ASE-9407 2018-01 Robot Manipulators: Modeling, Control and Programming

Assignment 2

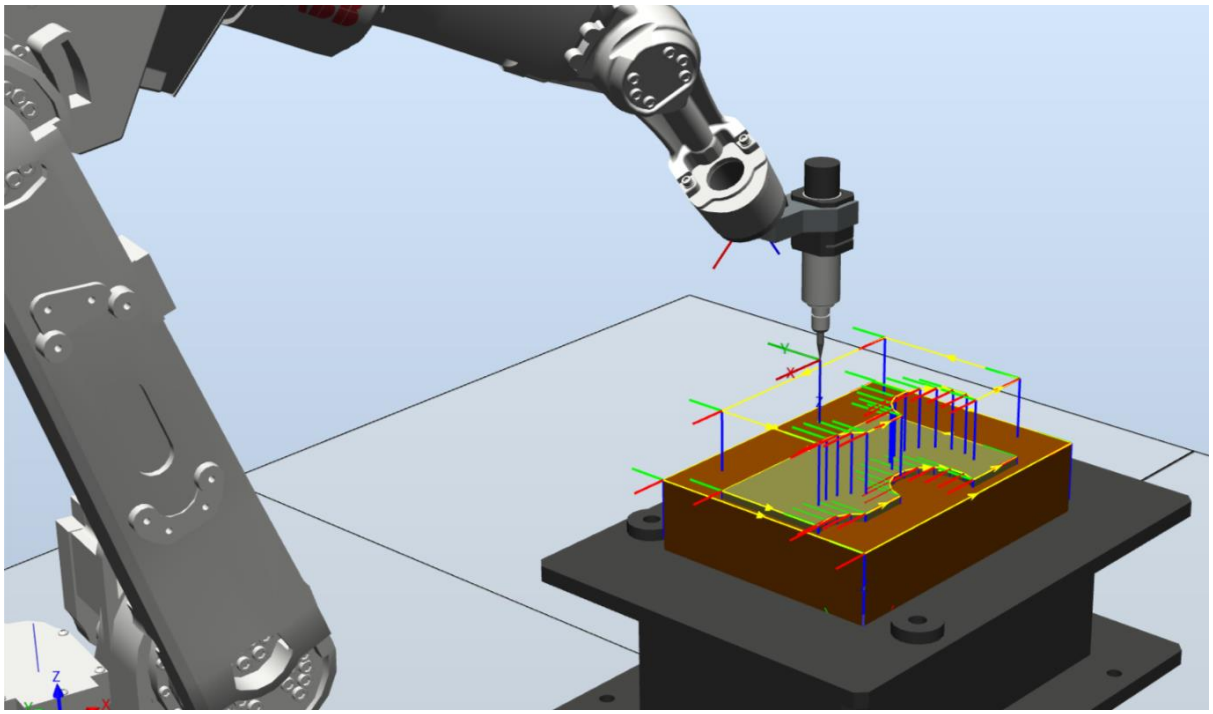
Authors:

Lorenzo Etchenique 282974
Jan Dierkes 282910
Christen Blomdahl 282803

Content

1. Welding routine

In this task, the contours of the workpiece (ochre) and its platform (brown), as well as a posterior workpiece contour placed above afterwards, are welded by a welding tool attached to an ABB IRB 1520 robot with 6 DOF. The planning paths are shown in Picture 1 presented by the yellow lines.

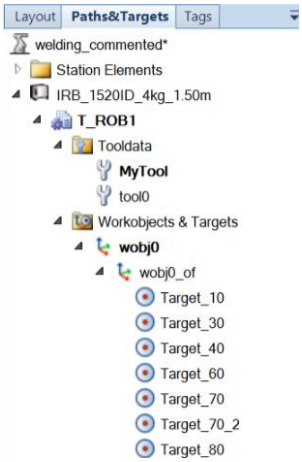


Picture 1. The welding procedure on the workpiece

First of all, the robot model is selected from the ABB library, after that, the robot tool which will be attached. Afterwards, robot pedestal and curve shape (workpiece) are loaded from the equipment tab.

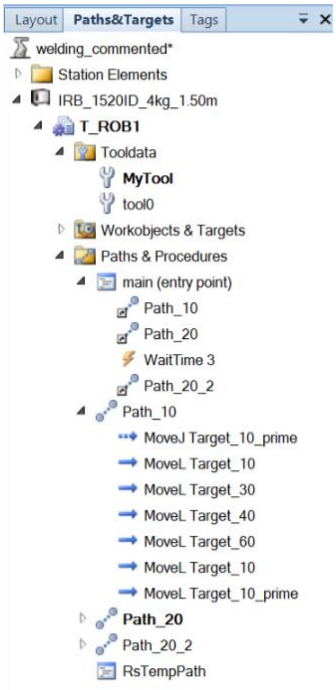
The workpiece is positioned over the pedestal using *snap object* and *part selection* tools.

Once the virtual objects are created and positioned, a Workobject is created and named by default as *wobj0.* Then the paths are created empty with *reference surface* regarding to the object. It is possible to create paths out of the edges of the objects in the virtual environment by using *autopaths*, then the target points are created automatically.



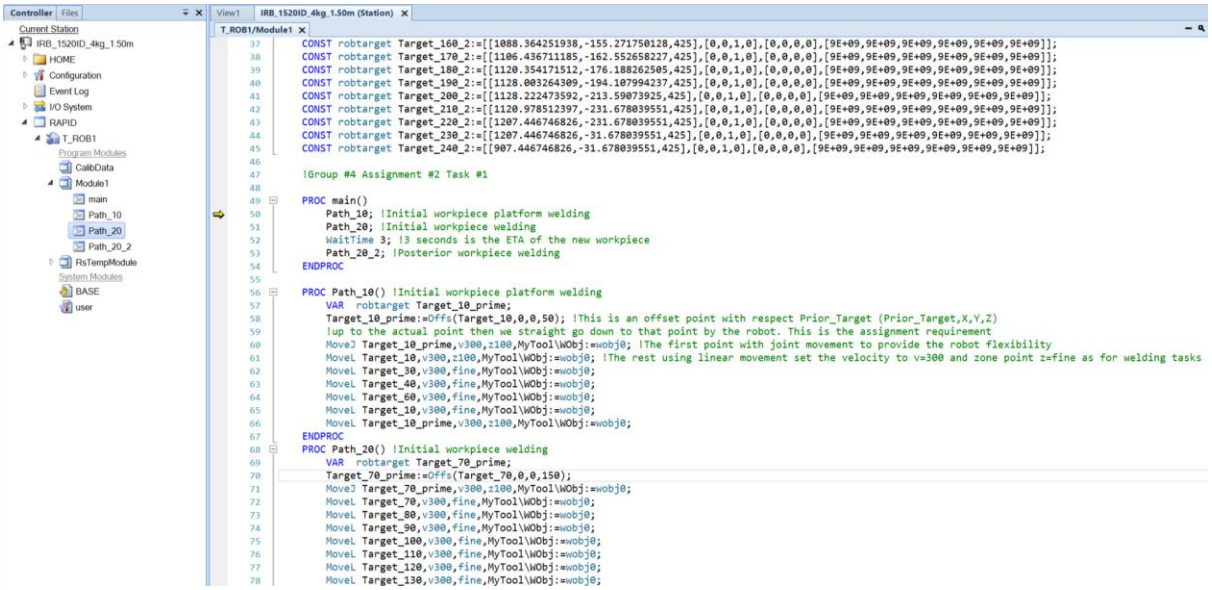
Picture 2. Tree of the workspace showing some targets

Once the targets are created, it is possible to define the paths out of them.
It is remarkable to say that the movements which strictly do not require linear path should be *MOVEJ* in order to ensure that the robot makes an higher performance for its joints. In addition, it is fundamental to indicate a *fine* path planning so the tool does not skip the corners.



Picture 3. Tree of the workspace showing the paths with their correspondent targets

Now that everything is defined and placed, the actions and features of the execution of the robot are transcripted into a RAPID code by means of the *Synchronize* action (RAPID>Synchronize>to RAPID).

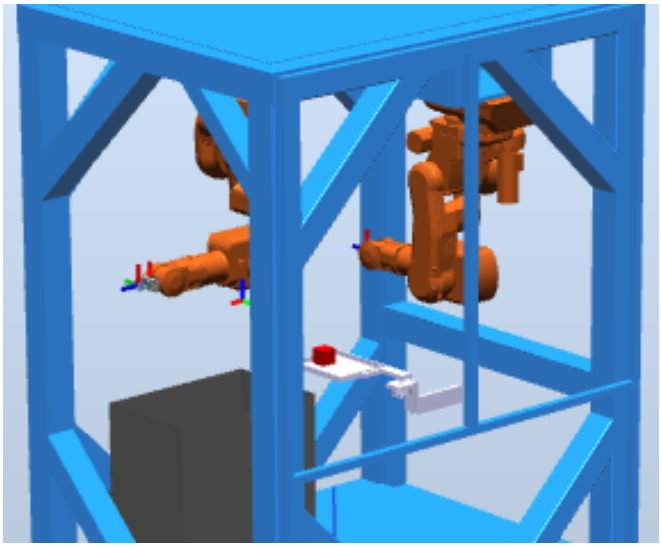


Picture 4. Fragment of the RAPID code

Notice that it is also possible to implement and/or modify actions and features to the model by adding or subtracting lines of code as well as modules. In order to make the new functions scripted visible in the RAPID code, the *Synchronize* action in a reverse way (RAPID>Synchronize>to Station) is a solution.

2. Pick and place

In the second task an ABB IRB 140 robot will execute a Pick and Place routine. We will first off test the robot offline with Robot Studio before testing it in the real environment. The workstation is composed of two ABB IRB 140 robots from which we will only use one. In the model we used the robot one to pick a red workpiece and place it in a basket located under the robot.



Picture 5. The workstation 3D model with the two robots

In the first place, we added the position of the Pick and Place *targets* to the ABB 140 model. Afterwards, we created a *path* and *synchronise* the model to the RAPID code.As a next step we implemented two more targets, pre_pick and pre_place, on top of Pick and Place targets respectively using offsets. The Z value of the offsets are negative because the frames of the targets are with its Z axes pointing downwards as well as the tool frame, and the offset targets are in the opposite direction.

In addition, it is relevant to indicate that the gripper actions work with digital I/O and SetDO RobU_Gripper_Reset,V and SetDO RobU_Gripper_Set,W are mutually excluents, that is, if V=0 then W=1 and viceversa. (U represents the # of the robot)

With regards to the sort of implemented movements, MoveJ is used for long distances between targets in which the trajectory of the path is not constrained The reason for this is that it makes the robot execute the movements with less workload in the joints. MoveL, on the other hand, is used for straight path constraints. It is feasible when the distances are not too long. MoveC is not used in this task though.

In the Path_10 module we find the following piece of code :

```
VAR robtarget pre_pick;
VAR robtarget pre_place;
pre_pick:=Offs(Pick,0,0,-100);
pre_place:=Offs(Place,0,0,-100);
```

Additionally created a function to open and close the griper:

```
PROC closegripper()
        SetDO Rob2_Gripper_Reset,0;
        SetDO Rob2_Gripper_Set,1;
        WaitDO Rob2_Gripper_Set, 1;
```

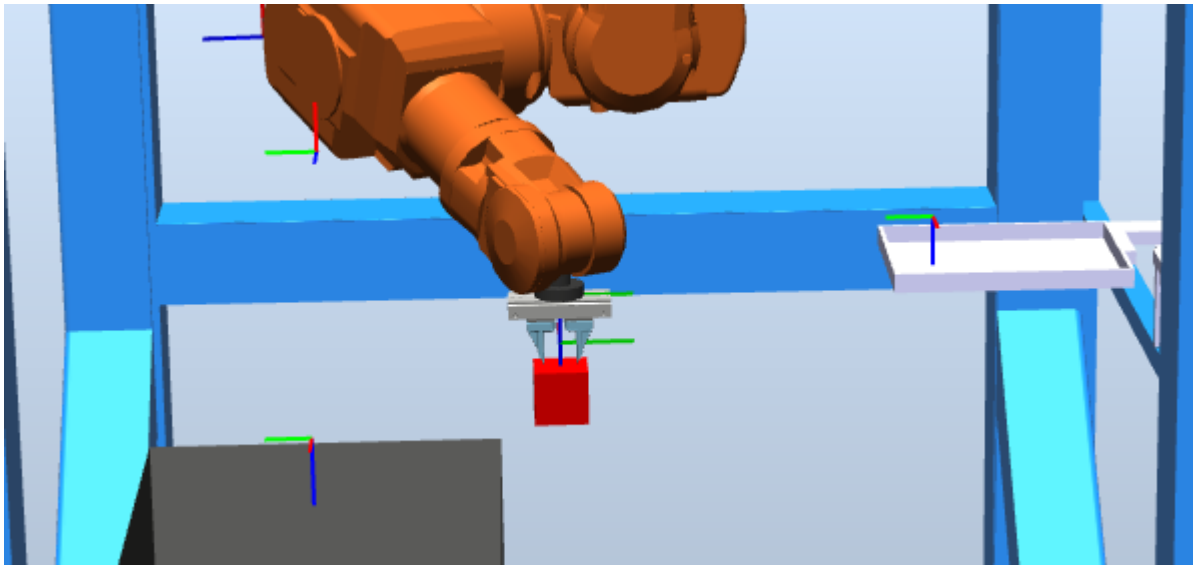
```
        WaitTime 0.5;
ENDPROC
PROC opengripper()
        SetDO Rob2_Gripper_Set,0;
        SetDO Rob2_Gripper_Reset,1;
        WaitDO Rob2_Gripper_Reset, 1;
        WaitTime 0.5;
ENDPROC
```

In the end, we applied the displacement described below in the RAPID code. It moves with linear movements between pre-pick/pre-place and the actual pick and place targets with a speed of V50. Nevertheless, it moves with free joint movement between pre-pick and pre-place targets with a speed of V500. The code for this can be found here :

```
        opengripper;
        !Move to pre_pick
        MoveJ pre_pick,v500,z0,Fingers\WObj:=wobj0;
        !Move linearly to Pick
        MoveL Pick,v50,fine,Fingers\WObj:=wobj0;
        !close gripper
        closegripper;
        !Move back linearly to pre_pick
        MoveL pre_pick,v50,z0,Fingers\WObj:=wobj0;
        !Move to pre_place
        MoveJ pre_place,v500,z0,Fingers\WObj:=wobj0;
        !Move linearly to Place
        MoveL Place,v50,fine,Fingers\WObj:=wobj0;
        !open gripper
        opengripper;
        !Move back linearly to pre_place
        MoveL pre_place,v50,z0,Fingers\WObj:=wobj0;
```

In the MainModule we just needed to put the robot in its home position in the beginning and in the end of the program and execute Path_10. The code is shown below.

```
PROC main()
        !Going to home position
        HomePosition;
        !executing Path_10
        Path_10;
        !Going to home position
        HomePosition;
ENDPROC
```



Picture 6. Frame shot of the robot executing the Path_10 module.

After being sure that the code properly works via simulation we implemented it in the hardware.

Appendix

RAPID code of Task 2 implementation in the hardware

```
MODULE Rob2MainModule

   CONST robtarget Pick:=[[99.00,-
637.37,806.69],[0.987347,0.00167527,0.00119748,0.158564],[-1,1,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
        CONST robtarget Place:=[[393.32,37.64,978.75],[0.999295,-0.00304885,0.0322908,-
0.0189102],[0,2,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];


   PROC main()
      !Going to home position
      HomePosition;
      !executing Path_10
      Path_10;
      !Going to home position
      HomePosition;
   ENDPROC

   PROC HomePosition()
                MoveAbsJ
[[0,0,0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]]\NoEOffs,v200,z50,tool0;
        ENDPROC

!Gripper opening
PROC closegripper()
   SetDO Rob2_Gripper_Reset,0;
   SetDO Rob2_Gripper_Set,1;
   WaitDO Rob2_Gripper_Set, 1;
   WaitTime 0.5;
ENDPROC

!Gripper closing
PROC opengripper()
   SetDO Rob2_Gripper_Set,0;
   SetDO Rob2_Gripper_Reset,1;
   WaitDO Rob2_Gripper_Reset, 1;
   WaitTime 0.5;
ENDPROC

PROC Path_10()
```

```
    !create pre_pick and pre_place positions
    VAR robtarget pre_pick;
    VAR robtarget pre_place;
    pre_pick:=Offs(Pick,0,0,-100);
    pre_place:=Offs(Place,0,0,-100);
    !open gripper
    opengripper;
    !Move to pre_pick
    MoveJ pre_pick,v500,z0,Fingers\WObj:=wobj0;
    !Move linearly to Pick
    MoveL Pick,v50,fine,Fingers\WObj:=wobj0;
    WaitRob \InPos;
    !close gripper
    closegripper;
    !Move back linearly to pre_pick
    MoveL pre_pick,v50,z0,Fingers\WObj:=wobj0;
    !Move to pre_place
    MoveJ pre_place,v500,z0,Fingers\WObj:=wobj0;
    !Move linearly to Place
    MoveL Place,v50,fine,Fingers\WObj:=wobj0;
    WaitRob \InPos;
    !open gripper
    opengripper;
    !Move back linearly to pre_place
    MoveL pre_place,v50,z0,Fingers\WObj:=wobj0;

ENDPROC



ENDMODULE
```