Test Report

Software Engineering and Project

University of Adelaide

Prospector Sea Floor Mapping System (PG04)

Navdeep Singh (1660360)
Liang Yuan (1679380)
Zeqi Fu (1680895)
Tao Zhang (1680974)
Lili Wu (1683229)
Yi Lin (1682781)
Yann Frizenschaf (1162562)

Semester 2, 2016

Revision History			
Date	Version	Reason for Change	Author
19th Oct 2016	0.1	Created a initial template	Zeqi Fu
20th Oct 2016	0.2	Added section 2	Zeqi Fu
28th Oct 2016	0.3	Added Regression Test and rearranged format	Lili Wu
28th Oct 2016	0.4	Format and content fix	Zeqi Fu
29th Oct 2016	1.0	Add unit test section and edit for final release	Yann Frizenschaf

Table 1: History

Contents

1 Introduction

1.1 Purpose and overview

The purpose of this document is to record the process and outcomes of testing performed on the Prospector Sea Floor Mapping (SFM) system. The test cases in this document were developed to ensure the software meets the requirements set out in [2], and verify the system as a close as practical to being fault-free. The tests documented here include system tests, unit tests and regression tests. The results documented here are up-to-date as of the version 1.0 release of the software.

2 Test items

2.1 Unit tests

Various parts of the software are covered by automated unit testing. These tests are designed to validate the output of the public methods available in a particular class, given a set of inputs and known state. These tests are primarily used to identify undiscovered bugs and also to ensure that new bugs aren't introduced when functionality is changed.

Unit test development was prioritised for classes which contain the most business logic, that is, the classes that contain the majority of the system's "smarts". Unit tests must be executable without connection to the real robot hardware, and as such are unsuitable for hardware calibration and testing.

The classes for which unit tests have been developed are:

MappingController: Contains logic for updating the map data model based on robot position, orientation, and sensor readings. Unit tests validate model update functionality under under a number of input conditions.

Parser: Converts internal map data model into XML document and vice versa. Unit test verifies that a pseudo-randomly generated map can be saved to file and read from the same file without loss of data integrity.

Navigation: Contains logic for performing A* search algorithm to determine suitable navigation path between two points. Unit tests validate generated navigation paths.

MovementController: Responsible for robot movement and location estimation based on user input, sensor input and navigation logic. Unit tests validate position estimation and destination derivation logic.

2.2 System Test

System tests aim to validate the behaviour of the system against the requirements defined in [2]. Table 2 contains details of the conducted systems tests including test steps, expected and actual results, and SRS requirements traceability.

Execution steps	Expected results	Actual results	Related functions
1.Put robot in start position; 2.Connect PC to robot; 3.Start robot moving forward.	1.There's an estimate of the SeaTank's current position shown on PC screen; 2.The robot doesn't go to any No-Go-Zone.	PASS	FRC001 FRC004
1.Turn the robot to autonomous mode; 2.Make the robot go forward a certain direction, exceeding the faultline.	The robot changes direction by a significant margin inside the faultline.	PASS	FRC002
1.Turn the robot to autonomous mode; 2.Make the robot go forward an obstacle.	The robot changes direction by an appropriate distance to the obstacle.	PASS	FRC003
1.Put robot in start position; 2.Connect PC to robot; 3.Manually make the robot move forward.	The robot moves forward.	PASS	FRC005

1.Specify a point in survey-area coordinate; 2.Make the robot move to the point.	The robot moves to the specified point avoiding any obstacle automatically.	The robot didn't move.	FRC006
1.Instruct the robot to begin the mapping operation; 2.Make the robot move to the point.	The robot moves to the specified point avoiding any obstacle automatically.	The robot didn't move.	FRC007
1.Make the robot move forward a certain direction.	1.The map data is output; 2.The estimated direction is informed at the same time.	PASS	FRC008
1.Provide the current map data model to the GUI	The update of display finishes in one second.	PASS	FRC009
1.Turn the robot to Move-to-point mode; 2.Make the robot reach the specified position.	The time consumed is within 20 minutes.	PASS	FRC010
1.Make the robot keep moving	1.The map is created appropriately. 2.The XML format file is created appropriately.	PASS	FRC011 FRC012 FRC013
1.Make the robot move towards an obstacle; 2.Make the robot move towards a faultline; 3.Make the robot move towards a boundary.	1.The map location error is no more than 10cm. 2.The map is created correctly.	The robot stopped when encountered an obstacle instead of turning around.	FRC014 FRC015

1.Make the robot keep moving and make sure the colour sensor detecting different colours.	The map created based on depths are correct.	PASS	FRC016
1.Make the robot keep moving.	The position estimate is corrected automatically.	The robot corrected the angle too much, it was turning around continuously rather than moving forward.	FRE001
1.Make the robot keep moving; 2.Manually input the position data.	The robot accepts the inputs.	PASS	FRE002
1.Make the robot move from the start position to a specified position.	The mapping operation compeltes within 10 minutes.	NA	FRE004
1.Make the robot move toward a boundary.	The robot avoids keeping moving ahead.	NA	FRE005

Table 2: System Test

2.3 Regression Test

Regression tests aim to re-validate basic functionality to ensure it has not been compromised by subsequent changes to code or configuration. The results of the final release round of regression testing are shown in Table 3.

Execution steps	Expected results	Actual results	Related functions
1.Specify a point in survey-area coordinate; 2.Make the robot move to the point.	The robot moves to the specified point avoiding any obstacle automatically.	PASS	FRC006
1.Instruct the robot to begin the mapping operation; 2.Make the robot move to the point.	The robot moves to the specified point avoiding any obstacle automatically.	PASS	FRC007
1.Make the robot move towards an obstacle; 2.Make the robot move towards a faultline; 3.Make the robot move towards a boundary.	1.The map location error is no more than 10cm. 2.The map is created correctly.	PASS	FRC014 FRC015
1.Make the robot keep moving.	The position estimate is corrected automatically.	PASS	FRE001

Table 3: Regression Test

3 Glossary

GUI Graphical User Interface

LeJOS The Lego Java Operating System

RMI Remote Method Invocation

SFM Sea Floor Mapping

SRS Software Requirements Specification

4 References

[1] Milanese, D and Weerasinghe, A 2016. *Software Engineering and Project - SeaFaults Mapping Robot*. Project specification version 1.0

[2] Prospector Team 2016. *Software Requirements Specification: Prospector Seafloor Mapping System*. Version 2.0.

[3] Buschmann, F, Henney, K and Schmidt, DC 2007. *Pattern-Oriented Software Architecture: On Patterns and Pattern Languages*, Volume 5. Wiley and sons, West Sussex, England.

[4] Hürsch WL and Lopes CV 1995. *Separation of Concerns*. Northeastern University, Boston, USA. Technical report NU-CCS-95-03.

[5] LEGO Education, no date. *EV3 Large Servo Motor*. Available online 1/10/2016 <https://education.lego.com/en-us/products/ev3-large-servo-motor/45502>

[6] LEGO Education, no date. *EV3 Medium Servo Motor*. Available online 1/10/2016 <https://shop.lego.com/en-US/EV3-Medium-Servo-Motor-45503>

[7] LEGO Education, no date. *EV3 Ultrasonic Sensor*. Available online 1/10/2016 <https://shop.lego.com/en-US/EV3-Ultrasonic-Sensor-45504>

[8] LEGO Education, no date. *EV3 Color Sensor*. Available online 1/10/2016 <https://education.lego.com/en-us/products/ev3-color-sensor/45506>

[9] LEGO Education, no date. *EV3 Gyro Sensor*. Available online 1/10/2016 <https://education.lego.com/en-us/products/ev3-gyro-sensor-/45505>