Software Engineering and Project

Software Design Document

UG-14

a1647816 Anthony Scaffidi
a1693014 Jingbin Zhang
a1710974 Jinshan Wang
a1727491 Lincoln Phillips
a1687940 Mitchell Phillips
a1696673 Nanye Zou
a1669907 Shaun Kraemer

0.3

21/10/2018

Version History

Date	Version	Changer	Description
25/08/18	0.1	Lincoln Phillips	Document created / Introduction section completed (awaiting review)
26/08/18	0.1	Lincoln Phillips	Human interface Design section complete
27/08/18	0.1	Lincoln Phillips	Components design section completed
21/10/18	0.2	Jingbing Zhang, Jinshan Wang	Modify document layout Inner Interface design description completed
21/10/18	0.2	Nanye Zou	Check the interface section and fixed
21/10/18	0.2	Shaun Kraemer	Modified the Human Interface Design section to more accurately describe the final GUI design
21/10/18	0.3	Lincoln Phillips	Rehashing of component design and additional changes
21/10/18	0.3	Mitchell Phillips	Added path finder algorithm documentation and updated system design.
21/10/18	0.3	Lincoln Phillips	Proofreading and spell check

1. Introduction

1.1 Overview

This purpose of this document is provide a holistic and detailed overview of both the system design and human interface for the autonomous rescue robot using the Lego NXT EV3 kit in conjunction with leJOS.

In the component design section details will be provided for each individual system required in the development and operation of the robot, as well as a connectivity diagram to illustrate how the respective subsystems will communicate with each other. These communication pathways will also be representative of the streaming data flow within the software to allow for graphical representation (in the human interface) of the purpose of the robot; to locate survivors in the process of disaster relief.

In the human interface section a screenshot of the graphical user interface (GUI) will be presented and subsequently have its individual components explained in accordance with the purpose of the robot. These components have been developed in accordance with client demands and user experience (UX), and seek to fulfill simplistic design that is easy to understand.

1.2 Intended audience

This software design documentation is intended for viewing by the members of the 2018 Software Engineering & Project group UG-14 as well as the client - The University Natural Disaster Search and Rescue Service (UNDSRS).

1.3 Glossary

Define any acronyms or terms used in this document

GUI	Graphical user interface
UX	User experience
leJOS	Java-based firmware for use on Lego NXT development kits
IPC	Intra-process communication

2. Components Design

2.1 Component Decomposition Description

Fig.1 of Appendix A illustrates a diagram of the decomposed system/subsystem. This diagram shows our design of a two-system model, which separates the responsibilities between a client and a server. Our LejOS EV3 robot operates as a TCP server, and a separate computer on the same local network acts as the remote client. The remote client primarily consists of a user interface, which is designed to be operated by the customer. The individual components of this diagram are explained below:

Server/Client Communication
Communication between client and server is done through TCP sockets using a custom intra-process communication (IPC) language. This is a text-based language, primarily consisting of the following simple commands:

- explore_next
    - Instructs the robot to explore the next available (i.e. reachable) grid space. This will instruct the robot to route to the location and scan it.
- encoded_map / encoded_map_end
    - Notifies that client that next series of bytes will represent the wire-format state of the current map. This contains all the exploratory data from the robot about its environment. The wire-format map is finalised by sending the "encoded_map_end" command, which notifies the client that the transmission is completed.

These messages are understood by the IPC clients present on both the robot (server) and client. The IPC client will decode the requests and issue the requested event to the correct component.

User interface / GUI
This system is the client's input portal, further explained by section 2 of this document. The user interface allows the user to enable automated exploration by the robot.

The User interface has two primary components - an input (to the subsystems) for client-controlled functions and processed data output for reporting and representation of data. The primary (and only) use case for the input of commands is via buttons built into the interface, however the processed data has several streams for status representation (as explained in *Human Interface Design*). The nature of these data paths will be via wifi or bluetooth (yet to be determined for the purpose of this draft).

Testing suite / software

Integration testing is achieved through dependency injection. We deliberately abstract the usage of the LejOS native hardware functions (as seen in the sensor and movement controllers), which allows us to inject stubs that emulate this behaviour. This allows us to have direct control over the signals that are sent to the remainder of the robot system.

Direct testing of the robot is possible without initialisation of the client side code. We can simply emulate the IPC channel directly, simulating a client's actions. We compare the responses of the server with an expected set of values. This allows for more direct control over signals sent to the driver program (detailed below) for more precise manipulation of factors in development, as well as debugging / process information to be shown while allowing for separate, independent creation of the user interface that will not need to be modified if bugs are discovered.

The data paths presented are similar, but more verbose than that from the user interface (detailed above). This will allow developers (i.e. our team) to more precisely manipulate the robot and receive additional dialogue and data from the robot during development and testing.

Dynamic Map

This storage component is responsible for providing a unified storage method between client and server. Our storage blobs are duplicated between client and server through the IPC to allow the client to directly render the content. This component stores information about the environment that the robot has ascertained through exploration. The map is responsible for storing information about obstacles, their type (obstacle or survivor), where the robot is currently located, and border provinces.

Sensor Controller

This is the interface over the sensors attached to the body of the device, and will send data through wiring to the Lego NXT EV3 controller.

Movement Controller

This is the interface over the driving motors responsible for the movement of the robot. These require no data reporting, as they are simply given an action by the Lego NXT EV3 controller. Henceforth they are only required to have one data path, to the motors themselves. This is delivered by hard wires from the controller above.

Path Finder

This component provides most of the functionality of the robot. It processes sequential exploration commands, where the client has requested an additional square to be explored.
A sequential exploration command is processed using the following algorithm. We always choose to observe the closest unknown location to the robot, based on manhattan distance. The path finder is responsible for planning a route to the square to be observed using the information present in the robot's map database. It then will proceed to actually traverse that route using the movement controllers, and then utilise the sensor controllers to scan the node to be explored. If the scan with the ultrasonic sensor detects an object, the path finder will attempt

to use the colour sensor to detect the type of the object (survivor or obstacle). It the ultrasonic sensor does not detect an object in the scanned square, then we carefully attempt to drive the robot to the scanned square. If, during the movement, we detect the presence of a border, we retreat to the last known good square and mark the observed square as a border. This algorithm is shown in detail in Figure 5, Appendix A.

3. Human Interface Design

3.1 Overview of the User Interface

The User Interface (GUI) is designed in a way that combines simple design with easy functionality. It was primarily designed with the client's requirements in mind (detailed in the following section) however use of colour, spacing, and positioning seek to create an interface that is intuitive and easy to use.

The GUI will make use of streaming data to present current location and orientation of the robot in the form of a navigational map, as well as all details regarding past discoveries and survivor information. This will fulfill the requirements of the robot, being differentiating between obstacles and survivors, location of both, and map boundaries, and present it in an intuitive and easy to understand format.

3.2 Detailed Design of the User Interface

Fig.2 in Appendix A illustrates an initial mockup of the Graphical User Interface (GUI) as first envisioned. This mockup incorporates a panelled design aiming to separate user input (left panel) with robot output (right panel). Staying true to this mockup the final design shown in Fig.3 in Appendix A, only differs slightly, consisting of the following components:

1. Left Panel - user input panel
    - 'CONNECT TO ROBOT' button
    - 'RUN AUTOMATIC' button
    - 'KEY' panel
        - 'Robot' label and image
        - 'Border' label and image
        - 'Obstacle' label and image
        - 'Survivor' label and image
        - 'Explored' label and image
        - 'Unknown' label and image
2. Right Panel - robot map output panel
    - Streaming map output panel
3. Top Panel - robot console output panel
    - Inform users of the process currently being performed by the robot.

The actions and reports of these components can be broken as follows:

3.2.1 Left Panel - User input panel

‘CONNECT TO ROBOT’ button
This button is the main source of connection to the robot - it allows for streaming data to be passed to and from the robot for manipulation and reporting criteria. This connection (and henceforth streaming data) will be passed to and from the robot over the bluetooth network.

‘RUN AUTOMATIC’ button
This button is the primary button for the robots operation. It systematically runs the two methods of the robot: Defining the map and detecting obstacles, and then defining the obstacles. As the robot performs these actions it will generate and populate the streaming map (right panel) with all of the required information for the client to process the robot’s report. This button is representative of R0001 in the Software Requirements Specification Document.

3.2.2 Right panel - robot output panel

Streaming map output panel
The purpose of this panel is to create a graphical representation of R0003 of the Software Requirements Specification - virtual mapping (storage). This panel creates a 2-dimensional representation of the area defined by the robot via streaming over bluetooth and populates it based on the discovery of boundaries, obstacles, and survivors by the robot. The purpose of this interface is to create intuitive and easy to reference information that can be used by disaster teams in their survivor recovery efforts, and will contain elements as defined by the **key** (left panel) as they are discovered. As elements are discovered by the various sensors of the robot, they are added to the map.

Key
The key is a reference to all objects as portrayed in the **Streaming map output panel** (right panel) and includes their meanings.

Map artifacts
Represented by Appendix A fig.4, the artifacts used in the graphical representation of the map in the user interface are as follows:

- **Border image:** Used to denote the ‘edge’ of the map, these artifacts are used to represent impassable sections of map
- **Explored image:** These artifacts denote grid spaces that have been explored, and contain nothing. These artifacts also serve to represent areas of the map that the robot can freely move through.
- **Obstacle image:** These artifacts denote identified obstacles on the map, and are considered impassable by (i.e. cannot be moved through, nor occupy the same space as) the robot.

- **Robot image:** This artifact denotes the current location of the robot.
- **Survivor image:** This artifact denotes identified survivors on the map, and are also considered impassable by the robot.
- **Unknown image:** These artifacts denote currently unexplored areas of the map, and are generated dynamically by the mapping algorithm upon the robot discovering new unexplored zones.

3.2.3 Colour and text representations in buttons

Both colour and text representations of status are implemented in the left (user input) panel of the user interface (GUI). This is to create a more interactive and user-oriented experience based on common design principles. The status of relevant buttons is altered upon the user interacting with them, giving indication of the change of state of the robot as well as limiting actions being performed simultaneously, or without prerequisite checks being done. The colour and text representation of status for the buttons are as follows:

Colours:
Grey: As seen in fig.2 of Appendix A, Procedural units panel, 'DEFINE OBSTACLES' button. Grey indicates this button is not able to be used at this stage, as prerequisite states (for example, lack of completion of object detection) have not yet been completed.

Green: As seen in fig.2 of Appendix A, 'RUN AUTOMATIC' button. Green indicates successful completion of the appropriate state.

Red: As seen in fig.2 of Appendix A, Manual override panel, 'MANUAL OVERRIDE' button. Red indicates that the button is available to be used by the user, however is not currently used.

Blue: As seen in fig.2 of Appendix A, 'CONNECT TO ROBOT' button. Blue is representative of a successful connection made, and is exclusive to the 'CONNECT TO ROBOT' button. This colour follows the UX idea that bluetooth (the proposed connection mechanism) is represented by the colour blue, however this may be changed in the final GUI.

Text:
Text is altered for the buttons based on their current status to give clear indication of the current state of the robot. For example, after a successful connection is made the 'CONNECT TO ROBOT' button will change text to 'ROBOT CONNECTED'. This alteration in text serves to have ambiguity of status lowered, by clearly defining the current state of the robot.

Appendix A

Fig.1, Component Decomposition Diagram

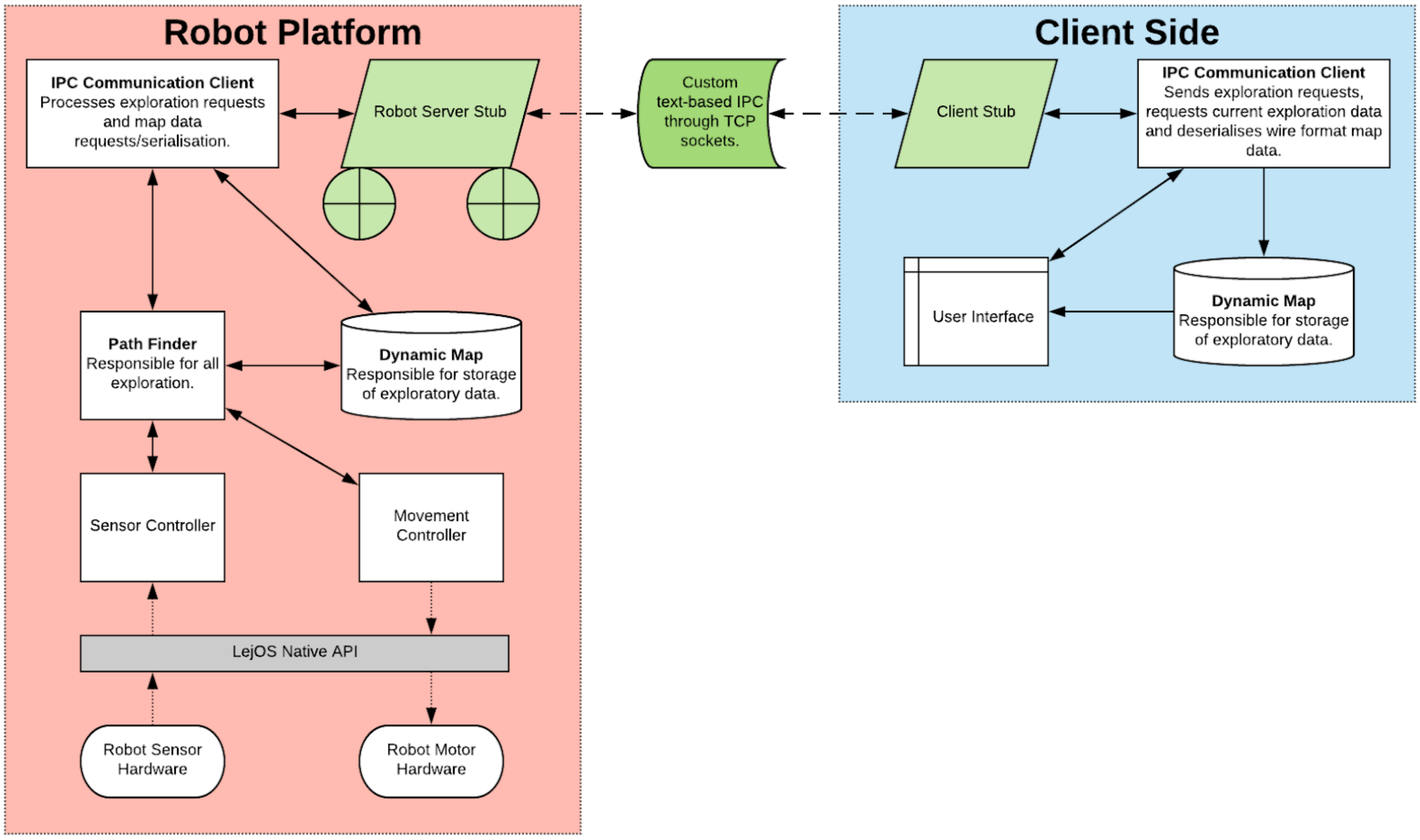

Robot Platform
IPC Communication Client
Processes exploration requests and map data requests/serialisation.
Robot Server Stub
Path Finder
Responsible for all exploration.
Dynamic Map
Responsible for storage of exploratory data.
Sensor Controller
Movement Controller
LejOS Native API
Robot Sensor Hardware
Robot Motor Hardware
Custom text-based IPC through TCP sockets.
Client Side
Client Stub
IPC Communication Client
Sends exploration requests, requests current exploration data and deserialises wire format map data.
User Interface
Dynamic Map
Responsible for storage of exploratory data.

Fig.2, User interface (GUI) mockup

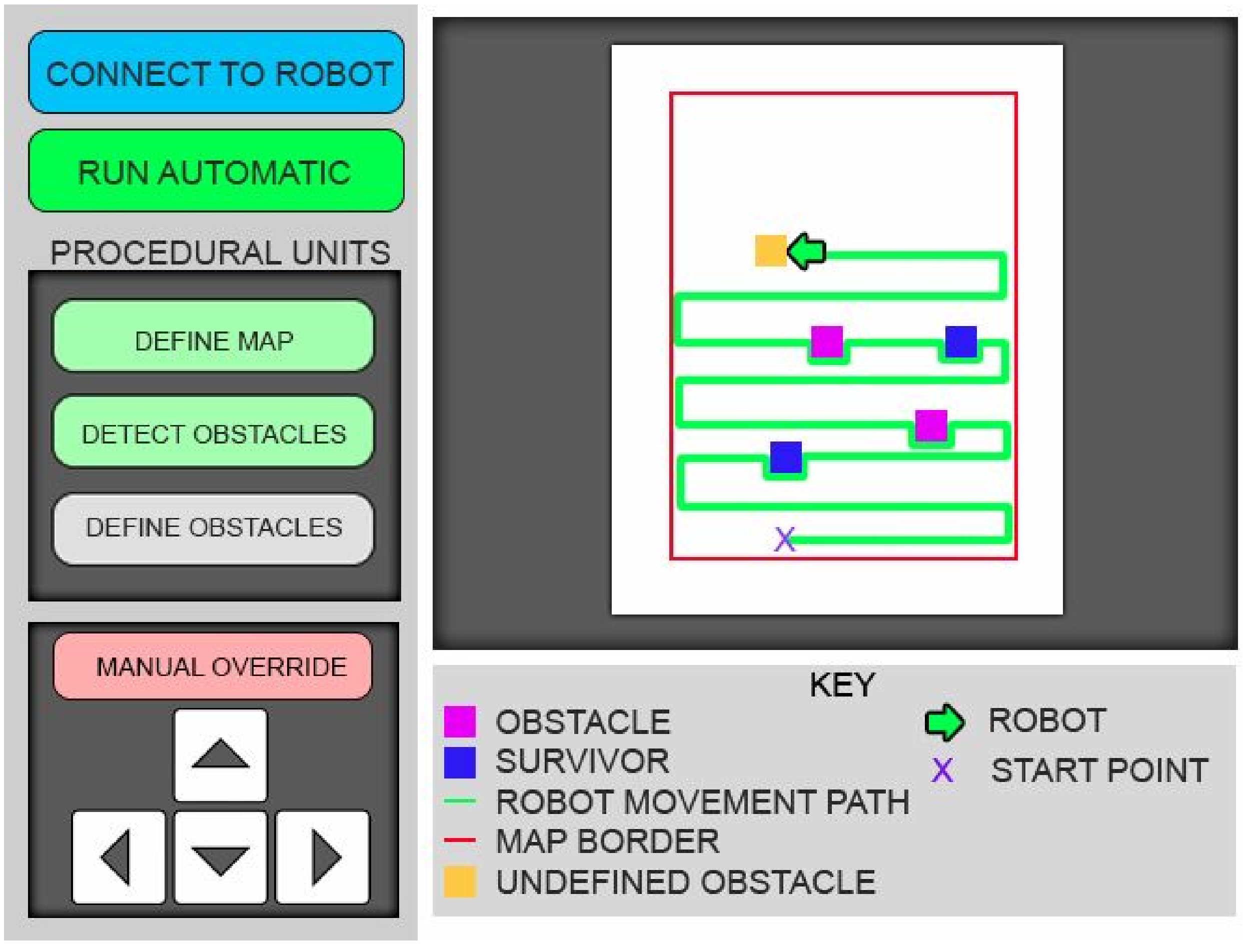


Fig.3, User interface (GUI) final

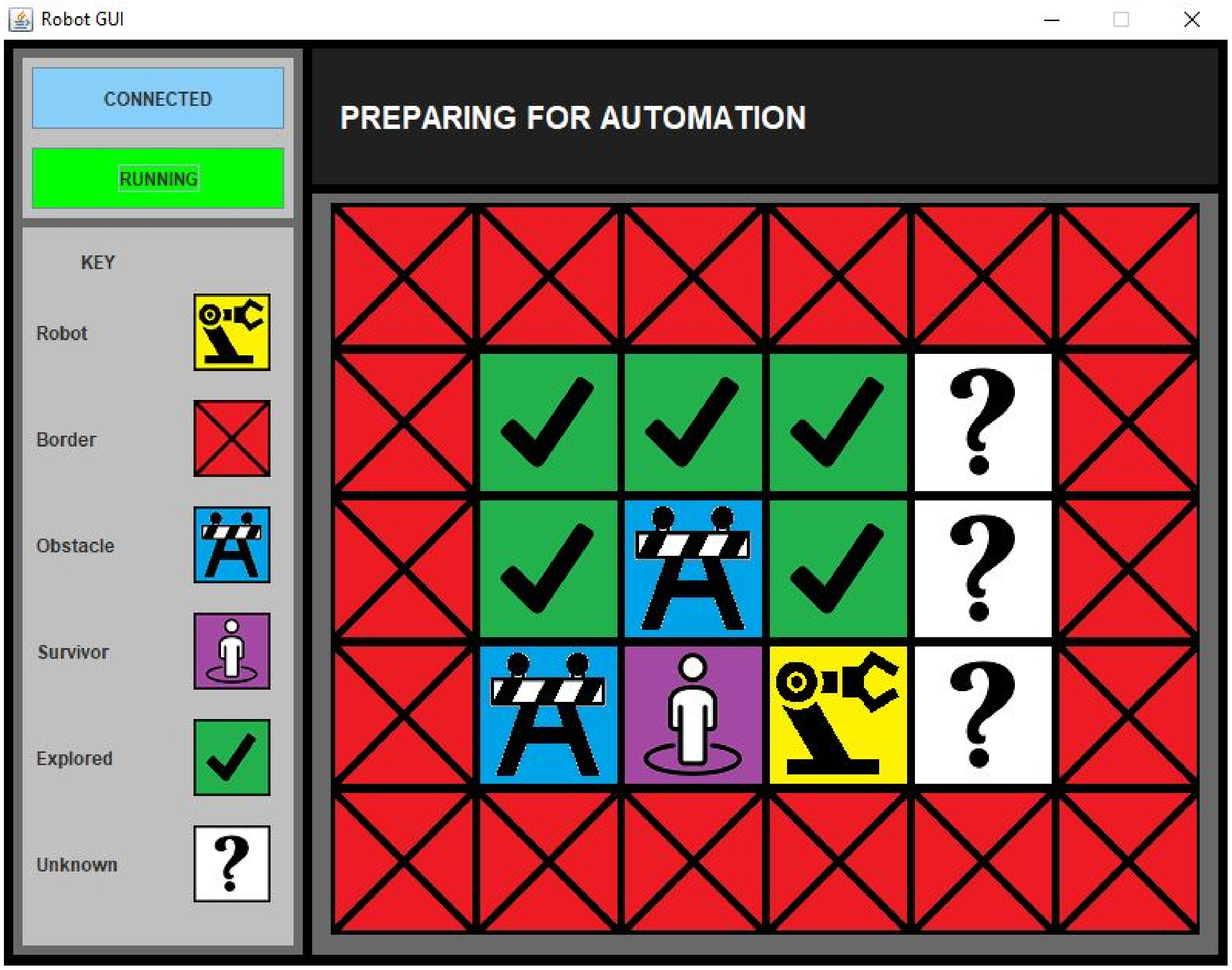


Fig.4, User interface (GUI) graphical representations

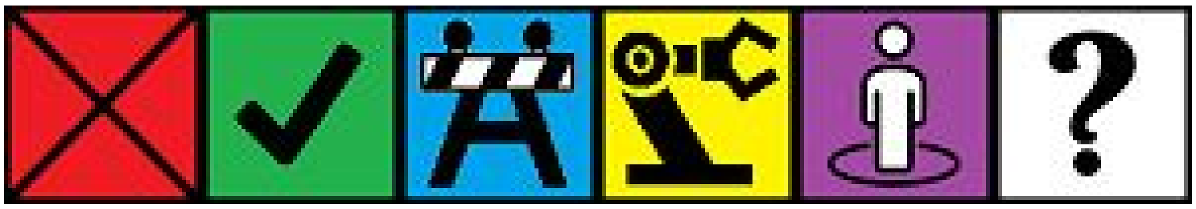
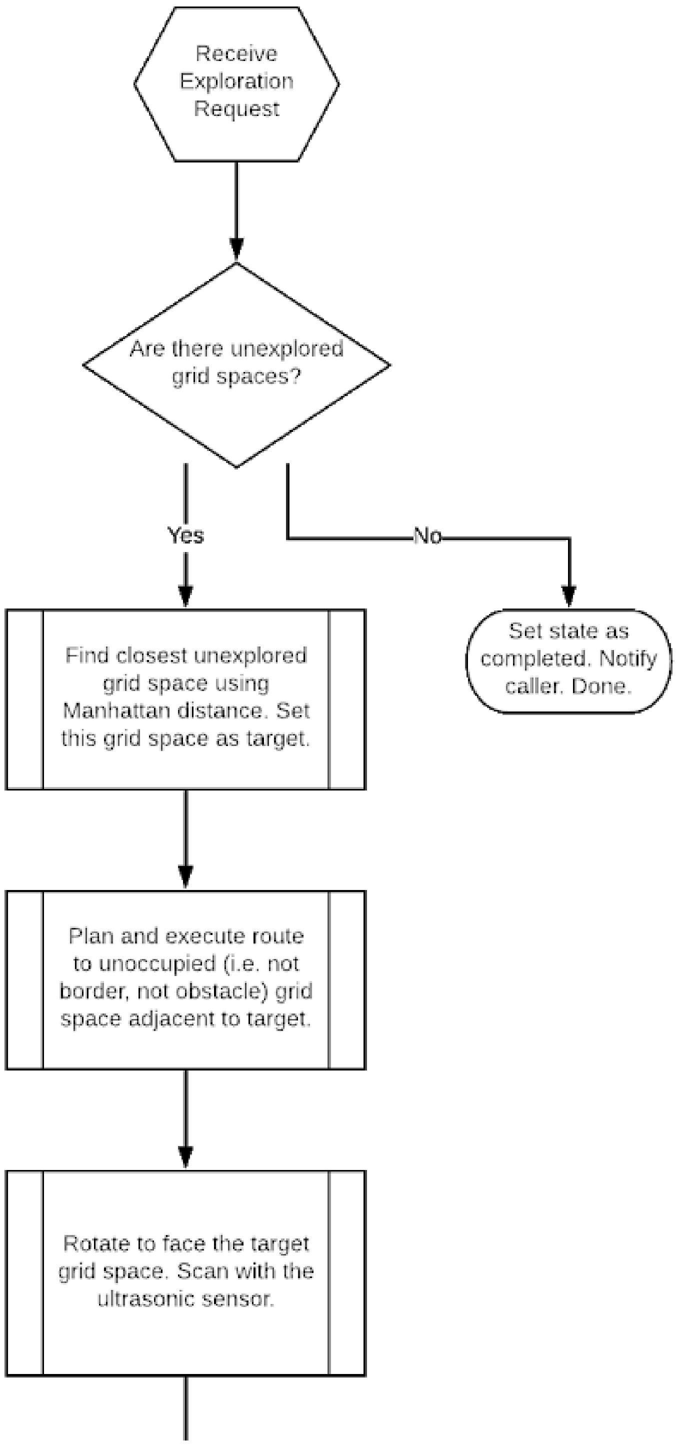
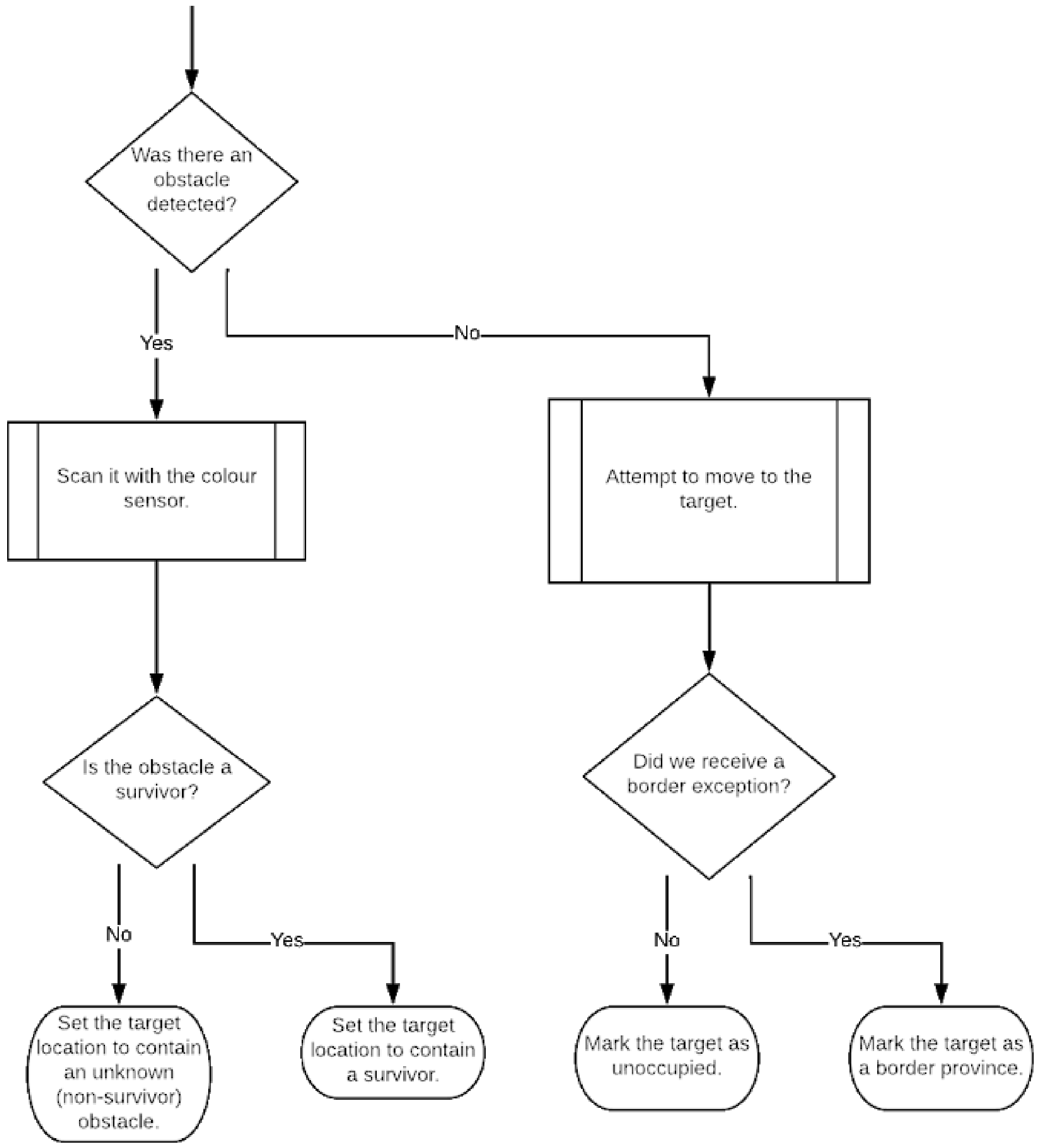


Fig.5, Path Finder Algorithm


Receive Exploration Request
Are there unexplored grid spaces?
Yes
No
Find closest unexplored grid space using Manhattan distance. Set this grid space as target.
Set state as completed. Notify caller. Done.
Plan and execute route to unoccupied (i.e. not border, not obstacle) grid space adjacent to target.
Rotate to face the target grid space. Scan with the ultrasonic sensor.


Was there an obstacle detected?
Yes
No
Scan it with the colour sensor.
Attempt to move to the target.
Is the obstacle a survivor?
Did we receive a border exception?
No
Yes
No
Yes
Set the target location to contain an unknown (non-survivor) obstacle.
Set the target location to contain a survivor.
Mark the target as unoccupied.
Mark the target as a border province.