Software Design Document

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 Overview

This document meant is to provide an overview of the robot's software functions as well as the functions of the user GUI.

1.2 Intended audience

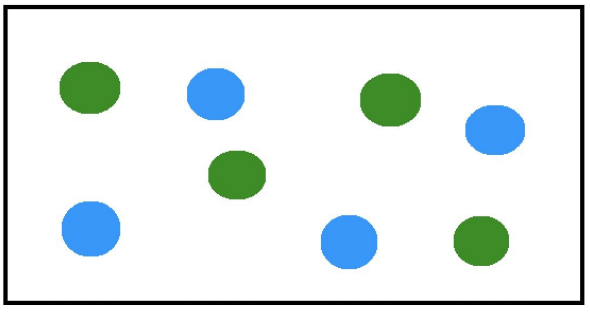
This document is aimed at the software development and management team. UNDSRS may also reference this document for project specification verification.

2. Components Design

To simulate survivor detection, a 2D map will be used with differently coloured markers, emulating obstacles and survivors. The robot will use sensors to detect and distinguish items in this map environment and behave accordingly. A manual override is also necessary. Hence, there are four distinct components: the map, the robot hardware, the controller software and the graphical interface. The following sections outline these components and interactions in detail.

2.1 Map

The map will simulate a real-world example of a disaster scenario where an environment is made up of a boundary, various obstacles and survivors to be found. As such, the prototype map will consist of a boundary painted on a large flat surface (e.g. the ground). Within the boundary, differently coloured markers will emulate obstacles and survivors. Such an example is in the diagram below.



For the robot to distinguish sensory encounters, each colour has a different meaning, as follows:

- Green - Survivor
- Blue - Obstacle
- Black - Boundary

On encountering these markers, the robot will behave accordingly, as outlined in Section 2.3.

2.2 Robot Hardware

The Robot Hardware will be constructed with the Lego Mindstorm EV3. It's modularity and easy part interchangeability is good for prototyping and will simulate a real-life version of the conceptual survivor detector robot.

2.2.1 Mobility

The robot will consist of an independent two-wheel drive system and trailing stabiliser wheel. In other words, the two front wheels run on independent motors while a rear "floating" wheel provides stability and maneuverability. With two independent motors on the front wheels, we can control the direction of travel by adjusting the power output of each wheel. E.g. a left wheel at 50% and right wheel at 70% will see the robot circling anti-clockwise (left). This also allows the robot to change its direction of travel on the same spot.

Surface friction differences often cause theoretical rotation angle and actual rotation angle to differ significantly. As a result, digital controlling and mapping will not be calibrated. To assist in this issue, a gyro will be added. The information from the gyro will be fed into the controller to help ensure rotation motor power is sufficient to achieve the desired rotation.

2.2.2 Sensor Array

In order to detect the coloured markers on the map outlined in Section 2.1, the robot will use a colour sensor. This sensor will be mounted on the front of the robot and face down to ensure the robot detects the markers before physically encountering them.

As outlined in Section 2.2.1, to facilitate better mobility (specifically rotation), a gyro will be added to the robot. Information from the gyro will allow the Controller to offset power output to the motors appropriately, allowing it to rotate the robot correctly for the surface it's operating on.

2.2.3 Connectivity

It is possible to link the robot hardware with a computer via cable or Bluetooth. While cable is reliable, convenient and good during the development process, Bluetooth allows autonomy. This autonomy is crucial for an autonomous robot. However, the ability to override this autonomy is equally important. I.e. switching to manual mode via the user interface (covered later in this document).

2.3 Robot Controller Software

The Robot Controller Software is responsible for processing input data provided by the robot hardware, specifically the sensors. The controller will also output data to the robot, allowing the robot to operate. The be definitive, the controller is the program that is uploaded to the robot hardware and interacts directly with the hardware during operation. This controller is

what provides the robot's autonomous capability. It will define how to interpret the sensor data and what to do next. The behaviours are defined below.

2.3.1 Green Encountered

A green marker emulates a survivor. When the colour sensor encounters green, the program should log and store the location of the marker, while indicating its location on the digital map (reference Section 2.4). An avoidance routine should then be activated to ensure continual searching without running over the marker (survivor).

2.3.2 Blue Encountered

A blue marker emulates an obstacle. When the colour sensor encounters blue, the program should log and store the location of the marker, while indicating its location on the digital map (reference Section 2.4). An avoidance routine should then be activated to ensure continual searching without running into the marker (obstacle).

2.3.3 Black Encountered

A black boundary emulates the operating area boundary. When the colour sensor encounters black, the program should log and store the location, while indicating the location on the digital map (reference Section 2.4). An avoidance routine should then be activated to ensure continual searching without exiting the operational area.

2.3.4 Manual Override

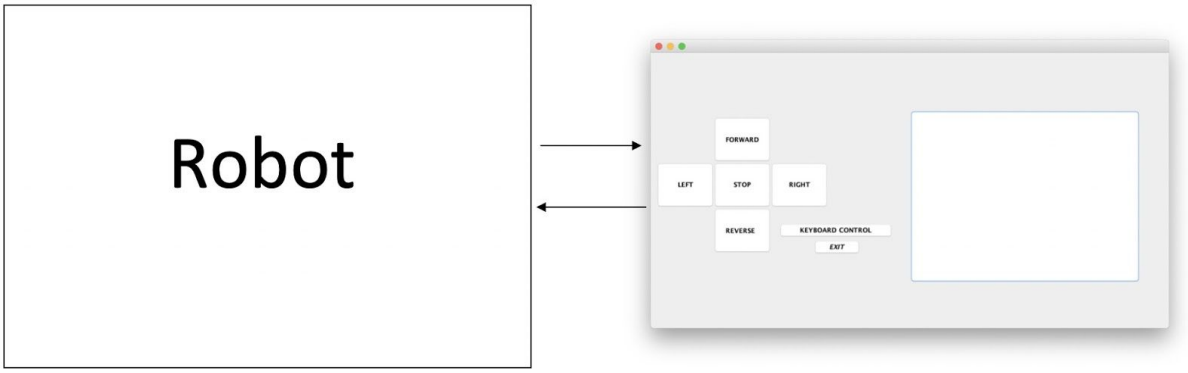
It must be possible to manually override the autonomy of the robot via graphical interface. When such an override command is given, the robot should immediately terminate its current motion and execute the command received via cable or Bluetooth. Manual override is outlined in more detail in Section 2.4.

2.4 Graphical Interface Software

The Graphical Interface is a means for the operator to remotely track and interpret data collected by the robot. It should also boast the ability to manually override current autonomous operations. The interface is outlined in detail in Section 3.

2.4.1 Manual Override

A Manual override capability is crucial; it allows the operator to interrupt any autonomous operations currently underway. It will also allow manual movement control, while at the same time logging any objects it has found.



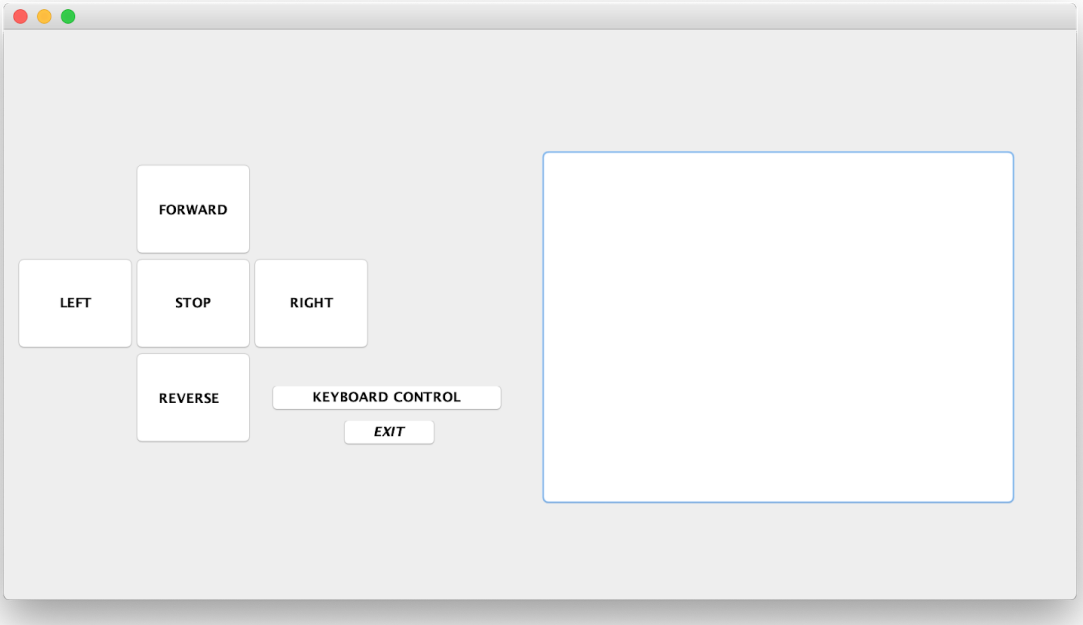
User GUI

3. Graphical Interface

The graphical interface will be situated on the operators terminal (computer). It will allow the operator to manually override autonomous robot operations as well as see the location of robot and the objects it has detected.

3.1 Design

Our design for the interface is shown below:



3.2 Graphical Operations & Features

The interface will feature a number of buttons and information displays. This will include directional buttons, the robot populated map, with the area it has traversed, its current location, as well as the location of objects, as well as keyboard control, stop, and exit buttons.

3.2.1 Actions

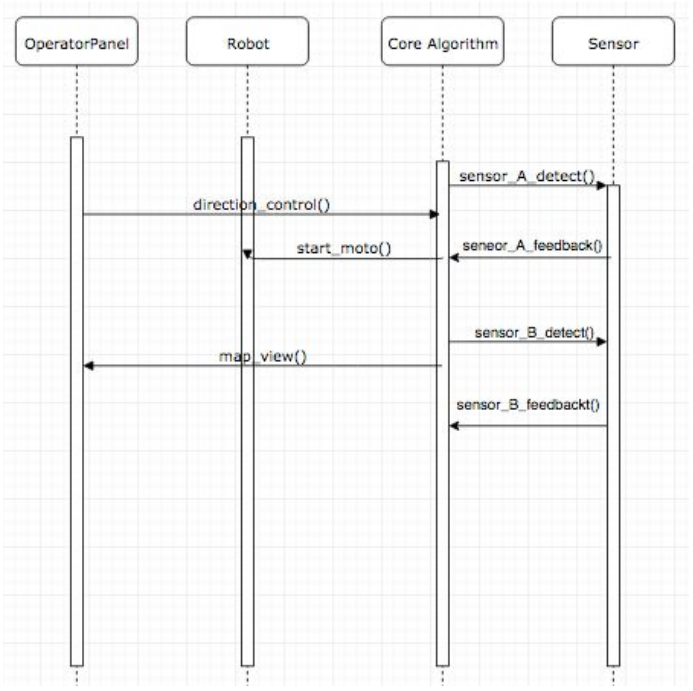
The following are interactable items displayed on the interface and each button's purpose is detailed below:

- Keyboard Control: Allows for the keyboard directional keys to control the robot.
    - Up-Forward
    - Down-Backwards
    - Left-Left
    - Right-Right
- Exit: Exits manual robot control.
- Movement Buttons: Instructs the robot to move in the relevant direction.
    - Includes the buttons Left, Right, Forward, and Reverse
- Stop: Forces the robot to stop its current movement.

4. Data Design

4.1 Operator Interaction

The following is an example of the interaction process between operator and robot during manual control.



4.2 State Diagrams

This diagram explains the basic functions of the robot and what it is supposed to do.