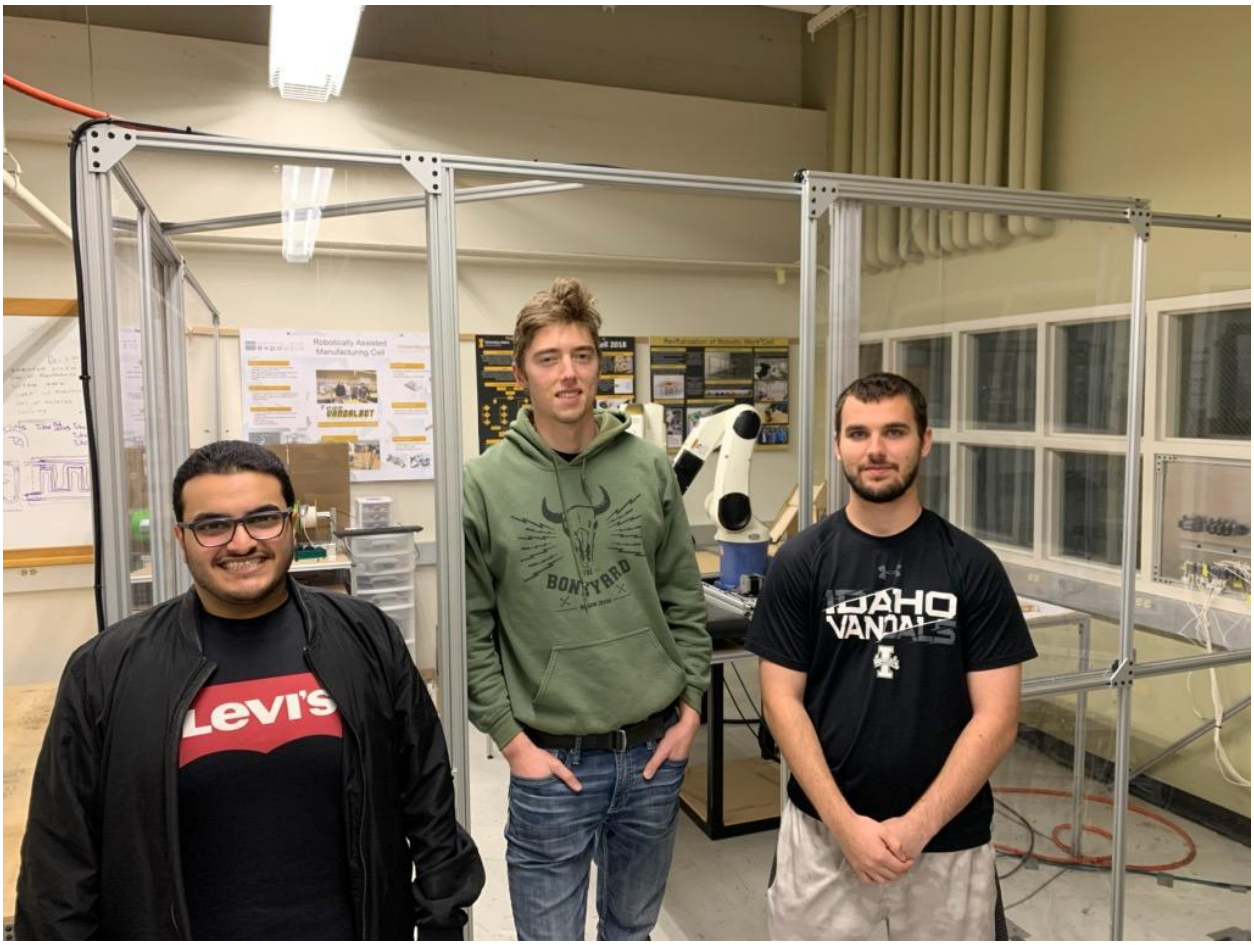


ROBO SHOP

SYNTHESIZED MANUFACTURING CELL

SALEM ALHAJRI| JOSHUA KOHL | BRAD HUMMEL

Contents

Problem Definition

Our team seeks to expand the assembly capabilities in the robotic manufacturing cell in the senior design suite. Our mission is to add fastener handling and installation to the existing part handling, part scanning, and part assembly features. The upgraded cell will have a wider range of simulated manufacturing operations that can be shown to prospective students and will have expanded infrastructure that can be used to enhance technical electives in industrial automation. Our process, if implemented into a real factory floor could replace 2-3 workers from mundane and routine work.

Objectives

- Pick up screw
- Fasten screw
- Identify parts
- Manipulate parts

Background

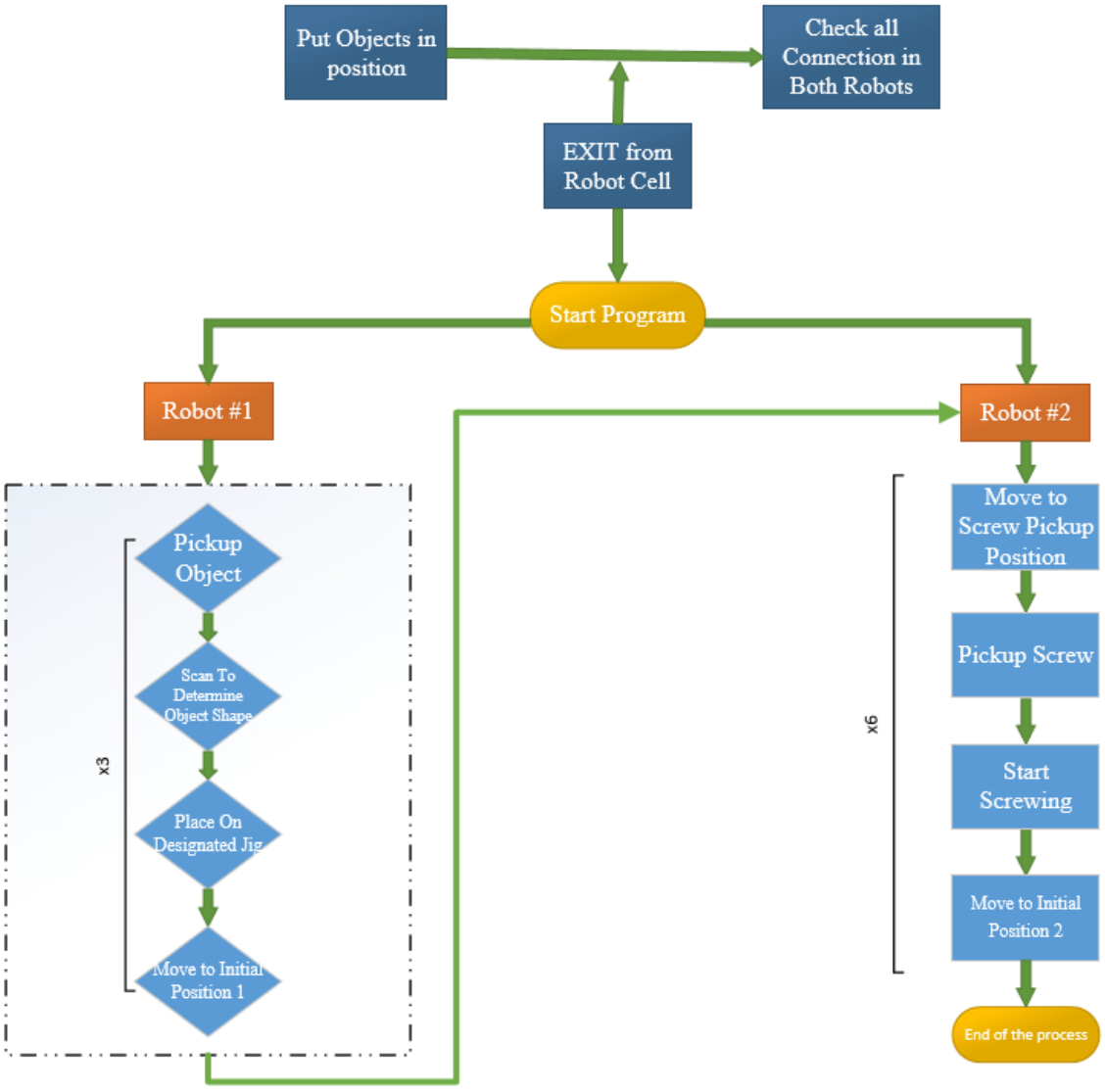
2013-2014 In Spring 2013 The Boeing Company donated DENSO robotic arms to The University of Idaho College of Engineering. That fall Mechanical and Electrical Engineering students, as a part of Team Roboshow, were tasked to learn basic programming of the robotic arms and create a work-cell for the arms. Their work-cell incorporated multiple safety features as well as a clear poly-carbonate enclosure for public demonstration. The team was able to successfully program the robot to use a dry-erase marker to create logos and patterns on a white board.

Summer-Fall 2016 During the Summer and Fall of 2016 Do All Robotics completely redesigned the robotic manufacturing cell. They made the cell larger which allowed for operators to have easier access to the robots for things like maintenance or changing end effectors. They also made the work cell fully modular allowing future teams to rearrange the cell for their specific needs. New pedestals for the robots were also designed because the originals were very unsteady. The team also performed stress analysis on the new pedestals to ensure they wouldn't break or fall over during operation. The team also programmed the robots to do tasks such as stack cups and write with a marker on a board. Finally the team also created a simple beginners manual that included all the basics needed to run the robot.

Summer-Fall 2017 The next year, the design team for the Robotic Arm Manufacturing Cell, Team CSRM, integrated a simple vision system involving a scanner, so that the arms could sort through two varied parts. These parts, a rectangle and a hexagon, were created by the team members, as well as the fixtures these parts were based in. Their process involved one of the robotic arms picking up pieces from a randomized mixture, then scanning the object to sort between the two. To pick the objects up, the robot used a pneumatic gripping mechanism.

Summer 2018-Fall 2018 The group that worked on this period were made of four Mechanical Engineering, and the name of the group is CyberCrew. There goals for their project is to integrate the two robots to work together in one assembly line, and create an assembly line that combine two female and male parts together. they made the two robotic arms work together in one assembly line, their assembly line were to make the first arm grab the part, and then scan that part. After the scanning, the arm will put female part first in a specific area, and then the same arm grabs the male part and put it above the female part. The Second arm will use pressure power to combine the two parts together.

PROCESS OVERVIEW



User setup

- Turn on both robots
- Plug in air compressor hose
- Make sure black scanner cable is plugged into the blue Net Gear box

- Place the parts onto one of the three towers. It is critical the part is placed in the correct orientation to be in line with the gripper lips of robot#1. For most consistent operation the square part should be in position 3.
- Load fasteners into screw holder.
- After the parts are placed and the fasteners are loaded into the screw holder, the worker should step away from the table, and start the programs simultaneously.
- The programs are titled RS2CODE3 in robot #1 and ROBOSHOPSCREW in robot #2. Robot #1 should be at 10% speed and robot #2 should be at 7% speed. The automated process takes slightly less than 12 minutes.
- The worker should have their hand on the emergency stop on whichever robot is working.

System Software

Start new code:

The process to start a new code for the user to start creating his own program:

- Connect Ethernet between laptop and robot that you want create a code for.
- Open WINCAPS III
- Choose Projects from top list in WINCAPS III.
- Choose create new project.
- Then choose get information from controller.
- Check the IP address.

  Note: each robot has IP address written on controller for example

  Robot 1 (192.168.0.210).

- Then choose do nothing.

  Note: this will make your code not connected to other codes in the robot.

- Write project name.

After all that user project will be created, then the user should create a new program to put his code on, and the steps for that are:

- Choose project from the top list in WINCAPS III.
- Choose add new program.
- Write the program name.

Send code to the controller:

When the user is finished creating the code, the user must send it to the controller so the robot can read it and start the process. The steps to do that are:

- Connect Ethernet cable between laptop and robot that you want the code sent to.
- Open WINCAPS III.
- Choose connect from the top list in WINCAPS III.
- Choose connect setting.
- Check if the laptop is connected to the right controller.
- Then press accept.
- Choose transfer data from the same connect list.
- Choose the file that you want.

  Note: Choose only the file that you want to transfer.

- Wait until it is done loading the whole code.

  Note: You can now disconnect the Ethernet cable from the laptop

Compile the code:

After transferring the code from the laptop, the user must compile the code so the robot can use it. This process is done in the controller that the code has been transferred to, and the steps for this process are:

- Make sure that the controller is in manual mode.
- Choose the programs.
- Choose the code that has been transferred.
- Choose edit and check the code.
- Choose save and then compile.

Coordinates:

The software that is being used for the two Robotic arms is WINCAPS III, and this software uses coding to make both robots work. The robots do not have any sensors to know directions or to know where the objects are so the user has to gather coordinates by using the robot controller. There are two specific coordinates that the user can use to move the robot, the first one is using joint (j) and the second one is by using position (p). The joint system of the robot is a set of six joints, and the coordinates are degrees of rotation for each joint. The position coordinate is X, Y, Z, RX, RY, and RZ where X, Y, Z are coordinates for the robots position and RX, RY, and RZ are the rotation of the joint 6. As an example for each one:

- j01 = (63.03, 1.53, 147.98, 3.37, -38.36, 231.67) (Joint)

- p01 = (58.00, 287.11, 305.14, -179.02, -10.75, 85.40) (Position)

Start and end:

The code usually starts with (TAKEARM), and (GIVEARM) but these are not necessary. These two codes are just saying that the robot will take the lead of the process, and the other one says that the robot will leave leading of the process to the second robot or the user. The command to end the code is (END). This ends the process of the robots. As an example:

- TAKEARM
- GIVEARM
- END

Movements:

When the user finishes gathering all the coordinates, the next step is making the robot move to the coordinate that the user wants. This is will not be hard if the user has the specific coordinates. There are two ways to make the robot move. The first is using the position system, this will make the robot choose the fastest way to get to the desired coordinates. The second way is by using the L shape movement system, and that will make the move in an L shape to the coordinates that the user wants. As an example:

- MOVE P, @E j00
- MOVE L, @E j01

The meaning of this code is that the robot will move (MOVE) by using the system that the user choose (P or L), to the (@E) coordinate the user wants (j00 or j01).

There are two other codes for movements, but these movements work differently from the other two. The first is (APPROACH), this command instead of going directly to the desired coordinate goes to a position above the coordinate, at a distance that the user desires, then slowly lowers to the actual coordinates. The second is (DEPART), this code works similarly to APPROACH but starts at set coordinates and goes up from those coordinates to a height set by the user. As an example:

- APPROACH P, j01, @E 55
- DEPART P, @E 55

The meaning of this code is that the robot will move close or away (APPROACH or DEPART) to the coordinate the user wants (j00 or j01) with this much of units (@E 55).

Speed and Delay:

Sometimes the user wants the robot to wait or delay for a few seconds to make the other robot do some subjects before the first robot do his next movement, and also it can help

the user to know if the robot is in the right position before he starts his next process. The code that will help in that is (DELAY), and the user can choose how many seconds he wants the robot to delay his movement. As an example:

- DELAY 1000

The meaning of this code is that the robot will delay his movements (DELAY) for this much of milliseconds (1000 = 1 sec).

The user can control the speed of the robot, and that will help him to make the most detailed process or hardest part of his process to be done with slowest speed to avoid trouble for both process and robot. The code for this process is (SPEED), and the user adjust the speed that he wants and the adjustment depends on the speed that the robot starts with from the controller. As an example:

- SPEED 75

The meaning of this code is that the robot will decrease from 100% to 75% or increase the speed from below 75% to 75%.

Close and open end effector:

The first have a pressure system that helps to close and open the end effector that user put in the robot 1, and the user can control this pressure system using the code. the code for that is (SET) to activate the pressure in the pipe and (RESET) to deactivate the pressure in the pipe, and there are two pipes that the robot 1 use for pressure system, the first one is (IO65) this is the back pipe and (IO64) this is the front pipe. As an example:

- for closing :

  SET IO64

  RESET IO65

- for opening :

  SET IO65

  RESET IO64

Connection between robots:

The robots and the scanner are connected to each other by ethernets, and the user can each of the robots and scanner get a signal that will help to know when to start working or to identify the object that the robot is delivering. The connection in the codes are a number of the line for the specific connection. The first connection is (#8) this line is connecting between robot 1 and scanner, and the second connection is (#10) this line is connecting between robot 1 and robot 2. As an example:

- connecting between robot 1 and scanner:

```
COM_ENCOM #8
INPUT #8, S1
COM_DISCOM #8
```

- connecting between robot 1 and robot 2:

```
COM_ENCOM #10
INPUT #10, I3
COM_DISCOM #10
```

The meaning of this code is that the robot will open the connection (COM_ENCOM) and wait for the signal from other robot or scanner then will put the information that has been sent from the other side to the robot that waiting for the signal (INPUT), and then when the signal has been identified the robot will close the connection and begin the process (COM_DISCOM).

Scanner:

As we explained before the line that connects the scanner to the first robot is (#8), and the scanner will send signal with the information to the robot. the scanner's job is to scan the barcode to identify the object that the robot is carrying, and that will help the robot to continue his process. The barcode can be created by an online generator, and create the barcode that user needed. example for the barcode is (SQ) and that means square, and An example of the scanner code:

- IF S1 = "SQ" THEN

The meaning of this code is that (IF) the barcode that the scanner has scanned (S1) is equal to (SQ), then (THEN) this object is square.

IF statement:

The user will definitely use the if statement to program the robot for specific process, and this code will help to make robot identify an object or to start a process after getting signal from the other robot. In our project we used this if statement code many times, and an example for it:

- code components:
  - IF
  - ELSEIF
  - THEN
  - END IF
- Robot 1:
  - IF S1 = "SQ" THEN

    APPROACH L, j05, @E 130

    MOVE L, @E j05

    DELAY 500

    'OPEN GRIPPER & DEPART

    SET IO65

    RESET IO64

    DELAY 500

    DEPART L, @E 100

This code explains that (IF) the scanned barcode is for the square (THEN) move the object to its own jig place.

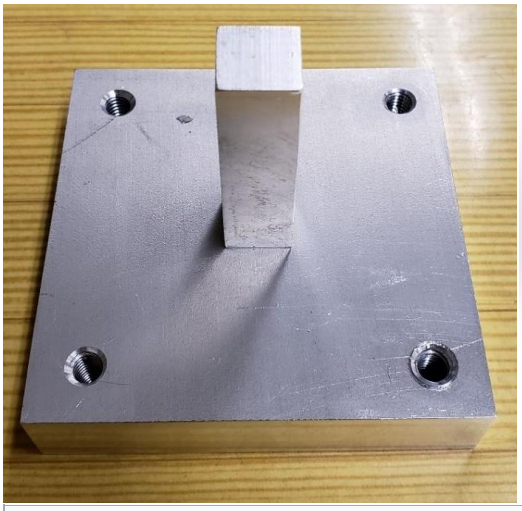
- Robot 2:
  - IF I3 = 3 THEN

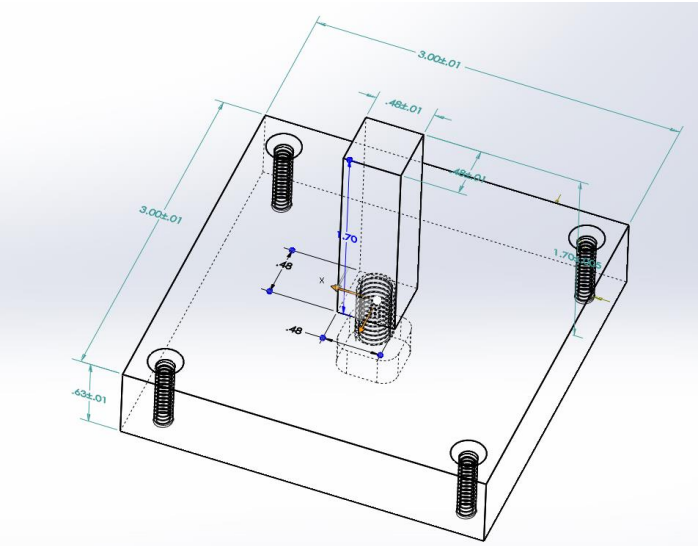
this code from robot 2 explains that (IF) the signal that has been sent from the robot 1 is (I3 = 3) , (THEN) robot 2 will start screwing.

SYSTEM HARDWARE

Parts

Our parts went through multiple iterations. First we 3D printed them, changed the dimensions several times to optimize. To add more weight and more applicability to real world situations we machined the final parts with Aluminum. The holes were threaded and counter sunk at ½ in @ 90 degrees.



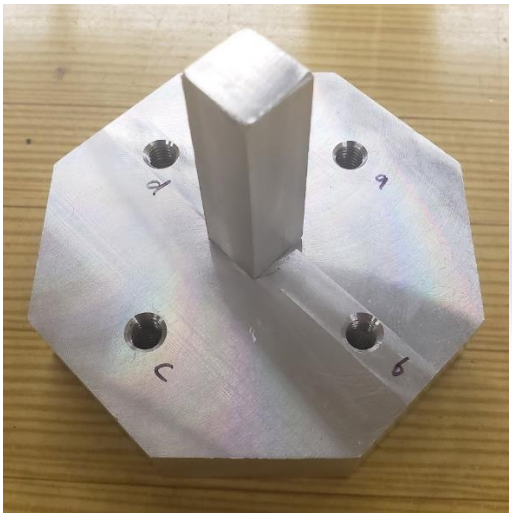


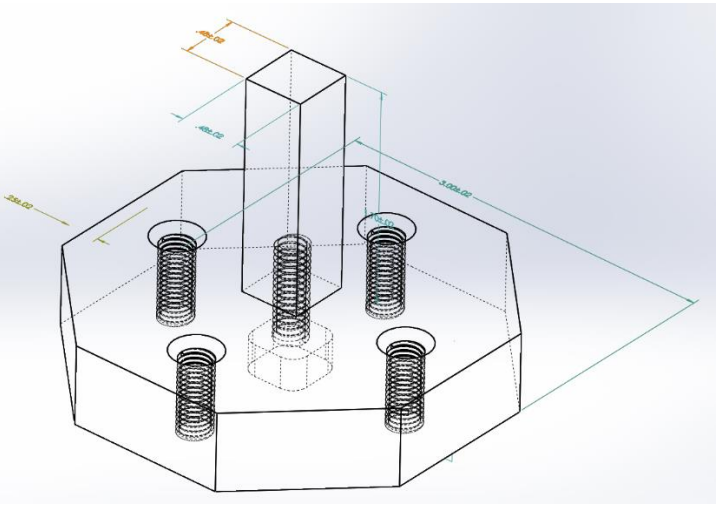
The template is 2" x 2" and 1/2" thick.

The extruded part is 2" tall and a 1/4" x 1/4" square.

There is a 1/4" x 1/4" x 1/4" square cut out of the bottom to place on a peg so we can guarantee screw hole locations.

We changed to using Aluminum parts instead of plastic. We were having an issue with the parts being picked up after the screw was inserted. Using Aluminum significantly increased the weight of the part and has helped eliminate the part being lifted. Also, Using a metal part has allowed us to tap the part and eliminate the need of metal inserts.

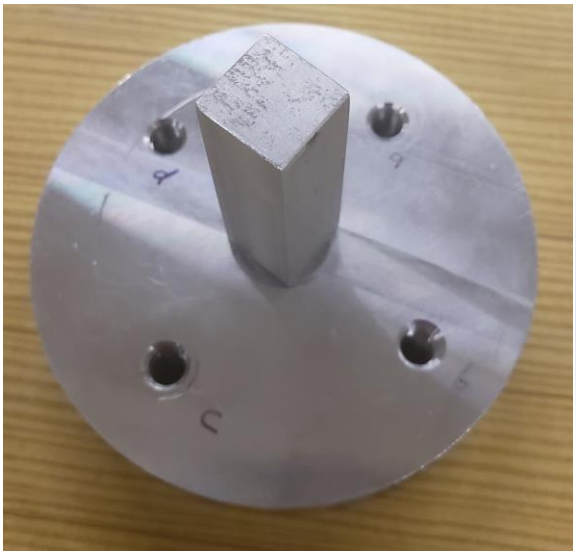


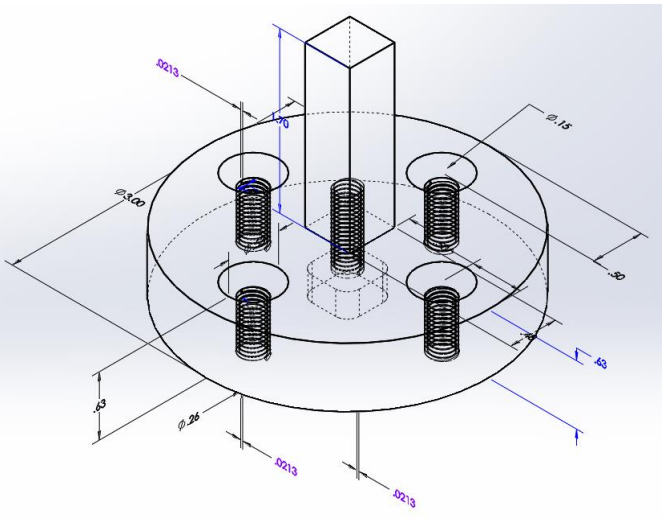


The octagon has a side length of 1 1/4".

The extruded part is 2" tall and a 1/4" x 1/4" square.

There is a 1/4" x 1/4" x 1/4" square cut out of the bottom to place on a peg so we can guarantee screw hole locations.



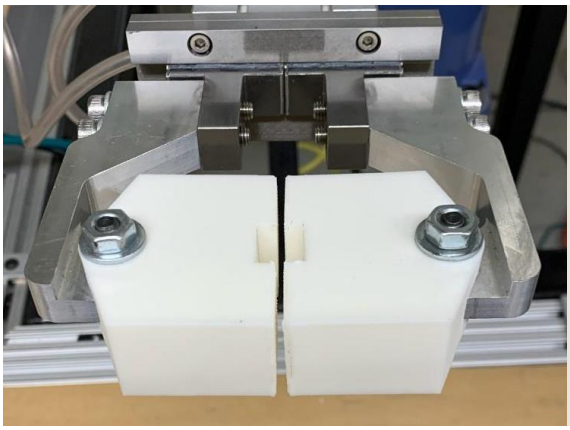


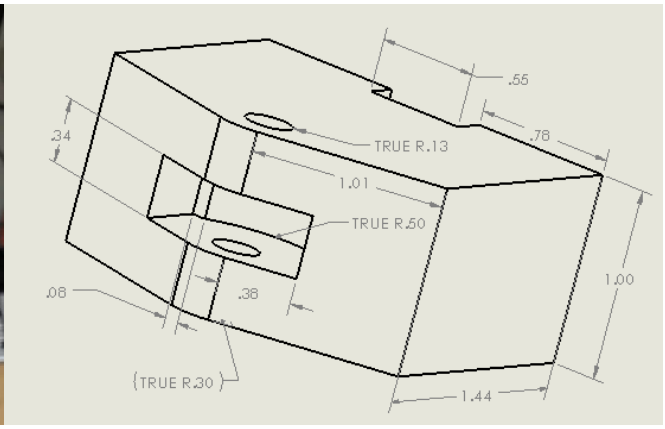
The Circle is a 3" Diameter with the extruded part being 2" tall and a 1/4" x 1/4" square.

There is a 1/4" x 1/4" x 1/4" square cut out of the bottom to place on a peg so we can guarantee screw hole locations.

Fixtures

Gripper Lips

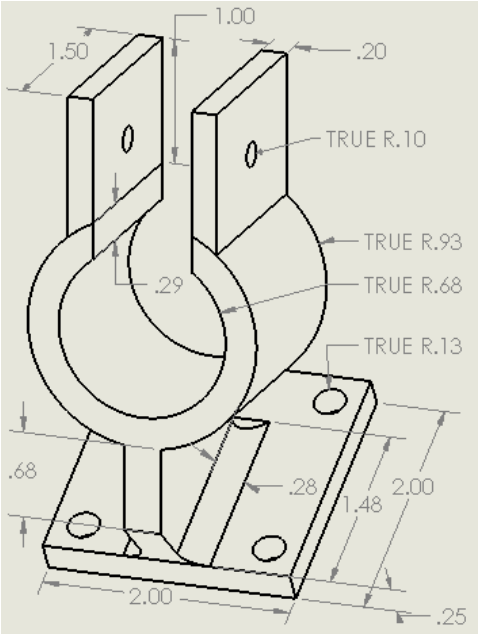




End effector for robot#1. These grippers are air pressurized so they can open and close.

They are used for the handling of mock parts.Gripper lips have a simple design. We have to continually tighten the single bolt as it gets loosened through multiple operations. We could have redesigned to either add a screw or add walls to lock into the tongs, but it was considered low priority. Also having a cheap piece of plastic as the part of the system to break first was a good thing.

End effectors



Updated end effector for for Robot #2. We changed to a clamping end effector because the thumb screws were tearing the rubber on the torque screwdriver and therefore moving the screwdrivers location. Using a clamp fixes this problem

a coding error, the fixture crashed into a part and has a small crack on the top side. So far this has caused no issues in operation though if necessary, it could be reprinted.

Tools

Dremel

GO 4-Volt Max Lithium-Ion Cordless Screwdriver with USB Charger and Insert Bits.

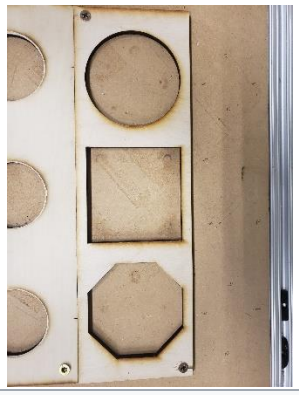
Pressure activated. It takes 2.5kg of pressing weight for the bit to begin turning.

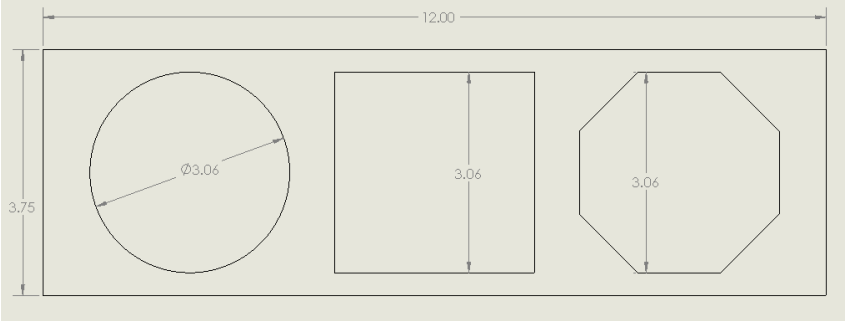
Fasteners

The screw we have chosen to use is a M6 x 1 with a threaded length of 1/2".
It is a Socket head cap screw (SHCS).
We chose a SHCS so that there were more orientations the screw could be in to be picked up.

Jigs

Part Holder



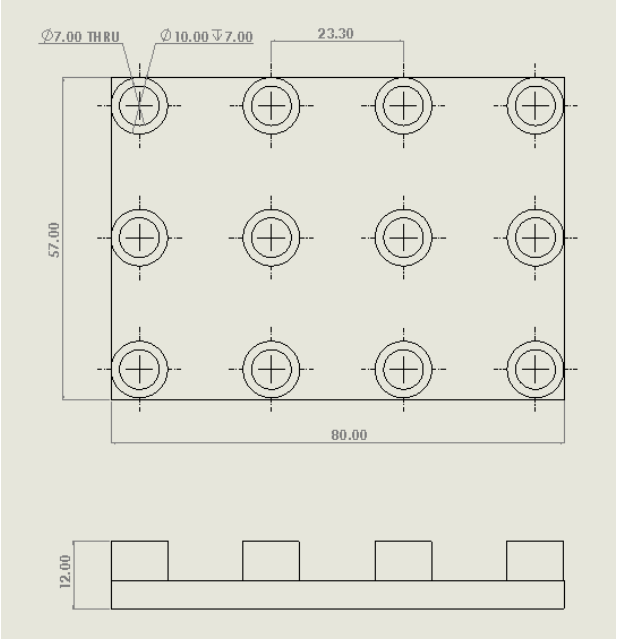


Current template design to hold 3 different shaped objects.

Template used so objects are in specific orientation for robot #2 screwdriver.

The cut outs are 1/16 of an inch bigger than the actual objects to allow for easier drop locations.

Screw Holder



Screw holder with set screw locations. We ended up needing to use set screw locations because the screw delivery system ended up being more advanced than expected. The diameter of the holes are about 2 mm larger than the screws. This allows the screws to be picked up without resistance from the holder while still being tight enough to not allow the screw to move around too much and able to get out of position to be picked up.

Drop off location testing

Test #	Square location	Square drop-off	Notes	Circle Location	Circle drop-off	Notes	Octagon Location	Octagon drop-off	Notes
1	2	yes	perfect	3	yes	hit front bounced in	1	no	left and back
2	3	no	front	1	yes	perfect	2	no	left and back
3	2	yes	perfect	1	yes	perfect	3	no	left and front
4	1	yes	perfect	3	yes	hit front bounced in	2	no	left and front
5	3	no	front	2	yes	perfect	1	no	left and back
6	1	yes	perfect	2	yes	perfect	3	no	left and front
7	2	yes	perfect	1	yes	perfect	3	no	left and front
8	1	yes	perfect	3	yes	hit front bounced in	2	no	left and back

We started with running the code 8 times to see what would happen and what needed to be changed. After those 8 tests we noticed that the octagon never worked, and the square did not work when started in location 3. Based on that we knew we needed to change the octagon drop-off location and pick-up location 3.

9	2	yes	perfect	3	yes	hit front bounced in	1	yes	right and front bounced in
10	3	yes	perfect	1	yes	perfect	2	yes	perfect
11	1	yes	perfect	2	yes	perfect	3	no	front
12	2	yes	bad contact on gripper lips	1	yes	perfect	3	yes	bad contact on gripper lips
13	2	yes	perfect	3	yes	perfect	1	no	bottom right corner
14	1	yes	perfect	3	yes	hit front bounced in	2	yes	right and bounced in
15	2	yes	perfect	1	yes	perfect	3	no	

After changing what we needed we ran the code 7 more times. After those tests we noticed that the square worked when starting in location 3 and the octagon

got better. After this test and having all 3 pick-up locations working for the square and circle we locked in the pick-up locations.

16	2	yes	perfect	3	yes	hit front bounced in	1	no	way right
17	3	no	front	1	yes	perfect	2	no	way right
18	1	no	front	2	yes	perfect	3	no	way right

After making another change to the octagon drop-off location we ran the code a few times. We noticed that things had somehow gotten worse for the octagon and square. We were puzzled by this and did not know what to do.

19	1	no	front	3	yes	hit front bounced in	2	no	left and back
20	2	yes	perfect	1	yes	perfect	3	no	left
21	1	yes	perfect	2	yes	perfect	3	no	left and back

We ended up going back to an old code and ran it a few times. We got the same result as before. We ended up thinking that maybe the way the octagon was approaching its drop-off location with coming close to the scanner was throwing it off. We changed the code so the octagon would stay up high after being scanned.

22	3	yes	perfect	2	yes	perfect	1	yes	perfect
23	1	yes	perfect	3	yes	perfect	2	yes	perfect
24	2	no	front awkward pick-up	1	yes	perfect	3	yes	perfect
25	2	no	front awkward pick-up	3	yes	perfect	1	yes	perfect
26	2	yes	perfect	1	yes	perfect	3	yes	perfect
27	2	yes	perfect	3	yes	perfect	1	yes	perfect
28	3	yes	perfect	1	yes	scraped lower right corner	2	yes	perfect
29	1	yes	perfect	2	yes	perfect	3	yes	perfect
30	2	yes	perfect	3	yes	perfect	1	yes	perfect
31	3	yes	perfect	2	yes	perfect	1	yes	perfect
32	3	yes	perfect	2	yes	perfect	1	yes	perfect
33	2	yes	perfect	3	yes	perfect	1	yes	perfect
34	3	yes	perfect	1	yes	perfect	2	yes	perfect
35	2	no	front	3	yes	perfect	1	yes	perfect
36	1	no	front	2	yes	perfect	3	yes	perfect
37	3	yes	perfect	1	yes	perfect	2	yes	perfect

We ran the code a bunch of times. In the current state the code works very well. There are still times that the parts get picked up awkwardly and do not go in. We have noticed that over time the gripper lips get bumped slightly out of position and that is what causes the awkward pick-up and failure of placing the part in its spot correctly.

Robot #2 Screwing in Test

Test #	Part Location	# of Magnents	Speed (%)	Slipped/Spun /Straight	Lifted Part?	Screwed in Fully?	Notes
1	Pushed into bottom left corner	2	1	Slipped	yes	yes	Original Code
2	Pushed into bottom right corner	2	1	Straight	yes	yes	
3	Pushed into bottom right corner	2	1	Spun	no	no	
4	Pushed into bottom right corner	2	1	Spun	yes	yes	
5	Pushed into bottom right corner	2	1	Barley Slipped	no	yes	
6	Pushed into bottom right corner	2	1	Barley Slipped	no	yes	
7	Pushed into bottom right corner	2	1	Slipped	yes	yes	
8	Pushed into bottom right corner	2	1	Slipped	yes*	yes	
9	Pushed into bottom right corner	2	1	Slipped	yes	no	
10	Pushed into bottom right corner	2	1	Straight	no	yes	
11	Pushed into bottom right corner	2	1	Slipped	yes	yes	
12	Pushed into bottom left corner	2	1	Slipped	yes	yes	
13	Pushed into bottom left corner	2	1	Slipped	no	no	
14	Pushed into bottom left corner	2	1	Slipped	yes	yes	
15	Pushed into bottom left corner	2	1	Spun	yes	yes	
16	Pushed into bottom left corner	2	1	Slipped	no	no	
17	Pushed into bottom left corner	2	1	slipped	no	no	
18	Pushed into bottom left corner	2	1	spun	yes*	yes	
19	Pushed into bottom left corner	2	1	Straight	yes*	yes	
20	Pushed into bottom left corner	2	1	Slipped	no	no	

We started with running 20 tests. After running these tests, we noticed that for the most part if the screw did not go in fully the part would not lift up. We also noticed that the screw rarely went straight into the hole.

21	Pushed into bottom left corner	2	7	Slipped	no	no
22	Pushed into bottom left corner	2	7	Slipped	yes	no
23	Pushed into bottom left corner	2	7	Guided in	yes**	yes
24	Pushed into bottom left corner	2	7	Guided in	no	no
25	Pushed into bottom left corner	2	7	Guided in	yes*	no
26	Pushed into bottom left corner	2	7	Guided in	yes	yes
27	Pushed into bottom left corner	2	7	Guided in	yes	yes
28	Pushed into bottom left corner	2	7	Guided in	yes*	no

We then wanted to see what would happen if we increased the speed that the robot was lifting up at from 1 to 7%. We ran 8 tests and it was not any better. We started guiding the screw straight in to eliminate a variable. We did like having the robot moving faster because it took less time.

29	Pushed into bottom left corner	2	7	Guided in	yes	yes
30	Pushed into bottom left corner	2	7	Guided in	yes*	yes
31	Pushed into top left corner	2	7	Guided in	yes	yes
32	Pushed into top left corner	2	7	Guided in	yes	yes
33	Pushed into bottom right corner	2	7	Guided in	yes	yes
34	Pushed into bottom right corner	2	7	Guided in	yes	yes

We edited the speed in the code to be even faster when lifting up even at the same 7% speed set on the controller. With this the part lifted up every time so we concluded that the faster speed did not help.

35	Pushed into bottom left corner	2	7	straight in	yes	yes
36	Pushed into bottom left corner	2	7	slipped	yes	yes
37	Pushed into bottom left corner	2	7	straight in	yes	yes
38	Pushed into bottom left corner	2	7	slipped	no	yes
39	Pushed into bottom left corner	2	7	slipped	no	yes
40	Pushed into bottom left corner	2	7	straight	yes	yes

We countersunk the hole so that the screw could be guided in and not have to slip into the hole. This helped the screw go into the hole better on its own. But we still needed to solve the problem of the part being picked up.

41	Pushed into bottom left corner	1 Big	7	straight	no	yes
42	Pushed into bottom left corner	1 Big	7	slipped	no	yes
43	Pushed into bottom left corner	1 Big	7	slipped	no*	yes
44	Pushed into bottom left corner	1 Big	7	slipped	no	yes
45	Pushed into bottom left corner	1 Big	7	slipped	no	yes
46	Pushed into bottom left corner	1 Big	7	Straight	yes	yes
47	Pushed into bottom left corner	1 Big	7	Straight	no	yes
48	Pushed into bottom left corner	1 Big	7	straight	no	yes
49	Pushed into bottom left corner	1 Big	7	straight	no	yes
50	Pushed into bottom left corner	1 Big	7	Straight	yes	yes
51	Pushed into top right corner	1 Big	7	straight	yes	yes
52	Pushed into top right corner	1 Big	7	Straight in	no	yes
53	Pushed into top right corner	1 Big	7	slipped	no	yes
54	Pushed into top right corner	1 Big	7	spun	no	yes
55	Pushed into top right corner	1 Big	7	slipped	yes	yes
56	Pushed into bottom right corner	1 Big	7	straight	no	yes
57	Pushed into bottom right corner	1 Big	7	straight	no	yes
58	Pushed into bottom right corner	1 Big	7	straight	no	yes
59	Pushed into bottom right corner	1 Big	7	straight	no	yes
60	Pushed into top left corner	1 Big	7	straight	no	yes

61	Pushed into top left corner	1 Big	7	slipped	no	yes
62	Pushed into top left corner	1 Big	7	slipped	yes	yes
63	Pushed into top left corner	1 Big	7	slipped	yes	yes
64	Pushed into top left corner	1 Big	7	slipped	no	yes
65	Pushed into top left corner	1 Big	7	slipped	no	yes

We ended up ordering different magnets for a different issue but decided to test out the screwing with one of these magnets. We put one of these magnets on and ended up having a 76% fully successful rate of not having the part being lifted up and screwed in all the way. This was up from the 10% fully successful rate that we started with.

Troubleshooting

Issue: Screws on the end effector for Robot #2 started to rip the rubber on the screwdriver changing the position of the screwdriver.

Solution: We redesigned the end effect to act as a clamp where we could tighten one bolt on the front of it and squeeze the screwdriver to keep it in its place.

Issue: Screwdriver not able to go up and down perfectly straight which is causing issues with picking up screws and fastening the screws

Solution: We want to be able to use approach/depart like we are able to do for robot #1. We did not think that this was possible because of the position of the end effector. Using approach/depart uses the face of the last joint and moves in that direction which was not the direction we wanted it to move in. We looked in the setting up manual for approach/depart and learned that the approach/depart only uses the z-axis of the tool coordinates, but we are able to change the axis and define the z-axis in whatever way we want.

We also learned that the command DEPART P, 70 is the same as MOVE P, P0+(0,0,-70)H. So we are going to try and use the move command instead of trying to figure out what we need to change the axis to and how to actually change them.

In the end we ended up using 2 sets of coordinates for each screw location. On set was above the screw/hole and one was at the bottom position of the screw/hole straight down from the above location. Doing this allowed us to use the move command first to the above location then straight down to the below then straight back up to the above.

Issue: Keep getting error that says value out of range.

Solution: We discovered that the way we were trying to name our variables was causing this error. We were attempting to name the variables J101 for one object then J201 for a different object but the code did not like that and wanted it to be J01,J02,J03,...,. When we started doing that the code worked as expected.

Appendix

Code for Robot 1:

```
'!TITLE "<Title>"
PROGRAM rs2code3

p1 = (58.00, 287.11, 305.14, -179.02, -10.75, 85.40) 'Shrink position 1
j01 = (-2.71, 56.94, 106.37, -0.98, 15.15, 180.88) 'Pick-up location 1
j02 = (-13.10, 57.70, 103.64, 1.56, 17.93, 166.31) 'Pick-up location 2
j03 = (-21.63, 60.53, 97.64, 0.74, 20.29, 160.32) 'Pick-up location 3
j00 = (-13.70, -9.19, 156.00, 3.54, 23.00, 167.31) 'shrink pos in J
p04 = (-7.67, -408.13, 286.39, -179.65, -0.24, -5.34) 'scanner
j05 = (-63.67, 63.29, 93.98, 3.19, 21.6, 114.54)'jig (SQUARE)
j06 = (-66.79, 71.31, 74.44, 1.87, 33.91, 115.07) 'jig (Circle)
j07 = (-59.23, 55.58, 111.08, 6.35, 13.49, 117.24) 'jig (Octagon)

'2782
TAKEARM

'Rest pos
MOVE P, @E j00

'OPEN GRIPPER
SET IO65
RESET IO64

'PICKUP POS 1
APPROACH P, j01, @E 55
```

```
MOVE P, @E j01
DELAY 1000

'CLOSE GRIPPER
SET IO64
RESET IO65
DELAY 1000
DEPART P, @E 45

'SAFE POS 2
MOVE P, @E j00

'SCANNER POS
MOVE P, @E P4

DELAY 1000

'ESTABLISHES CONNECTION WITH SCANNER, GATHERS BARCODE INPUT INTO S1
COM_ENCOM #8
 INPUT #8, S1
 COM_DISCOM #8

 'Square
 IF S1 = "SQ" THEN
  APPROACH L, j05, @E 130
 MOVE L, @E j05
 DELAY 500
```

```
'OPEN GRIPPER & DEPART
SET IO65
RESET IO64
DELAY 500
DEPART L, @E 100

  'Circle
ELSEIF S1 = "CR" THEN
   APPROACH L, j06, @E 130
  MOVE L, @E j06
  DELAY 500

'OPEN GRIPPER & DEPART
SET IO65
RESET IO64
DELAY 500
DEPART L, @E 100
  'Octagon
  ELSE
APPROACH L, j07, @E 130
  MOVE L, @E j07
  DELAY 500

'OPEN GRIPPER & DEPART
SET IO65
RESET IO64
DELAY 500
```

```
DEPART L, @E 100
  END IF

'BACK TO REST POS
MOVE P, @E j00

'2nd pickup
'Rest pos
MOVE P, @E j00

'OPEN GRIPPER
SET IO65
RESET IO64

'PICKUP POS 2
APPROACH P, j02, @E 55
MOVE P, @E j02
DELAY 1000

'CLOSE GRIPPER
SET IO64
RESET IO65
DELAY 1000
DEPART P, @E 45

'SAFE POS 2
MOVE P, @E j00
```

```
'SCANNER POS
MOVE P, @E P4

DELAY 1000

'ESTABLISHES CONNECTION WITH SCANNER, GATHERS BARCODE INPUT INTO S1
COM_ENCOM #8
  INPUT #8, S1
  COM_DISCOM #8

  'Square
  IF S1 = "SQ" THEN
    APPROACH L, j05, @E 130
  MOVE L, @E j05
  DELAY 500

'OPEN GRIPPER & DEPART
SET IO65
RESET IO64
DELAY 500
DEPART L, @E 100

   'Circle
ELSEIF S1 = "CR" THEN
    APPROACH L, j06, @E 130
  MOVE L, @E j06
  DELAY 500
```

```
'OPEN GRIPPER & DEPART
SET IO65
RESET IO64
DELAY 500
DEPART L, @E 100
  'Octagon
  ELSE
APPROACH L, j07, @E 130
  MOVE L, @E j07
  DELAY 500

'OPEN GRIPPER & DEPART
SET IO65
RESET IO64
DELAY 500
DEPART L, @E 100
   END IF

'BACK TO REST POS
MOVE P, @E j00

'3rd pickup
'Rest pos
MOVE P, @E j00

'OPEN GRIPPER
SET IO65
```

```
RESET IO64

'PICKUP POS 3
APPROACH P, j03, @E 55
MOVE P, @E j03
DELAY 1000

'CLOSE GRIPPER
SET IO64
RESET IO65
DELAY 1000
DEPART P, @E 45

'SAFE POS 2
MOVE P, @E j00

'SCANNER POS
MOVE P, @E P4

DELAY 1000

'ESTABLISHES CONNECTION WITH SCANNER, GATHERS BARCODE INPUT INTO S1
COM_ENCOM #8
 INPUT #8, S1
 COM_DISCOM #8

 'Square
 IF S1 = "SQ" THEN
```

```
   APPROACH L, j05, @E 130
  MOVE L, @E j05
  DELAY 500

'OPEN GRIPPER & DEPART
SET IO65
RESET IO64
DELAY 500
DEPART L, @E 100

   'Circle
ELSEIF S1 = "CR" THEN
   APPROACH L, j06, @E 130
  MOVE L, @E j06
  DELAY 500

'OPEN GRIPPER & DEPART
SET IO65
RESET IO64
DELAY 500
DEPART L, @E 100
  'Octagon
  ELSE
APPROACH L, j07, @E 130
  MOVE L, @E j07
  DELAY 500
```

```
'OPEN GRIPPER & DEPART
SET IO65
RESET IO64
DELAY 500
DEPART L, @E 100
  END IF

'BACK TO REST POS
MOVE P, @E j00

'Send Signal to Robot 2
I3 = 3
COM_ENCOM #4
WRITE #4, I3
COM_DISCOM #4

'Reset Variables
I3 = 0

GIVEARM

END
```

Code for Robot 2:

```
'!TITLE "<Title>"
PROGRAM RoboShopScrew
```

j01 = (63.03, 1.53, 147.98, 3.37, -38.36, 231.67) 'rest j

j02 = (71.98, 40.26, 147.42, 8.80, -69.34, 237.18) 'Well 1a

j03 = (71.98, 44.00, 146.12, 8.67, -71.76, 237.57) 'Well 1b

j04 = (73.10, 40.96, 144.10, 10.05, -66.73, 236.85) 'Well 2a

j05 = (73.10, 44.20, 142.97, 9.90, -68.81, 237.24) 'Well 2b

j06 = (74.10, 41.64, 140.73, 11.24, -64.08, 236.37) 'Well 3a

j07 = (74.10, 44.78, 139.63, 11.06, -66.08, 236.81) 'Well 3b

j08 = (68.36, 40.72, 146.32, 5.42, -68.76, 236.60) ’Well 4a

j09 = (68.35, 43.95, 145.19, 5.35, -70.86, 236.81) 'Well 4b

j10 = (69.68, 41.31, 143.06, 6.80, -66.10, 236.43) 'Well 5a

j11 = (69.68, 44.53, 141.93, 6.70, -68.17, 236.70) 'Well 5b

j12 = (70.66, 41.90, 139.78, 7.92, -63.41, 236.11) 'Well 6a

j13 = (70.66, 44.98, 138.69, 7.79, -65.39, 236.41) 'Well 6b

j14 = (65.60, 42.85, 144.43, 4.62, -69.35, 236.50) 'Well 7a

j15 = (65.60, 44.23, 143.94, 4.60, -70.23, 236.58) 'Well 7b

j16 = (66.95, 43.33, 141.33, 6.00, -66.70, 236.38) 'Well 8a

j17 = (66.95, 44.47, 140.89, 5.97, -67.42, 236.47) 'Well 8b

j18 = (68.20, 43.82, 138.08, 7.36, -66.70, 236.11) 'Well 9a

j19 = (68.20, 45.04, 137.64, 7.32, -64.75, 236.22) 'Well 9b

j20 = (62.24, 43.09, 143.03, 1.45, -68.22, 236.01) 'Well 10a

j21 = (62.24, 43.82, 142.77, 1.44, -68.68, 236.03) 'Well 10b

j22 = (63.71, 43.41, 139.95, 2.91, -65.45, 236.03) 'Well 11a

j23 = (63.71, 44.53, 139.54, 2.89, -66.17, 236.07) 'Well 11b

j24 = (65.13, 44.41, 136.60, 4.37, -63.10, 235.93) 'Well 12a

j25 = (65.13, 45.26, 136.29, 4.35, -63.64, 235.97) 'Well 12b

j31 = (50.74, 52.71, 109.33, -12.62, -44.46, 240.22) 'Circle 1a

j32 = (50.74, 54.00, 108.83, -12.46, -45.22, 239.98) 'Circle 1b

j33 = (50.27, 55.32, 102.48, -14.32, -40.36, 241.91) 'Circle 2a

j34 = (50.27, 56.67, 101.93, -14.09, -41.14, 241.29) 'Circle 2b

j35 = (46.89, 55.46, 102.95, -18.66, -41.29, 243.50) 'Circle 3a

j36 = (46.89, 56.60, 102.49, -18.42, -41.93, 243.18) 'Circle 3b

j37 = (47.12, 52.24, 110.09, -17.07, -44.98, 241.64) 'Circle 4a

j38 = (47.12, 53.64, 109.55, -16.83, -45.80, 241.29) 'Circle 4b

j41 = (54.75, 56.69, 99.39, -8.49, -38.26, 239.72) 'Square 1a

j42 = (54.75, 58.34, 98.69, -8.34, -39.23, 239.53) 'Square 1b

j43 = (57.49, 61.77, 88.73, -5.28, -32.58, 238.78) 'Square 2a

j44 = (57.49, 63.01, 88.16, -5.19, -33.26, 238.68) 'Square 2b

j45 = (53.62, 65.02, 81.06, -13.19, -28.53, 244.13) 'Square 3a

j46 = (53.62, 66.38, 80.41, -12.90, -29.23, 243.80) 'Square 3b

j47 = (50.58, 60.11, 91.97, -15.81, -34.71, 244.15) 'Square 4a

j48 = (50.58, 61.40, 91.37, -15.55, -35.39, 243.83) 'Square 4b

j51 = (60.05, 68.79, 73.81, -1.39, -24.58, 236.80) 'Octagon 1a

j52 = (60.05, 70.06, 73.14, -1.36, -25.19, 236.77) 'Octagon 1b

j53 = (59.34, 74.28, 62.84, -3.68, -19.11, 238.68) 'Octagon 2a

j54 = (59.34, 75.25, 62.25, -3.61, -19.50, 238.60) 'Octagon 2b

j55 = (56.71, 72.60, 65.76, -10.09, -20.55, 243.41) 'Octagon 3a

j56 = (56.71, 73.83, 65.06, -9.85, -21.07, 243.17) 'Octagon 3b

j57 = (57.32, 67.37, 76.25, -6.91, -25.69, 240.48) 'Octagon 4a

j58 = (57.32, 68.84, 75.51, -6.73, -26.42, 240.28) 'Octagon 4b

```
TAKEARM
SPEED 75

'REST POS
MOVE P, @E j01
'Receives Signal from Robot 1
        COM_ENCOM #10

        INPUT #10, I3

'Start Screwing
        IF I3 = 3 THEN

                'Fastener 1 pickup
                MOVE P, @E j02
                SPEED 6.6667 'change speed to 2%
                MOVE P, @E j03
                DELAY 1000
                SPEED 75
                MOVE P, @E j02

                'Back to REST POS
                MOVE P, @E j01

                'Circle hole 4
                MOVE P, @E j37
                SPEED 6.6667 'change speed to 2%
                MOVE P, @E j38
```

```
DELAY 1000
SPEED 75
MOVE P, @E j37

SPEED 50
'BACK TO REST
MOVE P, @E j01

'Fastener 2 pickup
MOVE P, @E j04
SPEED 6.6667 'change speed to 2%
MOVE P, @E j05
DELAY 1000
SPEED 75
MOVE P, @E j04

'Back to REST POS
MOVE P, @E j01

'Circle hole 2
MOVE P, @E j33
SPEED 6.6667 'change speed to 2%
MOVE P, @E j34
DELAY 1000
SPEED 75
MOVE P, @E j33

SPEED 50
```

```
'BACK TO REST
MOVE P, @E j01

'Fastener 3 pickup
MOVE P, @E j06
SPEED 6.6667 'change speed to 2%
MOVE P, @E j07
DELAY 1000
SPEED 75
MOVE P, @E j06

'Back to REST POS
MOVE P, @E j01

'Square hole 1
MOVE P, @E j41
SPEED 6.6667 'change speed to 2%
MOVE P, @E j42
DELAY 1000
SPEED 75
MOVE P, @E j41

SPEED 50
'BACK TO REST
MOVE P, @E j01

'Fastener 4 pickup
MOVE P, @E j08
```

```
SPEED 6.6667 'change speed to 2%
MOVE P, @E j09
DELAY 1000
SPEED 75
MOVE P, @E j08

'Back to REST POS
MOVE P, @E j01

'Square hole 2
MOVE P, @E j43
SPEED 6.6667 'change speed to 2%
MOVE P, @E j44
DELAY 1000
SPEED 75
MOVE P, @E j43

SPEED 50
'BACK TO REST
MOVE P, @E j01

'Fastener 5 pickup
MOVE P, @E j10
SPEED 6.6667 'change speed to 2%
MOVE P, @E j11
DELAY 1000
SPEED 75
```

```
MOVE P, @E j10

'Back to REST POS
MOVE P, @E j01

'Octagon hole 4
MOVE P, @E j57
SPEED 6.6667 'change speed to 2%
MOVE P, @E j58
DELAY 1000
SPEED 75
MOVE P, @E j57



SPEED 50
'BACK TO REST
MOVE P, @E j01

'Fastener 6 pickup
MOVE P, @E j12
SPEED 6.6667 'change speed to 2%
MOVE P, @E j13
DELAY 1000
SPEED 75
MOVE P, @E j12

'Back to REST POS
```

```
                MOVE P, @E j01

                'Octagon hole 2
                MOVE P, @E j53

SPEED 6.6667 'change speed to 2%
                MOVE P, @E j54
                DELAY 1000
                SPEED 75
                MOVE P, @E j53

                SPEED 50
                'BACK TO REST
                MOVE P, @E j01


        END IF

'Reset the Variables
I3 = 0

END
```