Valtteri Nikkanen	282688
Riku Pekkarinen	267084
Casey Jones	150598721

Robot Manipulators: Assignment 1 report

Assignment worked as an introduction to RobotStudio.

Task 1

In this task we used a 6-DoF ABB robot, IRB 1520 in this case. The task was to generate a program that follows external and internal parts of a given workpiece and “weld” these parts and then move to the side and wait 3 seconds, then presume another identical workpiece is placed on top of the first one and repeat the movements. A few considerations: the torch comes from above when starting the welding at the paths. The torch also has to leave vertically when finished.

We started this task by creating a work object on the corner of the workpiece and then teaching the robot a few target points which were used to create a path the torch moves in.

We had in mind two different methods for the program code when the new piece is brought on top of the first one. We could either repeat the movements with an offset the size of the piece or create a new work object with the same offset size. In this task we used both techniques to familiarize ourselves with these methods. For the outer lines just the offset was used, which is given in the function call, and for the inner lines the new work object was used.

The offset was a quick method, but the new work object reduced code somewhat. The inner piece path has circle shapes which could be done either by creating multiple targets to increase accuracy in these circle motions or use the `MoveC` command, which automatically follows the circle radius with fewer points needed. We used `MoveC` to reduce the number of targets.

The first task went smoothly after the orientations matched. Something we could have worked on more is the orientation of the tool in every target to make it smoother and the position not awkward, as in real life this is very important.

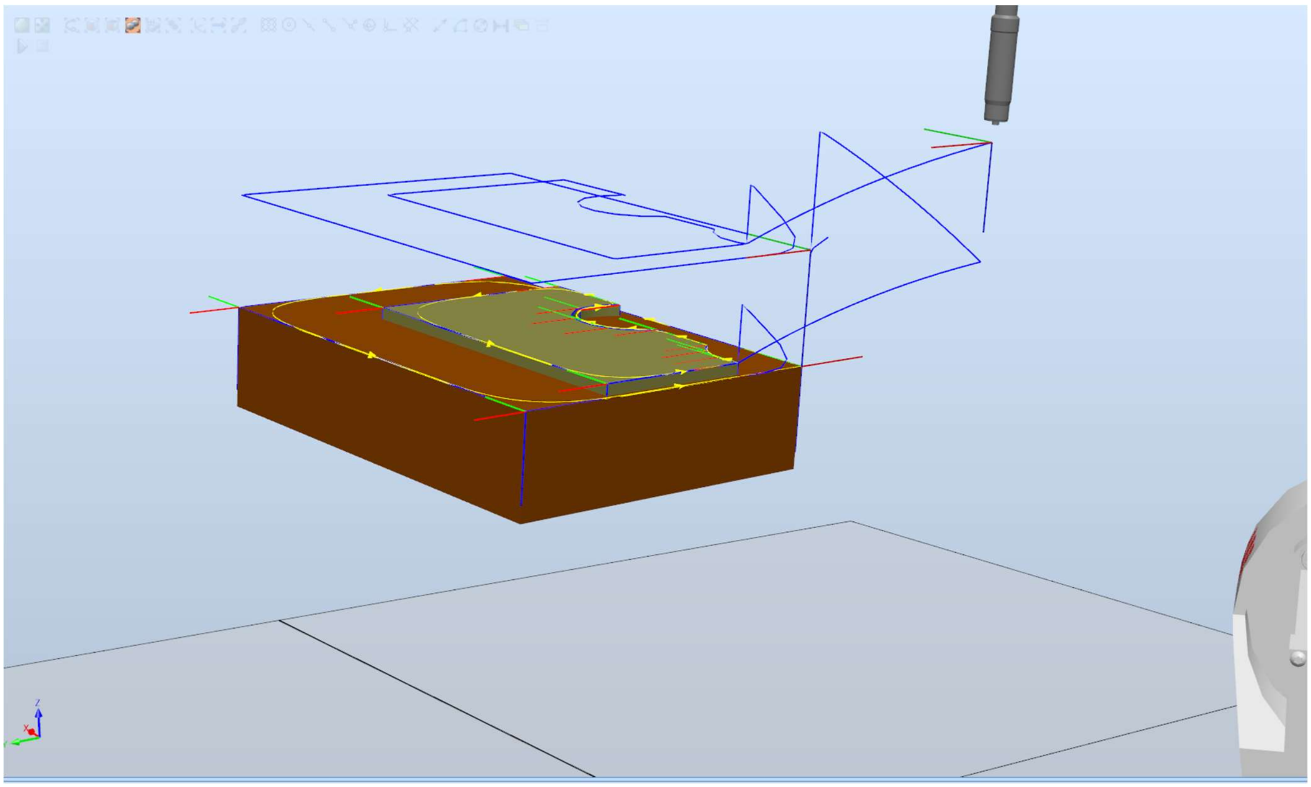

Figure 1: TCP trace of the path of the welding tool.

Task 2

This task required us to code an offline pick-and-place routine for the ABB IRB 140 robot, then upload and play our program on the actual hardware after manually teaching the targets. Our program used four targets: the initial and final positions and two waypoints.

When running our simulation, we ran into some small issues with zones, as well as with the correct sequence of commands for opening and shutting the gripper. We were able to overcome these by reducing the zones to zero and remembering to reset the gripper's opposing actuator before activating the one we want to fire.

Later, when running on the actual hardware, we experienced an issue in which the robot was moving to a point too high above the initial target we had taught. After tweaking various things this problem finally disappeared. We hypothesized that it may have been related to a previously saved work object. However, everyone seemed sure that the points we had taught were relative to our work object, so why this issue was occurring is still a bit of a mystery.

In general, despite these small challenges which we encountered, everything went well. Some logging could have come in handy, for example in helping to overcome our issues related to the gripper. Later we learned that the `ErrLog` command can be used for this purpose. For debugging a more complex program, it could be very helpful to take advantage of this capability.

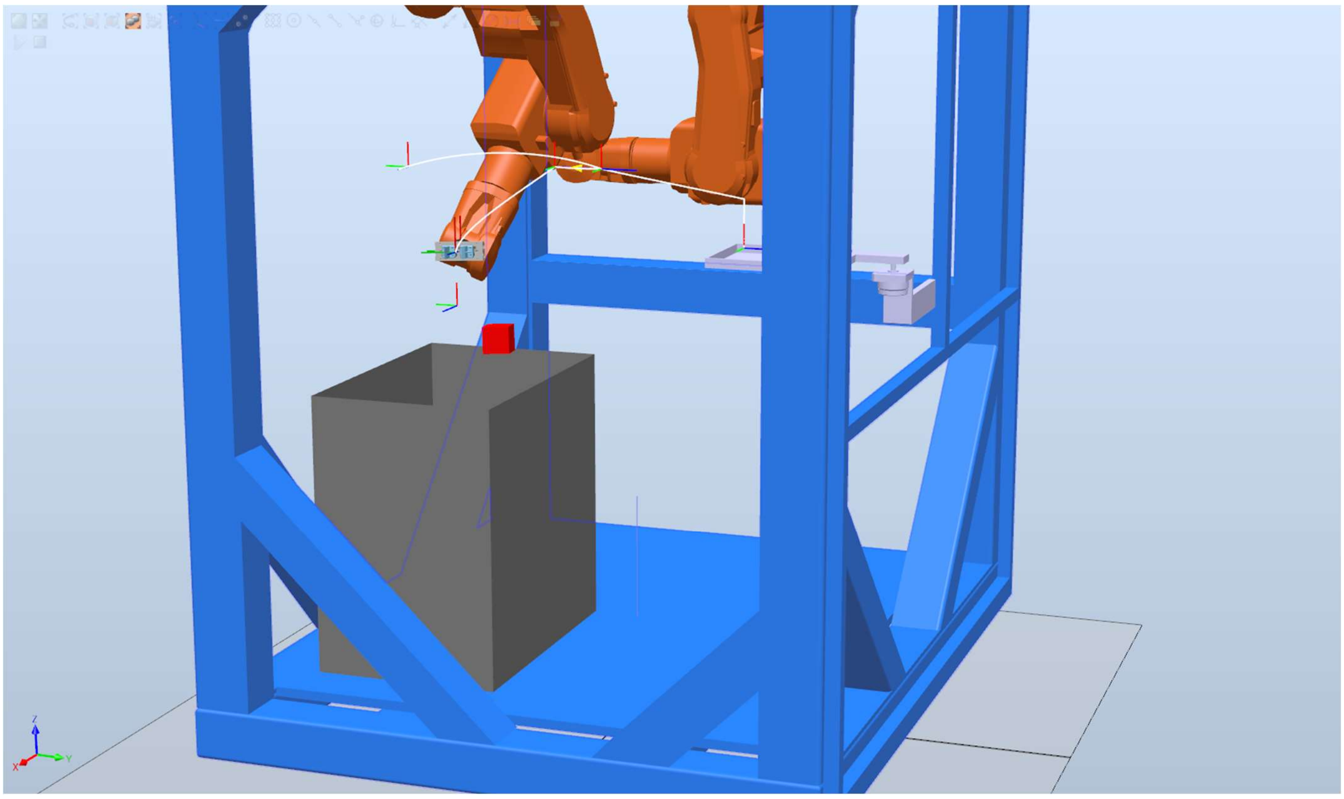

Figure 2: Robot arm moving the grasped cube to the place zone.

Task 3

The problem in Task 3 was to implement palletizing of boxes with an industrial robot. Our way to solve this was to count to six for each box placed and then increment our layer counter, which would then be used to decide where the boxes should be placed. It also had to make sure not to disturb boxes already placed on the pallet. The boxes were placed onto the pallet from above and then pressed into place to avoid knocking other boxes around and to ensure correct box placement. Digital inputs and outputs were used to control the end-effector.

All in all, we are quite happy with the solution that we came up with. There is some room to improve, especially in the way that the code was written. We could have used more functions and passed variables between them to make the code more readable. The use of global variables is also something that we would normally avoid in production, but as this was a small piece of code, we felt it was fine. Possibly reading more about RAPID code documentation could have also given us more insights into how the code could have been executed. Also, we spent some time fiddling with the offset values instead of just using the RobotStudio measurement tool which would have enabled us to just first measure and calculate where the boxes should be placed instead of spending time trying things out in the simulation.

We made one work object for the pallet. In that work object we created two target points in different orientations to the corner of the pallet. These were used to get the box orientation correct. Offsets were then used to get the boxes to the correct positions on the pallet. We could have also made more different work objects and gotten the boxes into the correct positions and orientations that way. We could have also made target points for all the possible box locations, but we felt that that would have required too many target points, which would have made the code quite confusing to read and memory inefficient.

Either way, in the end, our solution does what was required of it quite well.

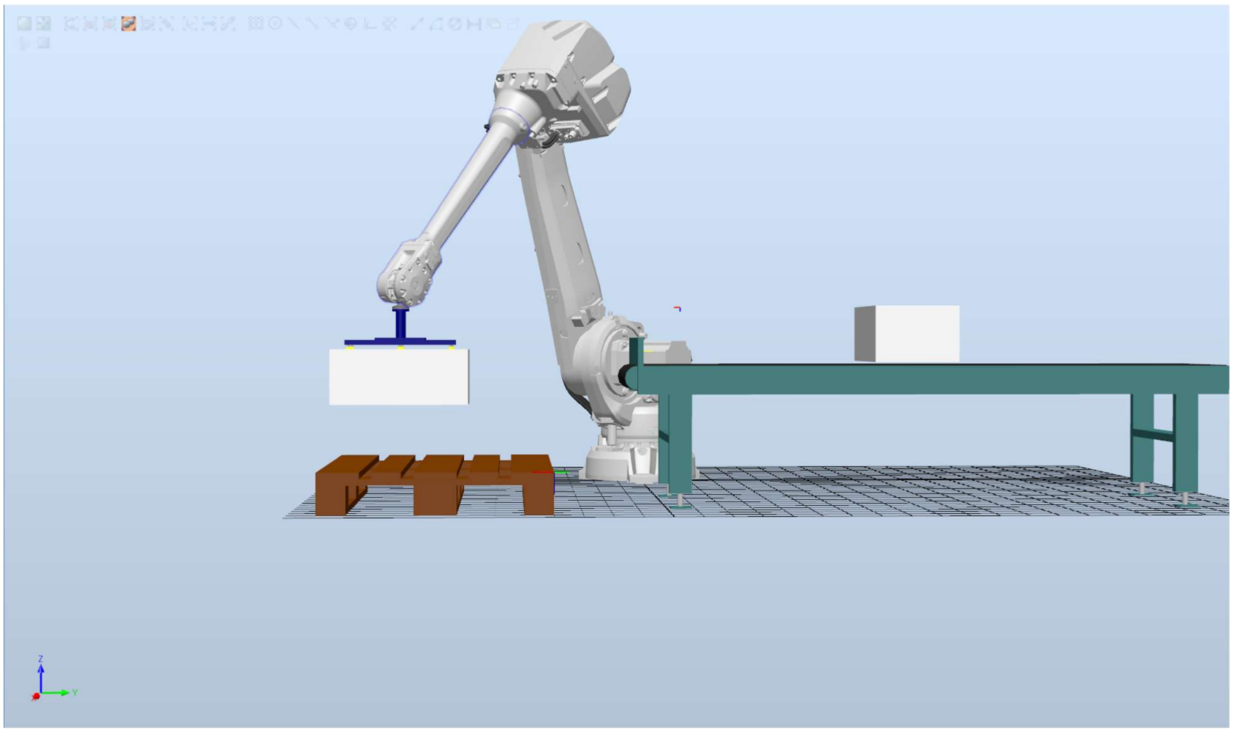
Figure 3: Palletizing robot placing a box onto the pallet.