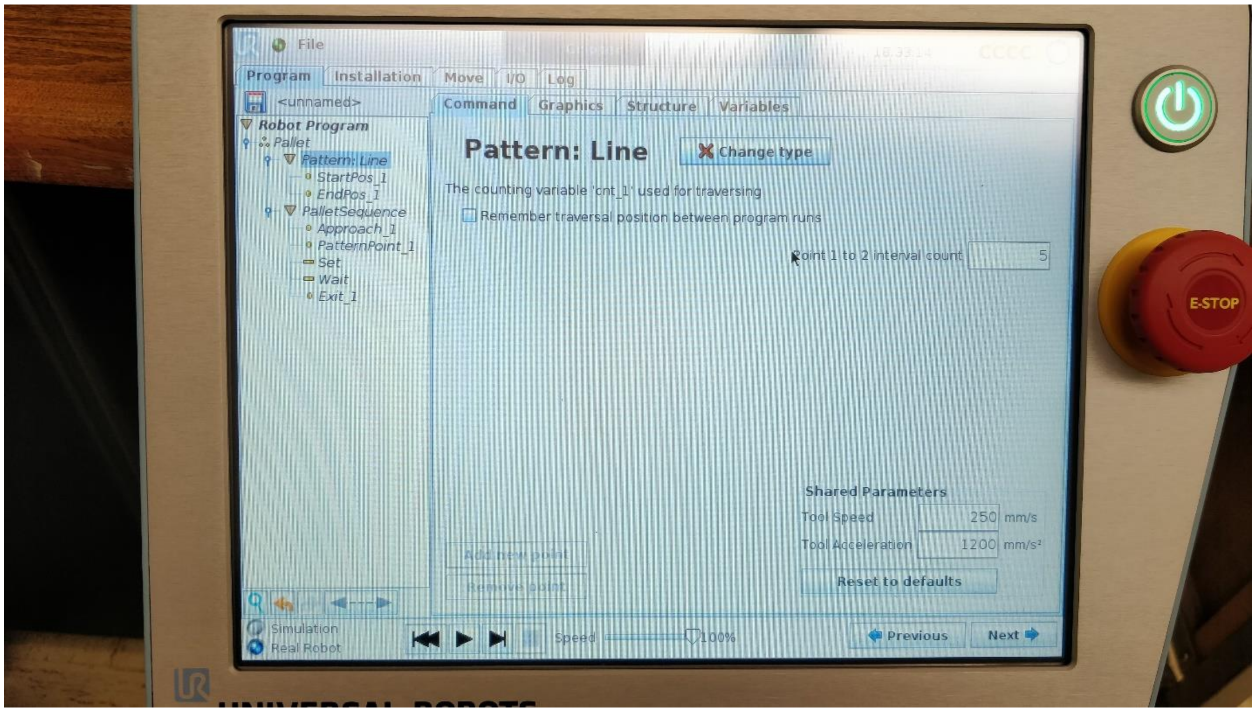
Pattern

A pattern is a group of positions to be cycled through.
Patterns can be used for making much more palletizing etc.

Positions on a line
Positions in a square
Positions in a box
A list of positions

Here you can select the pattern to be used. For this lab, select “Line.”

- A “StartPos_1” and “EndPos_1” are added to the Pattern folder of the program tree. Enter the number of times to iterate the pattern in the “Point 1 to 2 interval count” textbox.



Pattern: Line

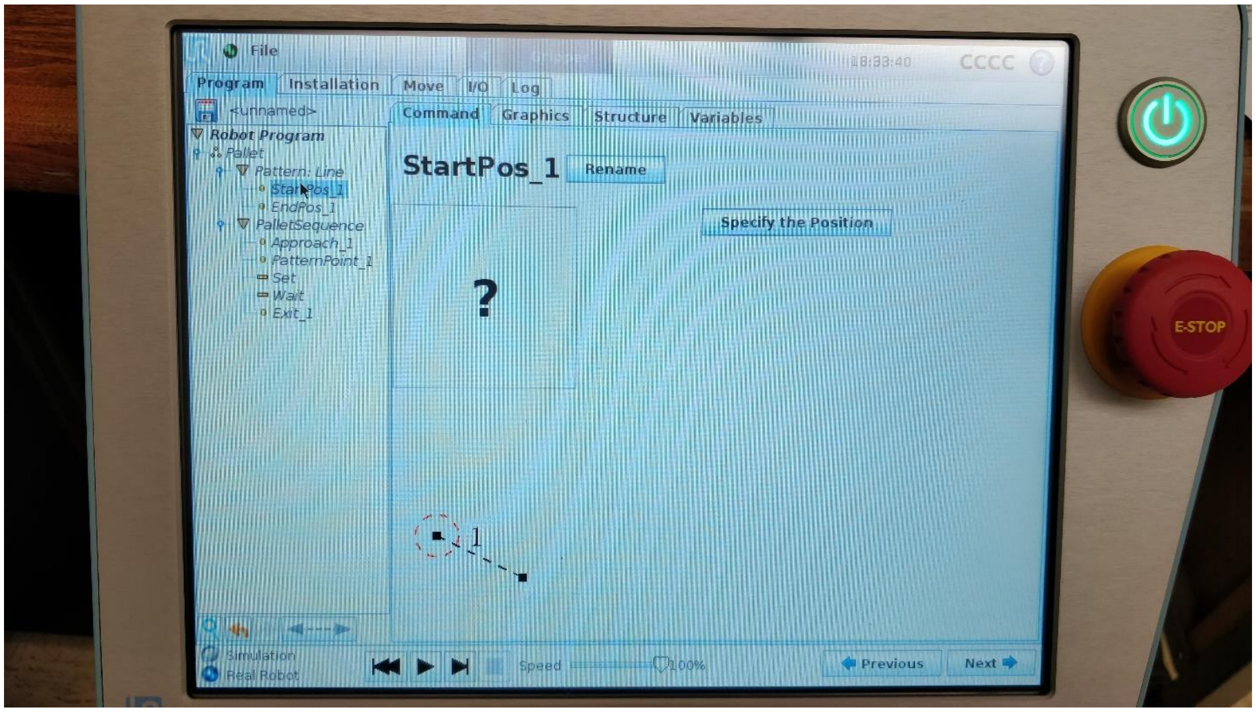
The counting variable 'cnt_1' used for traversing

Remember traversal position between program runs

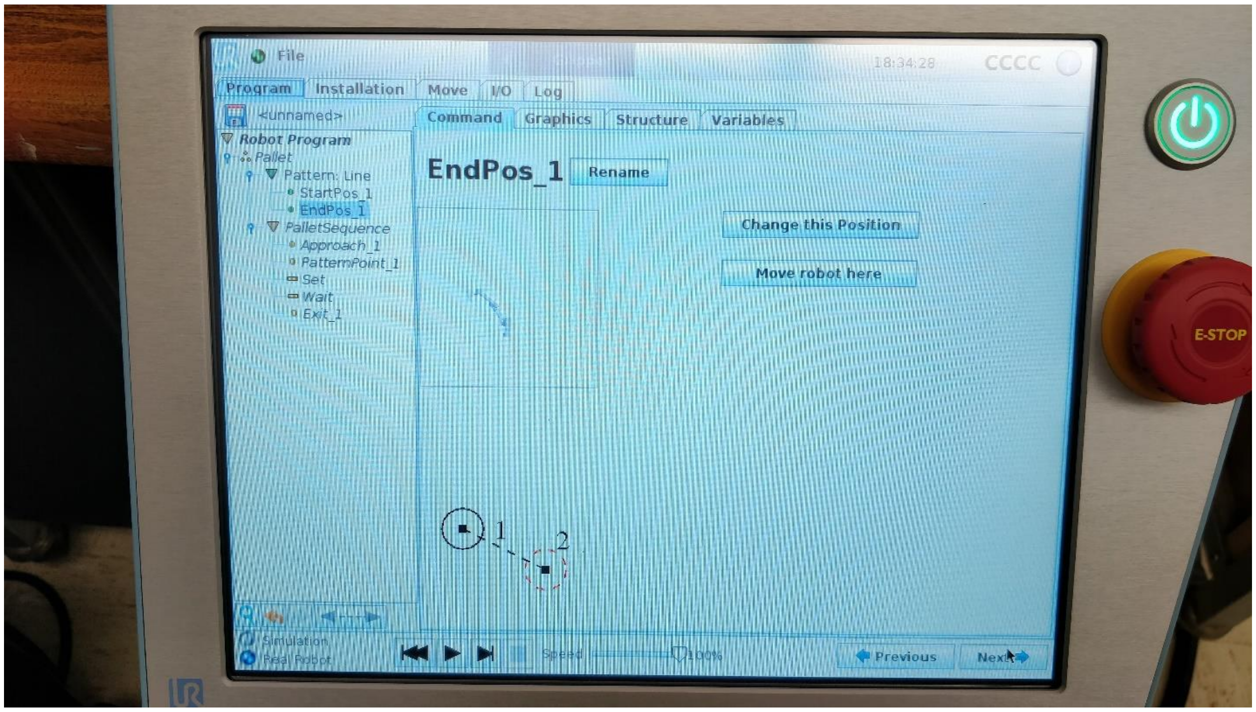
Point 1 to 2 interval count 5

Shared Parameters
Tool Speed 250 mm/s
Tool Acceleration 1200 mm/s²

- Select the “StartPos_1” waypoint in the program tree, then “Specify the Position”

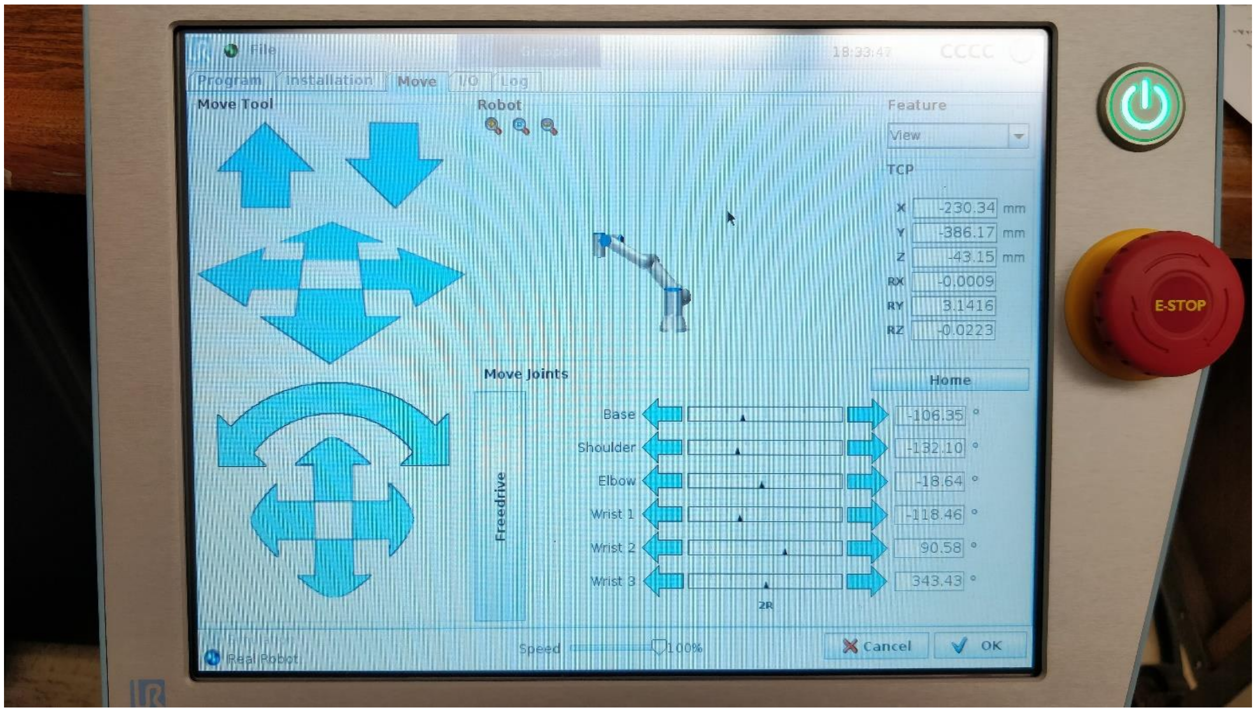


- Using the teach pendant controls or free drive, set the end effector to the starting position of the pattern, and press “OK.”
- Select the “EndPos_1” waypoint in the program tree, and assign it the location of the last position of the pattern, then press “OK.”



- The array of positions can visually be verified by selecting the “Graphics” tab

- Select “Approach_1” in the PalletSequence folder of the Program Tree. This waypoint is relative to StartPos_1 and establishes how the arm will move to each position of the pallet pattern. Place this waypoint a few inches above StartPos_1.

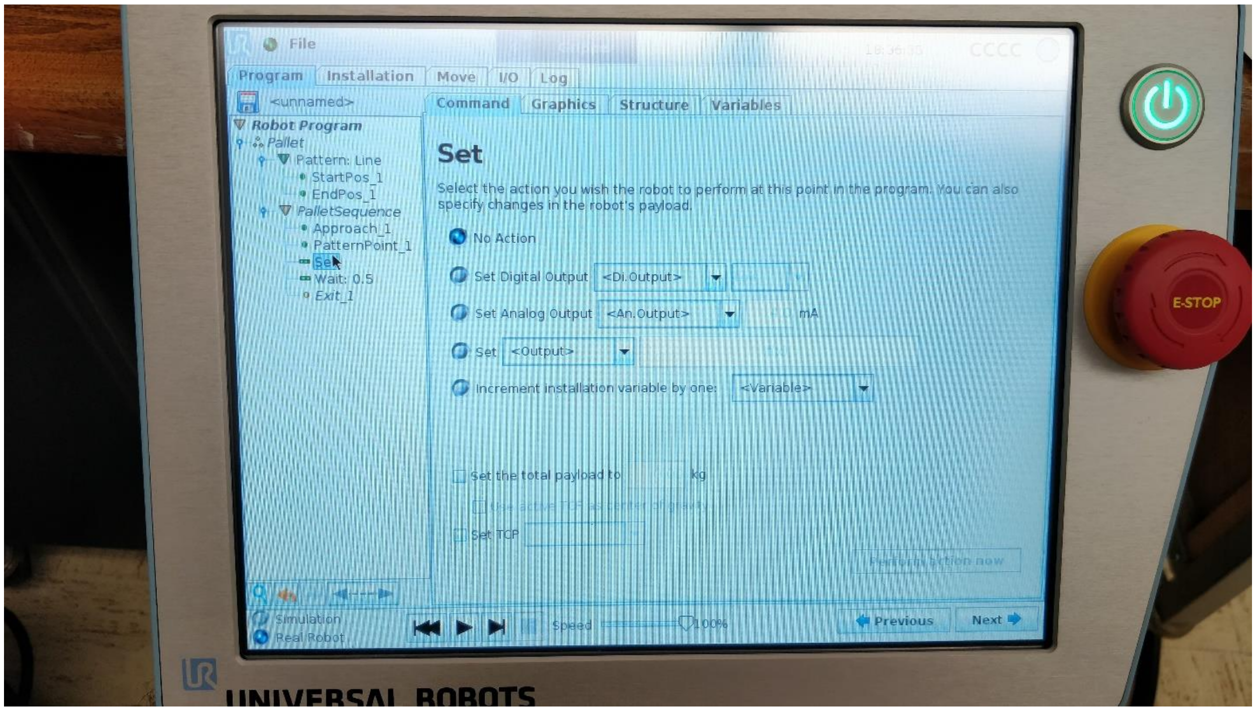


TCP

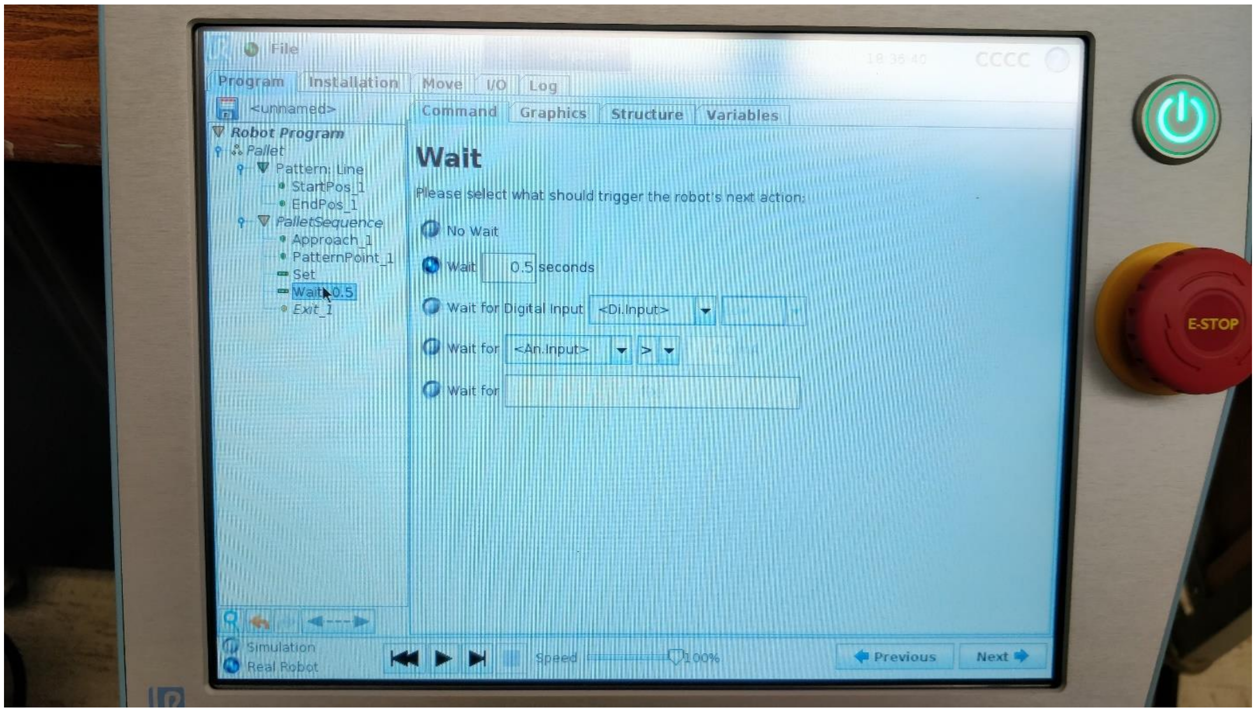
X	-230.34	mm
Y	-386.17	mm
Z	-43.15	mm
RX	-0.0009	
RY	3.1416	
RZ	-0.0223	

Base	-106.35 °
Shoulder	-132.10 °
Elbow	-18.64 °
Wrist 1	-118.46 °
Wrist 2	90.58 °
Wrist 3	343.43 °

- Similarly, “Exit” is a waypoint that the arm will move to after each pallet position. It is created relative to StartPos_1. Place this waypoint a few inches above StartPos_1 as well.
- “PatternPoint_1” represents the first waypoint of each pattern. Additional waypoints can be added to create the movements necessary at each pattern position.
- Select the “Set” action in the program tree. This action is performed at each position of the pallet pattern. It can be an action such as actuating a gripper to let go of an item. For this lab, select “No Action.”



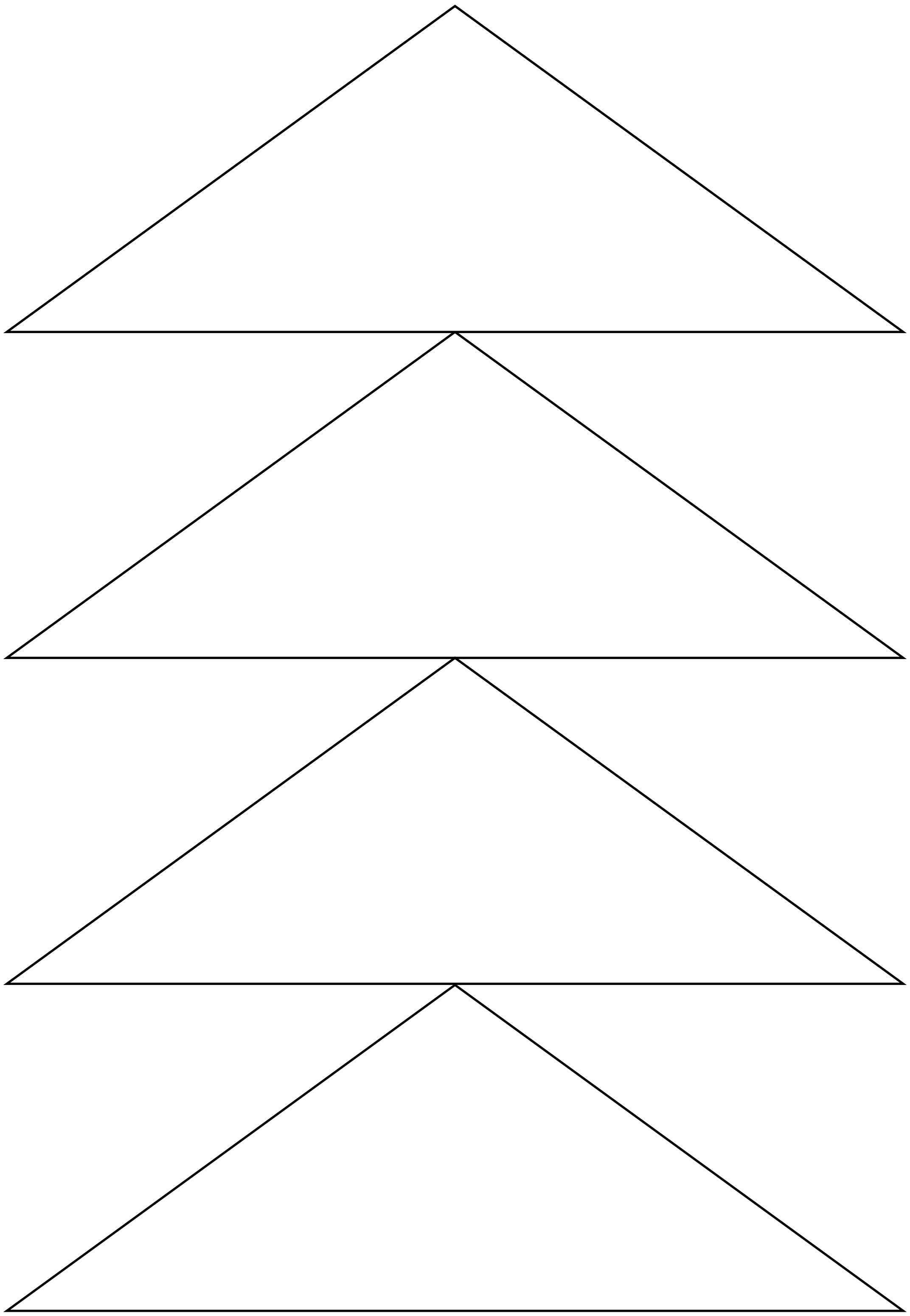
- Select the "Wait" action from the program tree. This action is used to allow the robot to dwell while the Set action is being performed. Set the robot to wait for 1 second.



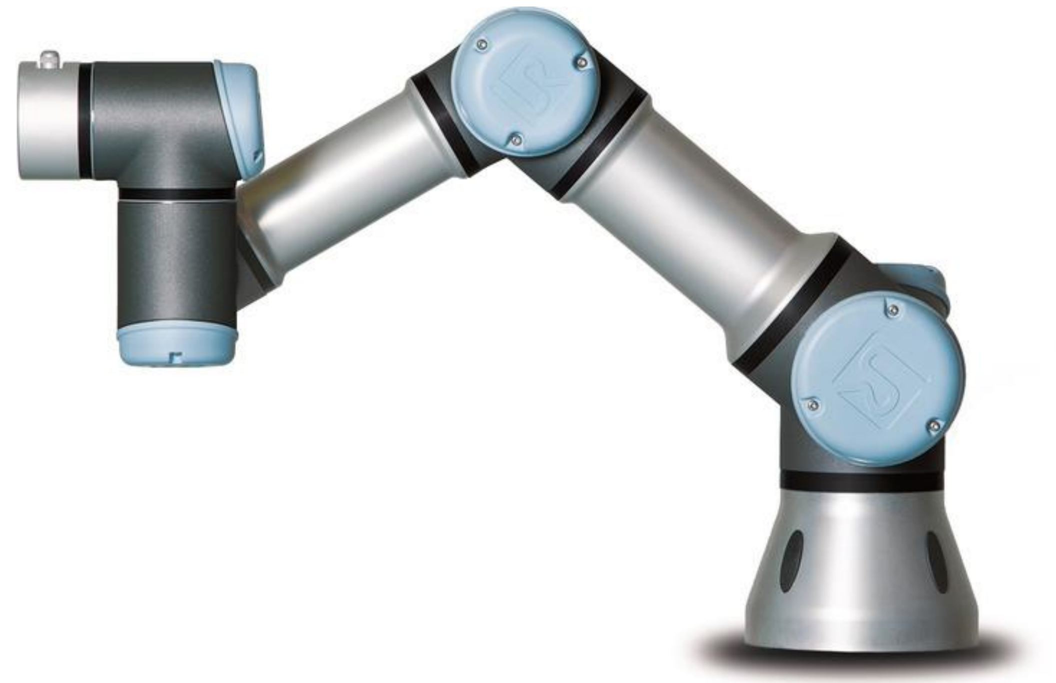
- The sequence created will apply to each position of the pallet pattern. Once the robot has ran through every position, it will return to the first position and iterate again. To stop the robot from looping:

    - Select “Robot Program” from the top of the Program tree, then the “Structure” tab.
    - Select the “Structure” tab, then the “Advanced” button, then the “Loop” button. A loop is added to the top of the program tree.
    - Select “Loop” at the top of the program tree, then set “Loop __ times” to the number of pattern positions
    - Select the “Structure” tab again, highlight the “Pallet” folder in the program tree, then select “Cut.”
    - Select “Loop” in the program tree, then the “empty” field below the loop folder, then select “Paste” to move the folder into the loop.
    - Select “Loop” in the program tree, then the “Structure” tab, the “Popup” button, then the “Command” tab.
    - In the text box, type in a message that will show up once the pallet sequence is finished.
- The program is ready to run, click the play button to start it.

Pallet Pattern 1



Simulation of UR3 Robotic Arm



California State Polytechnic University, Pomona

Mechanical Engineering Department

2017-2018

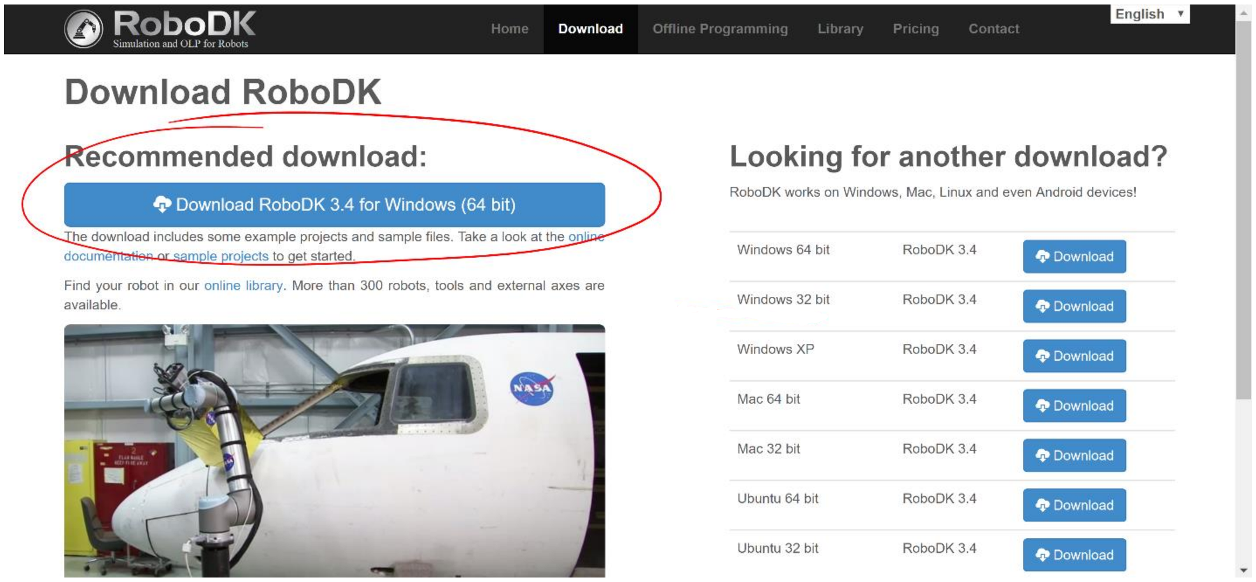
Objective: Develop the ability to program and simulate the UR3 robotic arm utilizing RoboDK software. Establish an understanding of the arm work envelope and singularities by creating path trajectories which work around them.

Equipment:

- 6 DOF robotic arm
- Control box
- Teach pendant
- RoboDK Bearing Module
- RoboDK Bearing Module - .STEP file

Procedure:

1) Download RoboDK software Bearing Module.STEP file
    a. RoboDK software available at: https://robodk.com/download
        i. Choose the "Recommended Download":



Download RoboDK

Recommended download:

Download RoboDK 3.4 for Windows (64 bit)

The download includes some example projects and sample files. Take a look at the online documentation or sample projects to get started.

Find your robot in our online library. More than 300 robots, tools and external axes are available.

Looking for another download?

RoboDK works on Windows, Mac, Linux and even Android devices!

Windows 64 bit	RoboDK 3.4	Download
Windows 32 bit	RoboDK 3.4	Download
Windows XP	RoboDK 3.4	Download
Mac 64 bit	RoboDK 3.4	Download
Mac 32 bit	RoboDK 3.4	Download
Ubuntu 64 bit	RoboDK 3.4	Download
Ubuntu 32 bit	RoboDK 3.4	Download

Figure 1: RoboDK Download Screen

        ii. Follow the installation wizard to complete the software install
    b. Obtain the Bearing module .STEP file from your professor

2) Open RoboDK software > File > Open Online Library > Find Universal Robots UR3 > Download UR3

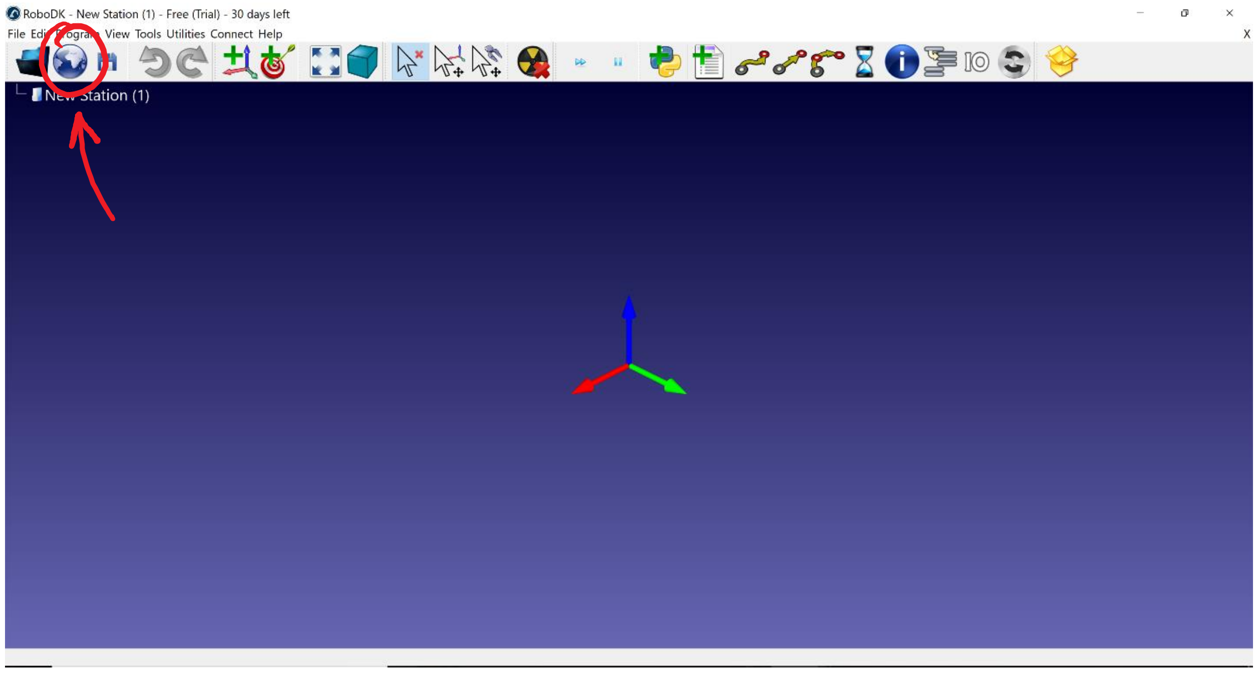

RoboDK - New Station (1) - Free (Trial) - 30 days left
File Edit Program View Tools Utilities Connect Help
New Station (1)

Figure 2: RoboDK Environment (Globe icon opens Online Library)

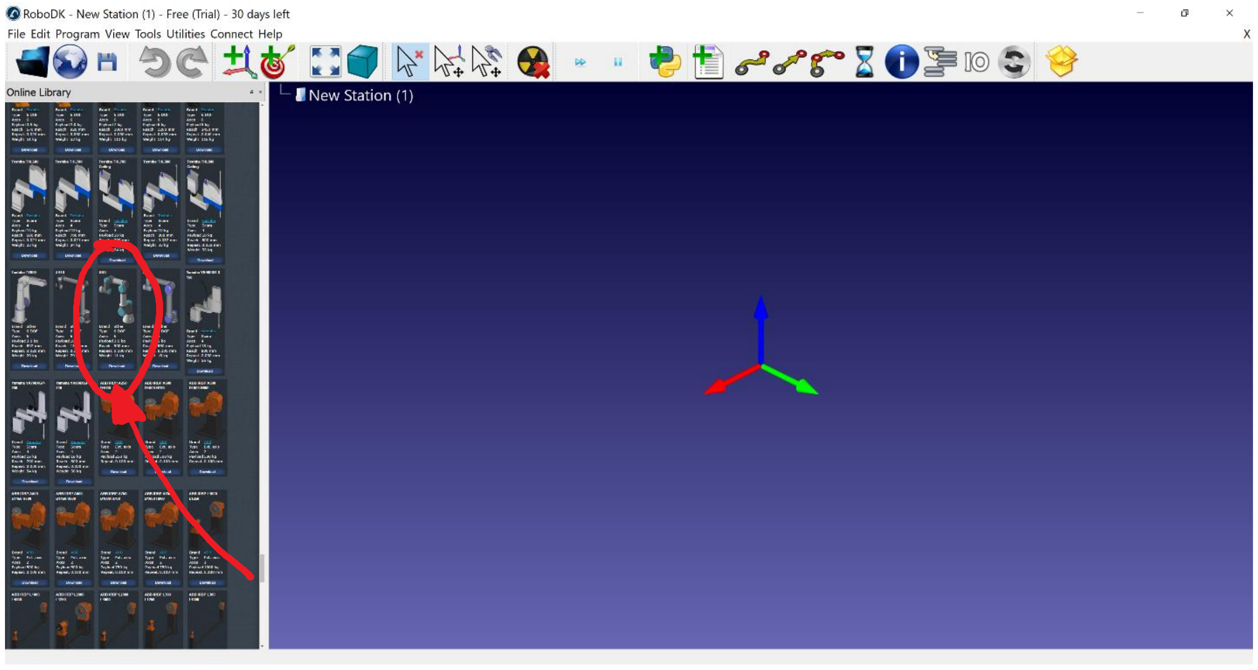

RoboDK - New Station (1) - Free (Trial) - 30 days left
File Edit Program View Tools Utilities Connect Help
Online Library
New Station (1)

Figure 3: Universal Robots UR3 in Online Library

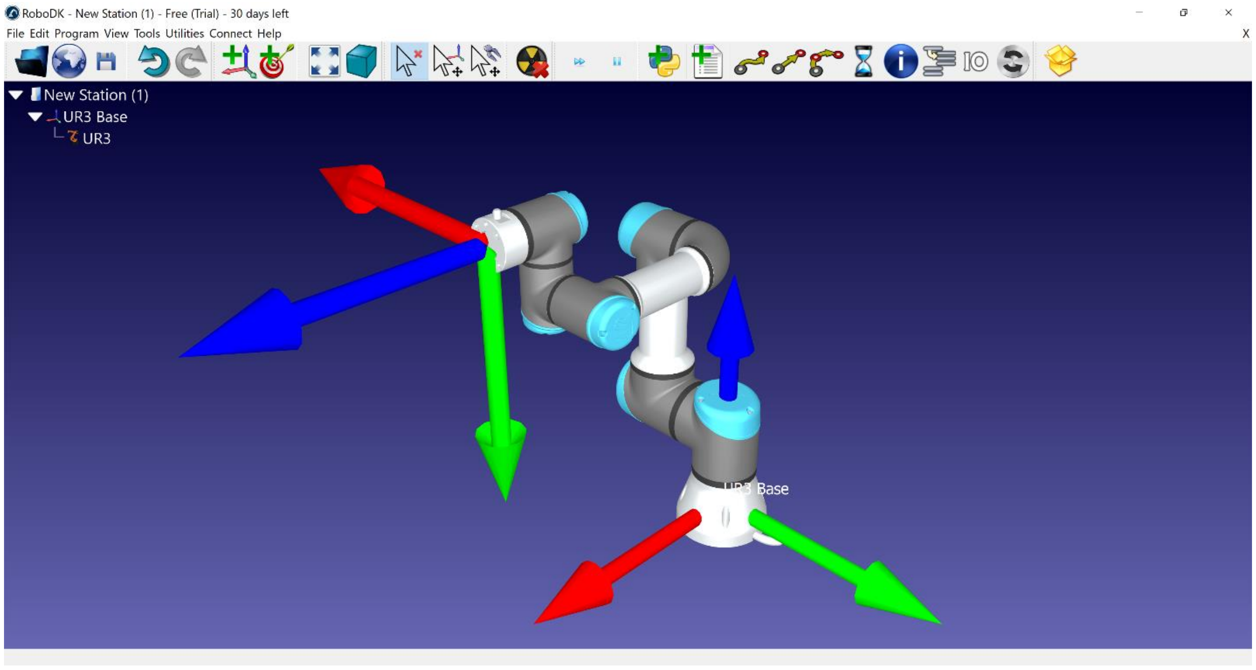

RoboDK - New Station (1) - Free (Trial) - 30 days left
File Edit Program View Tools Utilities Connect Help
New Station (1)
UR3 Base
UR3
UR3 Base

Figure 4: UR3 Loaded into Environment

3) In tree > right click UR3 > add TCP > modify translation and rotation values of TCP

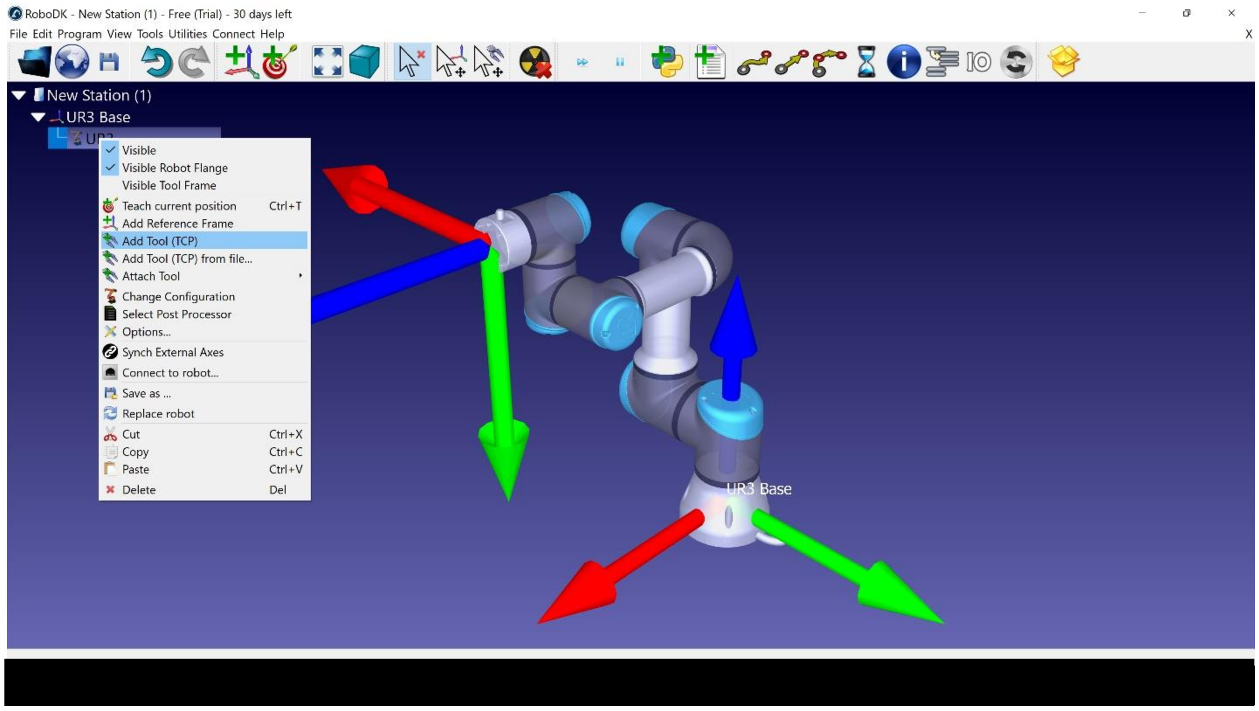


Figure 5: Add Tool (TCP)

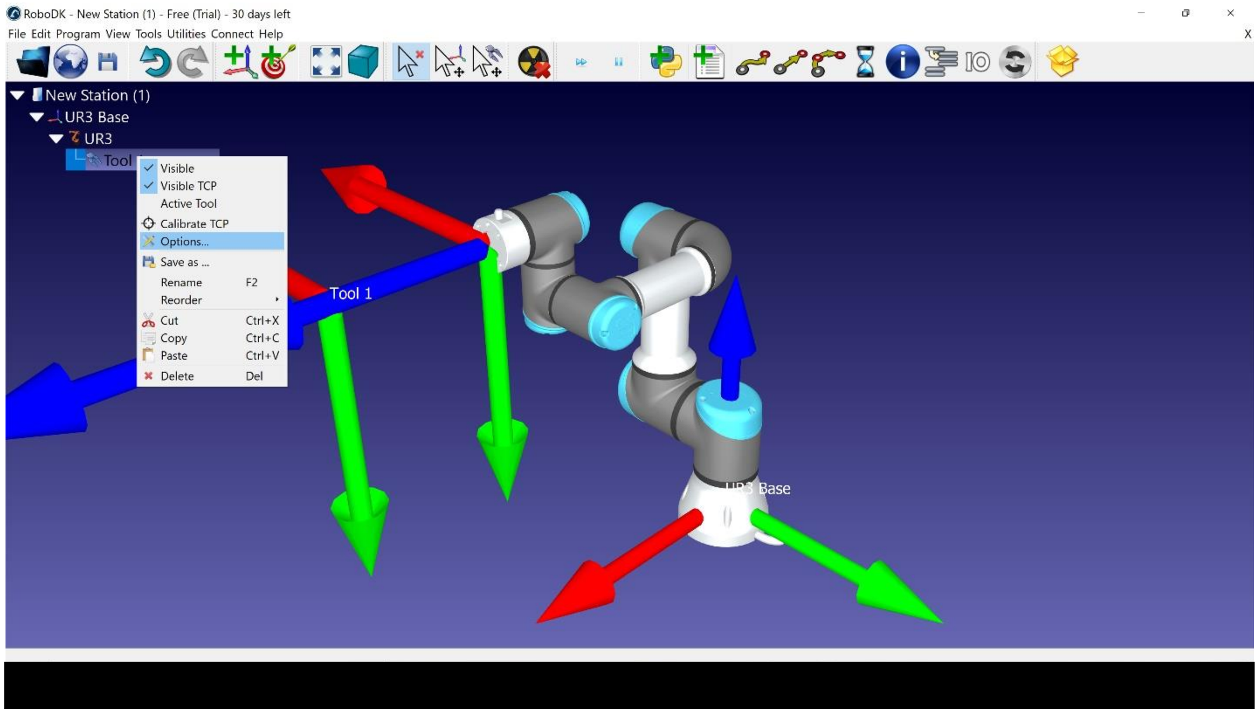
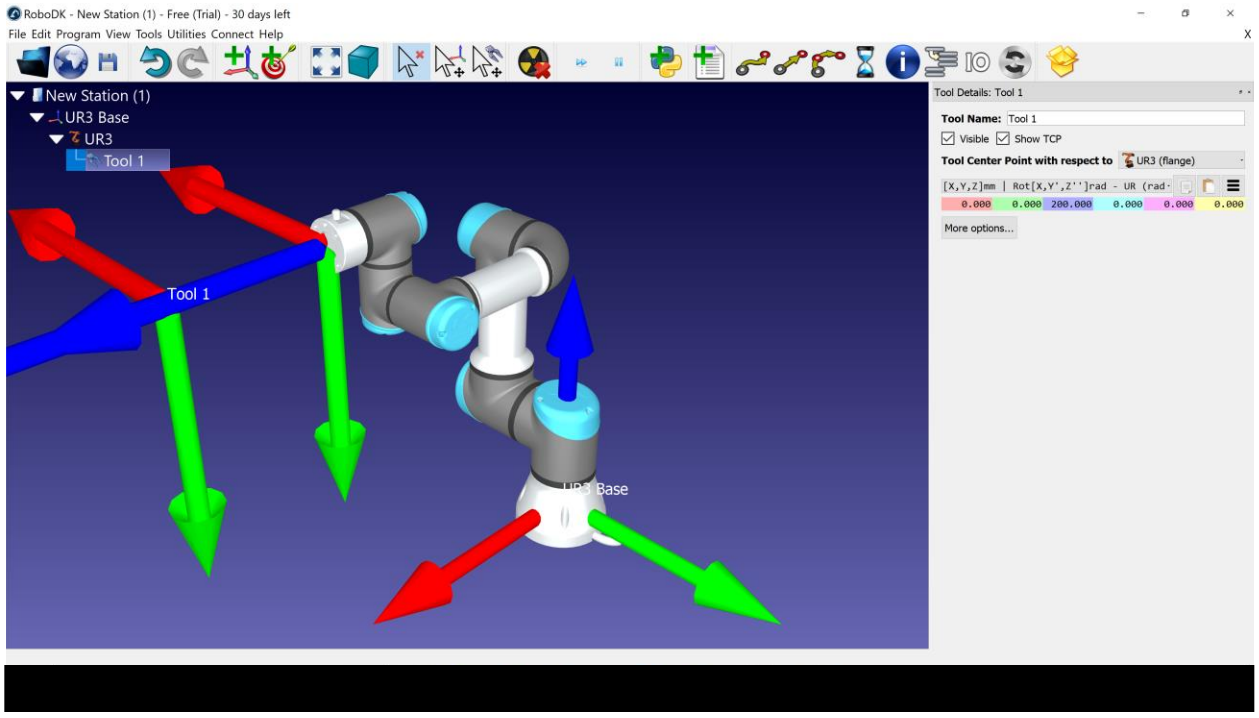


Figure 6: TCP tool options


RoboDK - New Station (1) - Free (Trial) - 30 days left
File Edit Program View Tools Utilities Connect Help
New Station (1)
UR3 Base
UR3
Tool 1
Tool 1
UR3 Base
More options...

Tool Details: Tool 1

Tool Name: Tool 1

☑ Visible ☑ Show TCP

Tool Center Point with respect to UR3 (flange)

[X,Y,Z]mm | Rot[X,Y',Z'']rad - UR (rad

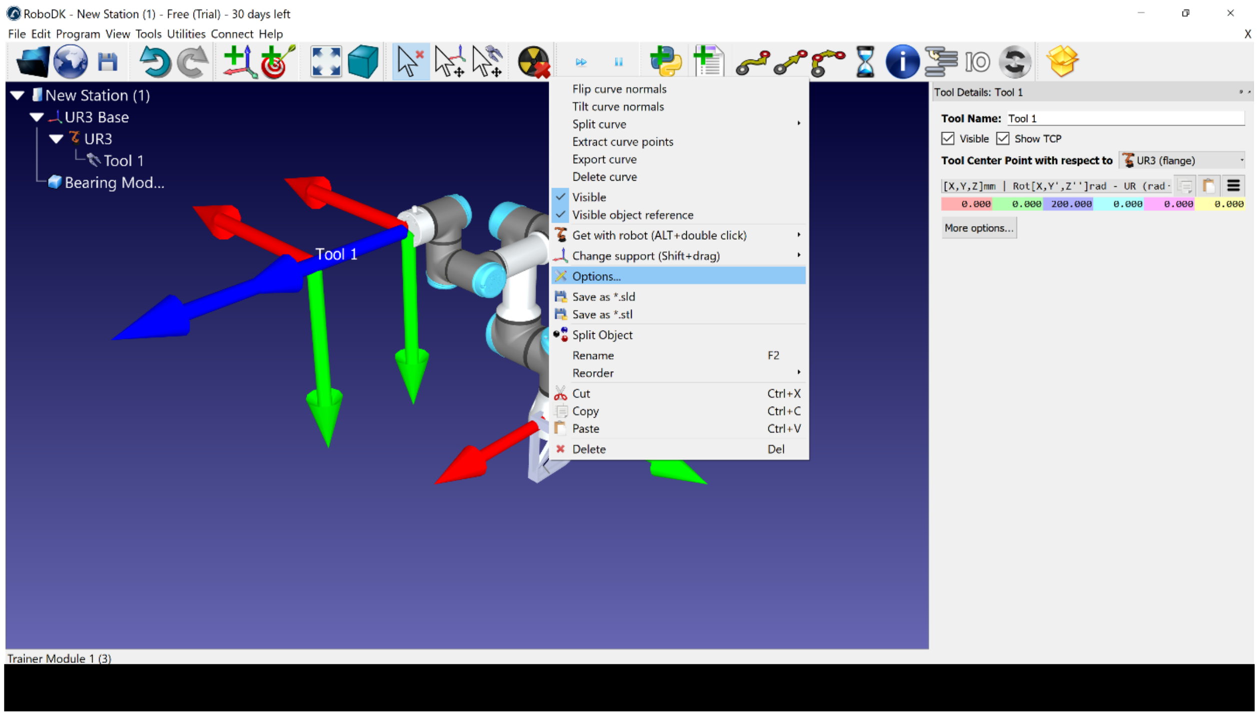
0.000	0.000	200.000	0.000	0.000	0.000

Figure 7: Setting TCP Parameters

4) File > Open > Locate "Bearing_Module_1.STEP" > Open

5) Right Click Bearing Module within Environment > Options

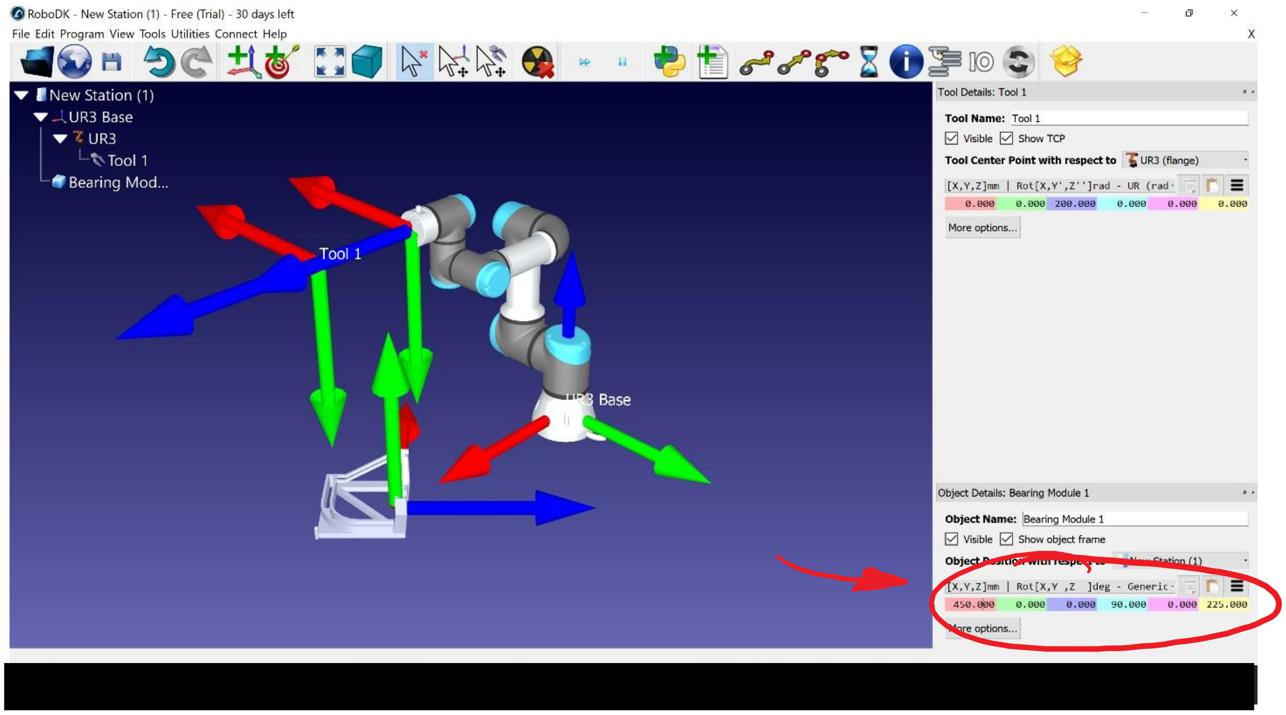
6) In Options Panel, adjust translation and rotation values to orient the module



Tool Name: Tool 1
☑ Visible ☑ Show TCP
Tool Center Point with respect to UR3 (flange)

[X,Y,Z]mm | Rot[X,Y',Z'']rad - UR (rad					
0.000	0.000	200.000	0.000	0.000	0.000

Figure 8: Inserted Bearing Module, Right Click for Options


RoboDK - New Station (1) - Free (Trial) - 30 days left
File Edit Program View Tools Utilities Connect Help
New Station (1)
UR3 Base
UR3
Tool 1
Bearing Mod...
Tool 1
UR3 Base
Tool Details: Tool 1
Object Details: Bearing Module 1
More options...

Tool Name: Tool 1
☑ Visible ☑ Show TCP
Tool Center Point with respect to UR3 (flange)

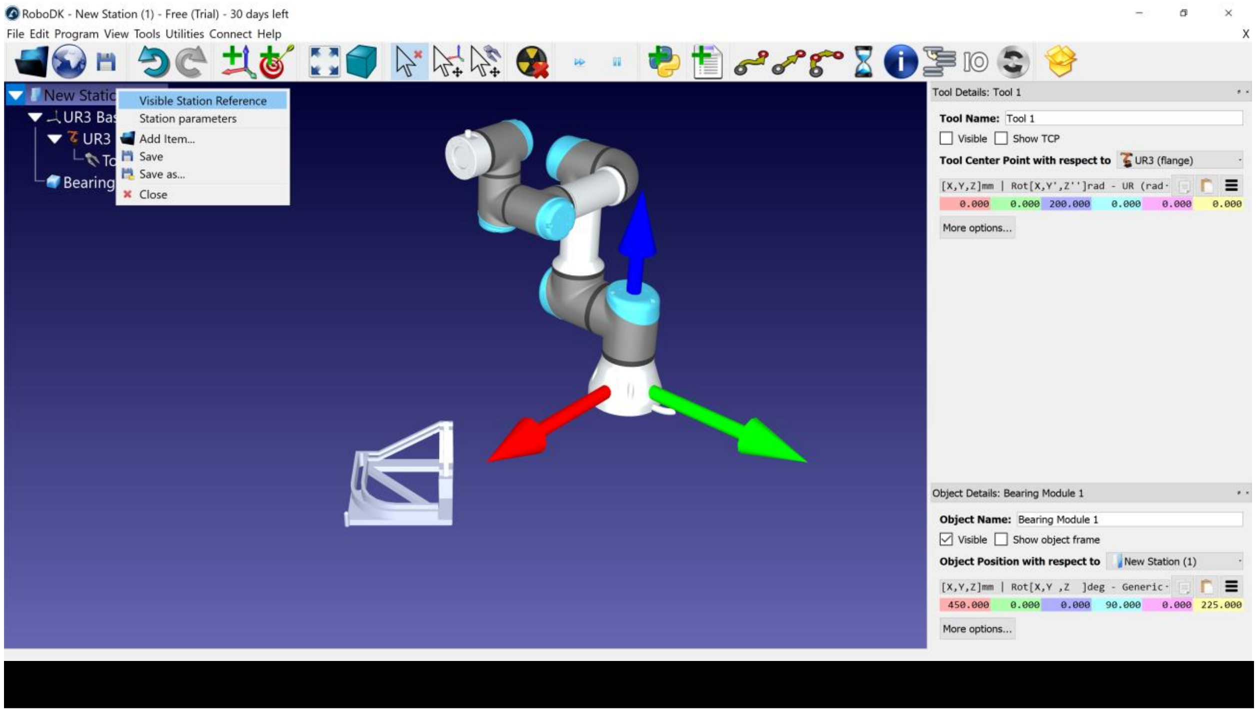
[X,Y,Z]mm | Rot[X,Y',Z'']rad - UR (rad					
0.000	0.000	200.000	0.000	0.000	0.000

Object Name: Bearing Module 1
☑ Visible ☑ Show object frame
Object Position with respect to New Station (1)

[X,Y,Z]mm | Rot[X,Y ,Z]deg - Generic					
450.000	0.000	0.000	90.000	0.000	225.000

Figure 9: Adjust Bearing Module Position to Approximate Location by Adjusting Values

Note: At this point, it may be helpful to right click each item in the tree and uncheck the frame visibility. This will declutter the environment space


RoboDK - New Station (1) - Free (Trial) - 30 days left
File Edit Program View Tools Utilities Connect Help
New Station
UR3 Base
UR3
Tool 1
Bearing Module 1
Visible Station Reference
Station parameters
Add Item...
Save
Save as...
Close

Tool Details: Tool 1

Tool Name: Tool 1

☐ Visible ☐ Show TCP

Tool Center Point with respect to UR3 (flange)

[X,Y,Z]mm | Rot[X,Y',Z'']rad - UR (rad)

0.000 0.000 200.000 0.000 0.000 0.000

More options...

Object Details: Bearing Module 1

Object Name: Bearing Module 1

☑ Visible ☐ Show object frame

Object Position with respect to New Station (1)

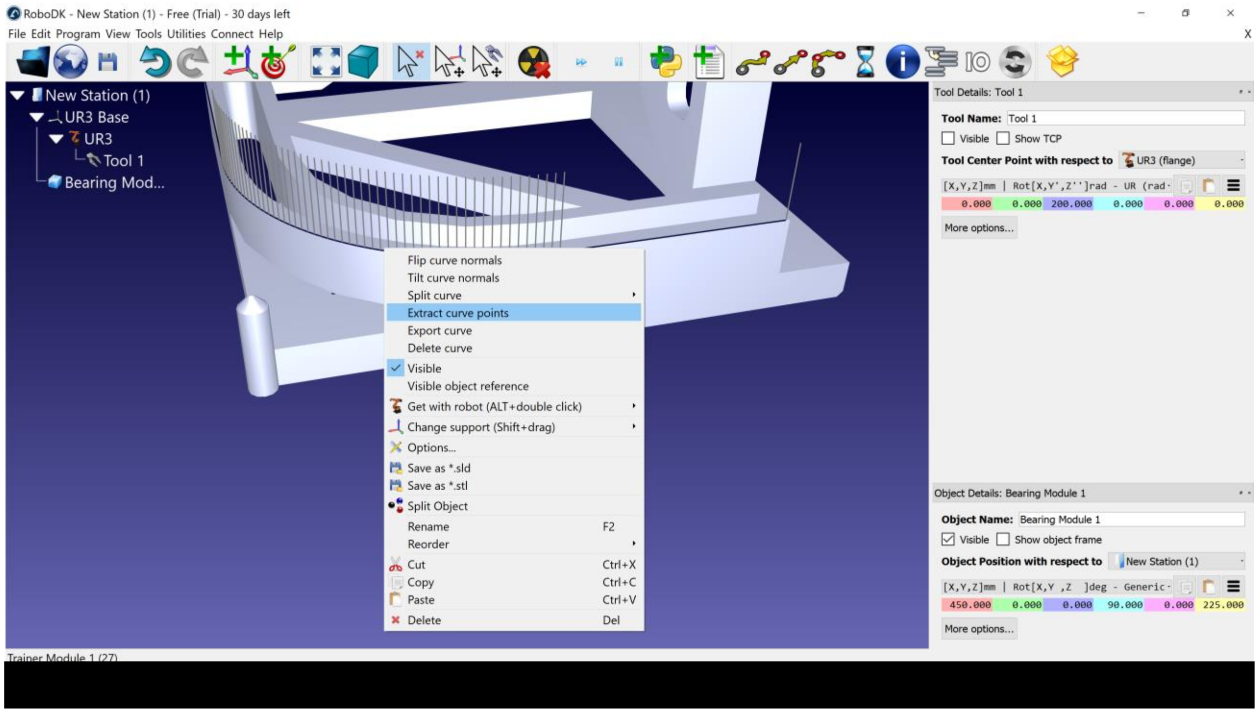
[X,Y,Z]mm | Rot[X,Y,Z]deg - Generic

450.000 0.000 0.000 90.000 0.000 225.000

More options...

Figure 9: Reference Frames Removed

7) Select a vertex on the Module with a right click > Extract Curve Points



Tool Details: Tool 1

Tool Name: Tool 1

☐ Visible ☐ Show TCP

Tool Center Point with respect to UR3 (flange)

[X,Y,Z]mm | Rot[X,Y',Z'']rad - UR (rad)

0.000 0.000 200.000 0.000 0.000 0.000

More options...

Object Details: Bearing Module 1

Object Name: Bearing Module 1

☑ Visible ☐ Show object frame

Object Position with respect to New Station (1)

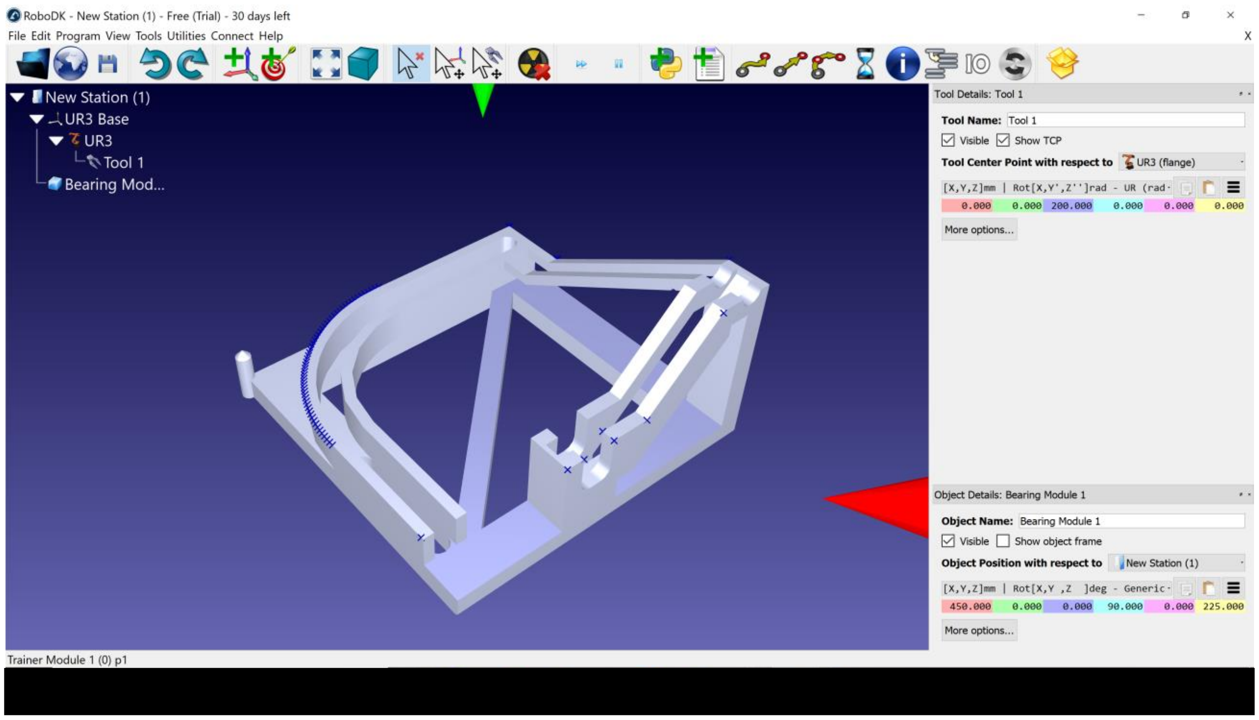
[X,Y,Z]mm | Rot[X,Y,Z]deg - Generic

450.000 0.000 0.000 90.000 0.000 225.000

More options...

Figure 10: Extracting Curve Points

8) Repeat for all vertices and edges to be used in program


RoboDK - New Station (1) - Free (Trial) - 30 days left
File Edit Program View Tools Utilities Connect Help
New Station (1)
UR3 Base
UR3
Tool 1
Bearing Mod...
Trainer Module 1 (0) p1

Tool Details: Tool 1

Tool Name: Tool 1

☑ Visible ☑ Show TCP

Tool Center Point with respect to UR3 (flange)

[X,Y,Z]mm		Rot[X,Y',Z'']rad - UR (rad)			
0.000	0.000	200.000	0.000	0.000	0.000

More options...

Object Details: Bearing Module 1

Object Name: Bearing Module 1

☑ Visible ☐ Show object frame

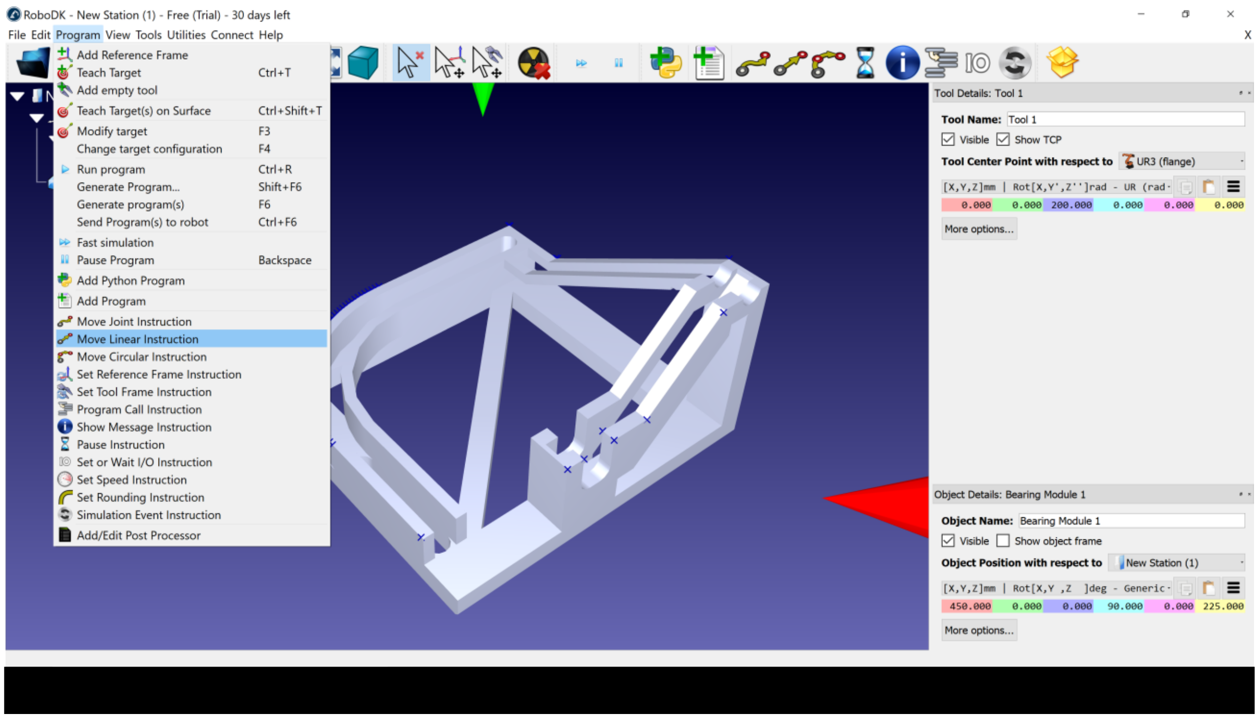
Object Position with respect to New Station (1)

[X,Y,Z]mm		Rot[X,Y ,Z]deg - Generic			
450.000	0.000	0.000	90.000	0.000	225.000

More options...

Figure 11: All Necessary Curve Points Extracted

9) Program > Program Linear Instruction


RoboDK - New Station (1) - Free (Trial) - 30 days left
File Edit Program View Tools Utilities Connect Help
Add Reference Frame
Teach Target Ctrl+T
Add empty tool
Teach Target(s) on Surface Ctrl+Shift+T
Modify target F3
Change target configuration F4
Run program Ctrl+R
Generate Program... Shift+F6
Generate program(s) F6
Send Program(s) to robot Ctrl+F6
Fast simulation
Pause Program Backspace
Add Python Program
Add Program
Move Joint Instruction
Move Linear Instruction
Move Circular Instruction
Set Reference Frame Instruction
Set Tool Frame Instruction
Program Call Instruction
Show Message Instruction
Pause Instruction
Set or Wait I/O Instruction
Set Speed Instruction
Set Rounding Instruction
Simulation Event Instruction
Add/Edit Post Processor

Tool Details: Tool 1

Tool Name: Tool 1

☑ Visible ☑ Show TCP

Tool Center Point with respect to UR3 (flange)

[X,Y,Z]mm		Rot[X,Y',Z'']rad - UR (rad)			
0.000	0.000	200.000	0.000	0.000	0.000

More options...

Object Details: Bearing Module 1

Object Name: Bearing Module 1

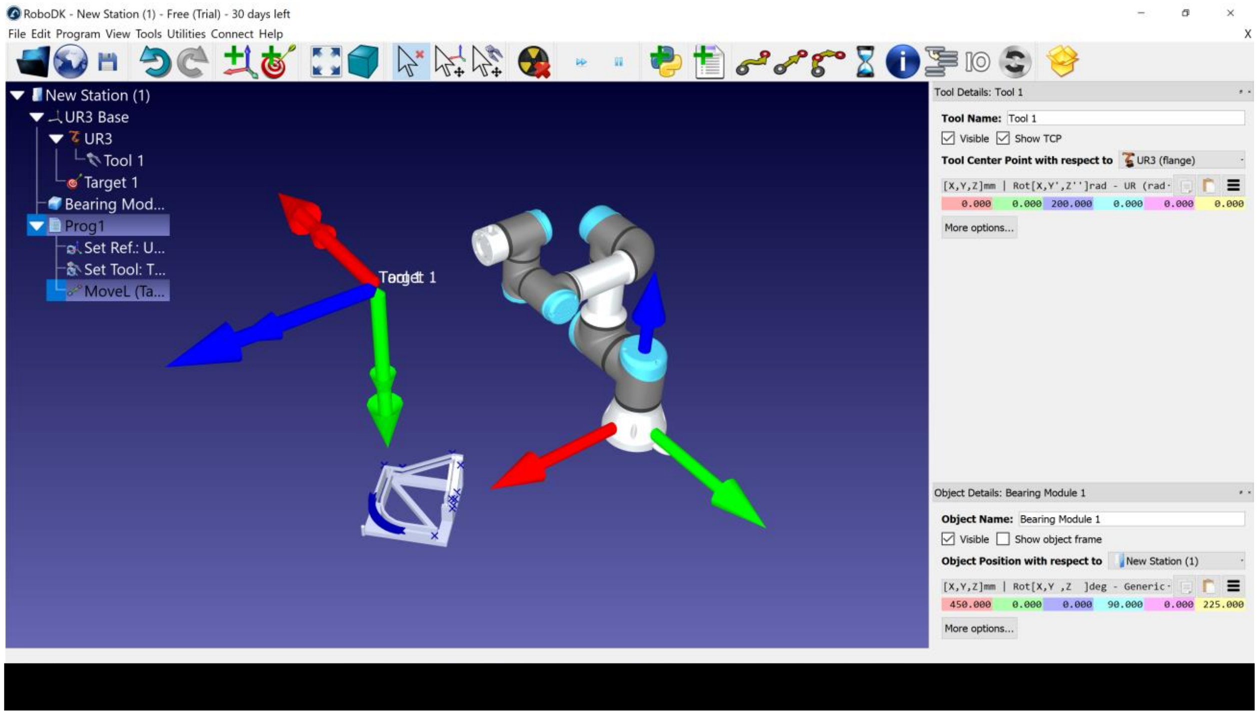
☑ Visible ☐ Show object frame

Object Position with respect to New Station (1)

[X,Y,Z]mm		Rot[X,Y ,Z]deg - Generic			
450.000	0.000	0.000	90.000	0.000	225.000

More options...

Figure 12: Move Linear Instruction



Tool Details: Tool 1

Tool Name: Tool 1

☑ Visible ☑ Show TCP

Tool Center Point with respect to UR3 (flange)

[X,Y,Z]mm | Rot[X,Y',Z'']rad - UR (rad

0.000	0.000	200.000	0.000	0.000	0.000

More options...

Object Details: Bearing Module 1

Object Name: Bearing Module 1

☑ Visible ☐ Show object frame

Object Position with respect to New Station (1)

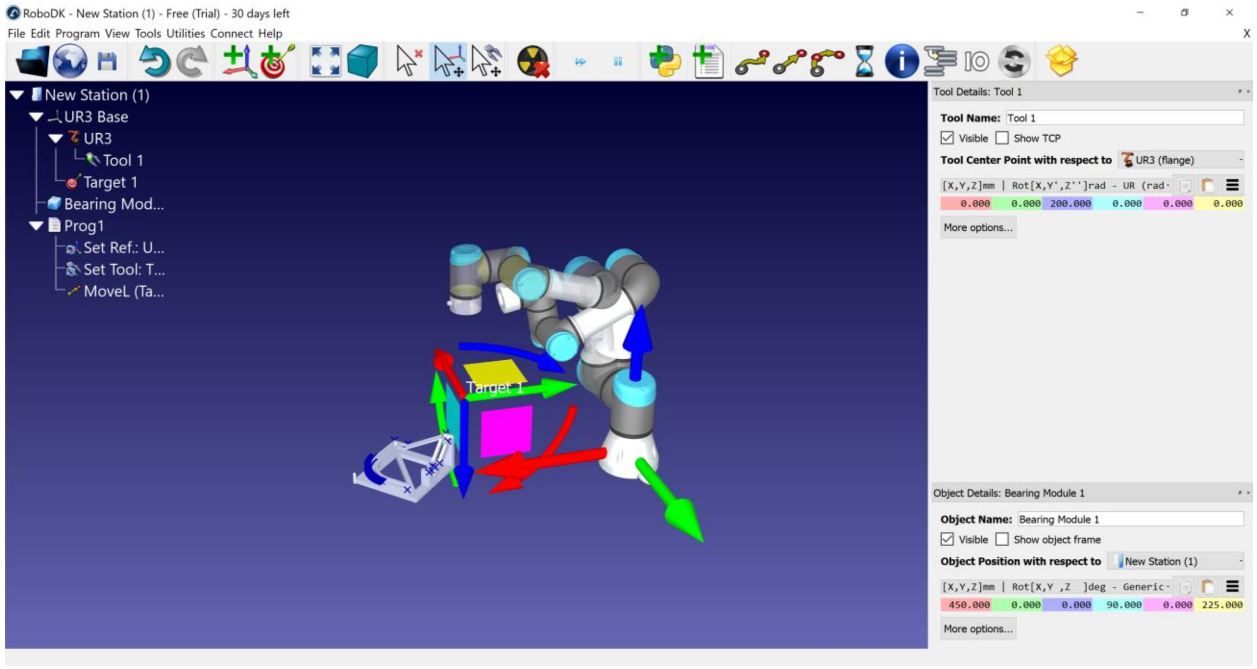
[X,Y,Z]mm | Rot[X,Y ,Z]deg - Generic

450.000	0.000	0.000	90.000	0.000	225.000

More options...

Figure 13: Linear Move Added to Tree

10) Hold ALT > click and drag on arrows to orient to a initial starting waypoint



Tool Details: Tool 1

Tool Name: Tool 1

☑ Visible ☐ Show TCP

Tool Center Point with respect to UR3 (flange)

[X,Y,Z]mm | Rot[X,Y',Z'']rad - UR (rad

0.000	0.000	200.000	0.000	0.000	0.000

More options...

Object Details: Bearing Module 1

Object Name: Bearing Module 1

☑ Visible ☐ Show object frame

Object Position with respect to New Station (1)

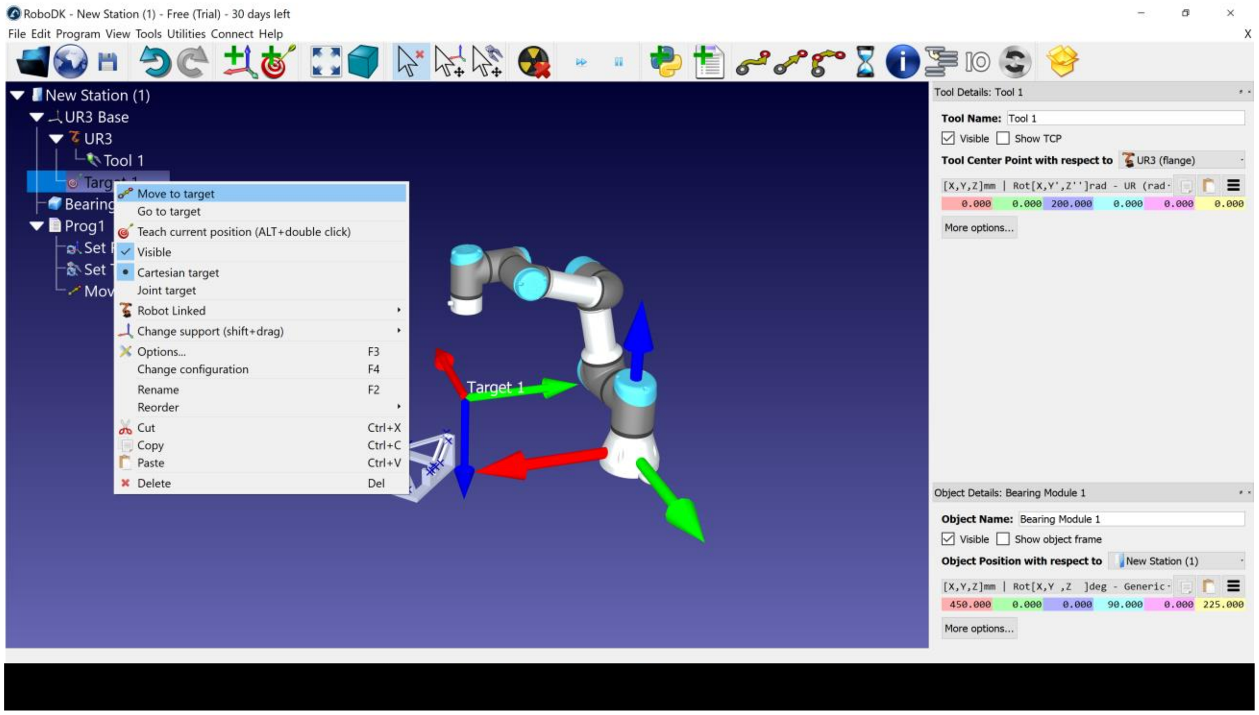
[X,Y,Z]mm | Rot[X,Y ,Z]deg - Generic

450.000	0.000	0.000	90.000	0.000	225.000

More options...

Figure 14: Creating and Re-orienting a New Waypoint

11)Right click target 1 in tree > move to target > teach current position



Tool Details: Tool 1

Tool Name: Tool 1

☑ Visible ☐ Show TCP

Tool Center Point with respect to UR3 (flange)

[X,Y,Z]mm	Rot[X,Y',Z'']rad - UR (rad)				
0.000	0.000	200.000	0.000	0.000	0.000

Object Details: Bearing Module 1

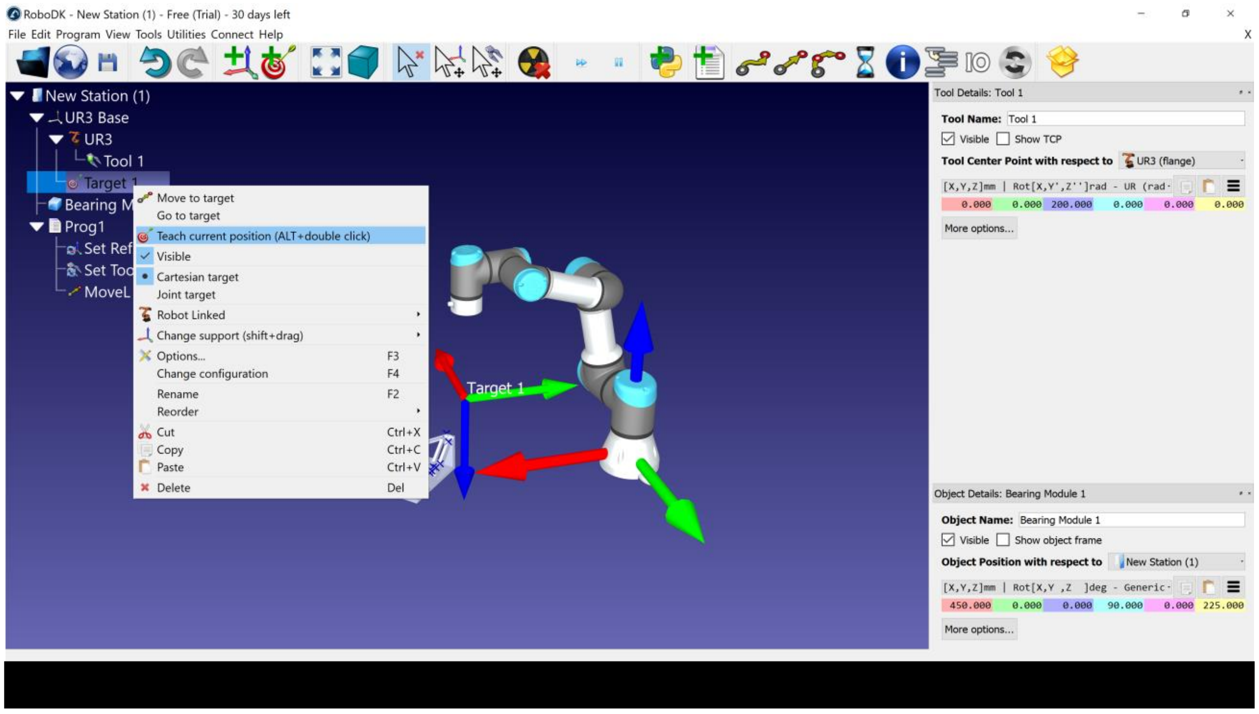
Object Name: Bearing Module 1

☑ Visible ☐ Show object frame

Object Position with respect to New Station (1)

[X,Y,Z]mm	Rot[X,Y,Z]deg - Generic				
450.000	0.000	0.000	90.000	0.000	225.000

Figure 15: Move to Target



Tool Details: Tool 1

Tool Name: Tool 1

☑ Visible ☐ Show TCP

Tool Center Point with respect to UR3 (flange)

[X,Y,Z]mm	Rot[X,Y',Z'']rad - UR (rad)				
0.000	0.000	200.000	0.000	0.000	0.000

Object Details: Bearing Module 1

Object Name: Bearing Module 1

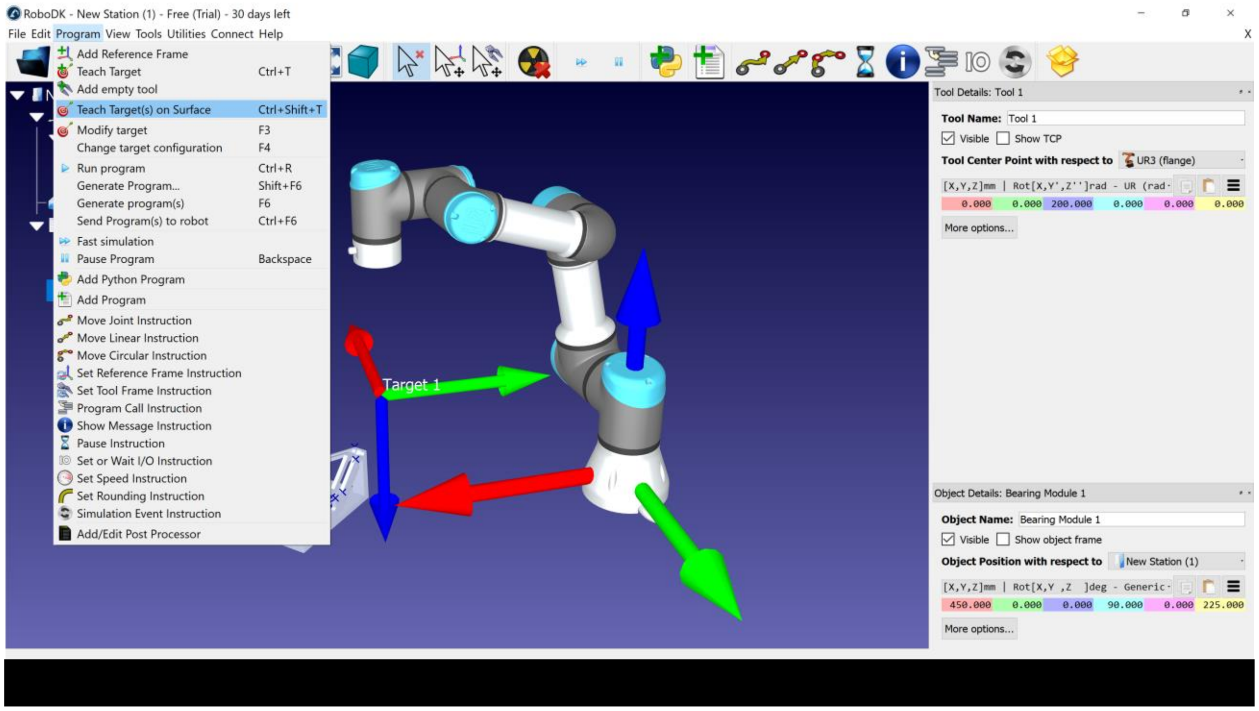
☑ Visible ☐ Show object frame

Object Position with respect to New Station (1)

[X,Y,Z]mm	Rot[X,Y,Z]deg - Generic				
450.000	0.000	0.000	90.000	0.000	225.000

Figure 16: Teach Current Position

12) Program > teach target on surface select point to be taught as a target



Tool Details: Tool 1

Tool Name: Tool 1

Visible [✓] Show TCP [ ]

Tool Center Point with respect to UR3 (flange)

[X,Y,Z]mm | Rot[X,Y',Z'']rad - UR (rad)

0.000 0.000 200.000 0.000 0.000 0.000

Object Details: Bearing Module 1

Object Name: Bearing Module 1

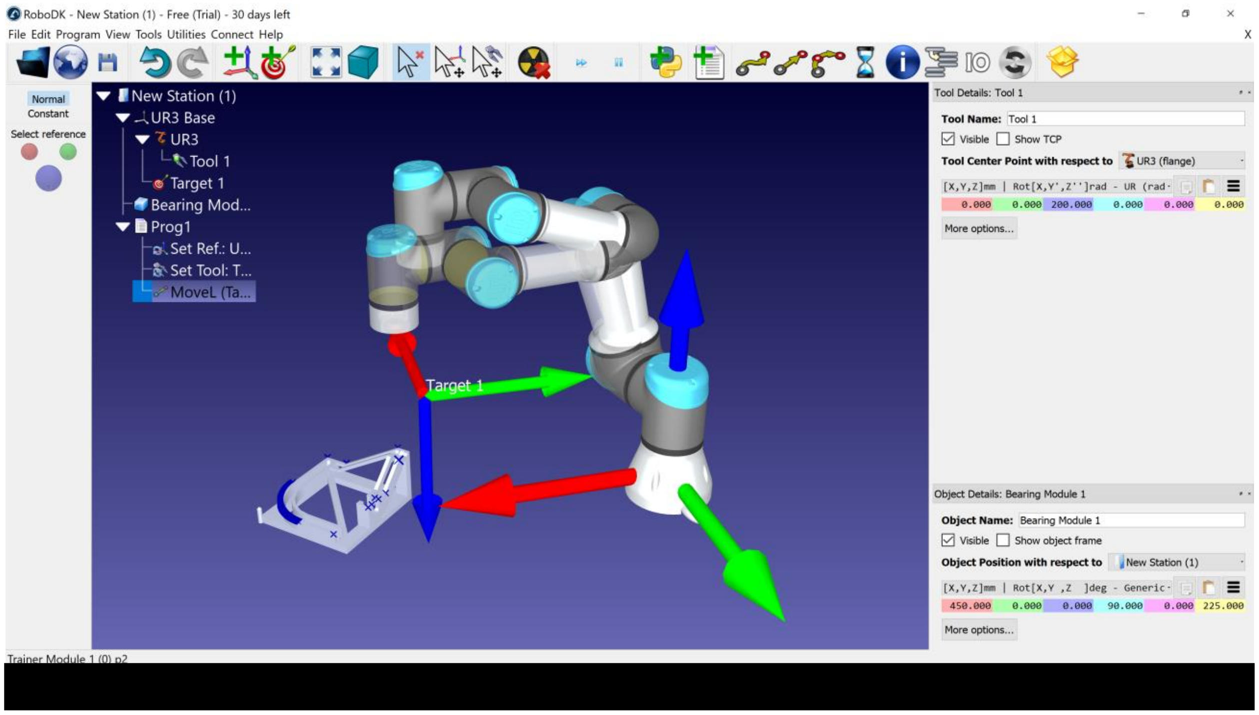
Visible [✓] Show object frame [ ]

Object Position with respect to New Station (1)

[X,Y,Z]mm | Rot[X,Y ,Z]deg - Generic

450.000 0.000 0.000 90.000 0.000 225.000

Figure 17: Teach Target(s) on Surface



Tool Details: Tool 1

Tool Name: Tool 1

Visible [✓] Show TCP [ ]

Tool Center Point with respect to UR3 (flange)

[X,Y,Z]mm | Rot[X,Y',Z'']rad - UR (rad)

0.000 0.000 200.000 0.000 0.000 0.000

Object Details: Bearing Module 1

Object Name: Bearing Module 1

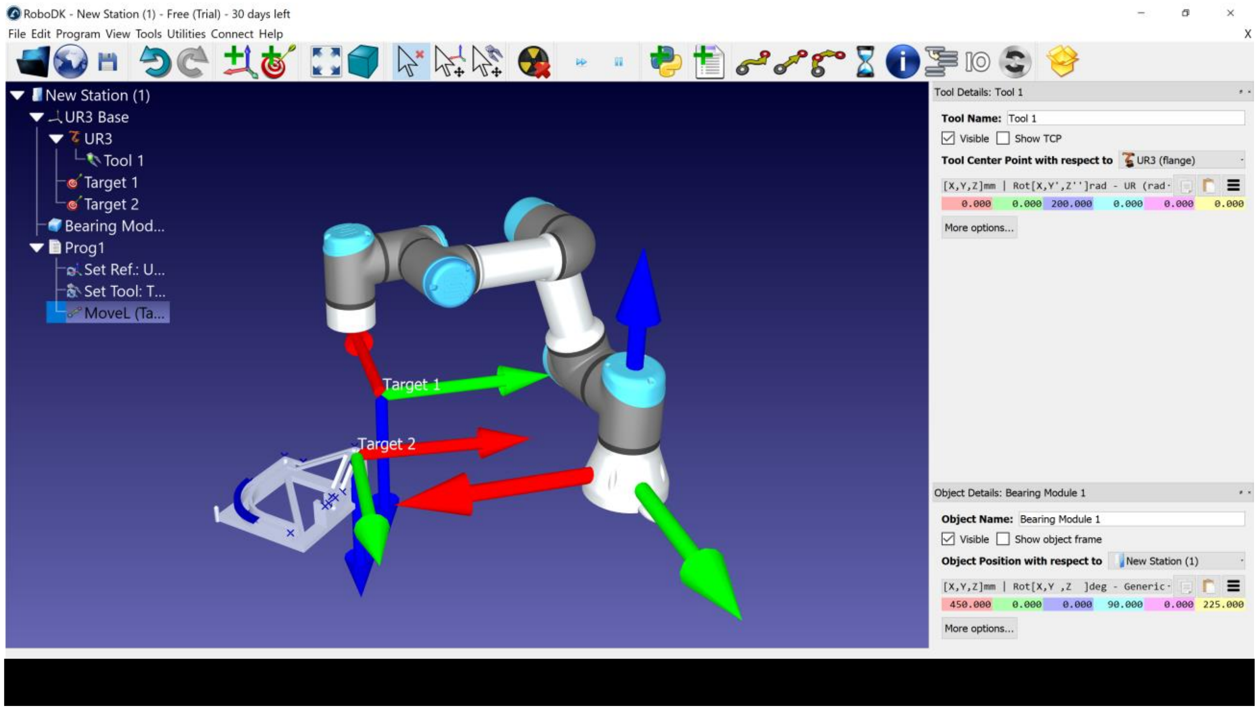
Visible [✓] Show object frame [ ]

Object Position with respect to New Station (1)

[X,Y,Z]mm | Rot[X,Y ,Z]deg - Generic

450.000 0.000 0.000 90.000 0.000 225.000

Figure 18: Selecting a Target on a Surface



Tool Details: Tool 1

Tool Name: Tool 1
☑ Visible ☐ Show TCP
Tool Center Point with respect to: UR3 (flange)

[X,Y,Z]mm | Rot[X,Y',Z'']rad - UR (rad					
0.000	0.000	200.000	0.000	0.000	0.000

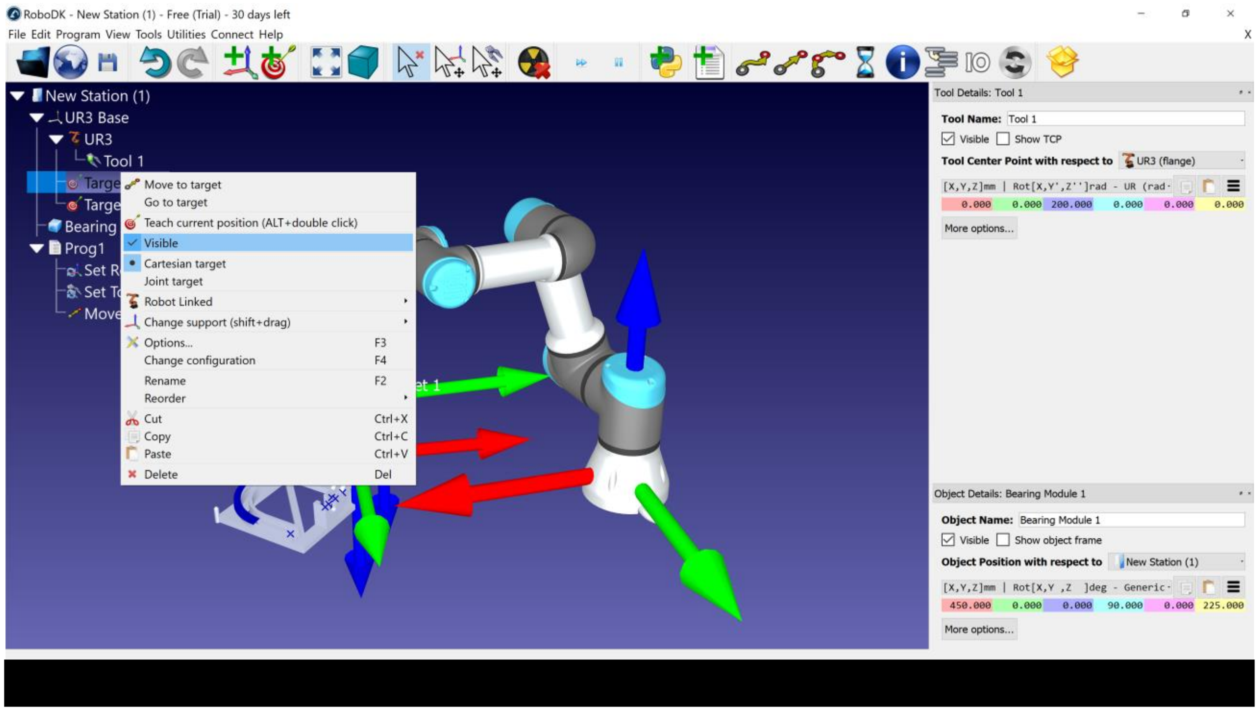
Object Details: Bearing Module 1

Object Name: Bearing Module 1
☑ Visible ☐ Show object frame
Object Position with respect to: New Station (1)

[X,Y,Z]mm | Rot[X,Y ,Z]deg - Generic					
450.000	0.000	0.000	90.000	0.000	225.000

Figure 19: Successfully Creating a New Target

Note: It may be beneficial again to right click targets as you go and unclick "visible" to declutter the environment



Tool Details: Tool 1

Tool Name: Tool 1
☑ Visible ☐ Show TCP
Tool Center Point with respect to: UR3 (flange)

[X,Y,Z]mm | Rot[X,Y',Z'']rad - UR (rad					
0.000	0.000	200.000	0.000	0.000	0.000

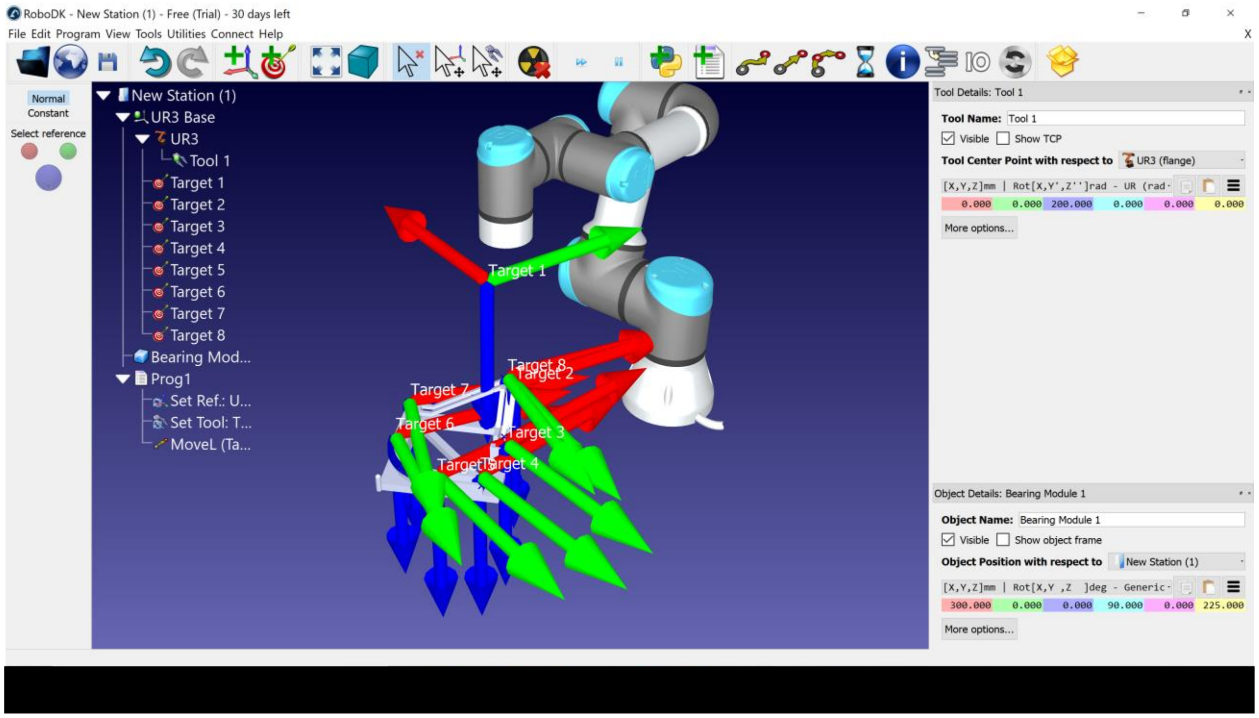
Object Details: Bearing Module 1

Object Name: Bearing Module 1
☑ Visible ☐ Show object frame
Object Position with respect to: New Station (1)

[X,Y,Z]mm | Rot[X,Y ,Z]deg - Generic					
450.000	0.000	0.000	90.000	0.000	225.000

Figure 20: Uncheck Visible to Hide Target Frames

13)Repeat step 12 for all points to be used in program



Tool Details: Tool 1

Tool Name: Tool 1

☑ Visible ☐ Show TCP

Tool Center Point with respect to UR3 (flange)

[X,Y,Z]mm | Rot[X,Y',Z'']rad - UR (rad

0.000	0.000	200.000	0.000	0.000	0.000

Object Details: Bearing Module 1

Object Name: Bearing Module 1

☑ Visible ☐ Show object frame

Object Position with respect to New Station (1)

[X,Y,Z]mm | Rot[X,Y ,Z]deg - Generic

300.000	0.000	0.000	90.000	0.000	225.000

Figure 21: Creating all necessary targets

Note: The colored wheels at the top left of the screen can be used to adjust target rotation.

Note: The software will not allow selection of waypoints that are out of the arm's work envelope. The part will need to be moved closer to the arm and waypoints adjusted accordingly.

14)Right click Move L in tree > Add Move Instruction > Select choice for next move

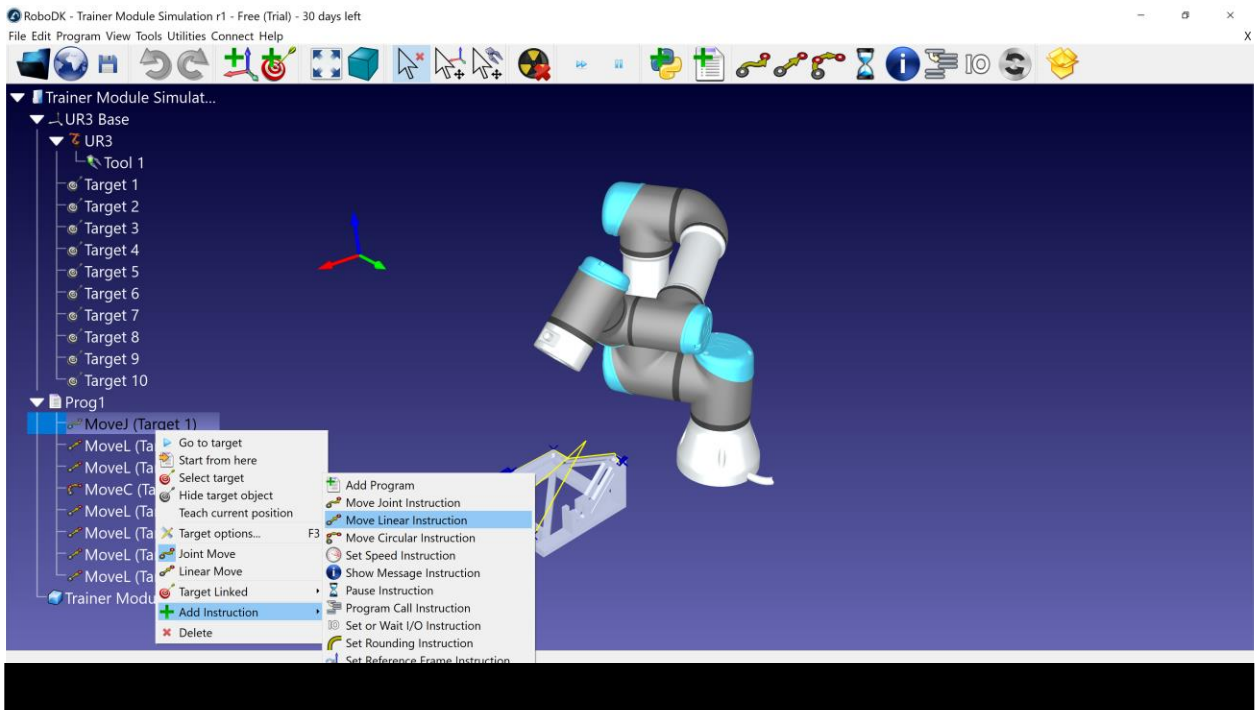


Figure 22: Adding a Linear Move

15) In the program tree > right click the newly added move > Target Linked > choose appropriate target

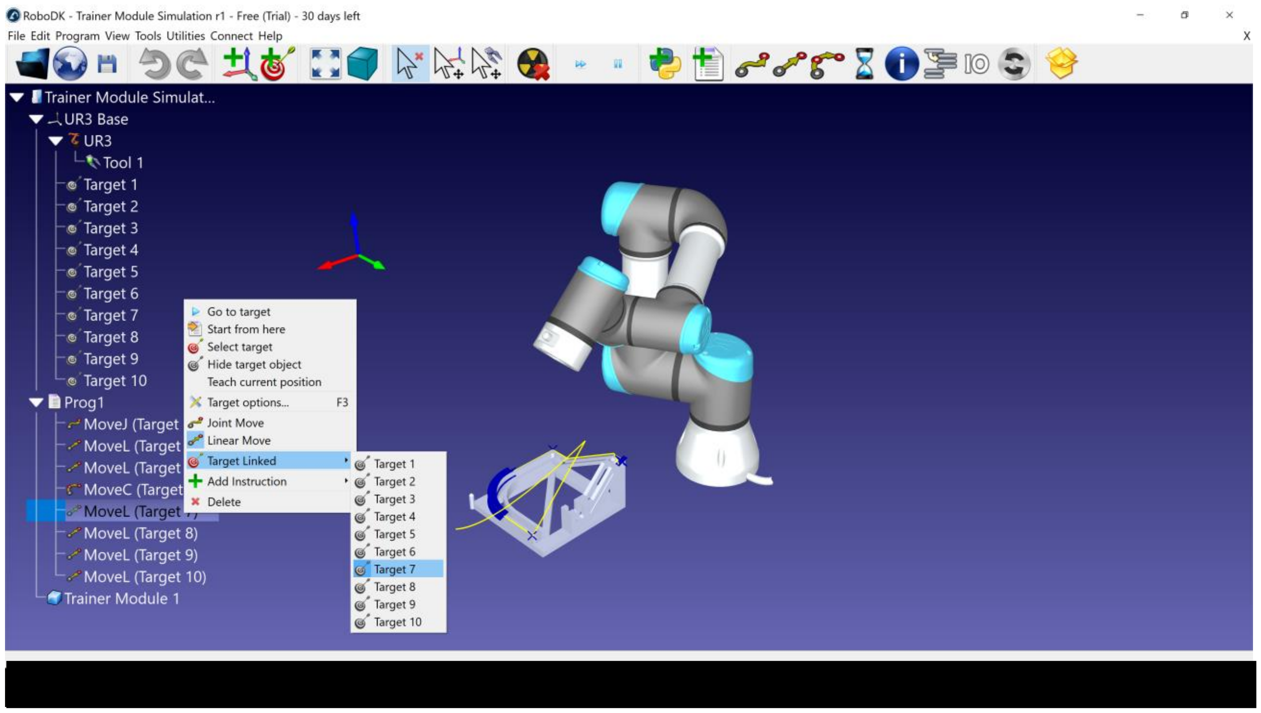
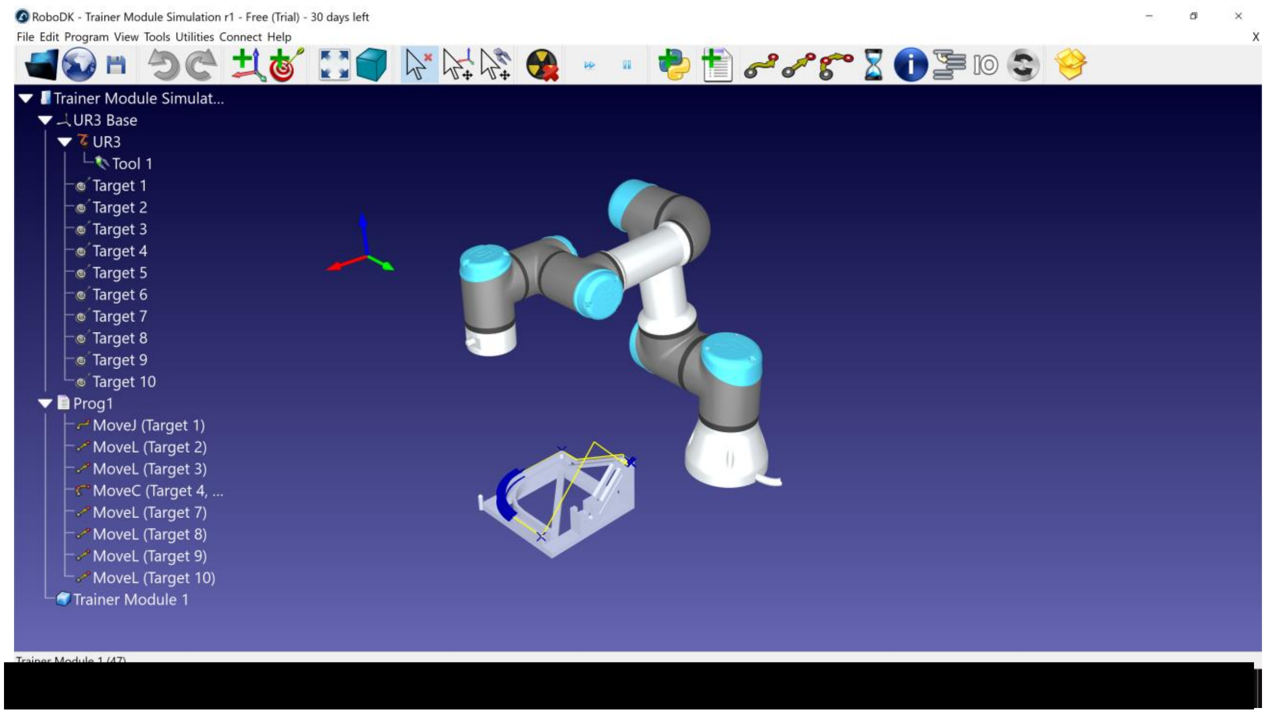


Figure 23: Associating Moves with Appropriate Target

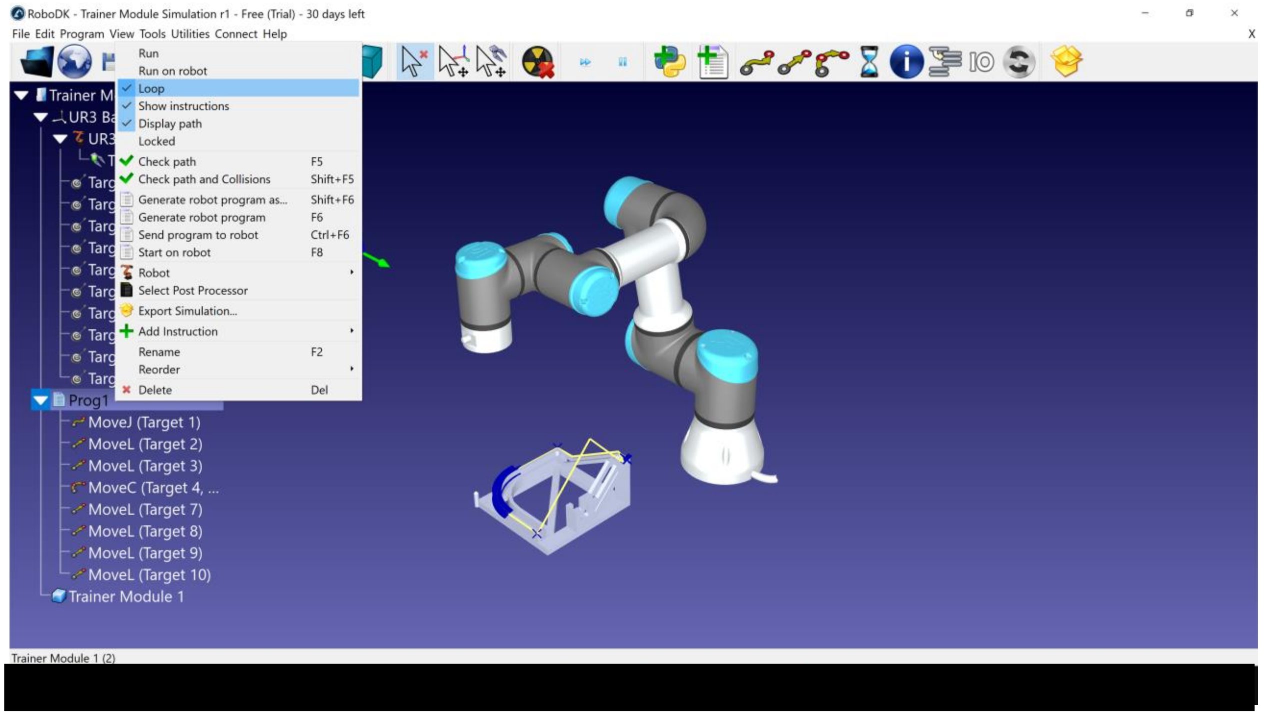
16) Repeat process for each waypoint in the program tree



[redacted]

Figure 24: Completed Robot Program

17) To Run program: Right click program > loop



[redacted]

Figure 25: Looping the Robot Program

Right click program > run

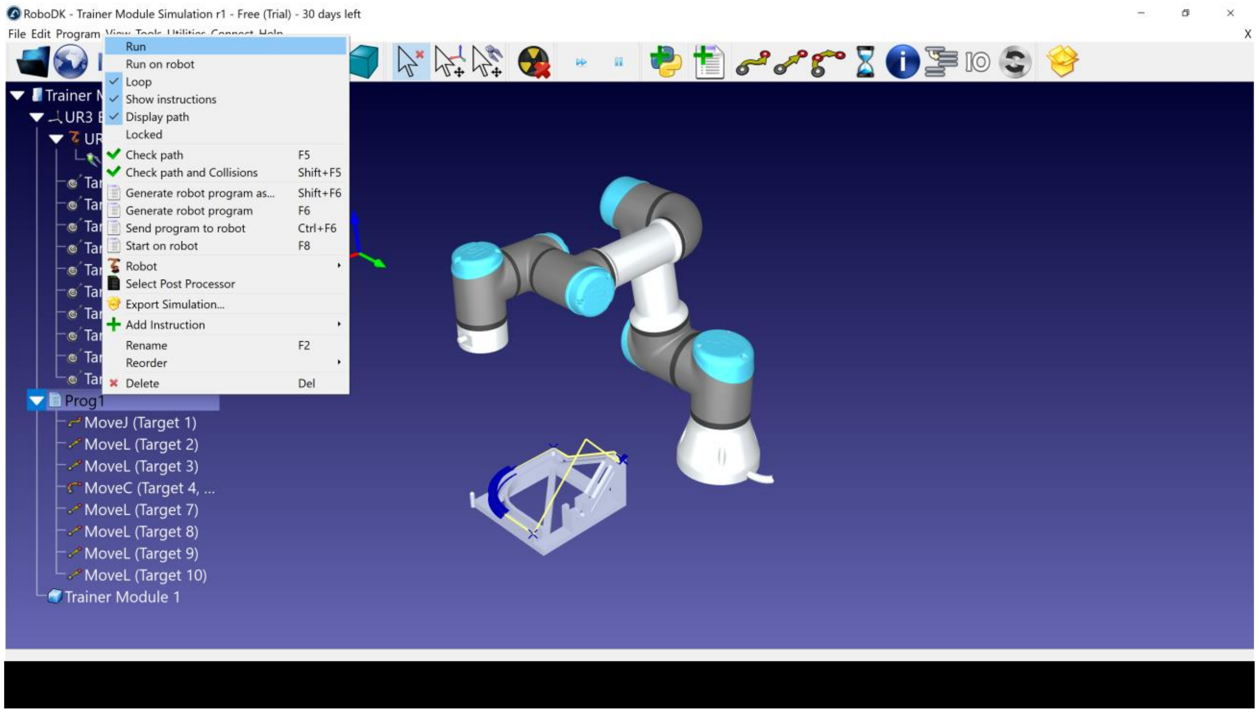
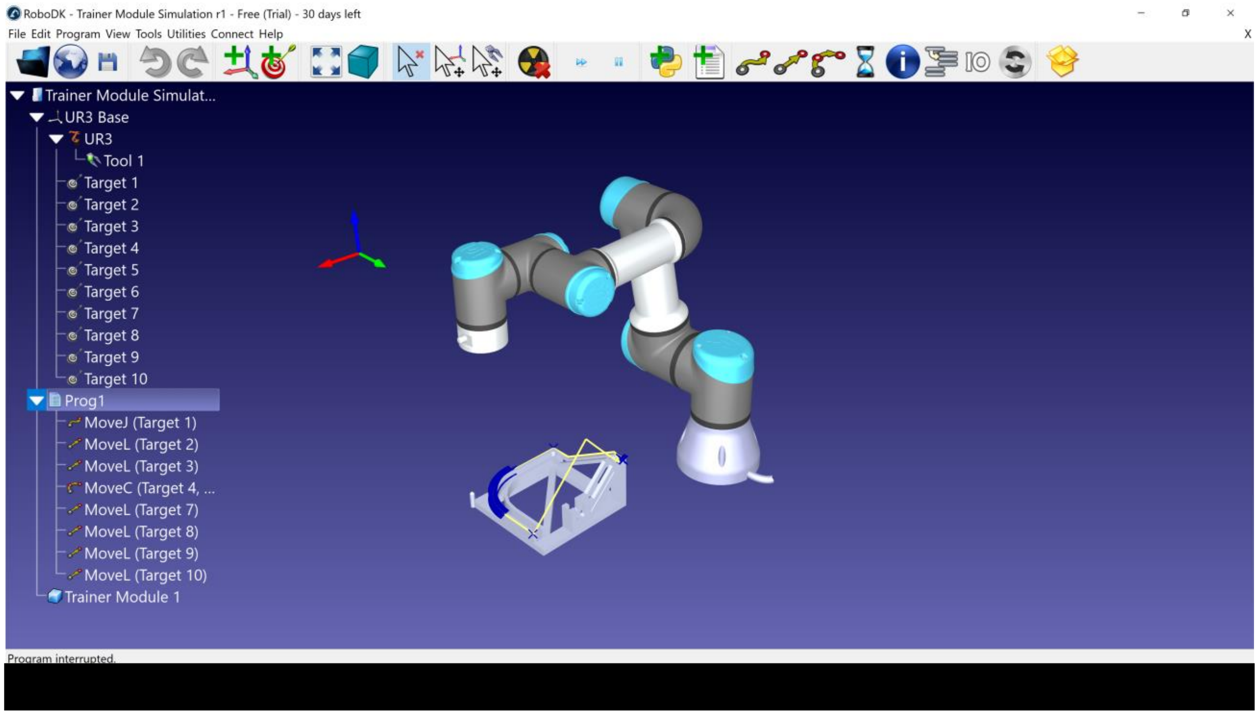


Figure 26: Running the Robot Program

Note: Program will stop at before any points outside of work envelope



Program interrupted.

Figure 27: Successful Program will Display Continuous TCP Trajectory

21) To Output Program:

a. Right click program > Generate robot program… > save to destination as a .SCRIPT file

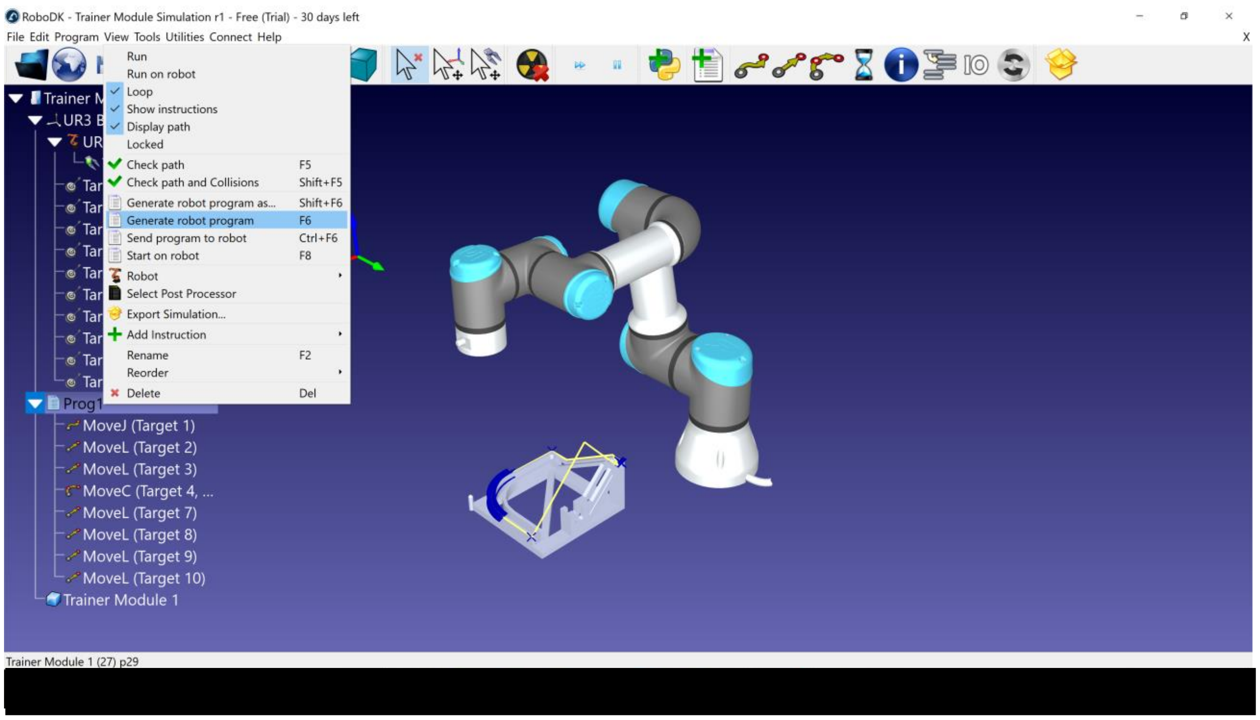


Figure 28: Generating Robot Program

Theory:

The RoboDK software allows programming and simulation of the UR3 robotic arm as well as many other industrial robots. The development process is similar to that of programming the arm using the teach pendant. Moves between waypoints are simulated by a UR3 avatar, and additional visual aids depict waypoint positions, orientations, and TCP trajectories. Software simulation permits non-destructive testing, high precision TCP placements, as well as development while away from the location of the physical arm. Once a process has been developed, the software outputs a .SCRIPT file which is native to the UR3 arm. This interpreted by the software onboard the UR3 controller and the process is run by the arm.

Results:

1) Download RoboDK software
2) Simulate process for bearing module on horizontal plane
3) Run process on UR3
4) Rotate the Bearing Module in the simulation to be 45 deg. to the horizontal plane
5) Adjust process to work within UR3 envelope while avoiding singularities
6) Re-run final process on UR3

Sample Program

```
def Prog1():
  # Default parameters:
  global speed_ms = 0.3
  global speed_rads = 0.75
  global accel_mss = 3
  global accel_radss = 1.2
  global blend_radius_m = 0.001

  # Add subprograms here
  # Start of main program
  # Program generated by RoboDK v3.4.3 for UR3 on 29/03/2018 01:49:28
  # Using nominal kinematics.
  movej([0.365402, -1.353555, -1.433878, -1.924454, 1.570604,
0.365402],accel_radss,speed_rads,0,blend_radius_m)
  movel([0.303292, -1.804946, -1.379636, -1.527294, 1.570636, 0.303292],accel_mss,speed_ms,0,blend_radius_m)
  movel([0.139436, -1.826125, -1.353346, -1.532387, 1.570722, 0.139436],accel_mss,speed_ms,0,blend_radius_m)
  movec([0.016003, -1.793668, -1.393952, -1.524232, 1.570788, 0.016003],[-0.061949, -1.635748, -1.580284, -
1.495821, 1.570830, -0.061949],accel_mss,speed_ms,blend_radius_m)
  movel([-0.084430, -1.298673, -1.876853, -1.536337, 1.570842, -
0.084430],accel_mss,speed_ms,0,blend_radius_m)
  movel([0.064763, -1.229286, -1.914460, -1.568117, 1.570763, 0.064763],accel_mss,speed_ms,0,blend_radius_m)
  movel([0.519778, -1.163362, -1.690897, -1.857672, 1.570535, 0.519778],accel_mss,speed_ms,0,blend_radius_m)
  movel([0.585144, -1.245767, -1.628930, -1.837244, 1.570500, 0.585144],accel_mss,speed_ms,0,blend_radius_m)
  # End of main program
end

Prog1()
```