The University of Queensland
School of Information Technology and Electrical Engineering
Semester One, 2016
CSSE3010 / CSSE7301 – Project 2
Due for demonstrations: your prac session - week 13 (31 May -3 June), 2016
Weighting: 25% of your overall mark (CSSE3010)
Weighting: 20% of your overall mark (CSSE7301)

Project 2 – Autonomous Rover Control

Objectives:

- To apply your knowledge and skills developed in stages 1 to 6, project 1 and milestone.
- To control a rover remotely and autonomously.
- To accurately and quickly track a moving rover using a pan and tilt webcam system.
- To estimate the position using Colour Markers and Sensors.
- Handle realtime radio communication telemetry messages, robustly and reliably.
- Integrate the Command Line Interface (CLI) library for control.

Introduction:

This project will build on your knowledge from stages 1-6, project 1 and milestone. You will be working individually for tasks 1.1 to 4.2. The project is designed to have you demonstrate your knowledge of practical content and ability to integrate new functionality into your program for autonomous remote control and realtime tracking of a moving rover.

Requirements:

1. This project **MUST** be implemented using **FreeRTOS**
2. This project MUST be built on the **Milestone** – the same CLI commands for Milestone must remain in project 2
3. **Version Control** – You **MUST** use version control as part of your development process. **HEAVY PENALTIES** apply for not obeying this rule. **READ** carefully the version control use requirements in the **ASSESSMENT** section of this document.

No excuses of malfunction of your system, Git (e.g. forgot to commit), etc. will be considered on the demo day.

Due Date:

Project 2 is due the prac session, in **week 13.** The project core tasks include task 1.1 to 4.2. In week 12, non-compulsory attendance help sessions will be available.

Core Tasks (35 Marks / 15% for CSSE3010/CSSE7301)
You may use the CLI library to implement additional (other than specified) commands, to control the different modes of your NP2 program.

Core Part 1: Rover Telemetry Download (7 Marks)
Establish a wireless connection with the rover using the nrf24L01plus radio. Send commands and receive responses, via the nrf24l01plus radio. See Table 1 for commands.

(2 marks) Task 1.1: GetPasskey Request – Implement the *getpasskey* and *rfchanset* commands as shown in table 1, using the CLI.

(3 marks) Task 1.2: Line Follower Sensor Request and Display – Implement the *getsensor* command as shown in table 1 and below. Display the received line follower sensor value (5 bits) in both terminal (with a timestamp) and using the LED Light Bar. You must write efficient bit manipulation code to display the value on the LED Light Bar.

(2 marks) Task 1.3: Timestamp Display – Implement the *gettime* command and display the command responses for *getpasskey* and *getsensor* with timestamps. Each sensor reading received must be displayed with a timestamp.

<timestamp><line follower sensor value>

Table 1: Telemetry Commands

Command	**Description**
getpasskey	Get a current passkey – must display in terminal
getsensor	Get the current sensor value – must display in terminal
gettime	Get the current time. The current time is the time since the NP2 was turned on. Time units should be seconds with 2 decimal point precision.
rfchanset	Set the RF channel of the nrf24l01p

Core Part 2: Rover Motion Control (10 Marks)
Control the angular and linear motion of the rover, using commands implemented with the CLI library. See Table 2 for commands.

(3 marks) Task 2.1A: Linear Motion Control – Implement the linear forward and reverse motion commands using the CLI library. The linear forward motion command should instruct the rover to move linearly in the forward direction, for a set distance. Similarly, the reverse command does the same but in the reverse direction. The distance must be specified on the command line as shown in table 2

(2 marks) Task 2.1B: Linear Motion Calibration – Implement the calibration CLI command, to allow the linear distance to be calibrated. The calibration CLI command can take any parameters that you may need.

(3 marks) Task 2.2: Rotation Motion Control – Implement the *angle* command (table 2), using the CLI Library. The angle command should move the rover to a certain angle in a particular direction (negative or positive).

(2 marks) Task 2.3: Accelerometer Control – Use the accelerometer (portrait and landscape status) to control the linear (tilt forward/backwards) and rotational (tilt left/right) motion of the rover. Accelerometer forward/backward controls linear motion (fixed minimum velocity of 20mm per second) and Accelerometer left/right controls rotational motion (fixed minimum angular velocity of 20 degree per second).

Table 2: Rover Motion Control commands

Command	Description
forward <distance>	Move the rover forward a specified distance in mm.
reverse <distance>	Move the rover in reverse a specified distance in mm.
angle <angle>	Move the rover to a certain angle (1 degree resolution).
Calibration <>	Calibrate linear motion of rover (parameters specified by user)

Core Part 3: Tracking Position Display (8 Marks)

Display the current distance and position of the rover (ORB output), using the laser on your pan and tilt module. See Table 3 for commands to implement. You can implement extra commands to control the tracking position display.

(5 marks) Task 3.1: Display – Display the current position of the rover, using the ORB output. You must use the laser on the pan and tilt to show the position on an A4 sheet grid, held vertically in the lid of your kit box. The laser must show the current position of the rover, as it moves. The A4 sheet must be 30cm away from the laser. The A4 (300mm x 200mm) sheet must correspond to the dimensions of the sandpit (900mm x 600mm).

(3 marks) Task 3.2: Distance Calculation – Implement the distance command to display the current distance of the rover from the starting edge of the boundary. Use the ORB output to obtain the distance. The distance estimation should be accurate within 20mm.

Table 3: Rover Motion Control commands

Command	Description
distance	Display the distance of the rover from the starting edge, in mm.

Core Part 4: (5 marks): Autonomous Control

Autonomously control the rover by making it drive to a waypoint in the sandpit and to follow a marker being moved in the sandpit. Implement the follower and waypoint CLI commands used for the autonomous control of the rover. Table 4 shows the commands implemented for task 4.1 and 4.2.

(2 marks) Task 4.1: Waypoint – The rover should drive towards a designated waypoint in the sandpit and stop. Implement the required CLI commands.

(3 marks) Task 4.2: Marker Follower – The rover should follow the position of the colour marker being move in the sandpit and stop. Implement the required CLI commands.

Table 4: Rover Motion Control commands

Command	Description
follower	Follow the marker
waypoint <marker ID>	Move to the waypoint defined by the marker ID.

(3 mark) Code Commenting, Style and Structure – Your code should conform to the following style:

(0.5 mark) Comments: Comments should be placed at the **front of each function** and **if/while/for loop**. Key sections of code should be commented.

(0.5 mark) Variable and Function Names: single letter names (ie. i, j, k) should only be used for counters. Each variable should have a name relevant to its use.

(2 marks) Code Structure: Structure your code into multiple c files.

(2 marks) Workbook– Your work will be marked according to the format outlined:
(0.5 mark) Objectives of Project
(0.5 mark) Hardware Implementation
(0.5 mark) Software Implementation
(0.5 mark) Testing Procedure

Total: 35 marks (20% of course marks for CSSE3010 and CSSE7301)

Make sure you keep updated on Piazza for any of hints or answers to questions that many students are having issues with. Also be sure to update **Git** regularly.

Challenge (10 Marks / 10% for CSSE3010 and 7.5 Marks / 5% for CSSE7301)

NOTE: You must pass all core tasks (1.1 to 4.2) and Milestone, in order to be assessed for the challenge. Failure to pass the core tasks will exclude you from being assessed for the Challenge.

The Challenge is an optional section for the remaining 10 marks. The following challenges are made to *challenge* you and they will NOT be marked easily. If you are not 100% on the rest of your project, it is strongly suggested you work on those first, as challenge marks are comparatively much harder to obtain.

You can choose to either one 10 Mark challenge or one 7.5 Mark with one 2.5 Mark challenges. For CSSE7301, you may only do one 7.5 Mark challenge.

You may include extra CLI commands for each challenge.

(7.5 marks) Challenge 1: Tracking Position Display Cursor – Turn the photodiode into a cursor for the tracking display. Place the photodiode on the tracker display and use the laser to detect the position of the photodiode (cursor). Once the cursor is located, guide the rover to a corresponding position (within 50mm) in the sandpit.

(7.5 marks) Challenge 2: Accelerometer based Menu select – Use the accelerometer portrait and landscape status to implement a simple menu for tasks 2.1 to 2.3, 3.2 and 4.1 to 4.2. Use kermusb to display a menu of items (commands in 2.1 to 2.3, 3.2 and 4.1 to 4.2). The accelerometer portrait status implements the up/down arrows and landscape status implements the left/right arrow. Use the onboard NP2 pushbutton as the enter button.

Each command in 2.1 to 2.3, 3.2 and 4.1 to 4.2 should be selected from the main menu (displayed as vertical list in kermusb). The cursor should highlight selected menu items in a specific colour. When selected from the main menu, a submenu should display the selected command options (e.g. forward -> distance -> 10, 20, 30, 40, 50). At most five options should be displayed for each command. Once a command has been selected, the main menu should be displayed again.

(10 marks) Challenge 3: MQTT interface – Create an MQTT interface to control the rover and implement tasks 2.1 to 4.2. This requires you to use the ethernet connection. The MQTT interface must be hosted on the NP2. You must be able to subscribe to distance or position and velocity MQTT topics, to display the current values. You must be able to publish to the forward, reverse and angle topics, to control the linear and rotational motion of the rover.

(10 marks) Challenge 4: Webpage interface – Create a webpage interface to control the rover and implement tasks 2.1 to 4.2. This requires you to use the ethernet connection. The webpage interface must be hosted on the NP2. Also display graphs of the current distance or position and velocity.

(7.5 marks) Challenge 6: Augment Accelerometer Control with Compass – Set up the magnetometer provided in the kit and use the raw magnetic X and Y values to implement the rover rotational motion. The accelerometer should use the raw accelerometer values to implement the rover linear motion (tilt forward/backward) and control the rover's speed (tilt angle). Note the compass values must be tilt compensated. Compute the approximate compass heading calibrated in degrees (360 scale) and display it at 1Hz (on the same line).

You will also have to compensate for directional errors caused by tilt motion, using the accelerometer. Note that magnetometers can be very sensitive to anything that can influence its magnetic field including tables, or even your mobile phone. You must apply magnetic declination and tilt correction in order to achieve full marks. You must be able to verify the heading using your (or the assessor's) smart phone. As always, you must use the CLI to enable this display mode.

(2.5 marks) Challenge 7: Use Doxygen to generate help documentation – Use doxygen to generate HTML based documentation of your code. You will have to follow the doxygen style guide, as well as remove any commented-out code. Note doxygen needs to be installed in the virtual machine environment. Installation details will be posted on BB. See the following for more details:

http://www.stack.nl/~dimitri/doxygen/

(2.5 marks) Challenge 8: Colourful CLI – Display a prompt for the CLI. The prompt should have a blinking underline or blinking cursor, as seen with other popular command lines. Also Enabled the text colours for Kermit. The prompt should be one colour and the user typing should be another and the display should be different. Implement the command line cursor (blinking underline), up, left and right arrow commands.

(7.5 or 10 Marks) Challenge 9: Create Your Challenge - Come up with your own challenge. Your idea **must be approved** before continuing. Your challenge can be based on using different hardware devices for rover control or implementing new features of the Netduino Plus 2 and FreeRTOS. Examples consist of IPv6 implementation of LwIP, touchscreen interface, using the Cortex M4F DMA unit or encryption unit.

Total: 10 marks (10% of CSSE3010 course marks) 7.5 marks (5% of CSSE7301 course marks)

ASSESSMENT RULES:

1.Marks will be applied as specified in the CORE and CHALLENGE sections above

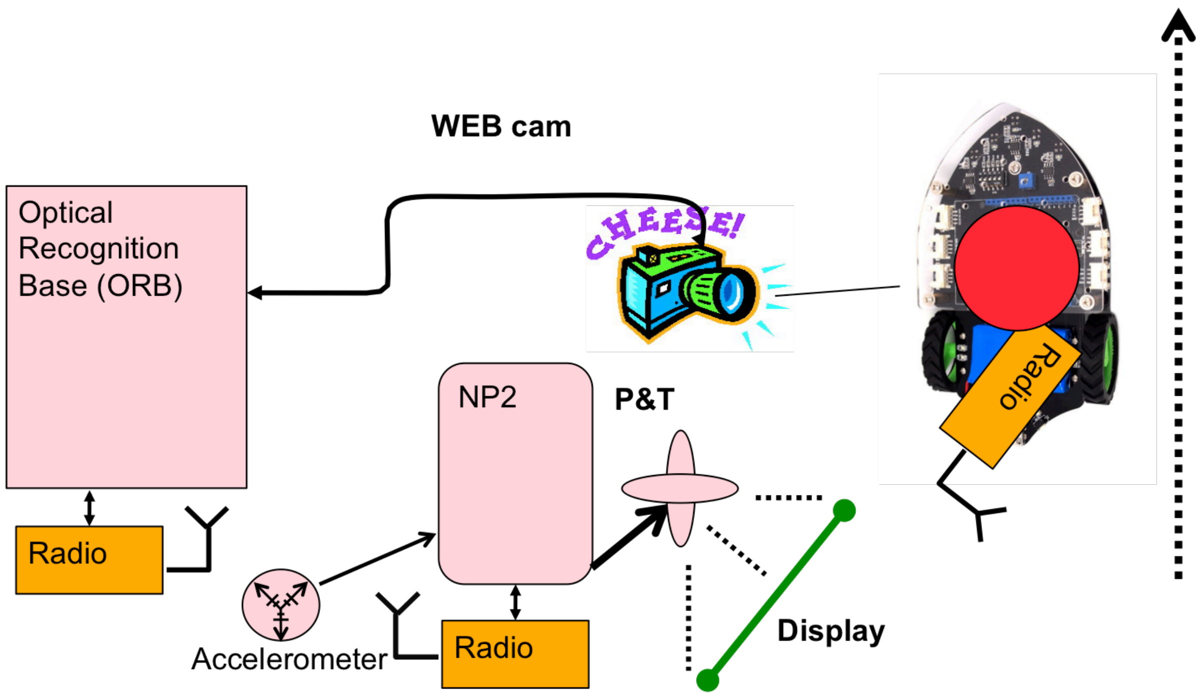
2. Git use requirements and potential mark deductions:

- You **MUST** have at least 4 Git commits:
- At least 1 commit from Milestone (can be committed week 11)
- At least 3 commits from Project 2 over the period week 11 to 13. (You can have as many commits as you want but you must identify which are to be considered in the assessment. There cannot be more than 2 considered in one week).
- The last commit must be the code of the final demo and the commit must be made BEFORE the assessment session, otherwise you will not be assessed and potentially you lose all marks from project2. .
- The last commit must include a text file titled PROGRESS_studentnumber, which must list 4 commits you want be considered and clearly identify progress in each of them.
- The considered commits must show enough progress in coding to be treated as 'genuine'.
- All your 4 commits will be checked for progress before accepting them as 'genuine'.
- The workbook must include a printout of the PROGRESS file stapled or glued to the last page of project2 entry.
- Marks will be deducted in the following cases: (in % of the course)
  - Lack of a commit (or any 'non-genuine' commit): -2% each
  - Lack of file PROGRESS: -2%
  - Lack of PROGRESS printout in the workbook: -2%
  - Lack of the final demo commit: -25

BEWARE: you can **lose** (3*4 +2 +2) = 10% in case of missing intermediate Git commits or 30% in case of lack of the final commit. SUBMIT OR PERISH.

3. Other cases of mark deductions:

- 4 mark deduction)- Reprogramming of NP2 is required for any task.
- 1 mark deduction) - Erratic servo control.
- 4 mark deduction) – Incorrect use of FreeRTOS, i.e. not using Tasks, Semaphores and Queues for implementation of functions
- 4 mark deduction) – Incorrect use of CLI, e.g. using CLI callback functions to implement Task or HAL functionality. CLI functions should signal Tasks with Semaphores and Queues to control the program flow.
- 4 mark deduction) - Incorrect use of library functions.
- 2 mark deduction) - Duplication of library code (i.e. copying parts of the radio driver into main code).
- 4 mark deduction) - mylib files are not complete.
- 4 mark deduction) - mylib files are not used correctly for initialization, set and get operations.
- 2 mark deduction) – mylbi formatting guidelines are not followed (i.e. No descriptive top comment).



Project 2 Overview