MTRN4230 Robotics Asst 2 Description S2 2018 Version 1 - 2018-08-14

Robot System Integration

Learning Outcomes

- 1. Learn about a robot environment and put it to use effectively and efficiently on a given task.
- 3. Implement good safety practices in the use of robots
- 4. Apply and evaluate image processing techniques in robotics
- 5. Apply engineering management and technical tools fluently and systematically

Aims

In your project groups, write a user interface for the Robot System in J18-213 which enables a user to drive the robot manually via keyboard or mouse or external controller to any possible configuration. Interface with the camera to allow point and click operation. In addition, all possible errors should be handled efficiently and safely. The user interface should be able to switch between controlling the real Robot System and a simulated Robot System constructed in RobotStudio and also allow the simulated Robot System to reflect the state of the real Robot System.

Create your own module named 'Group_X_Asst2', where your group number is visible in Appendix A and is also visible under 'Project groups' on Moodle. Make sure you delete all your own code from the real Robot Controller at the conclusion of each lab session.

Required Tasks

No:	Pts	Task
1.0		**RobotStudio (/4)**
1.1		Define and display in Robotstudio with correct orientation:
1.1.1	0.25	• World frame
1.1.2	0.25	• Base frame
1.1.3	0.25	• End effector frame
1.1.4	0.25	• Table camera frame
1.1.5	0.25	• Conveyor camera frame
1.1.6	0.25	• Table home frame
1.1.7	0.25	• Conveyor home frame
1.2	1.5	Display a recognisable CAD model of the Robot Cell and surrounding area. Include robot, walls, safety switches, light curtain, conveyor, tables, computer, flex pendant, conveyor panel.
1.3	0.25	CAD model is coloured.
1.4	0.5	Move robot in simulation through the Matlab GUI when not connected to the real Robot System.
2.0		**GUI:Startup and shutdown (/2.5)**
2.1	0.5	At the start of the program. Display all the safety information contained in the SWP and prompt the user for confirmation. Guide the user through the steps to operate the robot.
2.2		Provide a method to shutdown the robot:
2.2.1	0.5	• Programmatically move the robot to home position
2.2.2	1	• Avoid collision with the table.
2.2.3	0.5	• Programmatically set all Digital Outputs (DOs) to 0
3.0		**GUI:Robot status (/3)**
3.1	0.5	Give a live update on the status of the robot.
3.2	0.5	Display the current robot joint angles.
3.3	0.5	Display the current end effector position.
3.4	0.5	Display the current end effector orientation.
3.5	0.5	Display a list of messages sent to the robot system in a human readable format.
3.6	0.5	Display a list of messages received from the robot system in a human readable format.
4.0		**Safety system (/4)**
		Provide diagnostics on the GUI for all the conditions that prevent the robot from operating. The diagnostics need to be displayed until the problem have been resolved. Prevent move commands from being sent to the robot until the problem has been resolved.
4.1	0.5	• Emergency stop
4.2	0.5	• Light curtain
4.3	0.5	• Motors are off
4.4	0.5	• Hold to enable not pressed
4,5	0.5	• Motion task not running/ Execution Error

4.6	0.5	• Motion supervision triggered (Alternatively prevent the robot from being able to collide with the table).
4.7	0.5	• Conveyor is not enabled when trying to use the conveyor.
4.8	0.5	Set ConRun to 0 if ConStat is set to 0 and gracefully handle restarting of the conveyor.
5.0		**DIO (/2.5)**
		Display the status of Digital Input/Output (DIO) and enable control of DIO from GUI through RAPID
5.1	0.5	• Control vacuum pump
5.2	0.5	• Control vacuum solenoid
5.3	0.5	• Conveyor run
5.4	0.5	• Conveyor direction
5.5	0.5	• Conveyor enabled
6.0		**GUI:Video feeds (/6)**
6.1	0.5	Display a live video feed from the conveyor camera (min refresh rate: 1Hz).
6.1.1	1	• Outline the edge of any boxes.
6.1.2	1	• Visually indicate the orientation of any boxes.
6.2	0.5	Display a live video feed from the table camera (min refresh rate: 1Hz).
6.2.1	1	• Outline the edge of all visible blocks on the table video feed.
6.2.2	0.5	• Visually indicate the orientation of all visible blocks on the table video feed.
6.2.3	1	• Indicate which letter is visible on the block.
6.2.4	0.5	• Indicate if any block is unreachable
7.0		**GUI:Pose (/6.5)**
		Able to move the robot to a pose using matlab. Need to be accurate to within 5 mm for full marks.
7.1	1	• Move the end effector to a specified position relative to table home.
7.2	1	• Move the end effector to a specified position relative to conveyor home.
7.3	1	• Move to a specified pose (set of joint angles).
7.4	1	• Reorient the end effector without changing the position of the tip.
7.5	1	• Click on a position on the table video feed and move the end effector to that position (pointing down).
7.6	1	• Click on a position on the conveyor video feed and move the end effector to that position (pointing down).
7.7	0.5	• Able to move at different speeds.
8.0		**GUI:Jogging (/4.5)**
		Able to move the robot continuously using matlab with continuous human input. Eg: using mouse, keyboard or xbox controller. Motion need to be smooth for full marks.
8.1	1	• Move the end effector in linear mode with respect to the base frame.
8.2	1	• Move the end effector in linear mode with respect to the end effector frame.
8.3	1	• Reorient the end effector without changing the position of the tip.
8.4	1	• Jog any of the joints.
8.5	0.5	• Able to move at different speeds.
9.0		**RAPID:Motion (/6)**
		Able to move the robot to a pose using matlab, given a single command.
9.1	1	• Move the end effector to a specified position relative to table home.
9.2	1	• Move the end effector to a specified position relative to conveyor home.
9.3	1	• Move to a specified pose (set of joint angles).
9.4	1	• Reorient the end effector.
9.5	1.5	• Move in linear motion mode.
9.6	0.5	• Able to move at different speeds.
10.0		**GUI:Pause and resume (/1.5)**
10.1	0.5	• Pause the current motion task, stop the robot from moving.
10.2	0.5	• Resume the current motion task, continue the previous move command.
10.3	0.5	• Cancel the current motion task, stop the robot, prepare for new move command.
11.0		**RAPID:Pause and resume (/3)**
11.1	1	• Pause the current motion task, stop the robot from moving.
11.2	1	• Resume the current motion task, continue the previous move command.

11.3	1	• Cancel the current motion task, stop the robot, prepare for new move command.
12.0		**Error handling (/4)**
12.1	1	Handle all communication errors in RAPID without needing to restart tasks.
12.2	1	Handle all communication errors in Matlab without needing to restart the GUI (including incorrect IP address, socket timeout, unable to connect to the robot).
12.3	1	Advise the user if a move command is unreachable.
12.4	1	Advise the user if network connection is lost.
13.0		**Documentation (/5)**
13.1	1	Evidence of proper code version management. Showing each student's contribution. git commit log
		All code must be well structured and documented. Use appropriate variable and function names.
13.2	1	• Matlab – Include usage example and help command.
13.3	1	• RAPID
13.4	1	Document the communication protocol used to communicate between Matlab and RAPID. The structure of the message, what each field represents, valid value.
13.5	1	List of tasks assigned to each member and indicate percentage completion.
14.0		**Testing (/7.5)**
		Develop and document a test plan (to show to your demonstrator) and write test functions for
14.1	1	• **GUI:Robot status** (Hint: Jog robot to a position, check the display).
14.2	1	• **GUI:Video feeds** (Hint: Check block are displayed correctly).
14.3	1.5	• **Communication protocol** (Hint: print human readable information on flex pendent and matlab).
14.4	3	• **GUI:Pose and RAPID:Motion** (Hint: Do a move command and check the robot position).
14.5	1	• **DIO** (Hint: Test DIO).
		The test plan should include a number of test cases with procedure, required human input, and expected outputs. Testing should be easy to use and automated when possible. All requirements in each section need to be tested. More than one test case per requirement may be requested.

Total: 60 points scaled to 20 marks for the course.

Assessment and Due Date

Assessment will be by way of a demonstration during your lab timeslot in week 9. The demonstration must be completed by 5 min before the end of your time slot. No late submissions will accepted without special consideration (as discussed in the course outline). In addition, a zip file of all your code (RAPID, Matlab and anything else) and documentation (Tasks 13 and 14), named 'Group_X_Asst2_Submission', must be uploaded to Moodle by 11:55 pm Friday 21 September (week 9). You will also be required to complete a team evaluation, rating and commenting on the relative performance of your group members. This will be due at the same time and will be used to adjust your individual contribution relative to that of your peers.

Resources

1) http://arvc.umh.es/arte/practicals/pr5/practice5_english.pdf Page 3 of this document shows an example GUI for manual control of the robot. Other resources are also available from this site: http://arvc.umh.es/arte/index_en.html.
2) https://www.youtube.com/watch?feature=player_detailpage&v=VnFIpp1W0Vk#t=63 This video shows a very neat GUI for manual control of a simulated robot. The 3D simulation itself is NOT required in this assignment.
3) The folder of ABB Robot Manuals on the Moodle course homepage, in particular:
   The RAPID Reference Manual
   RobotStudio Operating Manual
   Section 8 on page 57 of the Robot Communication and I/O Control Application Manual
4) The RobotStudio Developer Centre
5) YouTube contains many unofficial RobotStudio tutorials, although many of these deal with older versions of RobotStudio. A RobotStudio Getting Started tutorial is available here.
6) Your demonstrators.
7) The Moodle discussion forum

Unless you decide to implement your GUI in C#, you should not need the RobotStudio SDK as all communication can be achieved through TCP/IP Socket programming.

It is anticipated that most solutions will utilise Matlab for the GUI and socket communication.

Mark Whitty and Zhihao Zhang

Appendix A - Group Numbers (also shown on Moodle)

my.UNSW Class	Lab Date and Time	Lab group number
6242	Fri 10:30 - 12:30	14
6243	Fri 12:30 - 14:30	15
6244	Fri 14:30 - 16:30	16
6245	Fri 16:30 - 18:30	17
6254	Mon 18:00 - 20:00	1
6247	Thu 10:30 - 12:30	10
6248	Thu 12:30 - 14:30	11
6249	Thu 14:30 - 16:30	12
6250	Thu 16:30 - 18:30	13
6258	Tue 12:30 - 14:30	2
6259	Tue 14:30 - 16:30	3
6260	Tue 16:30 - 18:30	4
6261	Tue 18:30 - 20:30	5
6262	Wed 10:30 - 12:30	6
6263	Wed 12:30 - 14:30	7
6264	Wed 14:30 - 16:30	8
6265	Wed 16:30 - 18:30	9