ENMT482 – Robotics – Assignment (25%)

UR5 Barista (well, part of a barista's job...)

The overall goal of this assignment is to use the UR5 to perform some basic coffee making tasks using the new interchangeable *barist-effector* tools.

Learning outcomes: The point of this assignment is to get some experience applying the transforms you have learned about in class to the real world and operate an industrial robot (without damaging it!).

Information: You will be provided with

1. Coordinates for two identified points on each piece of equipment (in the robot base frame).
2. Coordinates of the buttons, levers etc that you must actuate in local frames attached to each piece of equipment.
3. Coordinates of relevant *barist-effector* points in their local frame (note necessarily the same as robot TCP frame).
4. A model of the workspace in RoboDK including the *barist-effector* tools and UR5 robot to enable simulation and programming. This model will also contain several programs to complete tool pick-up/drop-off from set standoff points. Please use these, but don't modify them as you may damage the expensive tools. **

Note: There will be some things that you may need to measure or determine yourselves, e.g. how far to pull the lever on the grinder, how far to press a button, orientation of *barist-effector* required to complete a task, etc.

Tasks: You need to complete to following

1. Determine the transformation matrices between the coffee making equipment (coffee machine, grinder, tool stand, cup) and the robot base frame (=world frame).
2. Determine the tool-transformation matrices between the *barist-effector* and the robot's *tool centre point* (TCP).
3. Program the robot to complete <u>at minimum the following series of tasks</u> without colliding with itself or any other equipment within its workspace:
    a. Place Silvia (portafilter) tool under the Silvia grinder dosing head.
    b. Turn the grinder on, wait 3s, turn the grinder off.
    c. Pull the grinder dosing lever to deposit ground coffee in the porta-filter.
    d. Return to home position.
4. You can choose to complete 2-3 additional, more advanced tasks for additional marks (2-3 marks), for example:
    a. Picking up a coffee cup
    b. Placing a coffee cup in the correct location for the coffee machine
    c. Actually making Rodney a coffee with the setup...
    d. Additional tasks that you might discuss with us.
5. Demonstrate your program working on the physical robot.

Notes on the demonstration:

- Failure to successfully compile your code will result in 0 marks for this element of the assignment and no further opportunity to demonstrate – make sure your code works before you come to the demo!
- The task must be finished within 5 mins
- Penalties will be applied for any collisions

Demo day: To be confirmed

Resources: In addition to the simulation environment, each group will have up to three 20-minute slots on the UR5 for testing your code on the real platform. These sessions will be available during afternoons (14:00-17:00) of 8-19 October. There will be a sign-up sheet in the lab. Outside of these hours, if Rodney is in the lab and available to help, you can also test your code on the equipment.

Format: This task is completed in groups of two. However, each person must submit an individual report. Reports are to be submitted via Learn by 5pm Friday 19 October.

Report/project guidelines

For the UR5 report, the following are some guidelines on what to include, how it will be marked etc. While I won't impose a strict page limit, the report should probably be less than 8 pages, not including code in Appendix (This isn't a design report!)

Your report should include

- Introduction
- Frame assignment for the equipment, tools, and the robot (all joints + base) and the corresponding D-H tables (you don't need to use the DH parameters to complete the task, but I want to see them in the report). Remember to use appropriate referencing for sources.
- Determination of the transformation matrices between the equipment frames, tool frames, and the robot frame.
- Path planning and obstacle avoidance
- What you learned and what you suggest
- Code in an Appendix

Within this framework, you need to describe your methodology, why you chose it, and maybe reflect on the outcome/performance. You should also briefly explain things like the frame assignment and why it is important and how it relates to this assignment and the software running the robot, etc.

As you are all just about ready to go off into the world as professional engineers who will need to write reports for clients, colleagues etc, quality of writing is also very important. Spend some time making sure your report is concise and well written as this will also impact your mark.

Marking: You will be marked on both your report and the test run of your code. An approximate marking guide (out of 10) is given below:

10: Report is exceptionally well written and presented, and demonstrates excellent insight and effort on aspects such as optimising paths or use of certain commands to achieve rapid/optimal/fancy behaviour. Demonstration includes 2-3 advanced tasks in addition to the required tasks. Code is clever/cunning/elegant and demonstrates excellent knowledge of the robot and its programming environment. Code is well commented and easy to understand. Basically, going above and beyond…

7: Report is well written. Covers all the basic elements and shows a good degree of understanding of how to use the techniques taught in class (frame assignment, DH parameters etc) Good use of figures to illustrate aspects of the report. Code works and performs the minimum set of actions (task 3). Demonstration completes all required tasks, but none of the additional tasks. Code is commented and its flow/function can be easily understood.

5: Poorly written report, code works for required tasks, but is very basic.

2: Poorly written report. Code doesn't work

** Notes on programming:

In the RDK file *UR5 Robot Coffee Cart v1.rdk* there are several programs to perform final approach and tool attach/detach. You just need to get the robot to/from the appropriate point named *<Tool> Tool Entry* (e.g. Lux Tool Entry) and then call the required program (e.g. *Lux Tool Attach, Lux Tool Detach*). These entry points already exist in the .rdk file as targets.

YOU MUST NOT MODIFY THE PROGRAMS OR TARGETS ALREADY IN THE FILE!! Doing so may cause the robot to collide with the tools/equipment, resulting in very expensive damage.

I have added an example program, *Main*, that calls a couple of these sub-programs so you can see how to do it.

Any questions, please ask Rodney or myself.