Rescue on wheels project documentation

Team 2

Damian de Hoog 500780277
Yoshio Schermer 500760587
Mustafa Yücesan 500769574
Mohamed El Hadiyen 500777214

January 8, 2019

Contents

1 Introduction

This document contains all the information regarding "Metabot" our rescue on wheels project. This document will describe the design phase, working phase and post-launch phase consisting of:

- An analysis of the requirements for the project from the perspective of an user.
- A design model which explains in great detail how the system is built.
- Documentation containing important code segments with comments and explanations as well guidance for installation, operation and maintenance.

Metabot is a Raspberry Pi powered mobile robot designed to assist in rescue operations. The robot will be able to navigate difficult to traverse environments and explore areas as a sort of reconnaissance unit.
Metabot is equipped with a camera which will broadcast to a mobile device on which the user can control the robot as well as see said camera feed. The camera has a facial recognition functionality to assist in spotting survivors.
Whilst navigating the operating environment Metabot will map the area and ping locations on the map when survivors are found.

Metabot will be able to explore and create a map of the area with the locations of survivors. This way, having seen the environment and knowing the locations of the survivors, the rescue team will be able to conduct a swift and efficient rescue operation.

The following chapters will describe the robot in more details as well as it's design. We will begin with the analysis of the requirements for the Metabot.

2 Analysis

The analysis consists of the following parts:

- Epic
- User stories
- Use-cases
- Use-case diagram
- Domain model

The analysis answers the question of "How, in detail, is the product designed?"

The formulation of the requirements was performed based on user stories. These user stories were created based on an epic. An agile epic is a body of work that can be broken down into specific tasks (called "stories," or "user stories") based on the needs/requests of customers or end users[1].

After having created said requirements we will formulate the Use-case diagram(s) and the Domain model(s). These will visualize the user interaction with our system and our Metabot.

2.0.1 Terms and definitions

Below you'll find the terms and definitions used throughout this document.

Terms	**Definitions**
Rover	Remotely-controllable RC-car with sensors attached to it.
App	The mobile app through which the rover can be remotely-controlled.
Rover operator	An individual who operates the rover.
Web interface	The web interface through which the rover can be remotely-controlled.
Livestream	A livestream of the camera feed that is attached to the rover.

Table 1: Terms and definitions for this document.

[1] https://www.atlassian.com/agile/project-management/epics

2.1 Epic

A building has collapsed trapping the people inside. Some managed to get out in time but others weren't so lucky. Upon arrival at the scene our actor assesses the situation. The building is unstable and the trapped survivors need to be rescued as quickly as possible.

Normally a rescue crew would be assembled and they would slowly make their way through the rubble to search for people. This is a dangerous and time-consuming task. The crew has no idea whether or not there is a way to get to the survivors. Removing rubble could make the building collapse even further. These people need to be found and removed from the ruins before this happens.

Luckily our actor has just the tool for this job. The Metabot, a small remote controlled robot that can do the scouting for our actor. Our actor takes out his phone and boots up the app. Here, our actor can control the robot.

Our actor sends the robot into the ruins, slowly making it's way deeper and deeper into the building. Whilst moving, the robot sends out signals to its surroundings to map the area and to avoid collision with objects. Meanwhile, our actor can see what the robot is seeing through the live camera feed. Our actor uses this feed to try and find survivors while also looking around for possible routes to take with the rescue crew.

Once our actor has found a survivor, the robot's facial recognition software will help our actor detect the survivor. After this detection, the robot will ping the location of the survivor on the map so that our actor knows where in relation to the rest of the building the survivors are. After pinging the location, the robot will give a visual light signal to the survivors to let them know they have been found and will soon be rescued.

After mapping the area and locating the survivors our actor can create a rescue plan and execute it. Having performed reconnaissance safely and quickly with the Metabot, our rescue crew can now swiftly save the survivors.

2.2 User stories

From this epic story we could formulate user stories, this resulted in the following table:

M: As operator I want to be able to recognize a survivor's face, so I can get information about this survivor.	**M: As operator I want to visually explore the environment from a distance, so I can be better prepared for the rescue operation.**	**S: As operator I want to know the locations of the survivors in relation to the environment, so that I know where I can find the survivors.**	**S: As a survivor I want to know if I have been found, so I can be rescued.**
As operator I want a mobile robot to recognize a survivor's face, so I can get information about this survivor	As operator I want the mobile robot to show me the environment on a screen at a distance so that I can plan the rescue operation accordingly.	As operator I want a mobile robot to tell a mobile app where the survivor is in relation to the environment, so that I know where I can find the survivors.	As survivor I want the mobile robot to give me a sign, so I know I have been found.
As operator I want a mobile app to get the information that corresponds to the survivor's face, so I know who the survivor is.	As operator I want the mobile robot to stream the camera feed to a mobile device so I can get real time information from the environment.	As operator I want a mobile app to display where the survivors are in relation to the environment, so that I know where I can find the survivors.	
	As operator I want to control the mobile robot from a distance to plan the rescue operation accordingly.		

Table 2: User stories formed in accordance with the epic story.

2.3 Use-cases

Use Case ID:	1
Use Case Name:	Connect with rover
Primary Actor:	Rover operator
Secondary Actor:	Rover
Pre-conditions:	1. Rover operator has mobile app or address of web interface. 2. Rover is nearby.
Success Guarantee:	Rover operator successfully connected the mobile app/web interface to the rover.
Main Success Scenario:	1. Rover operator opens app or goes to the web interface. 2. Rover operator searches for rover to connect to within the app. 3. Rover operator selects rover to connect to. 4. Rover operator confirms to connect to the selected rover. 5. Rover operator successfully connected the app/web interface to the rover.
Exceptions	• Rover operator already has established a connection between the app/web interface and the rover. • Rover operator cannot connect app/web interface to the rover, because the rover is already connected with the same app/web interface from another rover operator.
Special Requirements	-

Table 3: Use-case 1

Use Case ID:	2
Use Case Name:	Look for survivors on flat surfaces
Primary Actor:	Rover operator
Secondary Actor:	Rover
Pre-conditions:	1. Rover operator has established a connection between the app /web interface and the rover. 2. Rover operator sees what the rover sees. 3. Rover operator has controller.
Success Guarantee:	Rover operator uses the rover to look for survivors on flat surfaces, that is, drives around.
Main Success Scenario:	1. Rover operator uses controller to drive either forward, backward, left, or right. 2. Rover operator uses the live stream to look for any survivors.
Exceptions	-
Special Requirements	-

Table 4: Use-case 2

Use Case ID:	3
Use Case Name:	Know who a survivor is and where they are
Primary Actor:	Rover operator
Secondary Actor:	Rover
Pre-conditions:	1. Rover operator has established a connection between the app/web interface and the rover. 2. Rover operator sees what the rover sees. 3. Rover operator has controller.
Success Guarantee:	Rover operator drives around with the rover. Through the livestream the rover operator can see who the survivor is and where they are to be seen.
Main Success Scenario:	1. Rover operator uses controller to drive either forward, backward, left, or right. 2. Rover operator uses the live stream to look for any survivors. 3. A survivorâĂŹs face is visible through the livestream. 4. Through the livestream the rover operator can tell who the survivor is and where the face of the survivor is.
Exceptions	-
Special Requirements	-

Table 5: Use-case 3

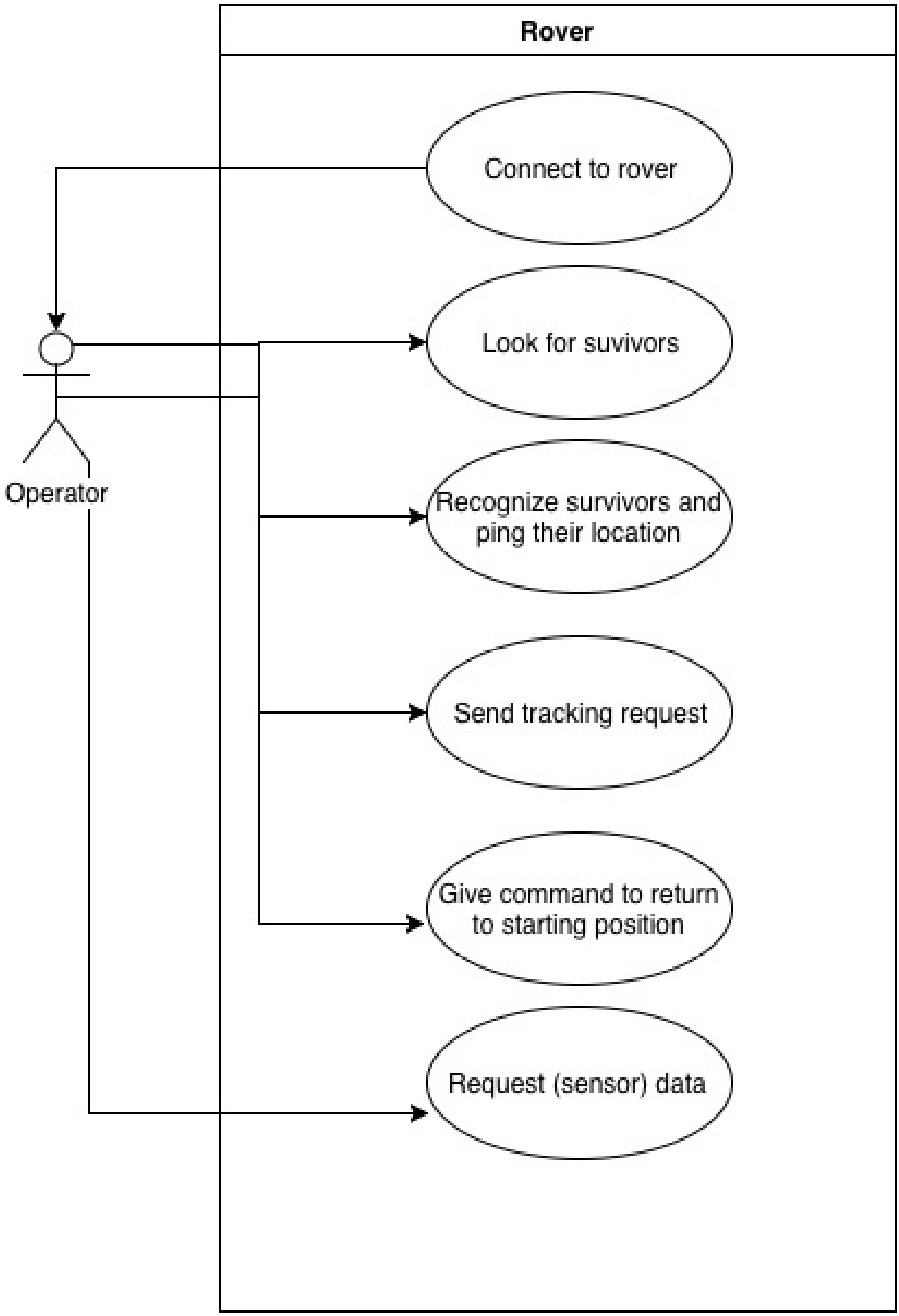
Use Case ID:	4
Use Case Name:	Tracking the location of the rover
Primary Actor:	Rover operator
Secondary Actor:	Rover
Pre-conditions:	1. Rover operator has established a connection between the web interface and the rover. 2. Rover operator has controller.
Success Guarantee:	Rover operator tracks the location of the rover through a map on the web interface
Main Success Scenario:	1. Rover operator uses controller to drive either forward, backward, left, or right. 2. Rover operator sees location of the rover changing in the same direction as he/she is moving it.
Exceptions	-
Special Requirements	When the rover is unable to move in a certain direction because of an obstacle then this should be reflected in the map as not having moved.

Table 6: Use-case 4

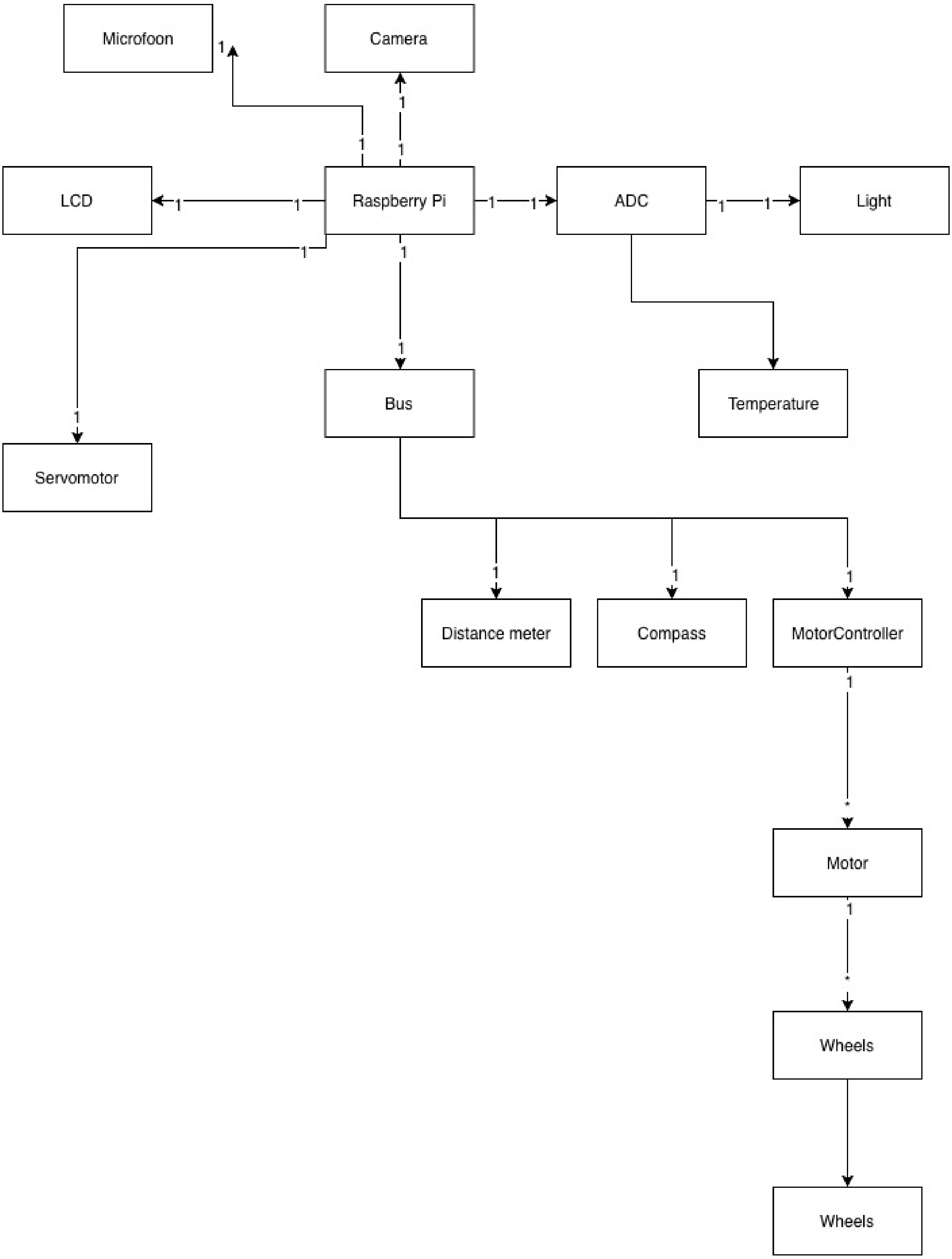
Use Case ID:	5
Use Case Name:	Backtracking the rover
Primary Actor:	Rover operator
Secondary Actor:	Rover
Pre-conditions:	1. Rover operator has established a connection between the web interface and the rover. 2. Rover operator has controller.
Success Guarantee:	Rover returns to initial position in the opposite way as it has moved to its current position.
Main Success Scenario:	1. Rover operator uses controller to drive either forward, backward, left, or right. 2. Rover operator turns backtracking on. 3. Rover goes back to its initial position.
Exceptions	-
Special Requirements	Moments where the rover stopped are ignored. So, the rover drives back directly without stopping.

Table 7: Use-case 5

2.4 Use-case diagram



2.5 Domain model



3 Testing

In the fourth sprint we conducted multiple tests:

3.1 General tests

ID	1
Testgoal	Proof that it can be remote-controlled successfully via the camera
Version	1
Testers	Mustafa, Yoshio
Date	21 Dec 2018 at 2:30pm
Location	AUAS, WBH3
Device(s)	MacOS, Chrome
Approach	Operator cannot see rover. He/she can only see the web interface. Therefore, the camera on the rover must only be used to control.
Proof	Video demonstration
Conclusion	Test successful. Operator is able to control the rover using only the camera on the rover.

Table 8: Testcase 1

ID	2
Testgoal	Proof that it can recognize faces
Version	1
Testers	Mustafa, Yoshio
Date	21 Dec 2018 at 1pm
Location	AUAS, WBH3
Device(s)	MacOS, Chrome
Approach	Mr Bean's face is trained and is the only face trained. We first show Mr Bean's face. His face should be recognized. Afterwards we show another person's face. That person's face should be unknown and most certainly not to be recognized as Mr Bean.
Proof	Video demonstration
Conclusion	The test was successful. Mr Bean's face was recognized at a percentage of 50%. Our own was considered to be unknown. There is, however, a significant delay of 1.5 seconds relative to real-time.

Table 9: Testcase 2

ID	3
Testgoal	Proof that light turns on when it's dark
Version	1
Testers	Yoshio
Date	21 Dec 2018 at 3:30pm
Location	AUAS, WBH3
Device(s)	Windows 10, Chrome
Approach	Cover the rover with something like a blanket. The light should then turn on. Removing the blanket should result in the light turning off.
Proof	Video demonstration
Conclusion	Test successful. Covering causes the light to turn on, removing cover turns it off.

Table 10: Testcase 3

ID	4
Testgoal	Proof that light turns on when it's dark
Version	1
Testers	Mustafa, Yoshio
Date	21 Dec 2018 at 2pm
Location	AUAS, WBH3
Device(s)	MacOS, Chrome
Approach	Type text into text field on the web interface. Subsequently check if text can be seen on the display.
Proof	Video demonstration
Conclusion	Test was successful. We followed the instructions, no problems arose.

Table 11: Testcase 4

ID	5
Testgoal	Proof that temperature works
Version	1
Testers	Mustafa, Yoshio
Date	21 Dec 2018 at 3pm
Location	AUAS, WBH3
Device(s)	MacOS, Chrome
Approach	Put finger on sensor. The web interface should display a higher temperature (finger temperature > room temperature). Also use a cold drink and push against the sensor, this should decrease the temperature.
Proof	Video demonstration
Conclusion	Test unsuccessful. Probably calculation mistake. The temperature decreases when temperature is higher and decreases when lower.

Table 12: Testcase 5

ID	6
Testgoal	Proof that distance sensor works
Version	1
Testers	Mustafa, Yoshio
Date	21 Dec 2018 at 2pm
Location	AUAS, WBH3
Device(s)	MacOS, Chrome
Approach	Put hand in front of sensor and move forward/backwards. It should be able to accurately determine the distance of objects between itself when the distance is between 25cm and 55cm. It should also remain constant when the object doesn't move.
Proof	Video demonstration
Conclusion	Test successful. Moving hand closer than 25cm gives inaccurate measurements and the same goes for distances larger than 55cm. In between the distance is accurately measured and constant.

Table 13: Testcase 6

ID	7
Testgoal	Proof that compass works
Version	1
Testers	Mustafa, Yoshio
Date	21 Dec 2018 at 2pm
Location	AUAS, WBH3
Device(s)	MacOS, Chrome
Approach	Turn rover 90 degrees. This should match a turn of a phone turning the same angle.
Proof	Video demonstration
Conclusion	Test unsuccessful. Probably a hardware/interference problem, because code seems to work for others.

Table 14: Testcase 7

ID	8
Testgoal	Proof that backtracking works
Version	1
Testers	Mustafa, Yoshio
Date	21 Dec 2018 at 3pm
Location	AUAS, WBH3
Device(s)	MacOS, Chrome
Approach	Drive forward, then a 90 degree angle to either left/right. Drive forward. Then backtrack. It should return to its initial position.
Proof	Video demonstration
Conclusion	Test was unsuccessful due to one or more wheels not working fully. Also only driving forward and then backtracking, causes it to go more backwards then forwards. No attention has been given to the fact that the motors are running at full power until the last millisecond.

Table 15: Testcase 8

ID	9
Testgoal	Proof that map corresponds to route taken
Version	1
Testers	Mustafa, Yoshio
Date	21 Dec 2018 at 3pm
Location	AUAS, WBH3
Device(s)	MacOS, Chrome
Approach	Drive forwards, then the operator should see that the line has moved forward. Drive an other direction then the operator should see that the line has moved into that direction.
Proof	Video demonstration
Conclusion	Line doesn't completely correspond to route taken due to the way the robot senses turns. The line is being generated though when it moves, but just not entirely accurate. Other approaches unfortunately weren't possible due to inconsistencies as well. Therefore, this is as of now the best we can do.

Table 16: Testcase 9

ID	10
Testgoal	Proof that when backtracking the rover stops moving when an object is behind it (25-55cm)
Version	1
Testers	Yoshio
Date	21 Dec 2018 at 4pm
Location	AUAS, WBH3
Device(s)	Windows 10, Chrome
Approach	Drive forward, then place an object behind it 25-55cm. Then activate backtrack. It should stop almost immediately.
Proof	Video demonstration
Conclusion	Test successful. It does, however, tend to drive a little too much backwards.

Table 17: Testcase 10

ID	11
Testgoal	Proof that when backtracking the rover moves the distance sensor via the servo so it can detect objects from different angles as well.
Version	1
Testers	Yoshio
Date	21 Dec 2018 at 3:45pm
Location	AUAS, WBH3
Device(s)	Windows 10, Chrome
Approach	Drive forward, then place an object behind it 25-55cm within a degree of 90. Then activate backtrack. It should stop almost immediately.
Proof	None
Conclusion	Test successful when little interference is present.

Table 18: Testcase 11

3.2 Acceptance tests

ID	1
Name	Connect with rover
Basis	• Use case: connect with rover • User story: connect app to rover
Figure(s)	1, 2
Pre-conditions	• Rover nearby • Mobile app opened • Connected through Wifi with rover hotspot
Test	Given: List of rovers to connect to When: Select rover from list Then: Selected and visual cue that rover is selected Given: List of rovers to connect to while rover already selected When: Select other rover from list Then: Selected and visual cue that other rover is selected Given: List of rovers to connect to while rover already selected When: Select same rover from list Then: Selection and visual cue remain the same Given: Selected rover from list When: Click on connect button Then: See Rover UI with rover no. of selected rover
Remarks	When I try to go through the list, the item I put my finger upon is automatically selected as well, even when my intention wasn't to.

Table 19: Acceptance test 1

ID	2
Name	See what rover sees
Basis	• Use case: look for survivors on flat surfaces • User story: see what rover sees
Figure(s)	1, 2
Pre-conditions	• Connected with rover
Test	Given: Rover UI of a rover Then: See what camera of rover sees through UI Given: Rover UI of a rover Then: See what rover number Given: Rover UI of a rover Then: See rover IP and port
Remarks	When I go to a dummy UI which has obviously no camera feed, then I get this ugly 404 error.

Table 20: Acceptance test 2

3.3 Use-case testing

Use-case id	1
Use-case name	Tracking the location of the rover.
Primary actor	Rover operator
Secondary actor	Rover
Pre-conditions	• Rover operator has established a connection between the web interface and the rover. • Rover operator has controller.
Success guarantee	Rover operator tracks the location of the rover through a map on the web interface
Main success scenario	• Rover operator uses controller to drive either forward, backward, left, or right. • Rover operator sees location of the rover changing in the same direction as he/she is moving it.
Exceptions	-
Special requirements	When the rover is unable to move in a certain direction because of an obstacle then this should be reflected in the map as not having moved.

Table 21: Use-case test 1

Use-case id	2
Use-case name	Backtracking the rover.
Primary actor	Rover operator
Secondary actor	Rover
Pre-conditions	• Rover operator has established a connection between the web interface and the rover. • Rover operator has controller
Success guarantee	Rover returns to initial position in the opposite way as it has moved to its current position.
Main success scenario	• Rover operator uses controller to drive either forward, backward, left, or right. • Rover operator turns backtracking on. • Rover goes back to its initial position.
Exceptions	-
Special requirements	Moments where the rover stopped are ignored. So, the rover drives back directly without stopping.

Table 22: Use-case test 2

Use-case id	3
Use-case name	See data about rover
Primary actor	Rover operator
Secondary actor	Rover
Pre-conditions	• Rover operator is on the control page/view.
Success guarantee	Rover operator can read the following in the interface: • Temperature at rover • Camera shown at rover • Position of servo of rover • Distance between rover and object in front of it • IP of rover • Port at which the operator is connected to • Rover number
Main success scenario	1. Rover operator reads temperature of from interface. 2. Rover operator reads what camera of rover he/she sees in the interface. 3. Rover operator reads position of servo. 4. Rover operator reads IP of rover. 5. Rover operator reads what port he/she is connected to. 6. Rover operator reads which rover this is.
Exceptions	-
Special requirements	-

Table 23: Use-case test 3

4 Design model

As with the analysis of our product the design model contains multiple elements. These elements are as follows:

- Class diagram
- Sequence diagram
- Communication diagram
- Hardware architecture

The design model answers the question of "how, in detail, is the product built?"

4.1 Class diagram

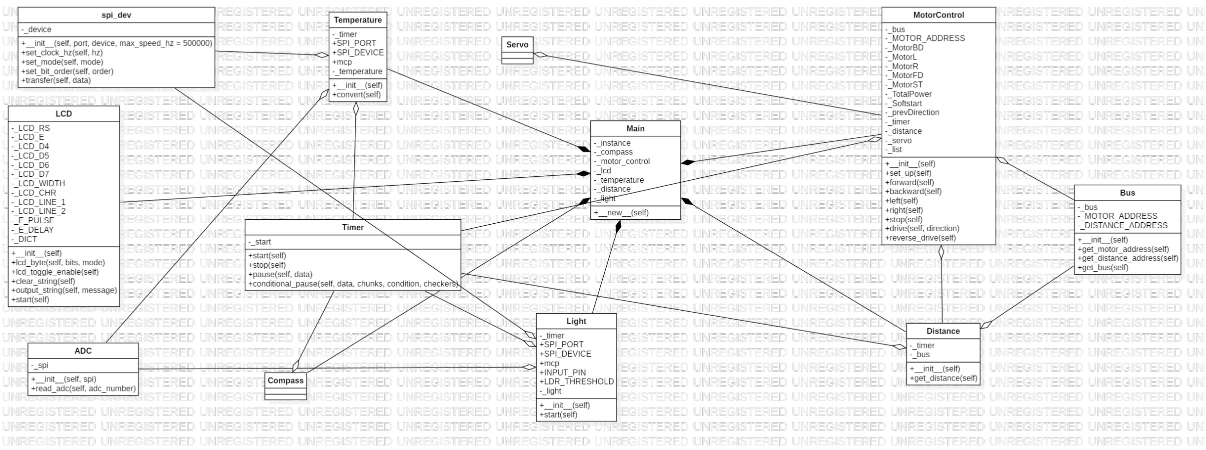


Figure 1: Class diagram for the mobile application

4.2 Sequence diagram

4.3 Communication diagram

Telecommands to rover

Event	Parameters	Description
direction	“forward” || “backward” || “right” || “left”	Forward/backward/right/left on rover
LCD	<a string up to 32 characters>	Displays text on LCD display
backtrack	N/A	Backtrack on/off

Telemetry from rover

Event	Parameters	Description
temperature	N/A	Returns temperature in C
compass	N/A	Returns compass heading
distance	N/A	Returns distance in cm

Examples (from (web interface / app) to rover)

Drive

```
socket.emit(“direction”, “left”);
```

Returns temperature in C

```
socket.emit(“temperature”, function(data){ // do something with data });
```

Figure 2: Diagram of the communication of our product

4.4 Hardware architecture

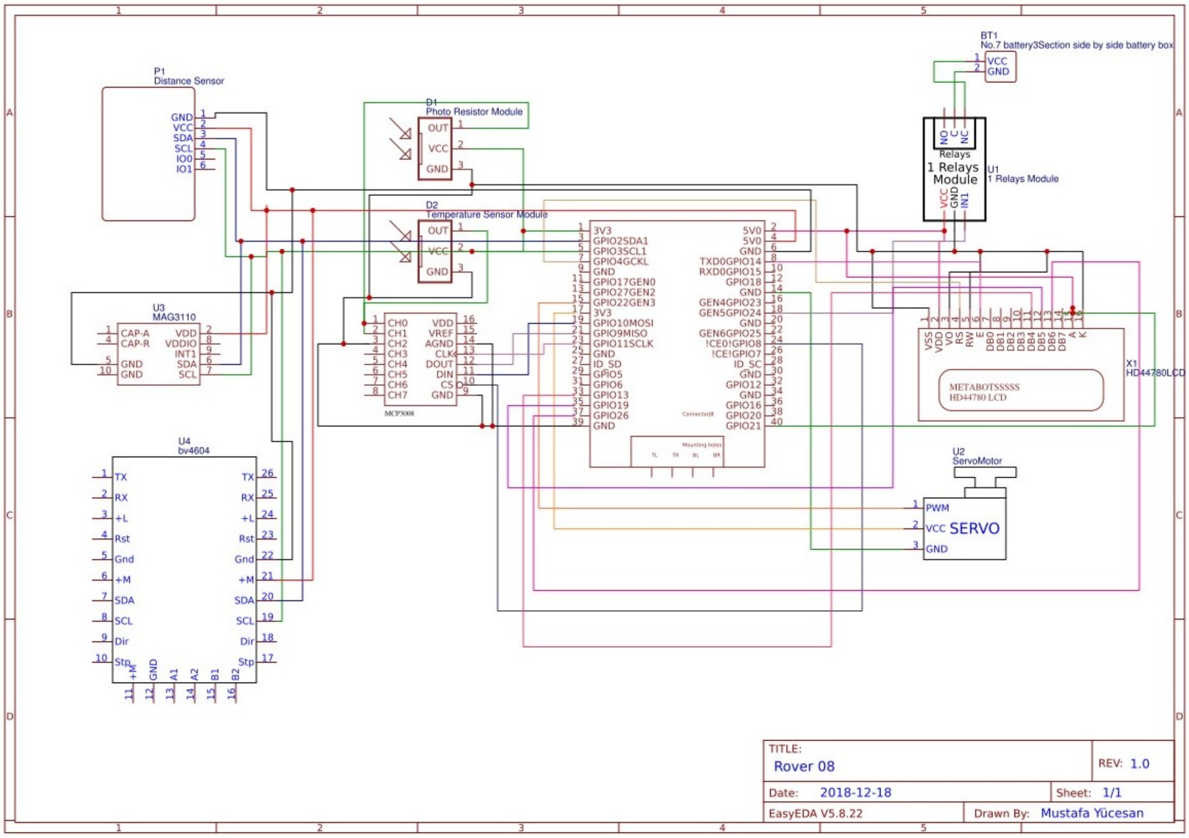


Figure 3: Diagram of our product's architecture

5 Installation manual

6 User manual