Software Project Management Plan

for

Lunar Rover Mapping Robot

Version 2.0

Group: UG12

THE UNIVERSITY
of ADELAIDE

Contents

Revision History

Version	Date	Reason for changes
1.0	30/8/2017	Initial draft
2.0	29/10/2017	Final version

List of Figures

List of Tables

1 Introduction

1.1 Purpose and Scope

This is the Software Project Management Plan(SPMP) for Lunar Rover Mapping Robot(LRMR). The project team members are undergraduate students at the University of Adelaide. This SPMP establishes roles, responsibilities, processes for the LRMR project. It also contains the Risk Management Plan, the Work Plan, the Configuration Management Plan, the Documentation Plan and the Quality Assurance Plan for the project, in addition to listing the tools, references, and procedures to be utilized.
The SPMP is a plan about how to successfully accomplish the goals of the project LRMR, which goals are to be set out in the Software Requirements Specification (SRS) document. The following system features will be included in the final version of the product. The system: shall be able to explore the survey area autonomously, possibly without any human intervention; can be remotely controlled by an operator from a distance; can transmit sensor data from the robot to the controller; can avoid colliding with objects, No-Go-Zones (NGZ), and radiation areas; transmit data through WiFi; import and export map data in XML format that satisfies the provided Document Type Definition (DTD); and traverse through a model of the survey area on an A1 sheet of paper.
The following features won't be included in the final version of the product: readiness to be deployed in the field, transmission of high-definition images and videos, and traveling 500 metres.

1.2 Assumptions and Constraints

Assumptions:
It is assumed that the focus of our implementation of the rover is its ability to survey (sense and map) its surroundings, and that the physical hardware and construction, as well as the communication protocols used are merely representative, and in line with the demo equipment provided. The sensors on a full scale rover would be suited to the obstacles found on the moon, but the type of information supplied would be usable by the control software used by the demonstration rover.
It is assumed that the physical A1 map to be used for the demonstration will be designed to be suitable for the size and capabilities of the rover and its sensors.
It is assumed that the DTD for the save format of the survey map will be provided during the course of the project, before final milestones are reached.

Constraints:
The DTD will be provided by a third party during the project. Until this has been delivered, support for the required format to interface between the internal representation of the survey and an external file cannot be added.

1.3 Project Deliverables

The Project Deliverables are summarised in table 1. For a further breakdown of Task-Resource Allocation, refer to table 3. For a further breakdown of documentation deadlines, refer to table 4.

Table 1: Project Deliverables

Week	Deliverable Title	Date
Week 7	Milestone 1 Freeze	5-09-17
Week 8	Milestone 1 Demo	12-09-17
Week 8	Milestone 2 Freeze	12-09-17
Week 9	Milestone 2 Demo	3-10-17
Week 12	Software Deadline	27-10-17
Week 12	Document Deadline	30-10-17

1.4 Evolution of the Plan

The SPMP will be updated during the developing progresses of the project. The following sections describe the update rules:

- References will be updated as necessary
- Definitions will be updated as necessary
- Project Organization will be updated while it has new roles and responsibilities.
- Risk Management Plan will be updated accordingly as the project progresses.
- Process Model will be updated if it is necessary.
- Work Plan will be updated according to the progress of the project.
- Supporting Plan will be updated if it is necessary

2 References

[1] Oracle. 1999. *Java Code Conventions*. [ONLINE] Available at: `http://www.oracle.com/technetwork/java/codeconvtoc-136057.html`. [Accessed 4 September 2017].

[2] Google Lunar XPRIZE. 2017. *Google Lunar XPRIZE*. [ONLINE] Available at: `https://lunar.xprize.org`. [Accessed 4 September 2017].

3 Definitions

Acronyms

API	Application Programming Interface
DTD	Document Type Definition
GUI	Graphical User Interface
NGZ	No-Go-Zones
SDD	Software Design Description
SPMP	Software Project Management Plan
SRS	Software Requirements Specification
UI	User Interface
XML	Extended Markup Language

4 Project Organization

4.1 Roles and Responsibilities

The project is separated into three modules. Since we have seven members, the team is divided into small groups. Two to three members are allocated to either the Handler-side (which handles high-level decision making), or the Robot-side (which translates high-level code to low-level motor actuation), as each these are complex but mostly independent modules. We have assigned our most experienced GUI programmer to the task of writing UI code. As some of the members have a heavy workload, those with lighter workload have been assigned the task of writing documentation. Table 2 lists out the roles and responsibilities of each team member.

Table 2: Roles and Responsibilities

Name	Role & Responsibilities
James Austin Kortman	Team Leader Motivate team members, assign duties to members and design the system architecture. Has experience in software architecture design and AI programming.
Adam Davidson	Documentation Manager Robot-side coding, take the minutes for the meeting, manage documentation. He has experience in running meetings and meeting minutes so seems well suited to this position.
Jeremy Tomas Hughes	Development Team Member Design robot architecture, take care of the robot-side building.
Cyrus Villacampa	Super Integration Manager Handler side coding
Ziang Chen	Development Team Member GUI design and coding, has experience in event driven programming in Java and GUI design.
Linlin Song	Assistant of documentation Writing documents
Lam Ravi Wing Fung	Development Team Member Handler side coding

5 Risk Management Plan

5.1 Introduction

The following is a list of risks that have been identified, classified, and given strategies to manage and minimize the risks.

5.2 Risk Monitoring Plan

To monitor risks, a list of risks (see below) will be compiled. Using this a guideline, team members most involved in the functional side of the software will monitor the components of the software for signs of problems that may impact assigned goals. When risks are identified, they will be shared with the whole team, so solutions can be discussed by the whole team.
Each risk identified will be classified as a Personnel Risk, Hardware Risk, or Software Risk, and be given a rating of severity and likelihood.
Ratings will be on a scale of Low/Moderate/High.
For Likelihood, low means that there is a chance it could happen but most likely will not, and high means that it is probably the risk will occur in any demonstrations.
For Severity, low means that while it is not ideal, it will not cause significant impact to the project, possibly due to easy of working around the problem, and high means that if the risk occurred, it would significantly impact the project with no easy work around.

5.3 Personnel Risks

Risk: Team members become unavailable to complete assigned tasks.
Severity: Moderate
Likelihood: Moderate
Indicator: No evidence of progress on task, or poor progress when approaching deadlines. Alternatively, notification from member in question.
Strategy: A minimum of two team members are assigned responsibility of any given component, and have a working knowledge of how that part is implemented. As members get busy, or become unavailable to make progress, the other member assigned to that component can take charge, and get assistance from other available team members if necessary.

5.4 Hardware Risks

Risk: Wireless Communication to rover fails on final demonstration
Likelihood: Moderate to Low
Severity: Moderate
Indicator: Any instance of connection difficulties when using the robot.
Strategy: Provide long USB cable, and an interface to use it, to directly connect to the robot to allow the other capabilities to be demonstrated.

Risk: Inaccuracy in the motors causing significant errors in maintaining an idea of position

and facing of the rover.
Likelihood: Low
Severity: Moderate to Low
Indicator: Autonomous tests resulting in significant difference in suggested location to physical location.
Strategy: Make use of the gyroscopic sensor to help keep the direction correct, and make use of sensors and previous survey data to periodically relocate the rover.

5.5 Software Risks

Risk: Communication delay between rover and operator causing poor responsiveness, and preventing the rover from reacting quick enough to avoid obstacles.
Likelihood: Moderate to Low
Severity: High
Indicator: Movement tests having unacceptable delays in response to movement commands.
Strategy: Build the handler program to minimize internal delays in pushing commands to the rover, build in reflexive safeguards, and make any safeguards act early as necessary to prevent collisions.

Risk: Path finding efficiency being poor, taking too long to identify a path, or choosing long or impossible routes
Likelihood: Moderate
Severity: High
Indicator: Poor performance on tests.
Strategy: Attempt to move directly to goal when no map information available, reacting to detected obstacles, otherwise do further research into existing path-finding algorithms.

Risk: Incompatibility with internal map representation with the DTD specification for the import and export map format.
Likelihood: Moderate
Severity: Moderate
Indicator: The DTD expecting types of data that are not tracked, or expecting a format hard to convert the internal representation to.
Strategy: Build the map system to support all the required client features, and with any features that we would include if we were producing the DTD, and being prepared to focus on fixing incompatibilities if they arise.

6 Process Model

The project follows the evolutionary process model with an agile approach, as shown as figure 1. The plan takes 13 weeks to develop and demonstrate the prototype of the Lunar Rover. Requirements are gathered and defined by clarifying the expectations of the client through in-person meetings. After gathering the requirements, the team will being to work out the demos for each part of the project: the GUI controller, the robot architecture, and the high-level functionality of the robot. The team will enter an iterative cycle, meaning demonstrations of the project will be given each week and feedback will be received from the client. Based on the feedback received, the team will refine and redesign the system. Two weeks before the deadline, the system design will be frozen, and the project will begin the final testing process.

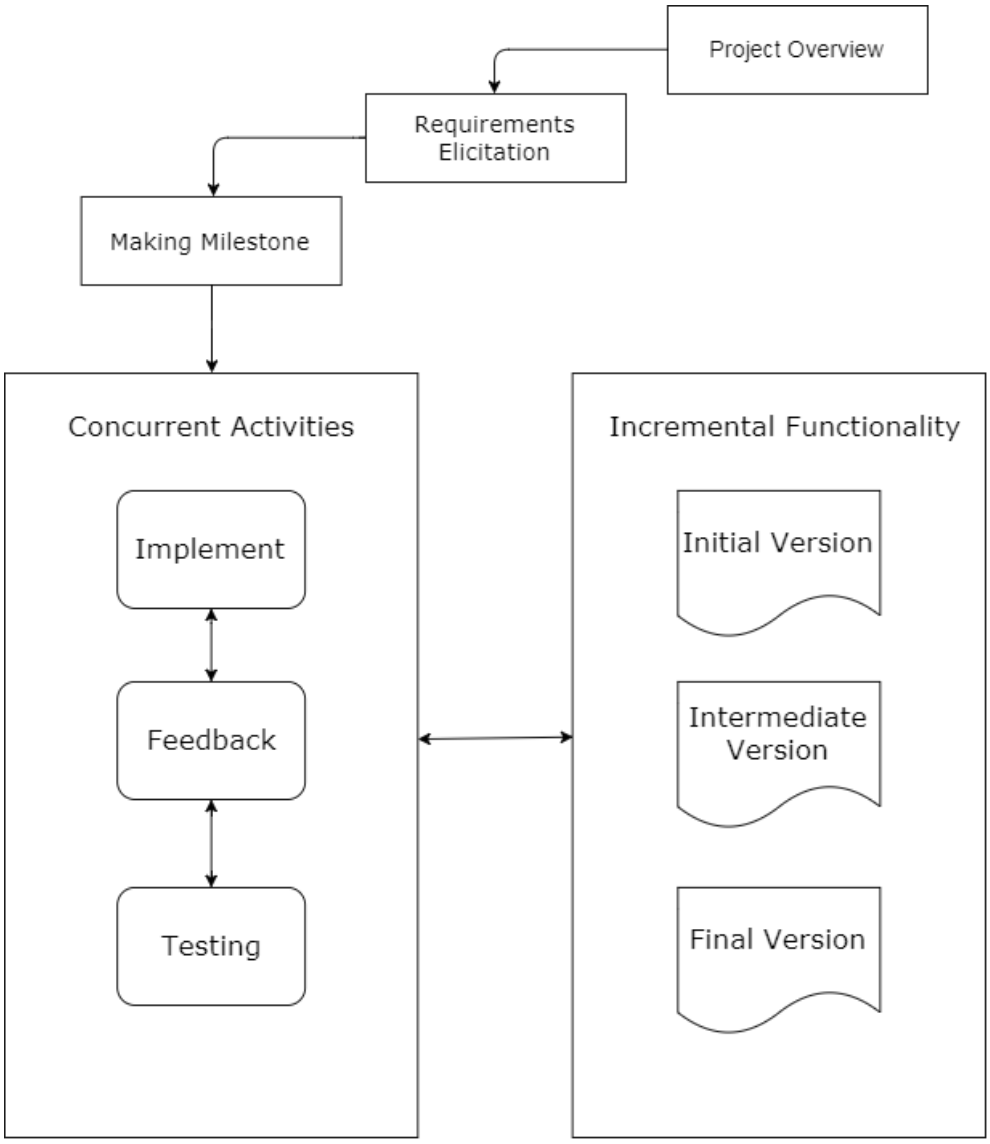


Figure 1: Process Model

7 Work Plan

7.1 Work Activities

Below are the list of work activities or tasks that have been identified to have workloads that can be completed within 1 to 2 weeks.

Title: Module communication queues
Description: Implementation of the interfaces that connects the GUI and the independent modules.

Title: Graphical user interface construction
Description: Involves the construction of a user friendly graphical user interface.

Title: Robot Assembly
Description: Involves the assembly of the robot hardware.

Title: Handler module implementation
Description: Implementation of the module that translates user-level command to high-level command and sends the translated command to the robot through a RobotMessenger interface.

Title: Robot module implementation
Description: Implementation of the module that translates and sends high-level command to low-level command that the robot can understand.

Title: Saving of sensor data and updating the map data structure
Description: Implementation of the algorithm that updates the map data structure using sensor data (such as colour, ultrasonic and possibly touch sensor) from the rover.

Title: Transmission of sensor data
Description: Programming the robot to send back sensible sensor data to the controller in a controlled manner.

Title: Map rendering on GUI
Description: Involves implementation of the technique in rendering of the map data structure onto a region on the GUI. This task is split into two, the first task involves rendering a static map and the second involves rendering a dynamic map.

Title: Robot localization algorithm implementation
Description: The implementation of the algorithm that locates the position of the rover on the current map.

Title: Path finding algorithm implementation
Description: The implementation of finding a safe and shortest route from the current position of the rover to a specified point.

Title: Importing/Exporting of map data
Description: This involves the implementation of taking an existing(partial or complete) map data in XML format and using it to move the rover and exporting a partial or complete map data into an XML file that adheres to the DTD.

Title: Autonomous mode exploration implementation
Description: This task involves the implementation of the autonomous mode of the robot, where the robot would explore the entire survey area without, possibly, any human intervention.

Title: Testing
Description: This involves testing of implemented features specified in the milestones. This task is spread throughout the entire software development life cycle.

7.2 Milestones

7.2.1 Week 6 Milestones

- The robot can be manually controlled by the user. Through buttons on the user interface or arrow keys on the keyboard, the robot can be directed to move forward, move backward, turn left, or turn right. It will move at a fixed speed.

7.2.2 Week 8 Milestones

- All necessary sensor data is sent by the robot and successfully received. This includes RGB information from the colour sensor, distance information from the ultrasonic sensor, and touch data for any touch sensors used, if any, and the gyroscopic sensor, if required.
- The robot will not collide with obstacles or move into radioactive regions or any other impassable regions. The robot will stop before moving into any of these obstacles or regions, even while under manual control.
- Sensor data received from the robot will be processed and stored into a data structure representing the landscape surrounding the robot. Any map information that the robot has sensed will be accessible in some form to the application by reading the map data structure.
- A testing map will be made including 3D obstacles, crater lines, and radioactive regions. It will be drawn on a piece of paper of size A1. crater lines and radioactive regions will be assumed to be black and green coloured, respectively.

7.2.3 Week 9 Milestones

- The map data structure can be rendered to an region of the UI. Features of the map, including crater lines, radioactive areas, obstacles, and any other map features are visible on the map if they have been detected by the robot. The map display is updated in real-time as the robot detects new features. They will be marked on the map using the same colours that those features use on the map.

- The robot module will communicate its status to the handler module (which issues high-level commands to the robot module). At any point in time, the handler module will know which commands have been completed, which command is currently being executed, and which commands have not begun to be executed by the robot.
- The robot is capable of traversing a path to any location on the map provided by the user if it is possible to get to, otherwise the system will inform th e user that the selected location cannot be reached. The robot will find a path around any obstacles or impassable areas. The location will be selected through the system GUI by selecting a point on the rendered map.

7.2.4 Final Milestones

- No-Go Zones can be drawn on the map. The robot will never move into a No-Go Zone, even when directed to.
- Maps can be loaded into the software in the form of XML files adhering to the provided DTD.
- Maps can be saved into an XML file adhering to the provided DTD, which can be read by the software to restore the map.
- While in automated mode, the robot will make decisions on where to move to in order to map the landscape within the area of interest. Without any human guidance, the robot will be able to map the entire area within 20 minutes. The user will still be able to direct the robot to particular locations using the user interface.

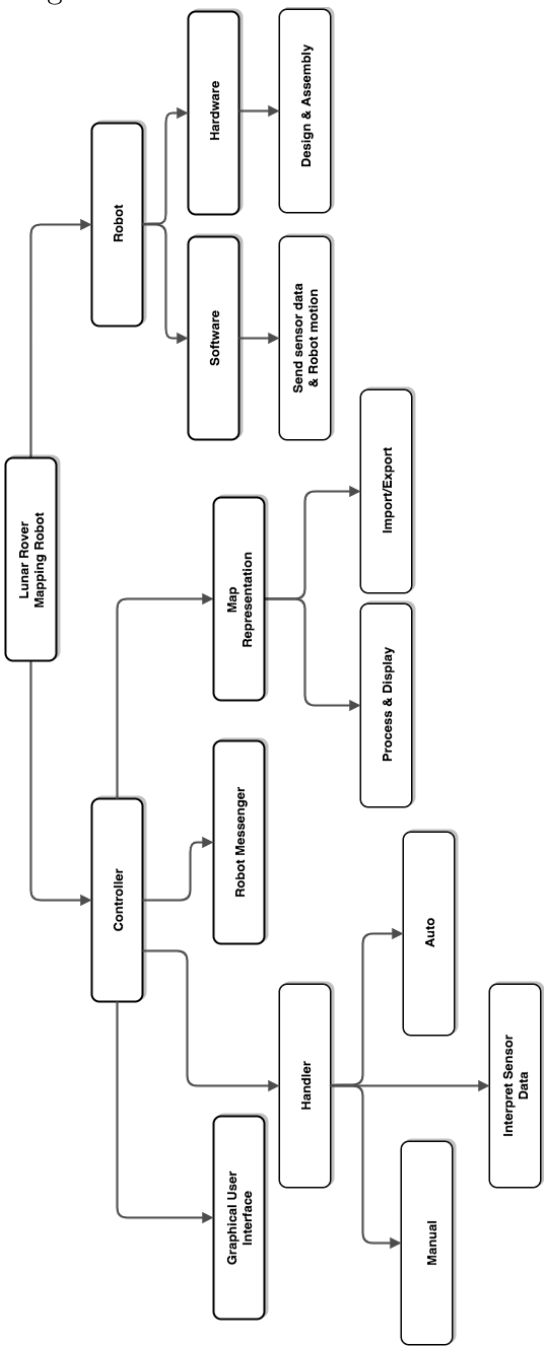
7.3 Schedule Allocation

The work breakdown structure (WBS) and the allocation of time on the identified activities are shown in figure 2 and 3 below respectively. A single activity is represented as a table with the name of the activity (first row), it's duration (second row) and it's start & end time (third row). Dependencies between tasks/activities is shown by an arrow and therefore tasks that are not directly/indirectly connected can be done in parallel. The tasks that are on the critical path is shown in red (−) line while the milestones are show in green (−) line.

7.4 Resource Allocation

The allocation of tasks to people (or resources) is mainly based upon two criteria. Firstly, the background and experience of the people involved. This is to ensure that at least one person is knowledgeable of the task at hand. Secondly, the difficulty and duration of a task. This is to ensure fair distribution of tasks among group members. A person with the necessary experience can voluntarily assign him/herself to particular task(s) or the team may decide to assign a task to a particular person. The allocations of tasks shown on table 3 below are both final and temporary, because some earlier tasks have already been completed (final) and more people might be assigned to future tasks if needed (temporary).

Figure 2: Work Breakdown Structure



8 Supporting Plans

8.1 Configuration Management Plan

Version iterations are numbered iterations, using the process described in Section 6, which describes our evolutionary process model. At a minimum, each client meeting where we present a new iteration of the software should be labeled with a new minor version numbering. For example, the first client meeting in which we present the software is version 0.1, so the next

Figure 3: Schedule Allocation Pert Chart

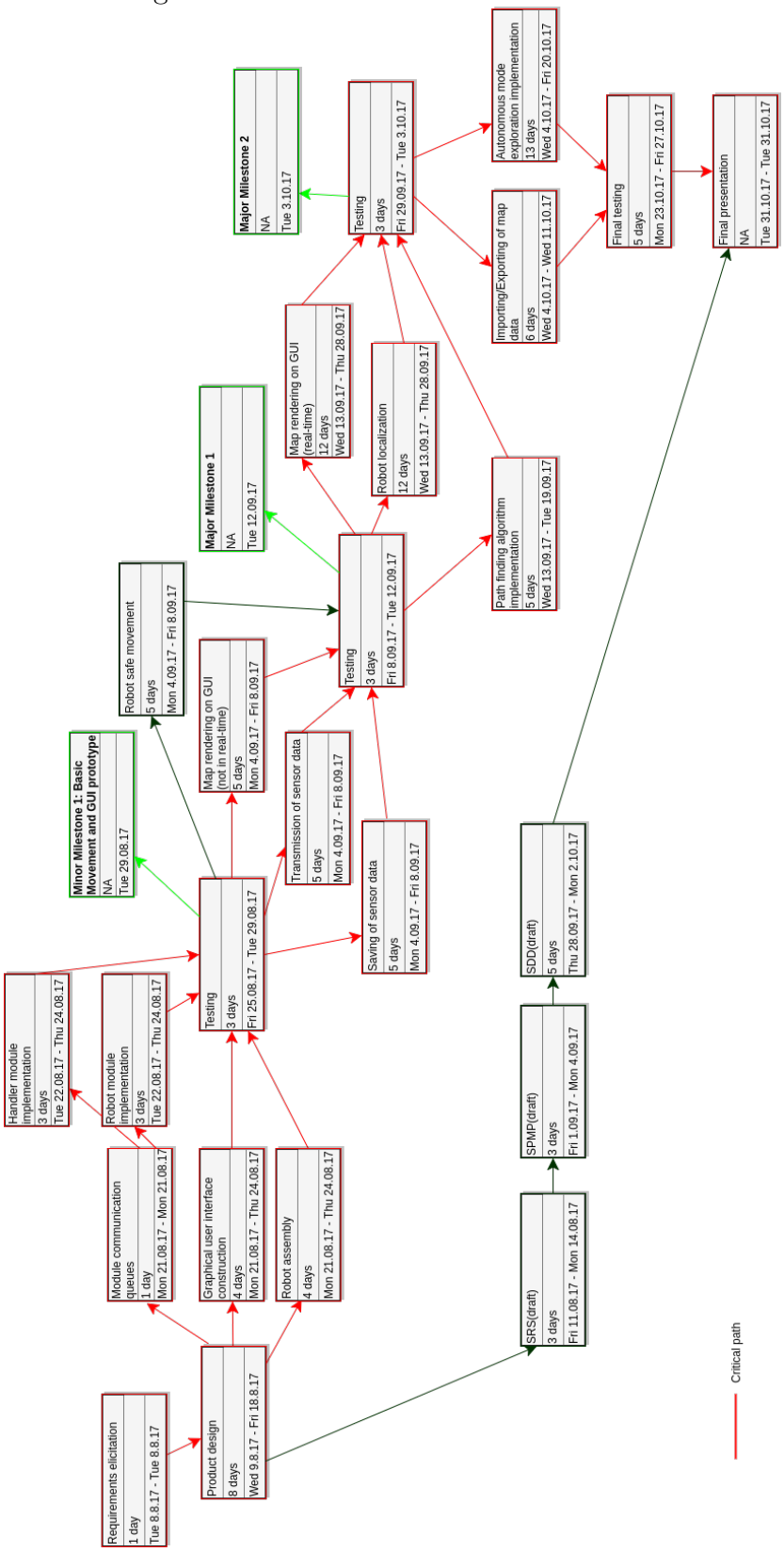


Table 3: Task-Resource Allocation

Task	**Assignee(s)**
Requirements elicitation	Everyone
Product design	Everyone
Module communication queues	James
Graphical user interface construction	Ziang
Robot assembly	Jeremy
Handler module implementation	Cyrus/Ravi
Robot module implementation	Adam
Testing	Everyone
Map rendering on GUI(not in real time)	Ziang
Transmission of sensor data	Jeremy/Linlin
Saving of sensor data	James/Cyrus/Ravi
Robot safe movement	Adam
Testing	Everyone
Map rendering on GUI(real time)	Ziang
Robot localization	Adam/Jeremy
Path finding algorithm implmentation	Cyrus/Ravi
Testing	Everyone
Importing/Exporting of map data	James
Autonomous mode exploration implementation	Cyrus/Jeremy
Final testing	Everyone
Final presentation	Everyone

client meeting may be version 0.2, and so on. In practice, we may have several internal versions which are fully-functional before the project is presented to the client. The code version may increment in this case. All increases in minor version numbering must be accompanied by a functioning software product.

Releases are labeled with major version numbers. As this is a short project, the initial pre-release versions will be labeled 1.x, and the final product will be labeled 1.0.

Change management will be handled through the use of GitHub commit logs. This allows us to view each code change as a unit, along with the author of the code change, and to roll back

any changes if necessary. These changes will not be labeled with any sequential numbering system (as is done for release and version numbering). GitHub branches are managed feature by feature.
Version and release change information will be compiled from GitHub logs and developer-written change information. JavaDoc comments may be used to determine what parts of the API have changed from previous versions when necessary. The GitHub repository structure is as shown in Figure 4.

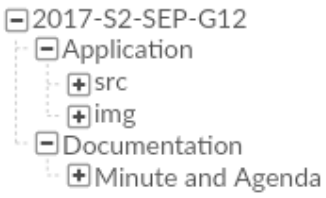


Figure 4: Repository Structure

8.2 Documentation Plan

8.2.1 Preparation process

Before a document is due, it is added to the next meeting's agenda. At the next meeting, the team discusses the required sections of the document and assign different team members to each section. This assignment is minuted and work is done on the respective sections to be ready for the internal milestone set in the meeting (usually the next meeting). At the next meeting progress is updated and reviewed and then the next milestone is set, so that the document is prepared on time.

8.2.2 Review Process

At each meeting, the whole document gets reviewed at a large scale. This is to make sure the document is heading on the right path and the authors are all going in the same direction. When the document is nearing completion, each section is allocated to a team member, who didn't write it, for review.

8.2.3 Template sources

The document templates we use are from MyUni. This template is then typed into an online, collaborative platform, from which the team can all work.

8.2.4 Documentation Schedule

The Documentation Schedule is shown in table 4.

Table 4: Documentation Schedule

Description	Status	Due date	Assignee
Software Requirement Specification Draft	Completed	22/8/17	Everyone
Software Requirement Specification Review	Completed	1/9/17	Adam
Software Requirement Specification Review Video	Completed	1/9/17	Adam
SRS Review Video Review	Completed	1/9/17	Adam
SPMP Draft 1	Completed	5/9/17	Everyone
Milestone Form	Completed	5/9/17	Everyone
Project Management Plan Review	Completed	12/9/17	James
Risk Management Plan Review	Completed	12/9/17	Jeremy
Configuration Management Plan Review	Completed	12/9/17	Linlin
SDD Review	Completed	10/10/17	Ziang
Code Review	Completed	10/10/17	Cyrus
Testing Review	Completed	10/10/17	Ravi
User Manual	Completed	30/10/17	Ziang, Adam
Final versions of all documentation	Completed	30/10/17	Ziang, Adam

8.3 Quality Assurance Plan

Our quality control process uses frequent quality review sessions to ensure that developed code and documentation adheres to our standards. These quality reviews will be held after the code and documentation for each milestone has been completed. Quality reviews will consist of a code inspection presented as a structured walk through by the writer of the code, and will be reviewed by the rest of the project team. After a quality review, code sections or documents will be assigned one of either *no action required*, *cleanup required*, or *redesign required*, along with further details provided to the author of the code or document. It will be the task of the code or document author to change their work to adhere to the quality standards of the project.

Code quality will be evaluated by it's adherence to code style and software architecture design. The coding style used for the project is the Java Code Conventions [1]. Additionally, we require JavaDoc comments to be written for every class and method in the project to ensure exhaustive API documentation, and unit test cases to be written for each function where it is practical. During quality reviews, the quality of the code implementation will be evaluated based on it's readability. The behavior of the code should be immediately obvious when the code is read; if it is not, short comments should be used to explain parts of the code.