UNSW
THE UNIVERSITY OF NEW SOUTH WALES

MTRN4230

Robotics

Assignment 3: Full System Demo

Group 15

Fraser Hamersley	z3416362
Daniel Thomas Cameron	z3414860
Hanxian Yu	z5060576
Hethu Venkata Siva Koushik	z5122812
Steven He	z5018243
Nur Liyana Othman	z5089623

Table of Contents

Table of Figures

1.0 Task Distribution

Task has been distributed among the group member accordingly aiming to complete the assignment requirement for a complete system demo. Each group members use slack as a communication medium and submit task progress through the group Github. Figure 1 shows the git commit log management. It indicates the contribution of each team member in the Github in progress of the assignment completion. Each group member uses a unique username which are:

Name	Username
Steven	GuineaBrown
Fraser	fhamersley
Nur Liyana	ONana20
Daniel	dtcameron
Koushik	koushikmaturi
Dennis	MeteroSF

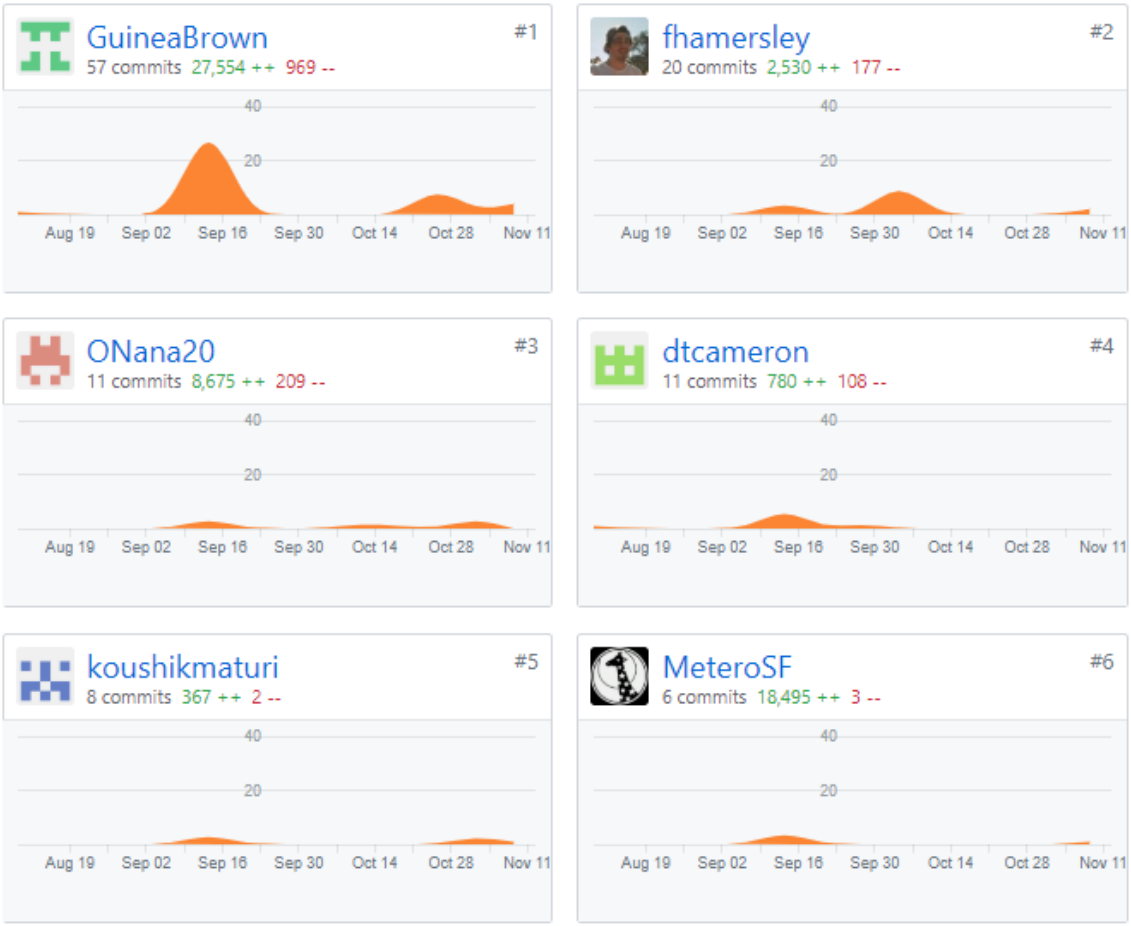

Figure 1: GitHub Management

2.0 Matlab code

This section shows Matlab code used to achieve the required task in this assignment. The Matlab task has been updated for the function listed below:

- Communication with RAPID
- GUI and movement
- Vision
- Path planning

2.1 Communication with RAPID

2.1.1 Command type

Status Check	(C)
Status Toggle	(G)
Change DIO	(D)
Toggle DIO	(T)
Set Pose	(P)
Jog Joint	(J)
Jog Linear	(L)
Pause Motion	(A)
Resume	(R)
Stop	(H)

2.1.1.1 Movement

The input movement data detailed is in section 2.1.2.

- Pose
  - specify a position in both con/table frames to move to (P)
  - specify a position of joint angles (P)
  - specify a end effector orientation (P)
  - click on a position in both feeds to move the end effector there (P)
  - speed set

- Motion
  - specify a position in both con/table frames to move to (P)
  - specify a pose of joint angles (P)
  - specify a end effector orientation (P)
  - reorient

- Jogging
    - manually move the robot in linear mode respect to base fram (L)
    - manually move the robot in linear mode respect to end effect (L)
    - reorient the end effector (L)
    - jog the joints (J)

2.1.1.2 Change status

The input Status Data is detailed in Section 2.1.2.

- Shutdown
    - move robot to home position, avoid collision, set all Dos
- pause
    - stop current command but remember what it is doing
- Resume
    - resume command
- Cancel
    - stop current command, remove commands from the queue

2.1.1.3 DIO change

The input Status Data is detailed in Section 2.1.2.

- VacSol
    - Solenoid starts the suck
- VacPump
    - Engages pre suck
- ConRun
    - Makes conveyer go
- ConDir
    - Conveyer Direction

2.1.1.4 Poll for status (No extra input)

- safety systems
- joint positions
- DIO

2.1.2 Command Input

This section details all the inputs required to give to the MATLAB functions, which will then be sent to RobotStudio. All inputs are listed in order, with their type specified and units.

When inputting, please put in the order specified below. Example string to send would look like:

EXAMPLE: Sending pose changing from MATLAB -> Robot (P) with velocity 100:

BYTE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	‘F’	‘P’	‘ d’	Pi/4		Pi/4		..		..		..		..	
	HEAD	TYPE	VEL	1		2		3		4		5		6	

2.1.2.1 Movement Inputs and Formats/Units

POSE CHANGING (P)

INPUTS	TYPE	SIZE	SEND TYPE	SIZE	Comments
Velocity (v100)	int	4	char	1	int converted to char
Joint Angles (radians)	float	4	2 chars	2	-6 floats converted to 6ints converted to 6 chars - rad * /(2*pi)*(2^16-1)) - reconvert on RS side

JOINT JOGGING (J)

INPUTS	TYPE	SIZE	SEND TYPE	SIZE	Comments
Velocity (v100)	int	4	char	1	int converted to char
Joint Number (radians)	int	4	char	1	int (as is)

LINEAR JOGGING (L)
No Inputs Required

JOG STOP (S)
No Inputs Required

2.1.2.2 Change Status Inputs

CHANGE STATUS (G)

INPUTS	TYPE	SIZE	SEND TYPE	SIZE	Comments
StatusType	string	x	string	4	string

Example: paused

2.1.2.3 Change DIO inputs (D)

INPUTS	TYPE	SIZE	SEND TYPE	SIZE	Comments
DIOType	string	x	string	4	string
BIOBinary			binary		

Example: VacSol
Example: BIOBinary

2.1.2.4 Status Check

No Inputs

2.1.3 Packet Protocol

For more information on the various types of inputs and how to structure this, refer to section 2.1.2.

BYTE	DESCRIPTION	VALUE	COMMENTS
1	Header	'F'	
2	Command Type	Status Check (C) Status Change (H) Set Pose (P) Jog Joint (J) Jog Linear (L) Jog Stop (S) Set DIO (D)	

	Data		See section 2.1.2 for things being sent from each type

2.1.4 Communications Method Protocol

This section details the communication interface for the Robot Studio which is the server with the MATLAB, which is the client. The method purpose was to send a command from MATLAB to the Robot Studio, Robot Studio then should receive the command and reply to the MATLAB.

1) MATLAB Send Command
- MATLAB send the first command.
- Run the internal timer to time out if there is any problem detected.

2) Robot Studio Reply
- Robot studio send confirmation bit checked.
- Execute the command.

3) MATLAB
- Matlab wait for command to finish.

4) Robot Studio
- Movement confirmation when done.
- Awaiting command to be TRUE.

2.1.5 MATLAB Function List

1) Movement Functions

- string cmd = setJointPose(int J1, J2, J3, J4, J5, J6)
  (times Joints angles by 1000)

- string cmd = startJointJogging(int joint, int vel)

- string cmd = stopJointJogging()

- string cmd = startLinJogging()

2) Pose Functions

- function SendPositionCommandButtonPushed(app, event)
    - This function is called when the pose required to be calculated for base frame and the target position co-ordinates are given w.r.t Table Conveyer home. The function uses transformation matrices to calculate pose and use inverse kinematics to calculate the required joint angles. The GUI also has a toggle switch to reorient the end effector which is also done through this function.

- function SendPoseCommandButtonPushed(app, event)
    - This is a basic input function where the joint angles input by the user are normalized and sent to the queue for the robot.

- function [x,y,z,roll,pitch,yaw] = EndEffectorPose()
    - This function is used to display the real-time position and orientation of the end-effector based on the joint angle data being received from the robot.

3) DIO Functions

- function string cmd = changeDIO(binaryStr DIOBinary)
    - Input is a binary string detailing on/off configurations of DOs
    - List of DIO:
      VacSol (1)
      VacPump (2)
      ConRun (3)
      ConDir (4)
    - For example:
      usage: changeDIO('0001')

- function string cmd = toggleDIO(string DIOType)
    - Sends DIO in question to change, creates a toggle command
      VacSol (1)
      VacPump (2)
      ConRun (3)
      ConDir (4)
    - For example:
    - usage: toggleDIO('VacSol')

4) Status Functions

- function string cmd = changeStatus(string statusType)
    - Sends status to change and new state of that status

5) Queue Functions

- function queue = sendToQueue(string[] queue, string cmd)
  - appends a command (string cmd) to the queue (string[] queue)
  - for example: "WOW"

- function queue = clearQueue(string[] queue)
  - cancels all commands in the queue

- function queue = removeFromQueue(string[] queue, int cmdInd)
  - remove from queue

6) Communications Functions

- function startConnection(IP)
  - opens a TCPIP socket to robot located at IP address 'IP'

- function prioritySendToRobot(socket, string cmd)

- function closeConnection(socket)
  - closes TCPIP socket

- function sendFromQueue(string[] queue, socket)

- function [status, safety, DIO, joints] = processMessage(string reply)
  - decodes the message coming back from RS
  - gives a status struct, binary safety and DIO, joints in raw data form

- function SafetySystem(app,safety)
  - displays error message and turns error lamp to red

- function joints = decodeJoints(char jointData)
  - decodes the joint data from char
  - returns a 1 x 6 array of floats

7) Helper Functions

- function string cmd = createCmdString(char cmdType, string data)
  - for creating communications strings

2.2 GUI and Movement

GUI is used to accomplish the required task of playing noughts and crosses between 2 human players on the central 3x3 section of the grid. Figure 2 shows the created GUI, where it enables the game to be played via AI or via Human. In addition, the movement can be paused by pressing the Pause button on the GUI, which will prevent user from making further move. The movement also can be resume by pressing the Resume button on the GUI and the game can be ended by pressing the End Game button on the GUI. This will indicate that the game is ready for a new game, any movement command will be cancelled and all blocks on the table will be packed back into the correct decks.

The blocks are indicated by green and red colors based on their availability. Once a move is played a particular block, it turns red and is disabled. So, that this block cannot be played in later.

Figure 2: GUI for Noughts and Crosses/ Tic-tac-toe game

2.3 Vision

- function blocks = detectBlocks(img)
    - Function call to detect a block

2.4 Path planning

- function ReadMe.txt
    - Contains List of Files and description

- function Astar_tutorial.pdf
    - Description of A* algorithm, examples

- function A_star1.m
    - This is the main file that contains the algorithm.
    - This needs to be executed to run the program.

- function dist = distance(x1,y1,x2,y2)
    - This is a function that calculates the distance between 2 nodes.

- function i_min = min_fn(OPEN,OPEN_COUNT,xTarget,yTarget)
    - This function takes the list OPEN as one of its arguments and returns the node with the smallest cost function.

- function n_index = node_index(OPEN,xval,yval)
    - This function returns the index of the location of a node in the list OPEN.

- function exp_array=expand_array(node_x,node_y,hn,xTarget,yTarget,CLOSED,MAX_X,MAX_Y)
    - This function takes a node and returns the expanded list of successors, with the calculated fn values. One of the criteria being none of the successors are on the CLOSED list.

- function new_row = insert_open(xval,yval,parent_xval,parent_yval,hn,gn,fn)
    - This function populates the list OPEN with values that have been passed to it.

- function InvOptiPath = pathplanning(MAP, xStart, yStart, xTarget, yTarget)
    - A* ALGORITHM Demo

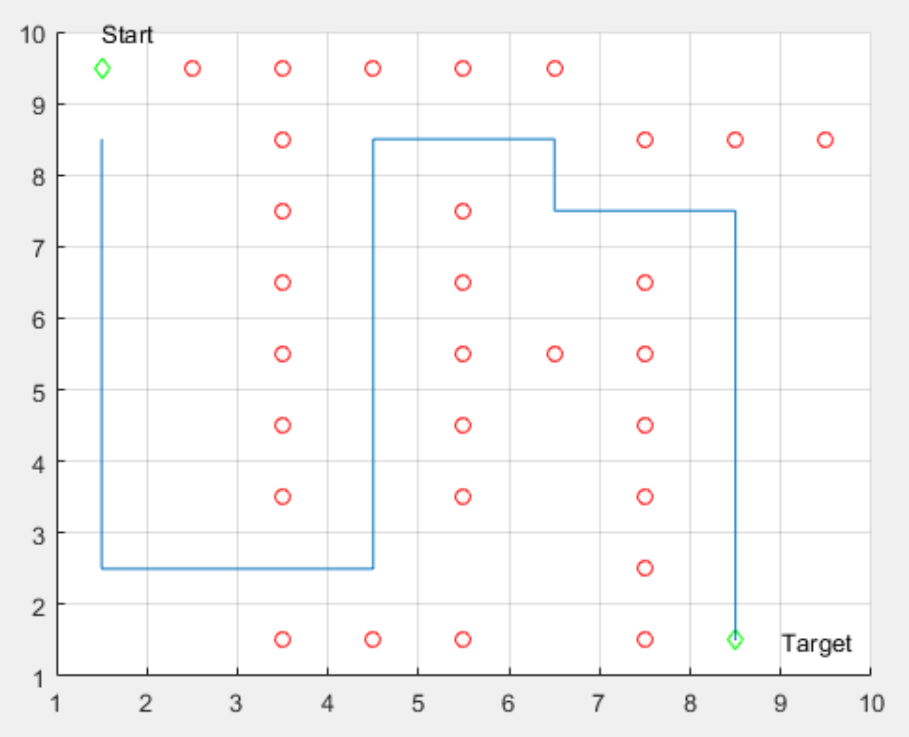

Figure 3: Target path planning

Figure 3 shows the result of the path planning when “testPP.m” is run. The idea of this path planning function is to find the shortest path from to the target.

3.0 RAPID Code

This section shows RAPID code used to achieve the required task in this assignment. The RAPID task has been updated for the function listed below:

- Communication with Matlab
- Robot studio Simulation

3.1 Communication with Matlab

3.1.1 Process Information Communications

- Incoming and outgoing byte
- Command type
- Subcommand type
- Queue number

3.1.2 Return Information Packet

- Waiting for command
- Status
- Pose
- Joints
- End effector
- Awaiting command

3.1.2 Safety System

- Emergency stop
- Light curtain
- Motors
- Hold to enable
- Motion task error

4.0 Minutes of Meeting

This section shows minutes of meeting for each day that group 15 set up for a meeting. The purpose is to keep track of each group member progress. Percentage completion for each required task assigned to each group member is shown beside each group member name for each week.

4.1 Meeting 1

Day: Friday
Date: 5/10/2018

Members in Attendance:

- Fraser
- Liyana
- Dennis
- Steven

On the Agenda:

- Assignment 3 briefing from tutor
- Distributing out roles among group member
- Determining the priorities for next week lab time

4.1.1 Agenda Item 1: Assignment 3 Briefing

Description:
Tutor went through the whole spec and tips on how to do it

Notes:
1) GUI

- Advised to have a system in place to manually label blocks
- Lists for reachable and non-reachable blocks
- Grid based selection system for BPs
- Conveyer changes: Insert vs Reload box options
- Image previews now only need to show snapshots (not live)

2) Occupancy Grid

- 9x9 area grid to denote playing area: will use a matrix in MATLAB to denote,
  9 x 9 x 4 where the z dimension will say the following:

Playing Area Occupancy Grid		
Name	**Data**	**Comment**
Status	Boolean	Denotes occupation 1 = occupied
Type	Integer	Denotes letter or shape 1 = Letter 2 = Shape
Orientation	Float	Block orientation only (no letter orient)
Position	vector	Denotes Block Position

- 2 6x1 grid to denote “decks” of players

Decks Occupancy Grid		
Name	**Data**	**Comment**

Status	Boolean	Denotes occupation 1 = occupied
Type	Integer	Needed to sort decks
Orientation	Float	To fix the positions
Position	vector	Denotes Block Position

3) Calibration

- Follow instructions on: Measuring Planar Objects (google and MATLAB instruct)
    - Need world coordinates of the following

Table			
Name	**Global**	**Local frame**	**Comments**
$\mathbf{P_{tableHome}}$		(0,0)	Home
$\mathbf{P_2}$		(,0)	Very Left (past decks)
$\mathbf{P_3}$			Very Right (past decks)
$\mathbf{P_4}$			Center Top of Board
Conveyer			
Name	**Global**	**Local frame**	**Comments**
$\mathbf{P_{conHome}}$		(0,0)	Home
P		(,0)	Defined by us
P			Defined by us
P			Defined by us

4) Path Planning

- Need a path finding algorithm, Dijkstra's or A*
- Takes in a 9x9 Boolean matrix and returns a 9x9 matrix
- Will be tested by having the EE 5mm above the board and tracing the good path through

5) Communications

- Changing to hexideimal encoding for Joints
- Introducing delimeters (incorporate on both sides)
- Robot Studio: give back a awaiting command status

6) Testing

- Robot Studio: Create a smart object that can be picked up (for now)
- Will require MATLAB part to test communication

7) Vision

- Strip the algorithm

    - Only need to identify between letters/blocks
    - Orientations are only block wise
    - Only need to mark centroids

- Strip the conveyer
    - Do not need box diagnostics, just Centroid of it

8) Movement

- Fix the movement functions for joints
  RS: get the XYZ EE straight from Robot Studio

4.1.1 Agenda item 2 – Distribution of Roles and Goals

1) Fraser: GUI and movement (5% complete)
    - Start stripping the GUI
    - Implement movement via callback

2) Steven: Communication in Matlab and vision (30% complete)
    - Start stripping the vision algorithm
    - Change communications protocol
    - Help write the new movement functions

3) Liyana: Simulation (5% complete)
    - Create game board in CAD
    - Get Smart Objects being placed in CAD (no need to get them working yet, just understand them)

4) Koushik: Calibration (15% complete)
    - Implement the MATLAB page for world coordinates
    - Modularize the movement functions

5) Dennis: Path Finding (5% complete)
    - Find some random ass path planning, and just outfit tit to input 9x9 and output 9x9 matrix

6) Daniel: Communication in RAPID (30% complete)

4.2 Meeting 2

Day: Friday
Date: 19/10/2018

Members in Attendance:

- Koushik
- Fraser
- Steven

- Dennis
- Liyana

On the Agenda:
- Fixing the communication between Robot Studio and Matlab
- Set up Smart Object in Robot Studio

4.2.1 Agenda Item 1: Group Member Start on Doing Task Assigned

Description:
Koushik has created a function to convert the pixel co-ordinates on the live camera feed to global co-ordinates, Fraser, Steven and Dennis are fixing the communication and Liyana has started on Robot Studio simulation on laboratory time.

Notes:
1) Koushik: (20% complete)
- Used pre-defined points and offsets to generate the rotation matrix, which was applied on the undistorted. The camera parameters were used to undistort the points.

2) Fraser, Steven and David (50% complete)
- Worked on the communication protocol between MATLAB and Robot Studio.

3) Liyana (40% complete)
- Aim to create blocks as a target (Robot Studio)
- Build 30x30x15 blocks according to the image captured using table camera and conveyor camera in lab last week
- Set work object and tool for smart component purposes
- Set path from robot to one block to test teaching robot without attaching end effector tool
- Set path for robot to reach a block and pick that block

4.3 Meeting 3

Day: Monday
Date: 29/10/2018

Members in Attendance:
- Koushik
- Steven
- Dennis

On the Agenda:
- Fixing task assigned for each group member to prepare for testing on laboratory time on Friday.

4.3.1 Agenda Item 1: Work on Each Part to Fulfil Task Assigned

Notes:
1) Koushik (80% complete)

- Create a GUI to be used for Noughts and Crosses / Tic-tac-toe game
- Create a button for AI and human player
- Create a movement button to set up the pause, resume and stop function

2) Steven (70% complete)

- Fixing the movement of the robot

3) Dennis (40% complete)

- Create an algorithm for a path planning and test the function created

4.4 Meeting 4

Day: Friday
Date: 2/11/2018

Members in Attendance:

- Steven
- Nur Liyana

On the Agenda:

- Fixing the communication of Matlab and RAPID
- Sending message from Matlab to RAPID
- Try to send message from RAPID to Matlab

4.4.1 Agenda Item 1: Work Out the Matlab-Robot Communication

Description:
Meeting at the robotic lab to fix and test the communication

Notes: (70% complete)

1) Fixing Matlab code on communication protocol
2) Successful in sending message from Matlab to RAPID when testing with the robot in the laboratory
3) Unsuccessful when RAPID try to respond to the Matlab after receiving the message

4.5 Meeting 5

Day: Monday
Date: 5/11/2018

Members in Attendance:

- Steven
- Fraser
- Daniel
- Nur Liyana

On the Agenda:

- Sending message from RAPID to Matlab
- Try to send message from RAPID to Matlab after successful communicate and receiving message

4.5.1 Agenda Item 1: Fixing Communication on RAPID

Description:
Meeting at the robotic lab to fix and test the communication

Notes:
1) Daniel, Fraser, Steven (80% complete):

- Fixing RAPID code on communication protocol

2) Liyana (85% complete):

- Build an end effector tool named vacuum solenoid using Solidwork
- Create path from robot to blocks to test teaching robot using robot tool and create path from target blocks to the target box
- Manage the documentation

4.6 Meeting 6

Day: Friday
Date: 9/11/2018

Members in Attendance:

- Dennis
- Daniel
- Koushik
- Fraser
- Steven

On the Agenda:

- Merge the necessary Matlab files and Robot Studio part and test it on the robot in the lab.

4.6.1 Agenda Item 1: Test the Function Created and Debugging

Description:
Each group member submit work that has been done for each part.

Notes:
1) Dennis (100% complete)

- Submit the path planning function files and explain it to the group member. The test run is performed successfully.

2) Daniel (100% complete)

- Working on RAPID side to fix the error. Daniel communicate with Fraser and Steven to try sync the communication between Matlab and RAPID.

3) Koushik (100% complete)

- Show the GUI function for playing the Tic-tac-toe game. The group member also aim to get extra mark by making human play the game with the AI function.

4) Fraser and Steven (100% complete)

- From the past few weeks, group member has been working on the communication and fix the communication error between Matlab and RAPID. They also aim to work on Matlab part to sync with RAPID side. At the end of this meeting, the Matlab and RAPID are able to communicate successfully