AUT.841 Robot Manipulators: Modeling, Control and Programming

Assignment 1 – Robot Programming

Group members:

Haider Ali, Pratyush Thapalia, Chathuranga De Silva

Student ID's:

50507210, 151110739, 150963990

Task 1:

Steps:

1. We created two borders around two objects that are the curve thing and the box under it.

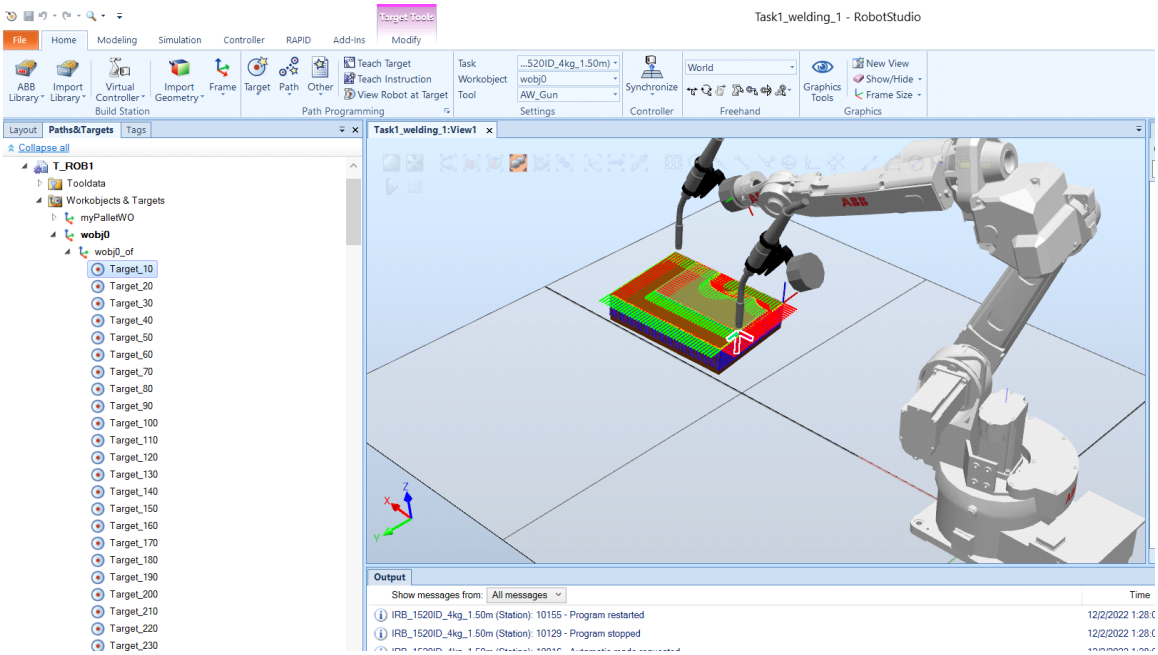
2. We made two paths, one on Curve little thing border and the other on the border of box under it.

3. The path we made, created a set of targets for us.

4. We chose the first target and tried out the option "View Tool On Target" . There we selected our "AW_GUN_PSF_25" on target.

5. Tool was not oriented properly, we fixed orientation by trying the option "Modify Target". In the suboptions we opened up the rotation and adjusted the Tool orientation. We copied the orientation and applied it to all other points.

The image below shows our set of targets and tool orientation on first target:



6. We syncronized the path with our rapid code. It created a procedure for us in Rapid.
7. In the rapid code. We created a while loop, Where we added our procedure call in the while loop.

8. We made changes on procedure as well. We added a Wait time Between Curve Path and Box Path.

Our code, PATH_10 is the procedure that moves the tool around the curve thing and PATH_20 is the path where our tool moves around outer object:

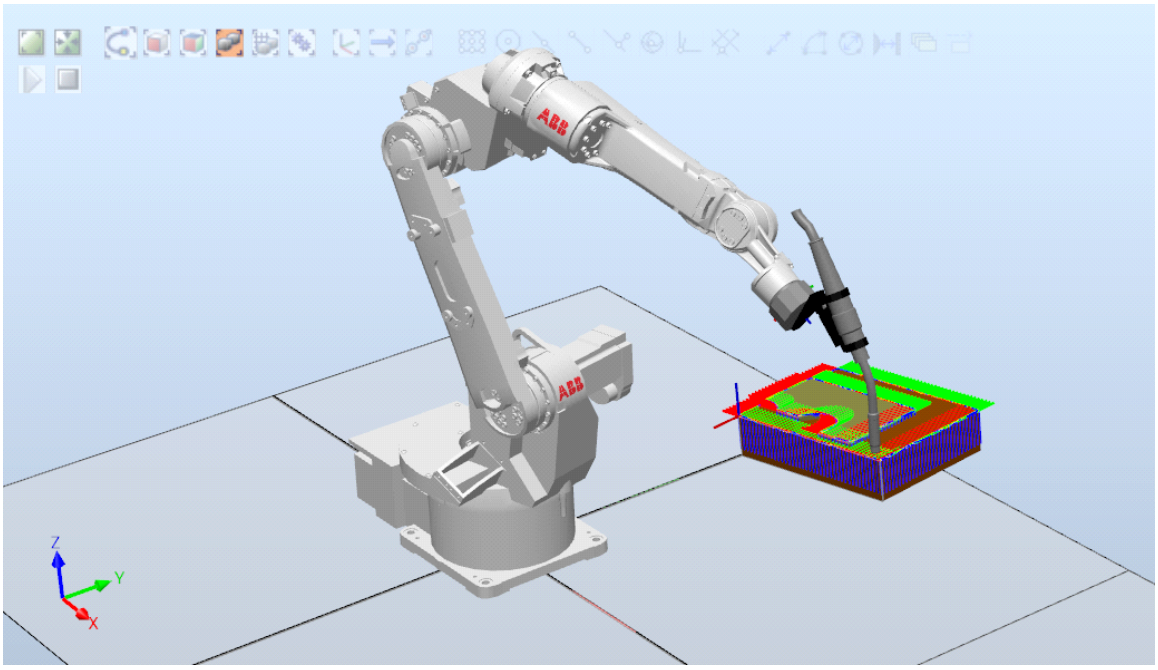
```
PROC main()

    VAR num sum;
    VAR bool bExit:=FALSE;

    WHILE NOT bExit DO
         Incr sum;
         IF sum = 10 THEN
             TPWrite "Sum :" \Num:= sum;
             bExit:=TRUE;
             ! ----------  Exit while loop and continue command
         ENDIF
         Path_10;
         WaitTime 3;
         Path_20;
         WaitTime 3;
    ENDWHILE


ENDPROC
```

9. When we ran our simulation, it made the correct movement on both object and stoped as we intended with our wait time.

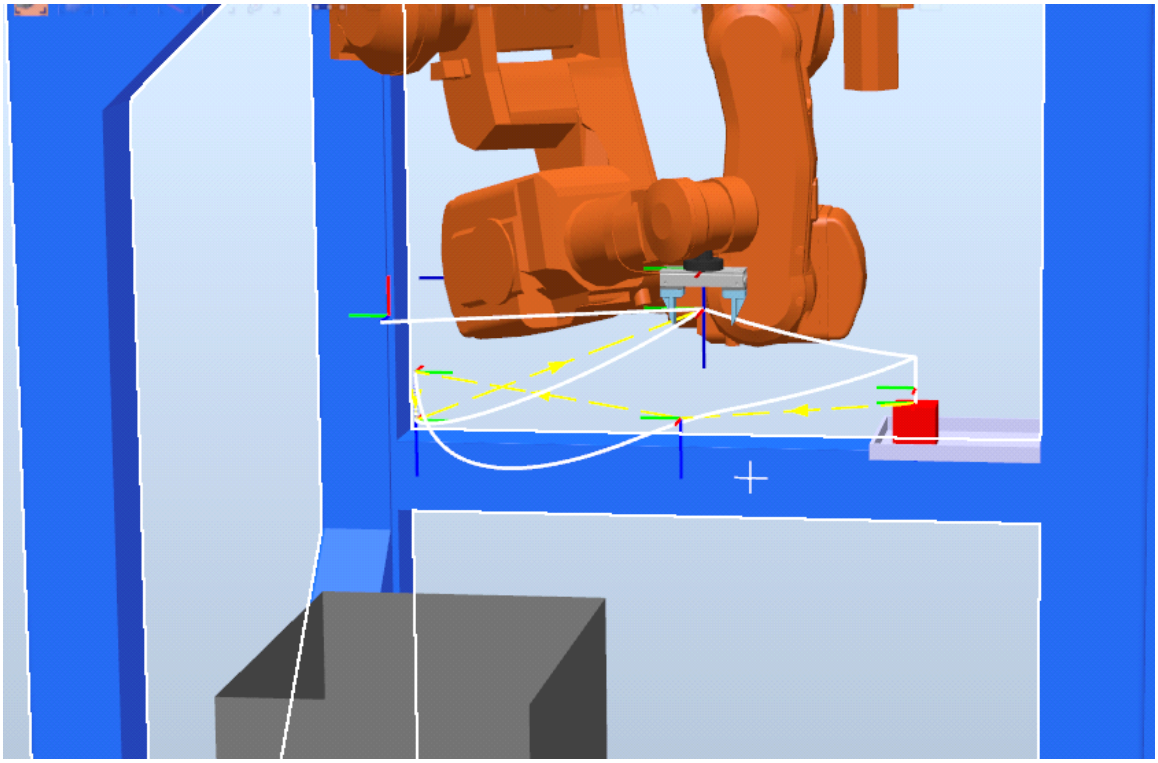


Task 2:

Steps:

1. We selected the pick target and then we created a prepick target which was slightly above it. It could have been done by using MoveJOff from pick target. Similarly, We created place and preplace targets. Similary it could have been done by using one point for place and chose MoveJOff method. Thirdly we chose a point between pre pick and pre place positions, in such a way that robot makes a circular movement between those points. In the end we created a home point where it goes after droping the cube.

The image below shows the path that our griper takes during pick and place:



2. We oriented our gripper and config our robot to the targets accordingly and we created a path from those targets. On the path tab we adjusted our motors speed and joint movements according to the requirements. Then we syncronized our path to the main module. There we added the gripper open and close code to open the gripper at prepick position and when it is on pick position it closes the gripper and wait for 2 seconds, then we added another movement in the code to go back vertically up to pre pick position. Then in the path our robots follow the middle point and pre place path. On pre place path which is few millimeters above place target. Our robot goes vertically down to place target and there is opens the gripper and then we added another wait method to stop for a second. After that our griper goes to home path.

Image of our code:

```
PROC Path_10()
    !Move to Pre Pick target
    MoveJ Pre_Pick,v500,z100,Fingers\WObj:=wobj0;
    !Griper Opening
    SetDO Rob1_Gripper_Reset,1;
    SetDO Rob1_Gripper_Set,0;
    WaitTime 2;
    !Move to Pick target
    MoveL Post_Pick,v50,z0,Fingers\WObj:=wobj0;
    !Griper Closing
    SetDO Rob1_Gripper_Set,1;
    SetDO Rob1_Gripper_Reset,0;
    WaitTime 2;
    !Move Back to Pre Pick target
    MoveL Pre_Pick,v500,z0,Fingers\WObj:=wobj0;
    !Move to Middle Target, the pre place target and then place target
    MoveJ Middle,v500,z100,Fingers\WObj:=wobj0;
    MoveJ Pre_Place,v500,fine,Fingers\WObj:=wobj0;
    MoveL Post_Place,v50,fine,Fingers\WObj:=wobj0;

    !Griper Opening
    SetDO Rob1_Gripper_Reset,1;
    SetDO Rob1_Gripper_Set,0;
    WaitTime 2;
    !Move Back to home target
    MoveJ Home,v500,z0,Fingers\WObj:=wobj0;
ENDPROC
```

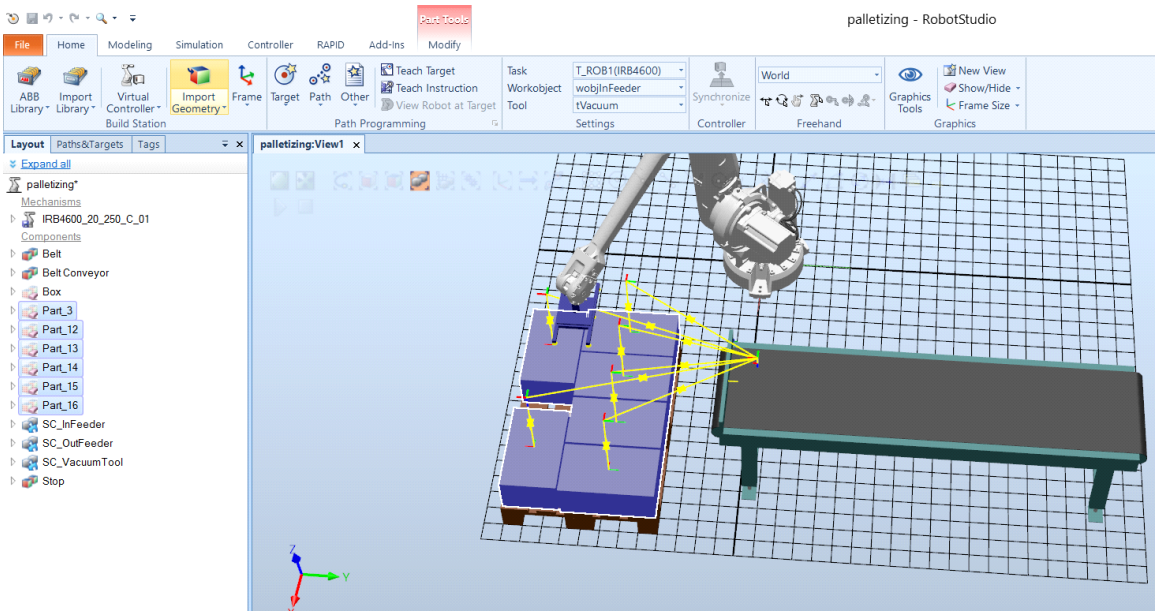
Task 3:

Steps:

1. For this task we opened the project and played around with the sensors. There we noticed, our vacum tool waits for Sc_Infeeder Line sensor. If we turn it on manually through I/0 in controllers before box arrives there, our vacum tool makes a move to the pick place. We noticed others sensors. One was on Sc_OutFeeder, which counts the box placed on Sc_OutFeeder, which can be used to stop the process or for any other thing we would like to do.

2. For targets we measured the box dimensions with measurement tool which were (490x290x190). We created 6 boxes for pallet patterns and put all the boxes on Sc_OutFeeder object according to pattern. Then we made targets on the boxes that were on the Sc_OutFeeder and above all those targets we created pre place targets which were right above our place targets, so our tool/object does not hit anything when Robot is moving object there.

The image below demonstrates the path and the hidden box objects that we used to target our place targets:



3. After syncronized our paths with targets. We added few more lines in code. Our Vacum waits for the box. When box reaches end of the line. It carries the box and place it on the Sc_OutFeeder.

This image is from actual simulation during vacum tool is placing the boxes on Sc_OutFeeder:

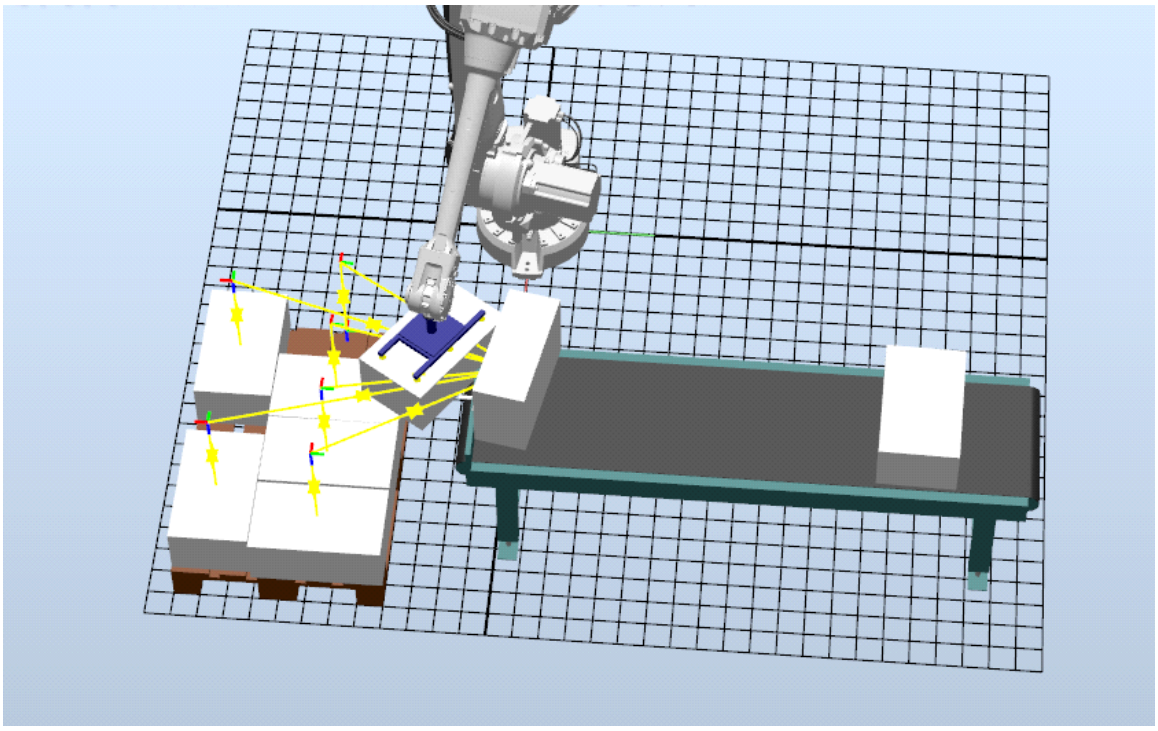


Image of our code:

```
PROC MyPath()
    MoveJ PrePick_10,v500,fine,tVacuum\WObj:=wobjInFeeder;
    WaitDI diBoxInPos,1;
    MoveL pPick,v300,fine,tVacuum\WObj:=wobjInFeeder;
    SetDO doVacuum,1;
    MoveL PrePick_10,v500,fine,tVacuum\WObj:=wobjInFeeder;
    !!First Box
    MoveJ Target_10,v500,fine,tVacuum\WObj:=wobjInFeeder;
    MoveL Target_20,v300,fine,tVacuum\WObj:=wobjInFeeder;
    SetDO doVacuum,0;
    MoveL Target_10,v500,fine,tVacuum\WObj:=wobjInFeeder;
    !!PICK PART
    MoveJ PrePick_10,v500,fine,tVacuum\WObj:=wobjInFeeder;
    WaitDI diBoxInPos,1;
    MoveL pPick,v300,fine,tVacuum\WObj:=wobjInFeeder;
    SetDO doVacuum,1;
    MoveL PrePick_10,v500,fine,tVacuum\WObj:=wobjInFeeder;
    !!Second Box
    MoveJ Target_30,v500,fine,tVacuum\WObj:=wobjInFeeder;
    MoveL Target_40,v300,fine,tVacuum\WObj:=wobjInFeeder;
    SetDO doVacuum,0;
    MoveL Target_30,v500,fine,tVacuum\WObj:=wobjInFeeder;
    !!PICK PART
    MoveJ PrePick_10,v500,fine,tVacuum\WObj:=wobjInFeeder;
    WaitDI diBoxInPos,1;
    MoveL pPick,v300,fine,tVacuum\WObj:=wobjInFeeder;
    SetDO doVacuum,1;
    MoveL PrePick_10,v500,fine,tVacuum\WObj:=wobjInFeeder;
    !!Third Box
    MoveJ Target_50,v500,fine,tVacuum\WObj:=wobjInFeeder;
    MoveL Target_60,v300,fine,tVacuum\WObj:=wobjInFeeder;
    SetDO doVacuum,0;
    MoveL Target_50,v300,fine,tVacuum\WObj:=wobjInFeeder;
    !!PICK PART
    MoveJ PrePick_10,v500,fine,tVacuum\WObj:=wobjInFeeder;
    WaitDI diBoxInPos,1;
    MoveL pPick,v300,fine,tVacuum\WObj:=wobjInFeeder;
    SetDO doVacuum,1;
    MoveL PrePick_10,v500,fine,tVacuum\WObj:=wobjInFeeder;
    !!Fourth Box
    MoveJ Target_70,v500,fine,tVacuum\WObj:=wobjInFeeder;
    MoveL Target_80,v300,fine,tVacuum\WObj:=wobjInFeeder;
    SetDO doVacuum,0;
    MoveL Target_70,v300,fine,tVacuum\WObj:=wobjInFeeder;

    !LAST Location
    MoveL PrePick_10,v300,fine,tVacuum\WObj:=wobjInFeeder;
ENDPROC
```

Things we considered while creating our simulation:

1. Orientation of the tool with respect to target was one major thing. We selected the orientations which would make our Robot make less rotations.

2. The movement between pre pick/post and pick/place were slow and "Linear". The movement between pre pick and pre place were fast and "Joint".

Things we didn't consider important at first:

1. During the live demo, we learned the important of z parameter in Move methods. So, z parameter means, if the target is between 2 points and z is z0, the it will move from there without and curve. If it is z100, it means it will curve when it tool is in 100 millimeter vicinity of the target. We can also use fine instead of z, which means the robot will go to exact target and then move the next one. We also learned about how the gripper variables work to signal open/close the gripper.

2. We also learned that we should chose less targets when we code, instead we should use offsets if possible. Because, those targets have to be reassigned on the actual tool.