Software Requirement Specification

SEP UG07

Software Engineering and Project
The University of Adelaide
21 October 2018

version 2.0

a1758818 Cameron Youngblood

a1693458 Jiming Li

a1687323 Cadeau Maniragaba

a1694970 Zaifeng Wang

a1696425 Chao Zhang

a1685651 Erik Praekelt

a1658197 Byeongjun An

Table of Contents

1. Introduction

1.1 Purpose and Scope

To create a prototype robot using the Lego Mindstorm EV3 kit that autonomously locates and digitally marks the positions of humans and map of its path in a designated operating area.

1.2 Intended Audience

The intended audience will be the client which is University Natural Disaster Search and Rescue Service team. The client will need to read through all the requirement specification. The other audience is the operator , with an aim on the use case and project description

2. Product Description

The robot needs to collect data of it position and path in order to search and digitally mark survivors as part of information gathering in a rescue operation. The operator must also have the ability to manually override and control the robot.

2.1 Core Features

The following core features are required.

1. A robot which moves within bounds and collects data
2. Identification and recording of objects found
3. Obstacle avoidance and map traversal within set bounds
4. A graphical user interface (GUI) displaying collected data
5. Mapping of survivor and terrain data on the GUI
6. Ability to manually override and control the robot via GUI

2.2 Functional Requirements

1. The robot program aims to get the location of human who is waiting to be rescued after a disaster and the map of the path it took to get there. However,the robot is not designed to rescue them.
2. Each count, the maximum runtime for the robot to search the whole map should less than 20 minutes.
3. The robot must search inside a given area (with GPS-painted boundary).
4. The robot program should provide a user interface that allow users to turn off the automatic driving mode to manually control the robot in four directions: left, right, front and back.
5. The program should allow users to set the boundary manually.
6. The robot must avoid the object, boundary and the human in the process of searching the given area.
7. The robot program should be able to search a minimum 2x2 city block.

2.3 Non-Functional Requirements

1. Given a reasonably computer literate user, the graphical user interface is easy to understand and use.
2. The robot program shall be reliable enough to avoid collision (e.g. hit the boundary/object).
3. Efficient algorithm to take action when get data from color sensor (< 100 milliseconds).
4. The complexity of the code should be less than O(n^3) so it's easier to maintain and debug.
5. The GUI interprets the code and the action of the robot on the physical map.

3. User Requirements

3.1 Robot Movement

3.1.1 RM001: Moving Autonomously Around a Given Area

Description: The robot should be able to autonomously move around and stay within a given area to detect survivors without a user giving it directions and return back to its starting point when it is done.
Rationale: The robot should be able to automatically find survivors.
Acceptance criteria: The robot should move around a given area without user intervention and stay within the boundary of the given area.
Source: Based on the description of project and the client meeting.
Priority: High

3.1.2 RM002: Control the Robot Manually in GUI

Description: The robot program should provide a user interface that allows users to switch the robot from autonomous driving mode to manual driving mode to allow the user to manual control the robot.
Rationale: This helps the allows for a user to free the robot in case it is stuck or to simply allow a user to control the robot themselves.
Acceptance criteria: This requirement can be verified by observing the action of the robot to see if it responds to a direction request from user For example, if the user clicks on the left button, the robot is should move left.
Source: Based on the description of the project.
Priority: High

3.2 Robot Object Detection

3.2.1 ROD001: Have the Robot Generate a Map

Description: The robot should generate a map to be shown on the GUI to show populate the area the robot has traversed, including survivors and obstacles.
Rationale: A map should be generated by the robot to help the UNDSRS team locate survivors when rescuing them, while avoiding obstacles in their way.

Acceptance criteria: A map with an accurate population of survivors and obstacles should be displayed in the GUI.
Source: Based on the description of project and the client meeting.
Priority: High

3.2.2 ROD002: Detecting and Identifying Objects

Description: The robot is supposed to detect all objects with the color sensor, and identify the objects using the different colors that represent each type of object.
Rationale: In order to avoid a collision with objects and survivors.
Acceptance criteria: The robot takes the information from the color sensor, and identifies what the object is based on its color.
Source: Based on the description of project and the client meeting.
Priority: Medium

3.2.3 ROD003: Send a Signal When a Survivor is Detected

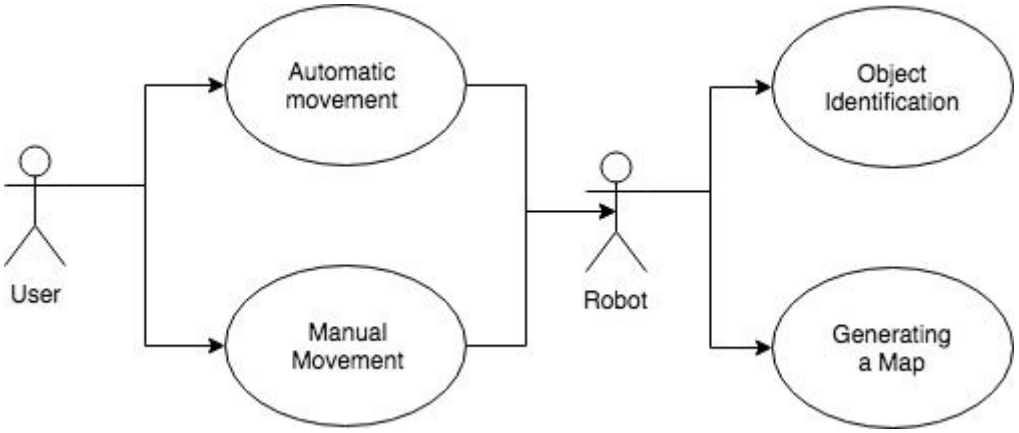
Description: When the color sensor detects a color that represents a survivor, the robot will beep and send the location of the survivor back to the robot operators.
Rationale: To help the UNDSRS team be able to locate and save survivors.
Acceptance criteria: The location of the survivor should be sent to the team and an audible beep should be heard from the robot.
Source: Based on the description of project and the client meeting.
Priority: High

3.3 Robot Quality

3.3.1 RQ001: Operating Time for the Prototype

Description: The robot should be able to automatically traverse the map within 20 minutes.
Rationale: The robot should be able to traverse a map in a reasonable amount of time to represent real world scenarios.
Acceptance Criteria: The robot finishes the map within 20 minutes.
Source: Based on the description of project and the client meeting.
Priority: Medium

4. Use Cases



4.1 Manual Robot Movement

Description: The robot can be operated manually by a user who wishes to manually search the area or move the robot around manually.
Actors: The user can be anyone who wishes to search the area to detect survivors manually or move the robot around manually.
Preconditions: This use case assumes the user can use the robot to detect survivors in the specific area.
Basic Flow

1. The user selects the manually robot mode.
2. The user moves the robot around using the corresponding buttons in the GUI depending on how the user wants the robot to move.
3. The robot logs an obstacle or survivor if it finds one and adds it to the GUI map.

Alternative Flows

- The user can move the robot back to its last autonomous point without losing any information

Exception Flows

- The user can switch the automatic or manual movement mode

4.2 Automatic Robot Movement

Description: The robot can be moved automatically in the specified area
Actors: The robot is supposed to move around within the boundary setup by user automatically.
Preconditions: This use case assumes the user can use the robot to detect survivors and obstacles automatically in the spespecified area.
Basic Flow

1. 1. The user uploads the search area to the robot.
2. The user selects the automatic movement mode.
3. The robot begins to move around the search area.
4. The robot collects the information of area throughout the automatic movement and sends the location of survivors and obstacles back to the GUI.

5. When the robot is finished traversing the map, it ends operation.

Alternative Flows

- The robot can go back to the starting point without losing any information

Exception Flows

- Dead Battery
- The robot loses its position

Post Conditions: The must upload a map to the robot for it to be able to search an area.

4.3 Object Identification

Description: The robot will identify all objects, which include obstacles and survivors, it encounters.

Actors: The objects are anything robot finds while searching for survivors, the objects being obstacles or survivors. This includes stone, metal, humans, etc.

Preconditions: This use case assumes robot can identify all objects found.

Basic Flow

1. The robot detects objects using the color sensor.
2. The robot collects the color of objects found with the color sensor, which signifies if the object is an obstacle or survivor.
3. The robot identifies objects and adds it to the map in the GUI.
4. The robot turns around to avoid the objects detected.

Alternative Flows

- The robot operates normally if there are no objects detected.

Exception Flows

- Dead battery
- The robot loses the position
- The color sensor is faulty.

Post Conditions: Robot must have the logic to understand which color signifies an object or not.

4.4 Generating a Map of Obstacles and Survivors by the Robot

Description: The GUI needs to show the path the robot takes and pinpoint the where the survivors and obstacles are located.

Actors: The objects are anything robot finds while searching for survivors, the objects being obstacles or survivors. This includes stone, metal, humans, etc.

Preconditions: This use case assumes robot can create the map using collected information and display it in the GUI.

Basic Flow

1. The robot searches the area.
2. The robot collects the information of the area it traverses.
3. The robot collects the information of any obstacles and survivors it finds.
4. The robot sends the information back to the terminal to displayed in the GUI in real time.

Exception Flows

- Dead battery
- The robot loses the position
- Manual control does not show the area the robot traverses.

Post Conditions: The robot must have an area to search and understand the different type of objects that can be detected.