ME3820 Industrial Robotics Lab

Assigned: March 14th
Due: March 18th at 11:59 pm

Overview

In this lab, you will program a 6-axis industrial robot to pick three blocks from a fixture and stack those blocks to form a structure. You will be provided with the robot, the blocks and the appropriate fixtures. You will be responsible for programming the robot and running your program. The blocks will initially be arranged as in Figure 1 and you must stack them as shown in Figure 2

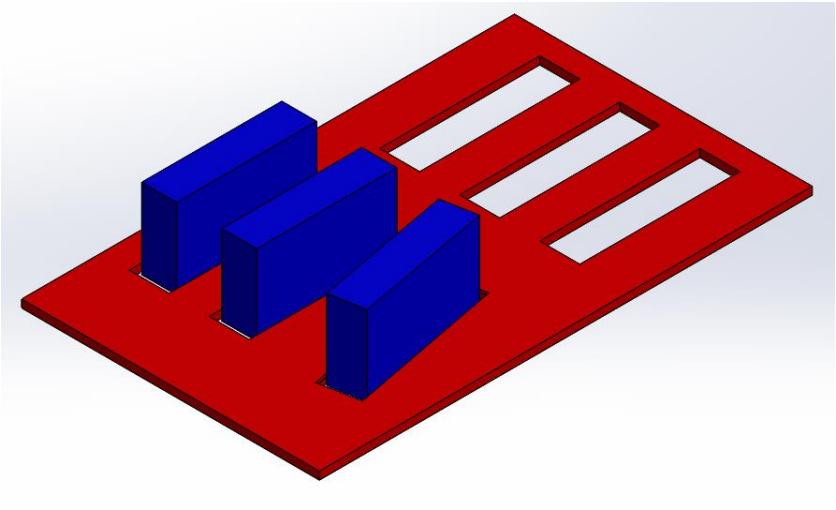


Figure 1: Block Starting Position

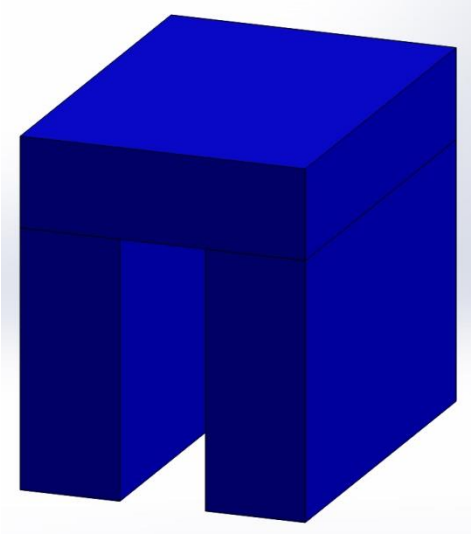


Figure 2: Block Ending Position

Getting Started

When you start this lab, have one of the course TA's/PLA's/Graders help you get started. They will show you how to turn on the robot, how to jog the robot and how to use the robot controller.

Programming the Robot

Create a New Program

The first step is to register a new program. For this lab exercise, create a program and save it on the controller in the folder for your group, which can be found under hd0a/16-63265/ME3820.

- Turn on the robot controller and wait for it to boot up
- Press the ABB logo (A) in the upper left-hand corner of the FlexPendant screen as shown in **Error! Reference source not found.**

Figure 3: ABB Startup Screen

- Select **Program Editor** from the menu of items. There may be a program currently loaded in memory, so you may see something like what is shown in Figure 4**Error! Reference source not found.**:

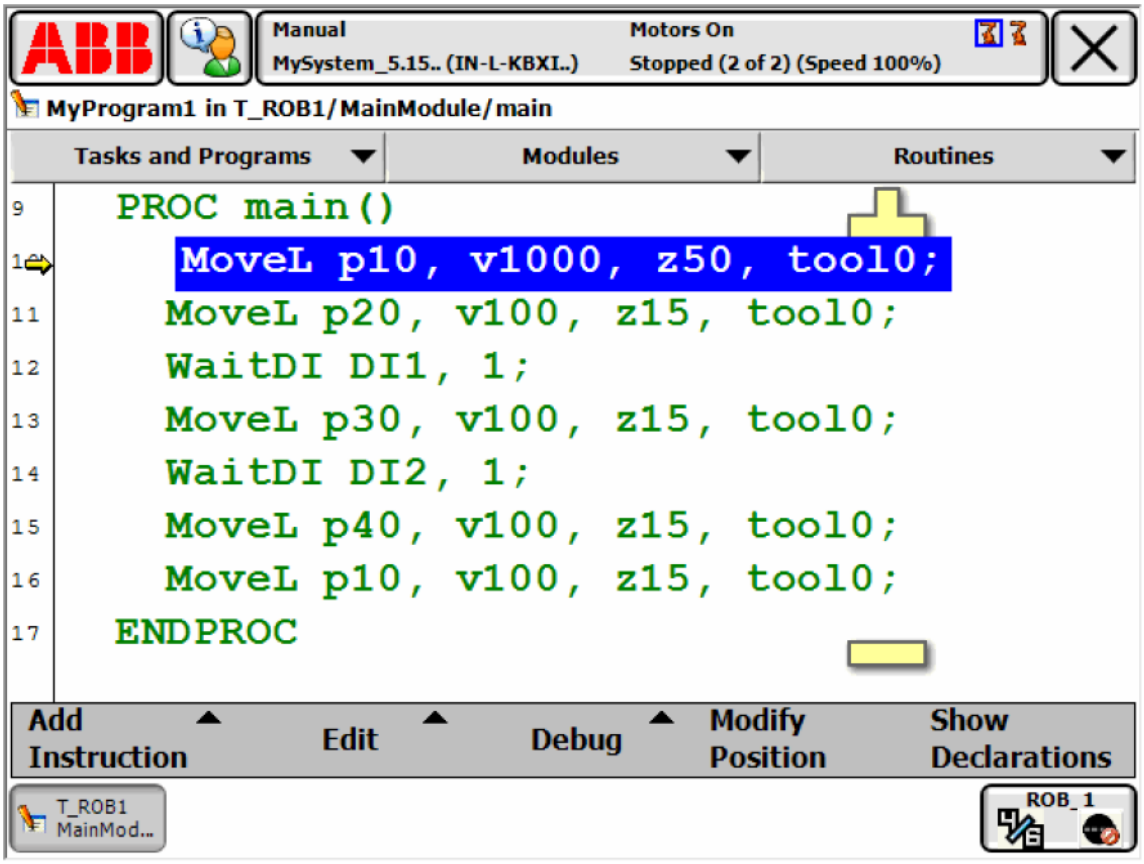


Figure 4: ABB FlexPendant Program Editor

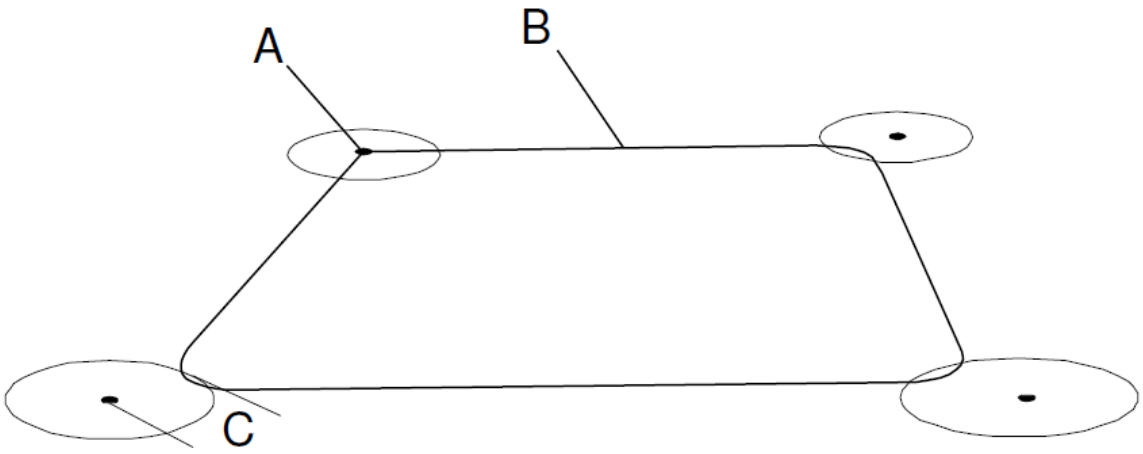
- If the loaded program is the correct one, then go ahead and edit / run it.
- If you want to create a new program, then:
    - Tap **Tasks and Programs**
    - Tap **File**, then **New Program.** If there was already a program loaded, a warning dialog appears:
        - Tap **Save** to save the loaded program
        - Tap **Don't Save** to close the loaded program without saving it (it will be deleted from the program memory)
        - Tap **Cancel** to leave the program loaded
    - Continue by adding instructions, routines, or modules
- If you want to load an existing program, then:
    - Tap **Tasks and Programs**
    - Tap **File**, then **Load Program.** If there was already a program loaded, a warning dialog appears:
        - Tap **Save** to save the loaded program
        - Tap **Don't Save** to close the loaded program without saving it (it will be deleted from the program memory)

        - Tap **Cancel** to leave the program loaded
    - Use the file search tool to locate the program file to be loaded (file type 'pgf'). Then tap **OK**. The program is loaded and the program code is displayed.
    - Continue by adding instructions, routines, or modules
- To save a program, then:
    - Tap **Tasks and Programs**
    - Tap **File** and then select **Save Program As…**
    - Use the suggested program name or tap … to open the soft keyboard and enter a new name. Then tap **OK**.
- To rename a loaded program:
    - Tap **Tasks and Programs**
    - Tap **File** and then select **Rename Program**. A soft keyboard is displayed.
    - Use the soft keyboard to enter the new program name. Then tap **OK**.

Add a New Pose (Target Location)

The easiest way to add a new pose is to manually jog the robot to the desired location and then record it. Make sure that you are in the Program Editor as shown in Figure 4.

Suppose you wanted to add four points in a square as shown in Figure 5. You would start by jogging the robot to the location shown as A. You will then add an instruction that records that point (and would move the robot to A if it were not already located there). You will then jog the robot to the next location and record that point, and so on. The speed with which the robot moves from point A to the next point is indicated by B in Figure 5. It's set to v50 which means the robot will move the TCP at 50 mm/s. Finally, note the circle drawn around each of the target points (the corners of the square). The value shown as C (z50) indicates a circle (or more accurately, a sphere) of radius 50 mm. Unless told otherwise, the robot controller will calculate spline curves to connect one of the straight-line paths to the next such that the robot must come within the specified zone radius of the target. This allows the robot to have smoother movements because it does not need to accelerate and decelerate as much near the target points.



A	First point
B	Robot movement Speed data v50 = speed 50mm/s
C	Zone z50 = (50mm)

Figure 5: Desired Path of the Robot

To create a program to perform the movement depicted in Figure 5, do the following:

- Jog the robot to the desired location
- In the program editor, tap **Add Instruction**
- Tap **MoveL** to insert a linear move instruction
- Repeat the above steps for the remaining locations
- Modify the first and last instructions of the program. Tap z50 in the instruction, tap **Edit** and then **Change Selected** to '**fine**'; then tap **OK**.
- Your program should like the following:

```
Proc main()
  MoveL *, v50, fine, tool0;
  MoveL *, v50, z50, tool0;
  MoveL *, v50, z50, tool0;
  MoveL *, v50, z50, tool0;
  MoveL *, v50, fine, tool0;
End Proc;
```

The pose here should be the same as the first one in order to close the square.

You could simply copy the instruction for the first pose.

Figure 6: Desired Path of the Tool

In the program shown above, notice the difference in the third (zone) argument of the MoveL instructions. In some of the instructions the zone argument is 'fine' whereas in others it is 'z50'. The argument 'fine' tells the robot controller to bring the EOAT to a full stop at the indicated pose before proceeding to the next instruction. If the zone is 'z50' as shown in the other instructions, the robot controller does not need to bring the EOAT to a full stop before proceeding. What happens in that instance is that the TCP need only get with a sphere of the indicated radius (50mm in this case) before proceeding on to execute the next instruction. A spline curve is generated to connect the two paths (the one approaching the indicated point and the one exiting from it).

Modify and Existing Point

You can modify an existing point anytime you want. This procedure describes how to modify positions, either by single-stepping to the positions or jogging. You can use the **Program Editor** or the **Production Window**, the functionality is the same.

- On the ABB menu, tap **Program Editor**
- Stop the program, if running
- Do you want to single-step to the position or jog?
    - If *single-stepping*, step through the program to the position you want to change. Make sure the correct argument is selected.
    - If *jogging*, use the **Jogging** view to make sure that the same work object and tool that are used in the instruction are selected
- Now jog the robot to the new desired location
- When using the jogging method, tap to select the position argument of the instruction you want to change
- In the Program Editor, tap **Modify Position**. In the Production Window, tap **Debug** and then **Modify Position**. A confirmation dialog appears.
- Tap **Modify** to use the new position, **Cancel** to keep the original
- Repeat step 3 through 7 for each position argument you want to change.

About Manual Mode

What is manual mode

In manual mode the manipulator movement is under manual control. The enabling device must be pressed to activate the motors of the manipulator, thereby enabling movement.

The manual mode is used when programming and for program verification. In some robots, there are two manual modes, the manual reduced speed mode and the manual full speed mode. (Note: Our robot does not have the manual full speed mode option.)

Safety in manual mode
When in manual mode the manipulator is operated with personnel in close proximity. Maneuvering an industrial manipulator is potentially dangerous and therefore maneuvers should be performed in a controlled fashion.

What is the manual reduced speed mode?
In manual reduced speed mode the movement is limited to 250 mm/s. The enabling device on the FlexPendant must be pressed to activate the motors of the manipulator.

WARNING

Whenever possible, the manual mode of operation shall be performed
with all persons outside the safeguarded space.

Operating the gripper from the program

You can command the gripper to close or open by controlling a digital I/O line – specifically a digital output line. This has been pre-wired for you in the robot controller; specifically, the solenoid (electric air value) that controls the gripper is D652_10_D01.

To close the gripper you would add the following line of code to your program:

SetDO D652_10_D01, 1;

where D652_10_D01 indicates the particular digital I/O line and the argument 1 indicates that the output should be set to the 'high' (24V) state. To open the gripper you would change the 2nd argument to be 0 (zero). You may also use the keywords 'high' and 'low' instead of 1 and 0.

Note that, unless told otherwise, the robot controller is 'looking ahead' and queuing up the next instruction to be executed as soon as possible after the current instruction has been executed. This presents us with a bit of a problem. Because the gripper takes some non-zero time to actuate, the program may command the EOAT to start moving before the gripper has had a chance to fully close (or open). We therefore need to insert a little delay to ensure that the gripper mechanism has fully closed (or opened) before moving on to the next instruction in the program.

Use the following instruction to insert some delay into your program:

WaitTime XXX;

where XXX is a value in seconds. For example, if you wished to delay program execution for a half-second you would use:

WaitTime 0.5;

Note that for this instruction to be effective the robot must already have come to a stop as a result of the previously executed Move instruction. This is achieved by using 'fine' as the zone argument (see the explanation in Add a New Pose (Target Location)).

A similar problem can occur as the EOAT is approaching a pose where the gripper will be commanded to open. In that case, you also want to use 'fine' as the zone argument to prevent the gripper from opening prematurely.

Running the Program

After you finish the program editing work, you can run it to see the result.

- Generally speaking, you will want to start execution at the very first line of the program. To do this you need to set the program pointer to that line of code.
- Assuming you are in the program editor, tap Debug

Then tap PP to Main; this will set the program pointer (the yellow arrow) to the first executable instruction in the main() routine

DANGER

Make sure that no personnel are in the robot working area.

- Squeeze the enabling device (deadman switch) on the FlexPendant. The amber light on top of the robot arm will illuminate; you will also see the status on the FlexPendant change from **Guard Stop** to **Motors On**
- If you wish for the program to execute, press the **Start** button on the FlexPendant (see E in Figure 7 below)
- The program will begin executing; pressing the **Stop** button (H in Figure 7) will halt program execution.

Debug the Program

You can also run the program one instruction at a time, i.e., single-stepping the instructions. This may be desirable when first executing the program to ensure that it is operating correctly.

- Assuming you are in the program editor, tap **Debug** and then tap **PP to Main**

DANGER

Make sure that no personnel are in the robot working area.

- Squeeze the enabling device (deadman switch) on the FlexPendant
- Press the **Step Forward** button on the FlexPendant (see E in Figure 7). The controller will execute the next instruction at the program pointer and then stop.
- Note that there is also a **Step Backwards** button (see F in Figure 7).

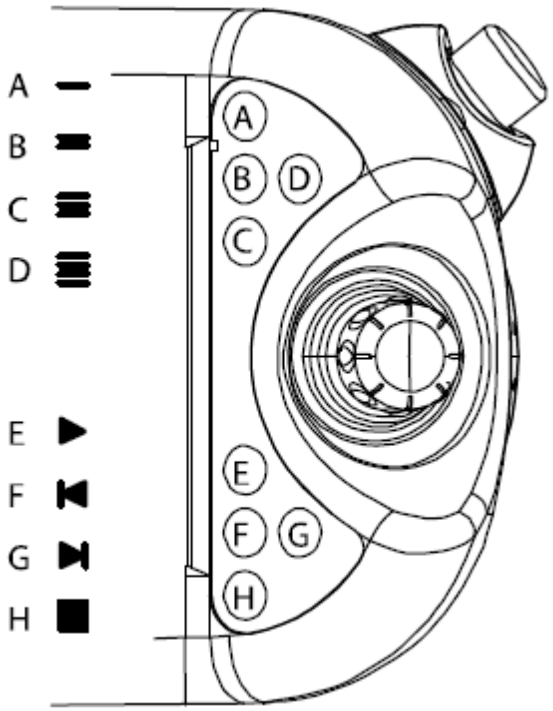


Figure 7: FlexPendant Program Control Buttons

Hints

- Make sure the plate used to hold the blocks is always aligned properly before starting operations. The plate is aligned so that most of the blocks are oriented parallel to the robot's base Y axis. [Use the supplied aluminum block to line things up properly.]
- Most of the block moves can be accomplished by gripping them from the top. Consider getting the gripper oriented vertically and then jog it (mainly) using linear moves in the base coordinate system.
- For some of the blocks, you may need more complex movements. Don't try to accomplish too much with a single move instruction. It's important to remember that, by default, the controller flag errors if you try to rotate an axis more than 90° in a single linear move (MoveL) instruction. Use multiple moves if necessary.

- There may be occasions where it might be more appropriate to use joint moves (MoveJ) instead of linear moves (MoveL). Joint moves are generally less likely to flag errors if you try to accomplish too much in a single move instruction.
- Remember to use the 'fine' zone argument to make sure the EOAT has come to a stop before opening or closing the gripper.
- Keep the speeds low – this will give you more time to react if something isn't right. This is particularly important when approaching the blocks to grip them.
- Don't forget that a zone argument such as 'z50' allows the robot to more smoothly connect multiple moves together – but be careful that 'cutting the corner' doesn't lead to problems. Make sure you always have enough clearance when moving near the other blocks.
- Work on getting the code for moving the first block working correctly. Once you have that written and debugged, you can then copy those lines of code and use them as a template for moving the second block, and so on. Once the lines of code are copied, all you will need to do is select the appropriate (copied) instruction, jog the robot to the correct pose for that instruction, and then click on Modify Position. Then repeat this process for the subsequent poses (moves) for the second block.
- The robot will automatically stop if you jam the gripper and a block against a plate. Depending on how hard the EOAT is jammed, you may have some difficulty in un-jamming the block. First try simply jogging the robot away from the jam. There will most likely be an error you will have to acknowledge before the robot can be moved. If you can't move the EOAT away from the jam, try releasing the block from the gripper – that may make it easier to get the block out. You may have to resort to tapping the plate out of the way in extreme cases.

Report and Submission

Upon completion of the lab, please either show the your program running to one of the TA's/PLA's/graders or submit a video pf you program running to the appropriate assignment on myWPI. Please also complete the post-lab assessment on myWPI.