

Rescue Robot

Ali Ahemd
Electronics Engineering
Hochschule Hamm-Lippstadt)
Lippstadt, Germany
ali.hisham-omer-ahmed@stud.hshl.de

Damilola Awosuru
Electronics Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
damilola-caleb.awosuru@stud.hshl.de

Johnson Darko
Electronics Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany: johnson-
kwabena.darko@sttud.hshl.de

Abstract—The purpose of creating a robot is to help in everyday human endeavors by providing security and working as a helping hand. In this paper, the ultimate focus of our system, the rescue robot, is to offer far possible security for human beings. We have discussed about a stand-alone, fully functional search and rescue robot that assist in the recovery and exploration of dangerous areas. The robot is able to operate on both land and water, can search for alive human bodies underneath a pile of debris, avoid obstacles, find the shortest path, stream live video of the surrounding environment wirelessly and assist people in search operations. This paper explains the capabilities of our rescue robot and the iterative process of making of making it.

Keywords—*rescue, search, assist, security*

I. INTRODUCTION

Today, due to unplanned civilization and industrialisation, environment pollution has become earths greatest enemy and a threat to our existence. We have never been more prone to natural calamities as we are today. Earthquakes, cyclones, tsunamis are becoming common incidents day to day. And often due to lack of proper disaster management and implementation of technology, people die in large numbers. In this particular area, robots can help save many valuable lives. Often, humans face a situation while rescuing that, they can't reach the affected people due to lack of proper path and there is always a risk of the rescuers getting hurt themselves. Robots on the other hand do not have these limitations, they can slip through even the smallest and slender paths, and they can operate without anyone getting hurt. The base station initiates and terminates the mission and has the ability to take over the system through a wireless communication system in case of emergency, even if the robot is 90 meters away, with the help of Wi-Fi technology. Microcontroller is used to control all operation. According to the motor operations, the robot will operate in specified directions. If the robot observes any obstacle, it will change its direction or it will stop. Our system is equipped with smoke sensor, gas sensor, a human identification and an alarm module for security purpose. The purpose of using a smoke and gas sensor is to monitor smoke to prevent fire, and leakage of gas that might be harmful to human health. If input voltages of these sensors exceed the threshold level, an alert/alarm is given. For example, in coalmine accidents such as fire damp explosion, ceiling board collapse, water disaster and traffic accidents happen frequently. The fire damp explosion is the most serious one and the scale of death toll caused by it is maximal. After an explosion accident such as gas or grime happens, the scene under the mine well becomes very abominable, abnormally complicated and unstructured. It is difficult for human rescuers to go into the mine well at the first time, so it delays precious rescue time. The accidents experts and decision makers cannot give estimation and make decisions without

required information and the trapped miners resultantly lose their life. In this situation after an explosion, our rescue robots can be immediately sent in, and feedback information in the form of sound, image and data to the control centre by wireless communication. It should be noted that throughout this paper the words "rescue robot" and "system" will be used interchangeably.

II. ROBOT MODELLING

Modelling covers the definition and simulation of the overall system functionality. The implementation aspects are not considered here. A system model built in system modelling language (SysML) and unified modelling language (UML) is used to capture the cross-domain dependencies. The model can be systematically linked with other domains (CAD, programming and dynamic analysis). As the modellers perform the design iterations, the update on cross-domain dependencies will be available through SysML model, containing the complete View. Many of the hidden complexities in the system become visible going from one abstraction level to another. Creating models at different abstraction level is necessary to unleash the hidden complexity present inside the system which is otherwise not visible while modelling at a single abstraction level. This is why in the modelling of our rescue robot, three SysML modelling diagrams and two UML modelling diagrams are used.

A. System modelling using SysML

SysML is a general-purpose language aimed to specify systems as concretely as possible. The design or creation of a system (which in our case is the rescue robot), requires that the designers or creators of the system to know the necessary information about the system (requirements, context, restrictions, parameters, activity etc.) to start the modelling process and the SysML provides such information. The SysML model contains the necessary information that a CAD tool, algorithm and Matlab/Simulink require, for the corresponding analysis to begin. The SysML provides a common knowledge base. The SysML model of our rescue robot was created, to provide information about the complete system in terms of its structure, its behaviour, its requirements, and constraint based behaviour. We will present a few excerpt of the SysML model to help visualise the system, and capture the cross-domain dependencies of the system. We will be showing the requirements diagram, block definition diagram (bdd) and the internal block diagram (ibd) of the system.

1. Requirements Diagram

This development process ensures that the requirements are captured based on the specified stake holder and market needs, which forms the foundation of the system to be developed,

Hence, everything from the system specifications, design ideas, to the final design implementation depends on this requirements. Figure 1 shows the properties, conditions or behaviours of the system that always have to be met. Such properties or conditions include:

- The robot must be competent to function for two hours.
- The maximum speed of the robot is limited to 40m/sec.
- The total weight of the robot must be 25kg.
- The robot should be able to operate on both land and water.
- The robot should be able to move forward, backward, turn left, turn right, turn around, release safety equipment and navigate on land and water.
- The robot should be able to rescue people out of water, which depends on the ability of the robot to locate humans in the water, know its location and navigate
- The robot should be able to be remotely controlled, as discussed previously, in case of emergencies.
- The robot should be able to rescue people in a fire accident which depends on the robot been resistant to heat to prevent melting of internal and external components and been able to release fire extinguishing material.

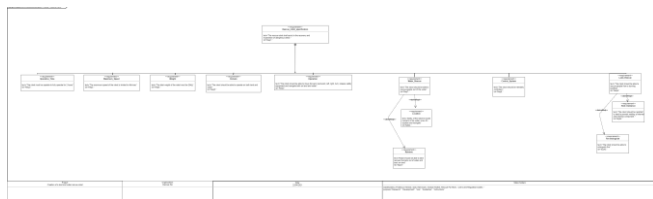


Figure 1: Requirements diagram of the rescue robot.

While referring to requirements, it is important to understand that the requirements cannot be regarded as fixed but are evolving during the project life time, either due to a market need, or from a change initiated by a modeller working in a detailed design tool.

2. Block Definition Diagram (bdd)

This development process shows and describes parts of the structure of the system, it defines types of physical entities (e.g. system, system component part, eternal systems, or items that flow through the system), as well as conceptual entities or logical abstraction.

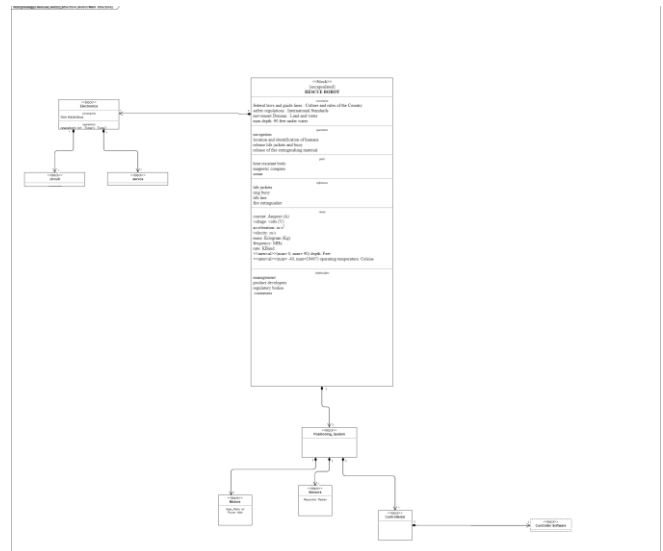


Figure 2: Block definition diagram (bdd) of the rescue robot

The part/value properties (compartment) in the bdd diagram contain the geometric data, which is designed, and iterated through the Solid Works Model. The heading for parts of blocks that are bound by means of a composition are parts. Properties that are of type block but not bound via composition are listed with the heading references in a compartment. In SysML all properties of a block have the visibility public. This means that it is not necessary to explicitly represent the visibility.

3. Internal Block Definition Diagram (ibd)

This development process shows how the properties of the block definition diagram are interconnected.

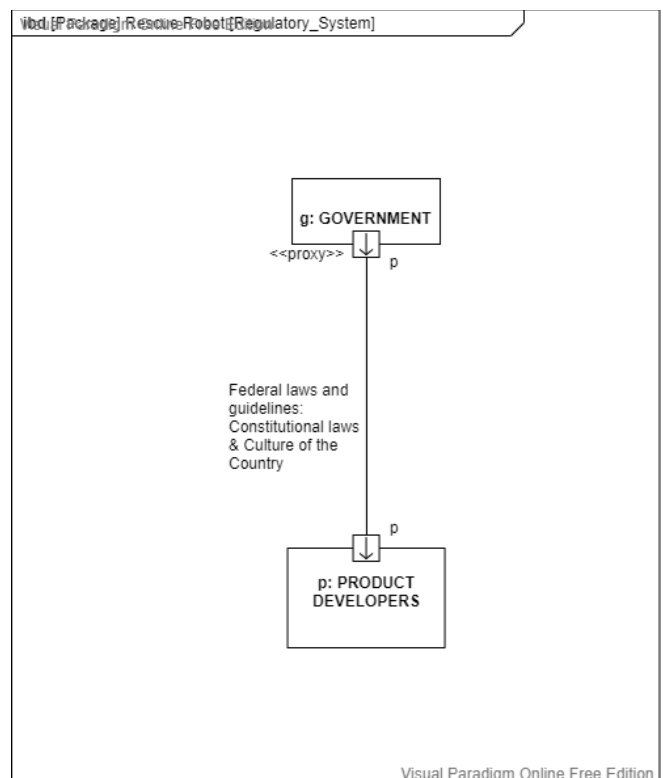


Figure 3: ibd Regulatory System.

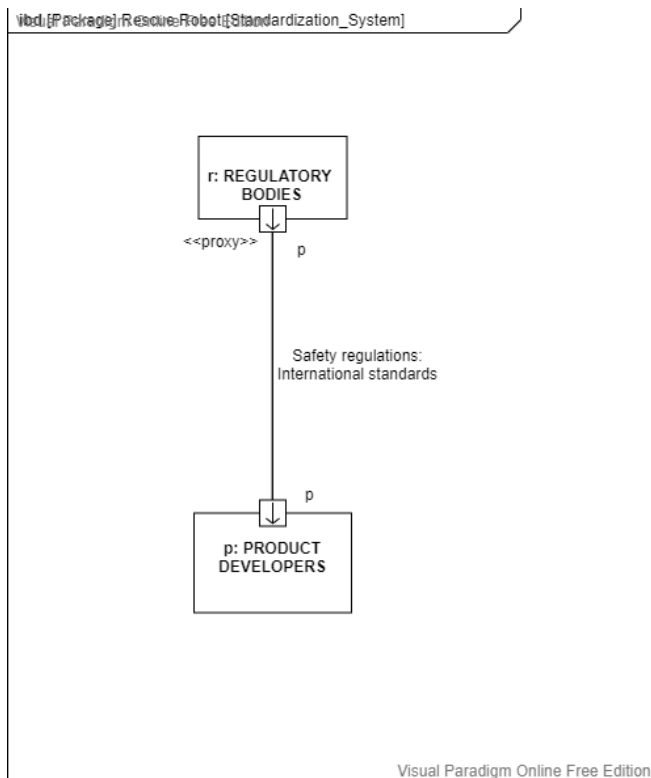


Figure 4: ibd Standardization System

The influence of governmental bodies and standardization bodies (or organizations) are felt throughout the entire process of creating the robot.

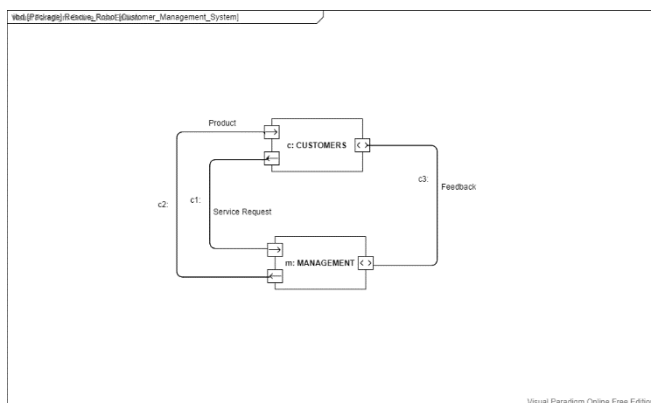


Figure 5: ibd Customer Management Interactions

Figure 5 show the connection or relationship between the customers and management. The management consist of all stakeholders and product developers. Figure 3, 4 and 5 show interconnectivity of the blocks or compartments in the block definition diagram (bdd).

B. System modelling using UML

We have chosen the UML for the modelling of the system. The UML is used in the modelling of the robot because of its representative and descriptive capabilities. The UML offers large possibilities for the improvement of dependability.

1. Use Case Diagram

Use Case diagrams are well dedicated to the capture of the user's (operator) possible interactions with the system. They

drive the designer during the development process and assist the functional testing of the system. Figure 6 shows the Use Case of the system (rescue robot). Use Cases may stand on their own, they may use capabilities specified in other Use Cases and other Use Cases may extend them. The relations ("use" and "extend") are represented by arrow. The actors are the operators or things that interact with the system. They may communicate with Use Cases by the way of interfaces (sensors or controllers).

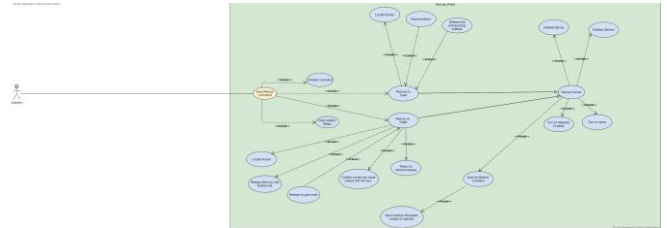


Figure 6: Use Case of the rescue robot.

A use case is denoted as an ellipse. The name of the use case is written inside or underneath the ellipse. The participating actors are connected with the use case by a solid line. Several actors being connected with a use case means that all of them participate in that use case. An include relationship describes that a use case is included in another use case and an extend relationship, is used when a use case conditionally adds steps to another first class use case. Figure 6 explains the behaviour of the system in collaboration with the user (operator). For the system to carry out it's function, the servos and the sensors must be initialized (turned on/active), the magnetic compass to help the system with navigation and the sonar to help the system with location and identification must also be activated, the system checks the medical condition of the human been rescued and immediately in real time, sends the medical information to the operator or base. When the operator sends a rescue command to the robot the validity of the command and the battery status of the robot are first checked, before any other process. Two of the various valid commands are "rescue on land" and "rescue on water". For the robot to be able to rescue on land, it locates the human (or humans), remove and or avoid debris (obstacles) and in case of a fire accident rescue, the robot extinguish fire. Also for the robot to rescue on water, it is be able to locate human, release life buoy with floating line attached, release oxygen mask, confirm if the human has made contact with the life buoy and the robot should be able to return to shore (or base).

2. Activity Diagram

This development process shows the activity of the system. Two activity diagrams where created for the model of our system; the activity diagram showing the general outlook of our system (Figure 7.); how the robot receives and process commands. And the activity diagram of the robot's activity, during a rescue mission (Figure 8).

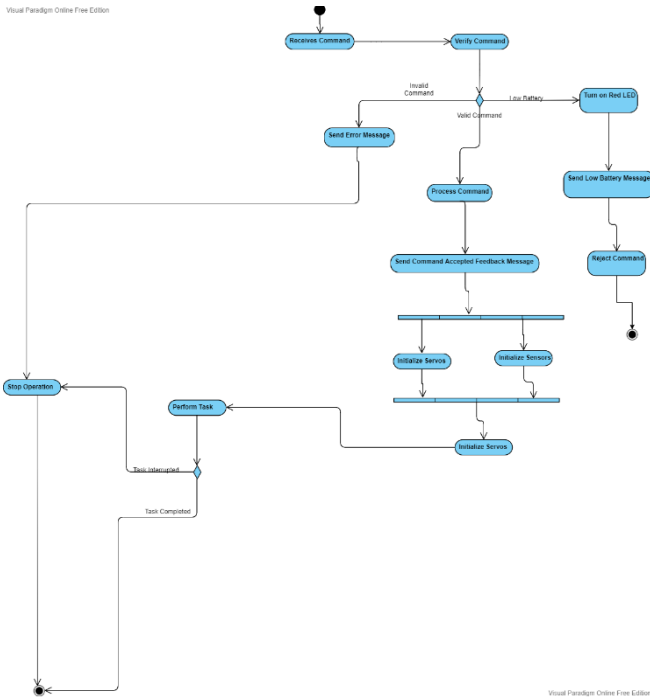


Figure 7: Activity diagram of the rescue robot (general outlook).

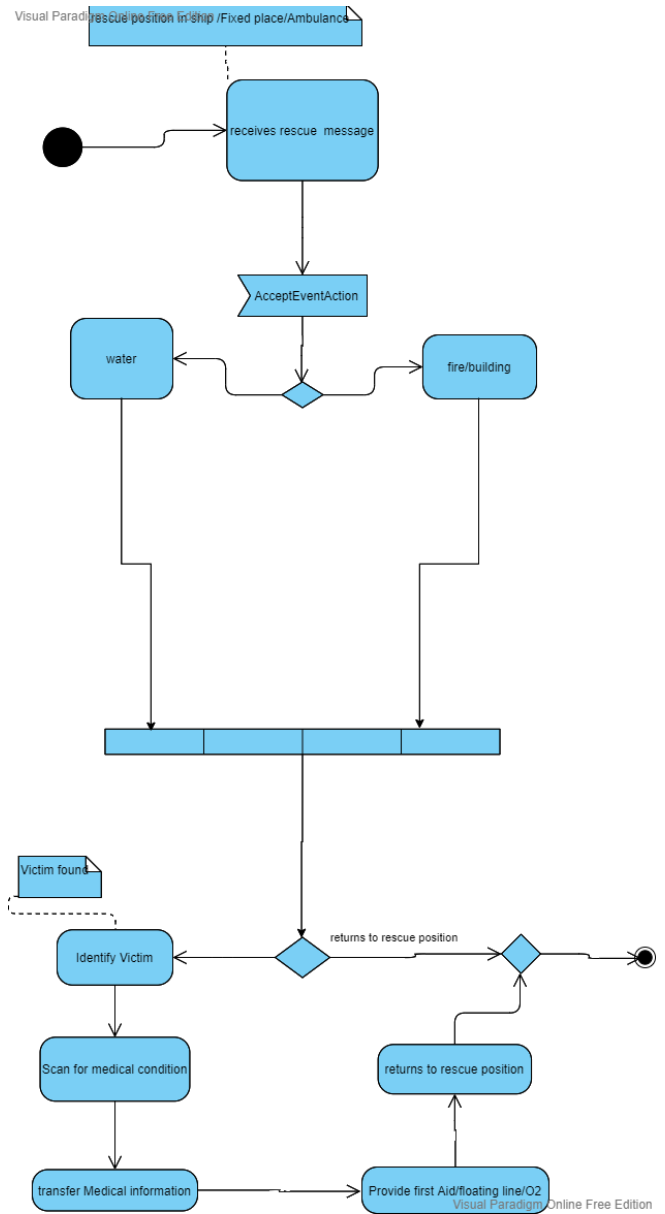


Figure 8: Activity diagram of the rescue robot (robot activity in a rescue mission).

III. SYSTEM ARCHITECTURE

The system architecture of our system is divided into four parts:

- A. The hardware architecture
- B. The software architecture
- C. The communications architecture
- D. The system architecture pattern

A. The Hardware architecture

Our system is a robot therefore it requires many different hardware interfaces. The hardware interfaces are responsible for the actual build of the system, and is concerned with the interfaces used to control and operate the entire system once the software is implemented on them.

B. The software architecture

The software of the system is responsible for the robots control, human detection, obstacle detection, exploration (movement) and many more.

C. The communication architecture

Communication is a crucial part of our system that handles the interaction between the different parts of the system. The network chosen is Wi-Fi and the approach implemented is TCP. TCP is chosen to ensure reliability. The data that will be sent over the network are images, signals from sensors and locations. To achieve this, the communication between the robot and base (operator) there exist a single client server model for the two to communicate.

D. The sytem architecture pattern

“High-level structural patterns are those that refer to the organization of the domains of the logical model, subsystems and components of the subsystem, and component view, and nodes (and the mapping subsystems and components to nodes) of the deployment view” (Douglass, 2009). In our system, the layered architecture pattern is used. “The layered pattern organizes domains into a hierarchical organization based on their level of abstraction” (Douglass, 2009).

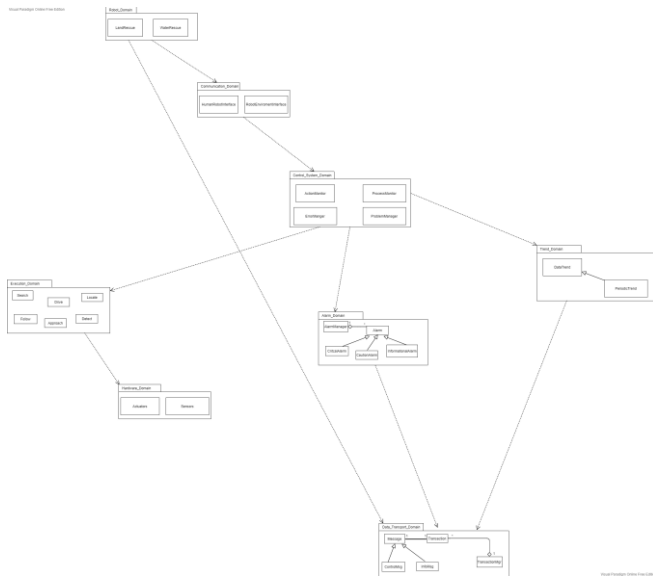


Figure 9: System Architecture pattern (The Layered Pattern).

IV. DETAILED DESIGN

This process involves the implementation of the system in real life by drawing and/or using a mechanical simulation tool (e.g. SOLIDWORKS)

A. Pen and paper model

This is the start of the modelling process for the rescue robot. The stakeholders come together to brainstorm ideas for the model that capture certain information. Figure 10 shows

the pen and paper model of the rescue robot that captures the important features and geometry of the robot.



Figure 10: Pen and paper model of rescue robot

In the pen and paper, model certain parameters (such as length, height weight etc.) are considered for the concrete system. Different properties such as the robot's geometry parameters, range of rotation and placement of robot inside the workspace are also considered. Therefore all information in figure 10 can be classified as design specifications that come to life. A list of properties is taken from the pen and paper model in figure 10 that will be designed and modelled in a CAD tool (e.g. SOLIDWORKS, FUSION etc.). It should however be noted that changes to the pen and paper model are sometimes made because of its implementation in a computer aided design tool.

B. CAD model

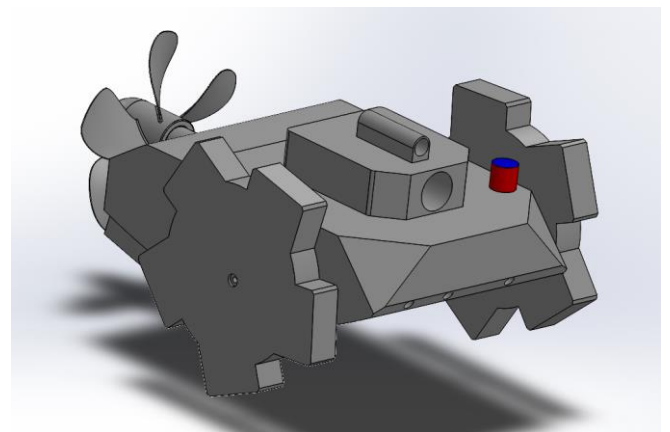


Figure 11: CAD model of rescue robot.

Figure 11 shows the physical representation of the rescue robot. The parts of the robot are:

1. Propeller

The propeller (figure 12) makes it possible for the robot to move on water, thereby making it possible for the robot to rescue on water (i.e. Sea, ocean etc.)

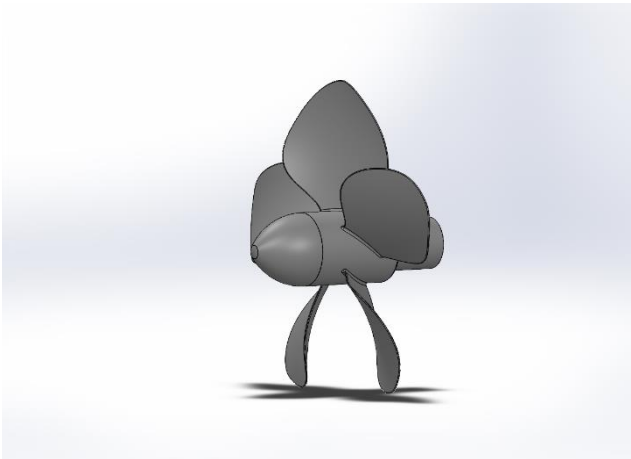


Figure 12: Rescue robot part: propeller.

2. Tracks

The tracks (figure 13 and 14) are very essential and efficient, because they make it possible for the robot to move on land especially rough surfaces. This is very important because after an accident, for example a mine or earthquake incident, the ground surface are usually not smooth.

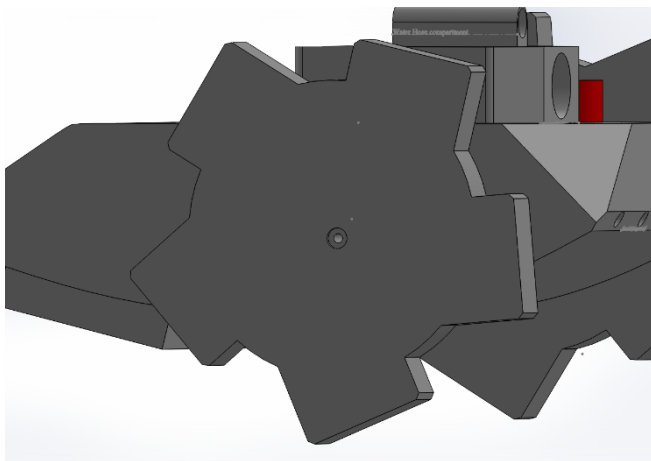


Figure 13: Rescue robot part: left sided track.

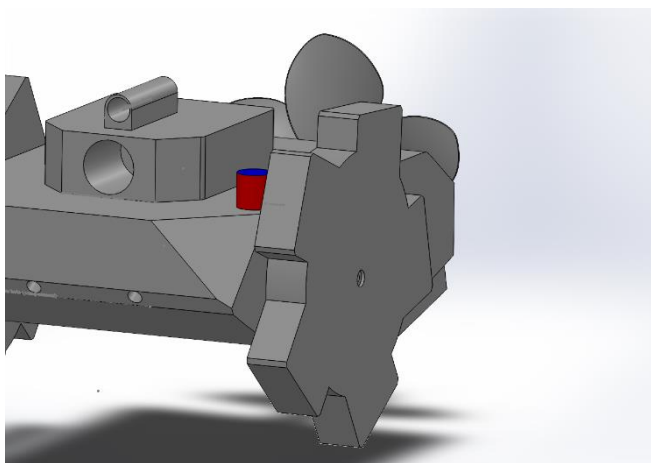


Figure 14: Rescue robot part: right sided track.

3. Life buoy compartment

This part of the robot is very essential for rescue on water (e.g. seas, oceans, beaches etc.) because this where the life buoy with floating line is released and other safety equipment that are essential for both land and water rescue.

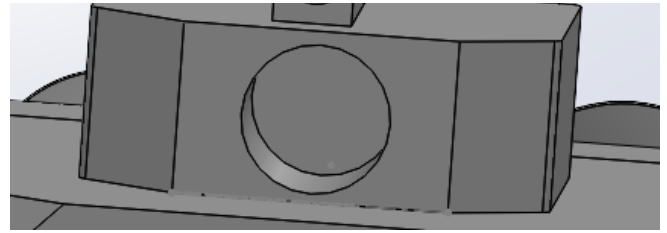


Figure 15: Rescue robot part: life buoy compartment.

4. Extinguisher compartment

Our robot has the capabilities to rescue during a fire incident, the extinguisher compartment (figure 16) is very essential during such incidents. A water hose or any fire extinguishing material is connected to the compartment and released to extinguish fire.

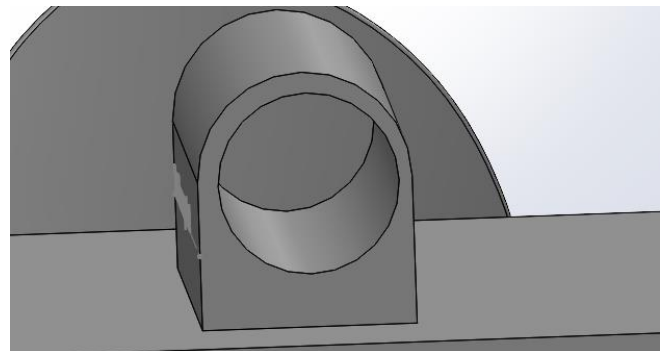


Figure 16: Rescue robot part: extinguisher compartment.

V. CODE

In this process, the C programming language is used. During this process, we gave our robot scenarios in a world representation and provided the respective the respective code to solve each rescue mission scenario.

A. *Scenerio 1*

In this scenario, a building has collapsed and our rescue robot has been deployed. They are walls represented as “#” as the border of the world. Our robot has been sent the command to rescue a human. Our robot gets a representation of the world it is navigating in. The representation is a character array in the C-programming language. In this real world representation the letter “O” represent empty spaces, the special character “#” represents walls. The letter “T” represents the human to be found and rescued. The letter “R” represent our robot. Based on the representation the robot is able to move north, south, east and west. For easy analysis, testing and debugging we have set a maximum number of steps of two hundred for the robot. We also set the system (robot) to stop, if it crashes into a wall “#”. For this scenario, the mission is considered successful if the human is found. Figures 17 to 22 are code excerpts of the solution to scenario 1.


```

def move_robot(world, robot):
    # Calculate the horizontal distance to the target
    horizontal_distance = abs(robot['x'] - target['x'])

    # Move the robot horizontally
    while horizontal_distance > 0:
        if robot['x'] < target['x']:
            robot['x'] += 1
        else:
            robot['x'] -= 1
        horizontal_distance -= 1

    # Calculate the vertical distance to the target
    vertical_distance = abs(robot['y'] - target['y'])

    # Move the robot vertically
    while vertical_distance > 0:
        if robot['y'] < target['y']:
            robot['y'] += 1
        else:
            robot['y'] -= 1
        vertical_distance -= 1

    # Return the final robot position
    return robot

```

Figure 17: Solution to scenario 1: part one.

```

def move_robot(world, robot):
    # Calculate the horizontal distance to the target
    horizontal_distance = abs(robot['x'] - target['x'])

    # Move the robot horizontally
    while horizontal_distance > 0:
        if robot['x'] < target['x']:
            robot['x'] += 1
        else:
            robot['x'] -= 1
        horizontal_distance -= 1

    # Calculate the vertical distance to the target
    vertical_distance = abs(robot['y'] - target['y'])

    # Move the robot vertically
    while vertical_distance > 0:
        if robot['y'] < target['y']:
            robot['y'] += 1
        else:
            robot['y'] -= 1
        vertical_distance -= 1

    # Return the final robot position
    return robot

```

Figure 21: Solution to scenario 1: part five.

```

def move_robot(world, robot):
    # Calculate the horizontal distance to the target
    horizontal_distance = abs(robot['x'] - target['x'])

    # Move the robot horizontally
    while horizontal_distance > 0:
        if robot['x'] < target['x']:
            robot['x'] += 1
        else:
            robot['x'] -= 1
        horizontal_distance -= 1

    # Calculate the vertical distance to the target
    vertical_distance = abs(robot['y'] - target['y'])

    # Move the robot vertically
    while vertical_distance > 0:
        if robot['y'] < target['y']:
            robot['y'] += 1
        else:
            robot['y'] -= 1
        vertical_distance -= 1

    # Return the final robot position
    return robot

```

Figure 18: Solution to scenario 1: part two.

```

def move_robot(world, robot):
    # Calculate the horizontal distance to the target
    horizontal_distance = abs(robot['x'] - target['x'])

    # Move the robot horizontally
    while horizontal_distance > 0:
        if robot['x'] < target['x']:
            robot['x'] += 1
        else:
            robot['x'] -= 1
        horizontal_distance -= 1

    # Calculate the vertical distance to the target
    vertical_distance = abs(robot['y'] - target['y'])

    # Move the robot vertically
    while vertical_distance > 0:
        if robot['y'] < target['y']:
            robot['y'] += 1
        else:
            robot['y'] -= 1
        vertical_distance -= 1

    # Return the final robot position
    return robot

```

Figure 22: Solution to scenario 1: part six.

```

def move_robot(world, robot):
    # Calculate the horizontal distance to the target
    horizontal_distance = abs(robot['x'] - target['x'])

    # Move the robot horizontally
    while horizontal_distance > 0:
        if robot['x'] < target['x']:
            robot['x'] += 1
        else:
            robot['x'] -= 1
        horizontal_distance -= 1

    # Calculate the vertical distance to the target
    vertical_distance = abs(robot['y'] - target['y'])

    # Move the robot vertically
    while vertical_distance > 0:
        if robot['y'] < target['y']:
            robot['y'] += 1
        else:
            robot['y'] -= 1
        vertical_distance -= 1

    # Return the final robot position
    return robot

```

Figure 19: Solution to scenario 1: part three.

```

def move_robot(world, robot):
    # Calculate the horizontal distance to the target
    horizontal_distance = abs(robot['x'] - target['x'])

    # Move the robot horizontally
    while horizontal_distance > 0:
        if robot['x'] < target['x']:
            robot['x'] += 1
        else:
            robot['x'] -= 1
        horizontal_distance -= 1

    # Calculate the vertical distance to the target
    vertical_distance = abs(robot['y'] - target['y'])

    # Move the robot vertically
    while vertical_distance > 0:
        if robot['y'] < target['y']:
            robot['y'] += 1
        else:
            robot['y'] -= 1
        vertical_distance -= 1

    # Return the final robot position
    return robot

```

Figure 20: Solution to scenario 1: part four.

The Robot was able to accomplish the mission successfully (scenario 1) without difficulty. Firstly the position of the robot was set to zero and the distance or the number of steps that the robot had to take, was determined by counting from the horizontal Target (set to zero also) and dividing by the number of walls (21 for horizontal as #). For the robot to reach the target (move horizontally), the target position of the horizontal from was subtracted from the target position of the robot. Same as for the robot to reach the target (move vertically), the target position of its vertical from was subtracted from the target position of the robot vertical position.

B. Scenorio 2

In this scenario, there are obstacles, such as walls, pipes and other materials from the collapsed building represented as “#” inside the world (map). The world also contains water, represented as “~”. To drives on water the driving mode is toggled. There are some conditions to this scenario, such as: the robot starts on land mode but can switch to water mode. If you are on land mode and wish to switch to water mode, the robot will stay in its current location and switch. The robot can only drive to it’s target location if it’s in is correct drive mode. The representation from scenario 1 still remain the same, only the addition of water for water rescue represented by “~”

C. Scenorio 3

In this scenario, the world gets more complex as there are more obstacles in the world restriction movement and bodies of water, so the robot must be able to avoid obstacles and change its mode more frequently to rescue the human. The solution to scenario 2 and 3 are combined. Three files were

needed for the solution of the scenarios: robo_world_2.c, robo_world_3.c, robot_rescue_pal.c and robot_rescue_pal.h.

```

1 // robot_rescue_pal.h
2 #ifndef ROBOT_RESUE_PAL_H
3 #define ROBOT_RESUE_PAL_H
4
5 #include "robo_world.h"
6 #include "robot.h"
7
8 void robot_rescue_pal(world_t *world, robot_t *robot);
9
10 #endif

```

Figure 23: Code for scenario 2 and 3: robot_rescue_pal.h

```

1 // robot_rescue_pal.c
2 #include "robot_rescue_pal.h"
3
4 void robot_rescue_pal(world_t *world, robot_t *robot)
5 {
6     // Calculate the distance from the robot to the target
7     int dx = robot->x - target->x;
8     int dy = robot->y - target->y;
9     int dist = sqrt(dx*dx + dy*dy);
10
11     // Update the robot's position
12     robot->x += dx/dist;
13     robot->y += dy/dist;
14 }

```

Figure 24: Code for scenario 2 and 3: robot_rescue_pal.c: part one.

```

1 // robot_rescue_pal.c
2 #include "robot_rescue_pal.h"
3
4 void robot_rescue_pal(world_t *world, robot_t *robot)
5 {
6     // Calculate the distance from the robot to the target
7     int dx = robot->x - target->x;
8     int dy = robot->y - target->y;
9     int dist = sqrt(dx*dx + dy*dy);
10
11     // Update the robot's position
12     robot->x += dx/dist;
13     robot->y += dy/dist;
14 }

```

Figure 25: Code for scenario 2 and 3: robot_rescue_pal.c: part two.

```

1 // robot_rescue_pal.c
2 #include "robot_rescue_pal.h"
3
4 void robot_rescue_pal(world_t *world, robot_t *robot)
5 {
6     // Calculate the distance from the robot to the target
7     int dx = robot->x - target->x;
8     int dy = robot->y - target->y;
9     int dist = sqrt(dx*dx + dy*dy);
10
11     // Update the robot's position
12     robot->x += dx/dist;
13     robot->y += dy/dist;
14 }

```

Figure 26: Code for scenario 2 and 3: robot_rescue_pal.c: part three.

```

1 // robo_world_2.c
2 #include "robo_world.h"
3 #include "robot.h"
4
5 void robo_world_2(world_t *world)
6 {
7     // Calculate the distance from the robot to the target
8     int dx = robot->x - target->x;
9     int dy = robot->y - target->y;
10     int dist = sqrt(dx*dx + dy*dy);
11
12     // Update the robot's position
13     robot->x += dx/dist;
14     robot->y += dy/dist;
15 }

```

Figure 27: Code for scenario 2 and 3: robo_world_2.c: part one.

```

1 // robo_world_2.c
2 #include "robo_world.h"
3 #include "robot.h"
4
5 void robo_world_2(world_t *world)
6 {
7     // Calculate the distance from the robot to the target
8     int dx = robot->x - target->x;
9     int dy = robot->y - target->y;
10     int dist = sqrt(dx*dx + dy*dy);
11
12     // Update the robot's position
13     robot->x += dx/dist;
14     robot->y += dy/dist;
15 }

```

Figure 28: Code for scenario 2 and 3: robo_world_2.c: part two.

```

1 // robo_world_2.c
2 #include "robo_world.h"
3 #include "robot.h"
4
5 void robo_world_2(world_t *world)
6 {
7     // Calculate the distance from the robot to the target
8     int dx = robot->x - target->x;
9     int dy = robot->y - target->y;
10     int dist = sqrt(dx*dx + dy*dy);
11
12     // Update the robot's position
13     robot->x += dx/dist;
14     robot->y += dy/dist;
15 }

```

Figure 29: Code for scenario 2 and 3: robo_world_2.c: part three.

```

1 // robo_world_2.c
2 #include "robo_world.h"
3 #include "robot.h"
4
5 void robo_world_2(world_t *world)
6 {
7     // Calculate the distance from the robot to the target
8     int dx = robot->x - target->x;
9     int dy = robot->y - target->y;
10     int dist = sqrt(dx*dx + dy*dy);
11
12     // Update the robot's position
13     robot->x += dx/dist;
14     robot->y += dy/dist;
15 }

```

Figure 30: Code for scenario 2 and 3: robo_world_2.c: part four.


```

1: for i in range(1, 10):
2:     print(i**2)
3:
4:
5:
6:
7:
8:
9:
10:
11:
12:
13:
14:
15:
16:
17:
18:
19:
20:
21:
22:
23:
24:
25:
26:
27:
28:
29:
30:
31:
32:
33:
34:
35:
36:
37:
38:
39:
40:
41:
42:
43:
44:
45:
46:
47:
48:
49:
50:
51:
52:
53:
54:
55:
56:
57:
58:
59:
60:
61:
62:
63:
64:
65:
66:
67:
68:
69:
70:
71:
72:
73:
74:
75:
76:
77:
78:
79:
80:
81:
82:
83:
84:
85:
86:
87:
88:
89:
90:
91:
92:
93:
94:
95:
96:
97:
98:
99:
100:

```

Output:

```

1
4
9
16
25
36
49
64
81

```

[illegible]

The screenshot shows a C++ program in a code editor. The program defines a string `world` and a loop that iterates 5 times. Inside the loop, the ASCII values of the characters in the string are printed. The output in the console shows the ASCII values for 'w', 'o', 'r', 'l', and 'd' repeated five times.

```

1 // C++ program to print ASCII values of characters in a string
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     string world = "world";
8     int i;
9     for (i = 0; i < 5; i++)
10     {
11         cout << "Character ASCII value is: ";
12         for (int j = 0; j < world.length(); j++)
13             cout << world[j] << " ";
14         cout << endl;
15     }
16     return 0;
17 }
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
```

[illegible][illegible][illegible][illegible][illegible]

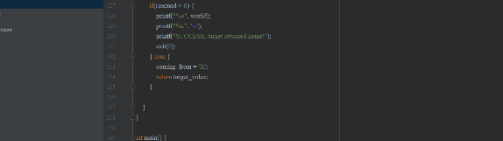
[illegible]

```

1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int n = 10;
6      int i = 1;
7      while (i <= n) {
8          int j = 1;
9          while (j <= n) {
1             if (j <= i || j >= n - i + 1) {
12                 cout << " * ";
13             } else {
14                 cout << "   ";
15             }
16             j++;
17         }
18         cout << endl;
19         i++;
20     }
21     return 0;
22 }

```

The screenshot shows a C++ IDE with a file named "diamond.cpp". The code defines a function `main` that prints a diamond shape of asterisks. The diamond has 10 rows and 10 columns, with a maximum width of 21 characters. The code uses nested loops to iterate over the rows and columns, and conditional statements to determine whether to print an asterisk or a space. The output is a diamond shape with 10 rows, 10 columns, and a maximum width of 21 characters.



```

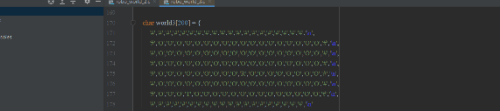
1  import numpy as np
2
3  class Layer:
4      def __init__(self, n_in, n_out):
5          self.n_in = n_in
6          self.n_out = n_out
7          self.weights = np.random.randn(n_in, n_out) * 0.01
8          self.bias = np.zeros((1, n_out))
9          self.z = None
10         self.a = None
11
12         def forward(self, x):
13             z = np.dot(x, self.weights) + self.bias
14             self.z = z
15             self.a = sigmoid(z)
16             return self.a
17
18         def backward(self, y, error):
19             # Calculate error gradients for weights and bias
20             dz = error * sigmoid_derivative(self.z)
21             dw = np.dot(x.T, dz)
22             db = dz
23             return dw, db
24
25         def sigmoid(self, z):
26             return 1 / (1 + np.exp(-z))
27
28         def sigmoid_derivative(self, z):
29             return self.sigmoid(z) * (1 - self.sigmoid(z))
30
31         def loss(self, y, a):
32             return np.mean((y - a) ** 2)
33
34         def loss_derivative(self, y, a):
35             return (y - a)
36
37         def train(self, x, y, epochs=1000):
38             for epoch in range(epochs):
39                 a = self.forward(x)
40                 loss = self.loss(y, a)
41                 dw, db = self.backward(y, error=self.loss_derivative(y, a))
42                 self.weights += dw
43                 self.bias += db
44                 print(f'Epoch {epoch+1}, Loss: {loss}')
45
46         def predict(self, x):
47             return self.forward(x)
48
49         def save(self, filename):
50             np.save(filename, self.weights)
51             np.save(filename + '_bias.npy', self.bias)
52
53         def load(self, filename):
54             self.weights = np.load(filename)
55             self.bias = np.load(filename + '_bias.npy')
56
57         def reset(self):
58             self.weights = np.random.randn(self.n_in, self.n_out) * 0.01
59             self.bias = np.zeros((1, self.n_out))
60             self.z = None
61             self.a = None
62
63         def __str__(self):
64             return f'Layer with {self.n_in} input nodes and {self.n_out} output nodes'
65
66         def __repr__(self):
67             return f'Layer({self.n_in}, {self.n_out})'
68
69         def __len__(self):
70             return self.n_out
71
72         def __getitem__(self, index):
73             return self.a[index]
74
75         def __setitem__(self, index, value):
76             self.a[index] = value
77
78         def __del__(self):
79             print('Layer object destroyed')
80
81         def __copy__(self):
82             return Layer(self.n_in, self.n_out)
83
84         def __deepcopy__(self, memo):
85             return Layer(self.n_in, self.n_out)
86
87         def __add__(self, other):
88             return Layer(self.n_in, self.n_out)
89
90         def __sub__(self, other):
91             return Layer(self.n_in, self.n_out)
92
93         def __mul__(self, other):
94             return Layer(self.n_in, self.n_out)
95
96         def __div__(self, other):
97             return Layer(self.n_in, self.n_out)
98
99         def __pow__(self, other):
100            return Layer(self.n_in, self.n_out)
101
102         def __matmul__(self, other):
103            return Layer(self.n_in, self.n_out)
104
105         def __rmatmul__(self, other):
106            return Layer(self.n_in, self.n_out)
107
108         def __iadd__(self, other):
109            return Layer(self.n_in, self.n_out)
110
111         def __isub__(self, other):
112            return Layer(self.n_in, self.n_out)
113
114         def __imul__(self, other):
115            return Layer(self.n_in, self.n_out)
116
117         def __idiv__(self, other):
118            return Layer(self.n_in, self.n_out)
119
120         def __ipow__(self, other):
121            return Layer(self.n_in, self.n_out)
122
123         def __imatmul__(self, other):
124            return Layer(self.n_in, self.n_out)
125
126         def __radd__(self, other):
127            return Layer(self.n_in, self.n_out)
128
129         def __rsub__(self, other):
130            return Layer(self.n_in, self.n_out)
131
132         def __rmul__(self, other):
133            return Layer(self.n_in, self.n_out)
134
135         def __rdiv__(self, other):
136            return Layer(self.n_in, self.n_out)
137
138         def __rpow__(self, other):
139            return Layer(self.n_in, self.n_out)
140
141         def __rmatmul__(self, other):
142            return Layer(self.n_in, self.n_out)
143
144         def __iadd__(self, other):
145            return Layer(self.n_in, self.n_out)
146
147         def __isub__(self, other):
148            return Layer(self.n_in, self.n_out)
149
150         def __imul__(self, other):
151            return Layer(self.n_in, self.n_out)
152
153         def __idiv__(self, other):
154            return Layer(self.n_in, self.n_out)
155
156         def __ipow__(self, other):
157            return Layer(self.n_in, self.n_out)
158
159         def __imatmul__(self, other):
160            return Layer(self.n_in, self.n_out)
161
162         def __radd__(self, other):
163            return Layer(self.n_in, self.n_out)
164
165         def __rsub__(self, other):
166            return Layer(self.n_in, self.n_out)
167
168         def __rmul__(self, other):
169            return Layer(self.n_in, self.n_out)
170
171         def __rdiv__(self, other):
172            return Layer(self.n_in, self.n_out)
173
174         def __rpow__(self, other):
175            return Layer(self.n_in, self.n_out)
176
177         def __rmatmul__(self, other):
178            return Layer(self.n_in, self.n_out)
179
180         def __iadd__(self, other):
181            return Layer(self.n_in, self.n_out)
182
183         def __isub__(self, other):
184            return Layer(self.n_in, self.n_out)
185
186         def __imul__(self, other):
187            return Layer(self.n_in, self.n_out)
188
189         def __idiv__(self, other):
190            return Layer(self.n_in, self.n_out)
191
192         def __ipow__(self, other):
193            return Layer(self.n_in, self.n_out)
194
195         def __imatmul__(self, other):
196            return Layer(self.n_in, self.n_out)
197
198         def __radd__(self, other):
199            return Layer(self.n_in, self.n_out)
200
201         def __rsub__(self, other):
202            return Layer(self.n_in, self.n_out)
203
204         def __rmul__(self, other):
205            return Layer(self.n_in, self.n_out)
206
207         def __rdiv__(self, other):
208            return Layer(self.n_in, self.n_out)
209
210         def __rpow__(self, other):
211            return Layer(self.n_in, self.n_out)
212
213         def __rmatmul__(self, other):
214            return Layer(self.n_in, self.n_out)
215
216         def __iadd__(self, other):
217            return Layer(self.n_in, self.n_out)
218
219         def __isub__(self, other):
220            return Layer(self.n_in, self.n_out)
221
222         def __imul__(self, other):
223            return Layer(self.n_in, self.n_out)
224
225         def __idiv__(self, other):
226            return Layer(self.n_in, self.n_out)
227
228         def __ipow__(self, other):
229            return Layer(self.n_in, self.n_out)
230
231         def __imatmul__(self, other):
232            return Layer(self.n_in, self.n_out)
233
234         def __radd__(self, other):
235            return Layer(self.n_in, self.n_out)
236
237         def __rsub__(self, other):
238            return Layer(self.n_in, self.n_out)
239
240         def __rmul__(self, other):
241            return Layer(self.n_in, self.n_out)
242
243         def __rdiv__(self, other):
244            return Layer(self.n_in, self.n_out)
245
246         def __rpow__(self, other):
247            return Layer(self.n_in, self.n_out)
248
249         def __rmatmul__(self, other):
250            return Layer(self.n_in, self.n_out)
251
252         def __iadd__(self, other):
253            return Layer(self.n_in, self.n_out)
254
255         def __isub__(self, other):
256            return Layer(self.n_in, self.n_out)
257
258         def __imul__(self, other):
259            return Layer(self.n_in, self.n_out)
260
261         def __idiv__(self, other):
262            return Layer(self.n_in, self.n_out)
263
264         def __ipow__(self, other):
265            return Layer(self.n_in, self.n_out)
266
267         def __imatmul__(self, other):
268            return Layer(self.n_in, self.n_out)
269
270         def __radd__(self, other):
271            return Layer(self.n_in, self.n_out)
272
273         def __rsub__(self, other):
274            return Layer(self.n_in, self.n_out)
275
276         def __rmul__(self, other):
277            return Layer(self.n_in, self.n_out)
278
279         def __rdiv__(self, other):
280            return Layer(self.n_in, self.n_out)
281
282         def __rpow__(self, other):
283            return Layer(self.n_in, self.n_out)
284
285         def __rmatmul__(self, other):
286            return Layer(self.n_in, self.n_out)
287
288         def __iadd__(self, other):
289            return Layer(self.n_in, self.n_out)
290
291         def __isub__(self, other):
292            return Layer(self.n_in, self.n_out)
293
294         def __imul__(self, other):
295            return Layer(self.n_in, self.n_out)
296
297         def __idiv__(self, other):
298            return Layer(self.n_in, self.n_out)
299
300         def __ipow__(self, other):
301            return Layer(self.n_in, self.n_out)
302
303         def __imatmul__(self, other):
304            return Layer(self.n_in, self.n_out)
305
306         def __radd__(self, other):
307            return Layer(self.n_in, self.n_out)
308
309         def __rsub__(self, other):
310            return Layer(self.n_in, self.n_out)
311
312         def __rmul__(self, other):
313            return Layer(self.n_in, self.n_out)
314
315         def __rdiv__(self, other):
316            return Layer(self.n_in, self.n_out)
317
318         def __rpow__(self, other):
319            return Layer(self.n_in, self.n_out)
320
321         def __rmatmul__(self, other):
322            return Layer(self.n_in, self.n_out)
323
324         def __iadd__(self, other):
325            return Layer(self.n_in, self.n_out)
326
327         def __isub__(self, other):
328            return Layer(self.n_in, self.n_out)
329
330         def __imul__(self, other):
331            return Layer(self.n_in, self.n_out)
332
333         def __idiv__(self, other):
334            return Layer(self.n_in, self.n_out)
335
336         def __ipow__(self, other):
337            return Layer(self.n_in, self.n_out)
338
339         def __imatmul__(self, other):
340            return Layer(self.n_in, self.n_out)
341
342         def __radd__(self, other):
343            return Layer(self.n_in, self.n_out)
344
345         def __rsub__(self, other):
346            return Layer(self.n_in, self.n_out)
347
348         def __rmul__(self, other):
349            return Layer(self.n_in, self.n_out)
350
351         def __rdiv__(self, other):
352            return Layer(self.n_in, self.n_out)
353
354         def __rpow__(self, other):
355            return Layer(self.n_in, self.n_out)
356
357         def __rmatmul__(self, other):
358            return Layer(self.n_in, self.n_out)
359
360         def __iadd__(self, other):
361            return Layer(self.n_in, self.n_out)
362
363         def __isub__(self, other):
364            return Layer(self.n_in, self.n_out)
365
366         def __imul__(self, other):
367            return Layer(self.n_in, self.n_out)
368
369         def __idiv__(self, other):
370            return Layer(self.n_in, self.n_out)
371
372         def __ip
```

[illegible]

```

1  #!/usr/bin/perl -w
2
3  use strict;
4  use warnings;
5
6  my $max = 1000000;
7  my $min = 0;
8
9  my $max_val = 0;
10 my $min_val = 0;
11
12 my $max_idx = 0;
13 my $min_idx = 0;
14
15 my $max_val_str = "0";
16 my $min_val_str = "0";
17
18 my $max_idx_str = "0";
19 my $min_idx_str = "0";
20
21 my $max_val_str = "0";
22 my $min_val_str = "0";
23
24 my $max_idx_str = "0";
25 my $min_idx_str = "0";
26
27 my $max_val_str = "0";
28 my $min_val_str = "0";
29
30 my $max_idx_str = "0";
31 my $min_idx_str = "0";
32
33 my $max_val_str = "0";
34 my $min_val_str = "0";
35
36 my $max_idx_str = "0";
37 my $min_idx_str = "0";
38
39 my $max_val_str = "0";
40 my $min_val_str = "0";
41
42 my $max_idx_str = "0";
43 my $min_idx_str = "0";
44
45 my $max_val_str = "0";
46 my $min_val_str = "0";
47
48 my $max_idx_str = "0";
49 my $min_idx_str = "0";
50
51 my $max_val_str = "0";
52 my $min_val_str = "0";
53
54 my $max_idx_str = "0";
55 my $min_idx_str = "0";
56
57 my $max_val_str = "0";
58 my $min_val_str = "0";
59
60 my $max_idx_str = "0";
61 my $min_idx_str = "0";
62
63 my $max_val_str = "0";
64 my $min_val_str = "0";
65
66 my $max_idx_str = "0";
67 my $min_idx_str = "0";
68
69 my $max_val_str = "0";
70 my $min_val_str = "0";
71
72 my $max_idx_str = "0";
73 my $min_idx_str = "0";
74
75 my $max_val_str = "0";
76 my $min_val_str = "0";
77
78 my $max_idx_str = "0";
79 my $min_idx_str = "0";
80
81 my $max_val_str = "0";
82 my $min_val_str = "0";
83
84 my $max_idx_str = "0";
85 my $min_idx_str = "0";
86
87 my $max_val_str = "0";
88 my $min_val_str = "0";
89
90 my $max_idx_str = "0";
91 my $min_idx_str = "0";
92
93 my $max_val_str = "0";
94 my $min_val_str = "0";
95
96 my $max_idx_str = "0";
97 my $min_idx_str = "0";
98
99 my $max_val_str = "0";
100 my $min_val_str = "0";
101
102 my $max_idx_str = "0";
103 my $min_idx_str = "0";
104
105 my $max_val_str = "0";
106 my $min_val_str = "0";
107
108 my $max_idx_str = "0";
109 my $min_idx_str = "0";
110
111 my $max_val_str = "0";
112 my $min_val_str = "0";
113
114 my $max_idx_str = "0";
115 my $min_idx_str = "0";
116
117 my $max_val_str = "0";
118 my $min_val_str = "0";
119
120 my $max_idx_str = "0";
121 my $min_idx_str = "0";
122
123 my $max_val_str = "0";
124 my $min_val_str = "0";
125
126 my $max_idx_str = "0";
127 my $min_idx_str = "0";
128
129 my $max_val_str = "0";
130 my $min_val_str = "0";
131
132 my $max_idx_str = "0";
133 my $min_idx_str = "0";
134
135 my $max_val_str = "0";
136 my $min_val_str = "0";
137
138 my $max_idx_str = "0";
139 my $min_idx_str = "0";
140
141 my $max_val_str = "0";
142 my $min_val_str = "0";
143
144 my $max_idx_str = "0";
145 my $min_idx_str = "0";
146
147 my $max_val_str = "0";
148 my $min_val_str = "0";
149
150 my $max_idx_str = "0";
151 my $min_idx_str = "0";
152
153 my $max_val_str = "0";
154 my $min_val_str = "0";
155
156 my $max_idx_str = "0";
157 my $min_idx_str = "0";
158
159 my $max_val_str = "0";
160 my $min_val_str = "0";
161
162 my $max_idx_str = "0";
163 my $min_idx_str = "0";
164
165 my $max_val_str = "0";
166 my $min_val_str = "0";
167
168 my $max_idx_str = "0";
169 my $min_idx_str = "0";
170
171 my $max_val_str = "0";
172 my $min_val_str = "0";
173
174 my $max_idx_str = "0";
175 my $min_idx_str = "0";
176
177 my $max_val_str = "0";
178 my $min_val_str = "0";
179
180 my $max_idx_str = "0";
181 my $min_idx_str = "0";
182
183 my $max_val_str = "0";
184 my $min_val_str = "0";
185
186 my $max_idx_str = "0";
187 my $min_idx_str = "0";
188
189 my $max_val_str = "0";
190 my $min_val_str = "0";
191
192 my $max_idx_str = "0";
193 my $min_idx_str = "0";
194
195 my $max_val_str = "0";
196 my $min_val_str = "0";
197
198 my $max_idx_str = "0";
199 my $min_idx_str = "0";
200
201 my $max_val_str = "0";
202 my $min_val_str = "0";
203
204 my $max_idx_str = "0";
205 my $min_idx_str = "0";
206
207 my $max_val_str = "0";
208 my $min_val_str = "0";
209
210 my $max_idx_str = "0";
211 my $min_idx_str = "0";
212
213 my $max_val_str = "0";
214 my $min_val_str = "0";
215
216 my $max_idx_str = "0";
217 my $min_idx_str = "0";
218
219 my $max_val_str = "0";
220 my $min_val_str = "0";
221
222 my $max_idx_str = "0";
223 my $min_idx_str = "0";
224
225 my $max_val_str = "0";
226 my $min_val_str = "0";
227
228 my $max_idx_str = "0";
229 my $min_idx_str = "0";
230
231 my $max_val_str = "0";
232 my $min_val_str = "0";
233
234 my $max_idx_str = "0";
235 my $min_idx_str = "0";
236
237 my $max_val_str = "0";
238 my $min_val_str = "0";
239
240 my $max_idx_str = "0";
241 my $min_idx_str = "0";
242
243 my $max_val_str = "0";
244 my $min_val_str = "0";
245
246 my $max_idx_str = "0";
247 my $min_idx_str = "0";
248
249 my $max_val_str = "0";
250 my $min_val_str = "0";
251
252 my $max_idx_str = "0";
253 my $min_idx_str = "0";
254
255 my $max_val_str = "0";
256 my $min_val_str = "0";
257
258 my $max_idx_str = "0";
259 my $min_idx_str = "0";
260
261 my $max_val_str = "0";
262 my $min_val_str = "0";
263
264 my $max_idx_str = "0";
265 my $min_idx_str = "0";
266
267 my $max_val_str = "0";
268 my $min_val_str = "0";
269
270 my $max_idx_str = "0";
271 my $min_idx_str = "0";
272
273 my $max_val_str = "0";
274 my $min_val_str = "0";
275
276 my $max_idx_str = "0";
277 my $min_idx_str = "0";
278
279 my $max_val_str = "0";
280 my $min_val_str = "0";
281
282 my $max_idx_str = "0";
283 my $min_idx_str = "0";
284
285 my $max_val_str = "0";
286 my $min_val_str = "0";
287
288 my $max_idx_str = "0";
289 my $min_idx_str = "0";
290
291 my $max_val_str = "0";
292 my $min_val_str = "0";
293
294 my $max_idx_str = "0";
295 my $min_idx_str = "0";
296
297 my $max_val_str = "0";
298 my $min_val_str = "0";
299
300 my $max_idx_str = "0";
301 my $min_idx_str = "0";
302
303 my $max_val_str = "0";
304 my $min_val_str = "0";
305
306 my $max_idx_str = "0";
307 my $min_idx_str = "0";
308
309 my $max_val_str = "0";
310 my $min_val_str = "0";
311
312 my $max_idx_str = "0";
313 my $min_idx_str = "0";
314
315 my $max_val_str = "0";
316 my $min_val_str = "0";
317
318 my $max_idx_str = "0";
319 my $min_idx_str = "0";
320
321 my $max_val_str = "0";
322 my $min_val_str = "0";
323
324 my $max_idx_str = "0";
325 my $min_idx_str = "0";
326
327 my $max_val_str = "0";
328 my $min_val_str = "0";
329
330 my $max_idx_str = "0";
331 my $min_idx_str = "0";
332
333 my $max_val_str = "0";
334 my $min_val_str = "0";
335
336 my $max_idx_str = "0";
337 my $min_idx_str = "0";
338
339 my $max_val_str = "0";
340 my $min_val_str = "0";
341
342 my $max_idx_str = "0";
343 my $min_idx_str = "0";
344
345 my $max_val_str = "0";
346 my $min_val_str = "0";
347
348 my $max_idx_str = "0";
349 my $min_idx_str = "0";
350
351 my $max_val_str = "0";
352 my $min_val_str = "0";
353
354 my $max_idx_str = "0";
355 my $min_idx_str = "0";
356
357 my $max_val_str = "0";
358 my $min_val_str = "0";
359
360 my $max_idx_str = "0";
361 my $min_idx_str = "0";
362
363 my $max_val_str = "0";
364 my $min_val_str = "0";
365
366 my $max_idx_str = "0";
367 my $min_idx_str = "0";
368
369 my $max_val_str = "0";
370 my $min_val_str = "0";
371
372 my $max_idx_str = "0";
373 my $min_idx_str = "0";
374
375 my $max_val_str = "0";
376 my $min_val_str = "0";
377
378 my $max_idx_str = "0";
379 my $min_idx_str = "0";
380
381 my $max_val_str = "0";
382 my $min_val_str = "0";
383
384 my $max_idx_str = "0";
385 my $min_idx_str = "0";
386
387 my $max_val_str = "0";
388 my $min_val_str = "0";
389
390 my $max_idx_str = "0";
391 my $min_idx_str = "0";
392
393 my $max_val_str = "0";
394 my $min_val_str = "0";
395
396 my $max_idx_str = "0";
397 my $min_idx_str = "0";
398
399 my $max_val_str = "0";
400 my $min_val_str = "0";
401
402 my $max_idx_str = "0";
403 my $min_idx_str = "0";
404
405 my $max_val_str = "0";
406 my $min_val_str = "0";
407
408 my $max_idx_str = "0";
409 my $min_idx_str = "0";
410
411 my $max_val_str = "0";
412 my $min_val_str = "0";
413
414 my $max_idx_str = "0";
415 my $min_idx_str = "0";
416
417 my $max_val_str = "0";
418 my $min_val_str = "0";
```

[illegible]



```
1 // Random Numbers
2 #include <iostream>
3 #include <cstdlib>
4 #include <ctime>
5
6 using namespace std;
7
8 int main()
9 {
10     // Seed the random number generator
11     srand(time(0));
12
13     // Generate and print random numbers
14     for (int i = 0; i < 10; i++)
15     {
16         for (int j = 0; j < 10; j++)
17         {
18             // Generate a random number between 0 and 100
19             int random_number = rand() % 100;
20
21             // Print the random number
22             cout << random_number << " ";
23
24             // Print a newline character after every 10 numbers
25             if (j % 10 == 9)
26                 cout << endl;
27         }
28     }
29
30     return 0;
31 }
```

```
1 // Create a new world from a given seed
2 void World::createWorldFromSeed(const unsigned int &seed) {
3     // Create a new world
4     createWorld();
5     // Set the seed
6     setSeed(seed);
7 }
8
9 // Create a new world
10 void World::createWorld() {
11     // Create a new world
12     createWorldFromSeed(0);
13 }
14
15 // Set the seed
16 void World::setSeed(const unsigned int &seed) {
17     // Set the seed
18     mSeed = seed;
19 }
20
21 // Get the seed
22 unsigned int World::getSeed() const {
23     return mSeed;
24 }
25
26 // Get the world
27 World * World::getWorld() {
28     return this;
29 }
30
31 // Get the world
32 World * World::getWorldFromSeed(const unsigned int &seed) {
33     // Create a new world
34     createWorldFromSeed(seed);
35     return this;
36 }
37
38 // Get the world
39 World * World::getWorldFromSeed(const unsigned int &seed, const unsigned int &steps) {
40     // Create a new world
41     createWorldFromSeed(seed);
42     // Run the world for a given number of steps
43     run(steps);
44     return this;
45 }
46
47 // Run the world for a given number of steps
48 void World::run(const unsigned int &steps) {
49     // Run the world for a given number of steps
50     for (unsigned int i = 0; i < steps; i++) {
51         // Run the world for a given number of steps
52         update();
53     }
54 }
```

Figure 36: Code for scenario 2 and 3: robo_world_3.c: part seven.

Figure 36: Code for scenario 2 and 3: robo_world_3.c: part eleven.

In order to drive through the map and the robot been able to change its way of movement in water and on land, initial energy was set to zero and rescued target was also determined

as zero. When the robot moves, it means energy was created to toggle to a new position (line). And to be able to reach the target index for a direction, for example South, there has to be increase in the steps of the robot position (index) as the width increases).

D. Shortest path finder

Time is of essence in a rescue mission, and one way to save time is by finding the shortest path. In our rescue robot, the A* search algorithm is used to find the shortest path. The A* search algorithm is one of the best and popular technique used in path finding.

VI. CIRCUIT DIAGRAMS

In this process, the schematic diagram showing the representation of the elements of the system and the graphical representation of the circuit and circuit elements are drawn.

A. Smoke and Gas sensor

The smoke and gas sensor is used in rescue missions by the robot to warn against fire, and emission of gases that are harmful to the human health.

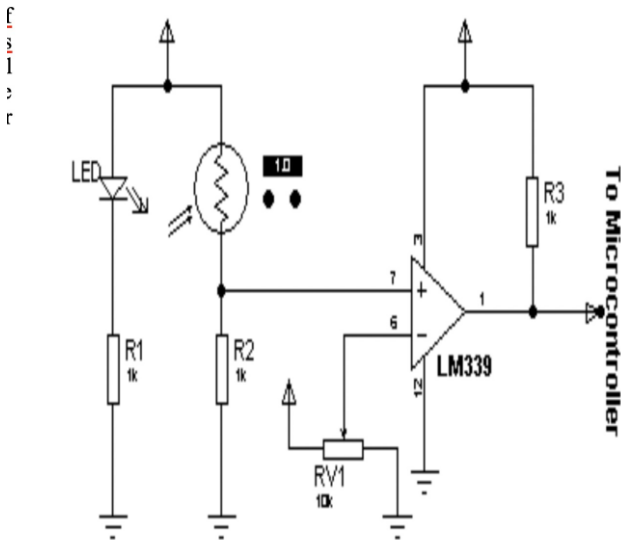


Figure 37: Circuit diagram smoke and gas sensor (Dey, Hossen, Noor and Ahmmed, 2013).

B. Ultrasonic sensor

An ultrasonic sensor is an instrument that measures the distance to an object (or obstacle) using ultrasonic sound waves, this electronic elements prevent the robot from crashing into a wall or obstacle. The ultrasonic sensors consists of a set of ultrasonic receivers and transmitter, which operates at the same frequency.

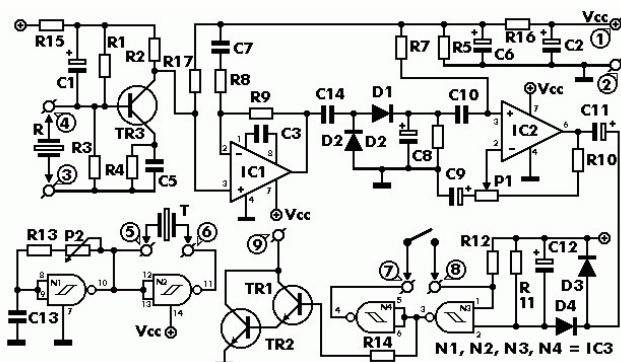
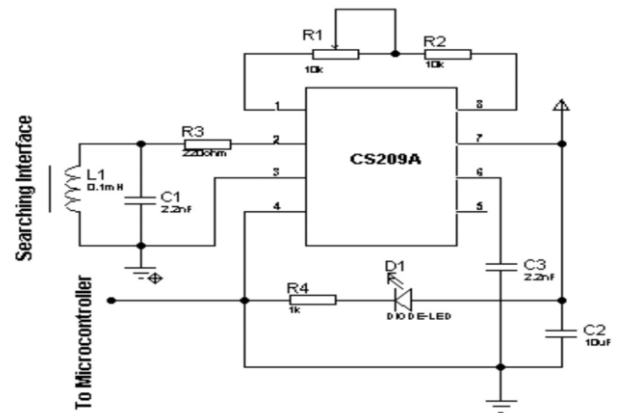


Figure 38: Circuit diagram of ultrasonic sensor (Ultrasonic Sensor Circuit, 2015)

C. Metal Detector

Metal detectors work on the principle of transmitting a magnetic field and analysing a return signal from the target and environment.



B. Human detection

Another property of our rescue robot is obstacle identification.

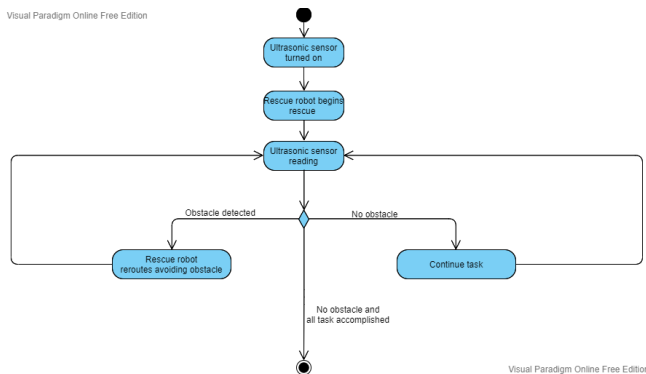


Figure 42: Obstacle detection activity diagram.

VIII. DYNAMIC BEHAVIOUR MODELLING IN MATLAB AND SIMULINK

Dynamic analysis and controller design of the system (rescue robot) is possible in Matlab and Simulink. What is or maybe imperative are the necessities and data that are required to construct the kinematic model in matlab and simulink, such data is gotten from the modelling languages (UML and SysML) used in the modelling of the system and also from the Solid works model. The information gotten from the Solid work model is the starting point for modelling in Matlab/Simulink along with the controller design objectives gotten from SysML. The simulation and analysis in MATLAB and Simulink will result in: finalising the power requirements for each motor based on optimising the controller, recommendations on sensor precision and possible updates to the requirements of the system (the rescue robot) (Qamar, Wikander and Törngren, 2010).

IX. CONCLUSION

Accidents are the third leading cause of deaths worldwide (FastStats, 2021), there is an estimation of four million (4000, 000) deaths every year due to the natural disasters, there are an estimated 236,000 annual drowning deaths worldwide. “Global estimation may significantly underestimate the public health problem related to drowning” -World Health Organization (W.H.O). To reduce and eventually, totally eradicate this global issue, the development of search and rescue robots, that assists in the recovery of persons involved in accidents, natural disasters and help in the exploration of dangerous areas are created. This paper highlights the complications in designing the robots and its capabilities. With all said and done, it is safe to say the rescue robot can be successfully designed based on the content of this paper and can rescue humans, after an accident has occurred.

X. TASK

The section shows who wrote what part in the paper:

Name	Task
Ali Ahemd	IV
Damilola Awosuru	Abstract, I, II, III, IV, VI, VII, VIII, IX.
Johnson Darko	V

ACKNOWLEDGMENT (Heading 5)

Ali Ahmed:

No acknowledgement.

Damilola Awosuru:

I will like to thank my LORD and PERSONAL SAVIOUR: JESUS CHRIST, and THE HOLY SPIRIT (MY BEST FRIEND).

Johnson Darko:

I will like to thank Ali Ahemd and W3 school.

REFERENCES


- [1] Douglass, B., 2009. *Real-time design patterns*. Boston: Addison-Wesley.
- [2] Dey, G., Hossen, R., Noor, M. and Ahmmed, K., 2013. Distance controlled rescue and security mobile robot. 2013 International Conference on Informatics, Electronics and Vision (ICIEV),...
- [3] McComb, G., 2018. *How to make a robot*. San Francisco: Maker Media.
- [4] Electroschematics. 2015. Ultrasonic Sensor Circuit. [online] Available at: <<https://www.elctroschematics.com/ultrasonic-sensor-circuit/>> [Accessed 16 July 2021]
- [5] Medium. 2021. Review: FCN — Fully Convolutional Network (Semantic Segmentation). [online] Available at: <<https://towardsdatascience.com/review-fcn-semantic-segmentation-eb8c9b50d2d1>> [Accessed 16 July 2021].
- [6] GeeksforGeeks. 2021. A* Search Algorithm - GeeksforGeeks. [online] Available at: <<https://www.geeksforgeeks.org/a-search-algorithm/>> [Accessed 16 July 2021].
- [7] Qamar, A., Wikander, J. and Törngren, M., 2010. Integrating multi-domain models for the design and development of mechatronic systems <https://www.researchgate.net/publication/264195025_Integrating_multi-domain_models_for_the_design_and_development_of_mechatronic_systems> [Accessed 16 July 2021].
- [8] Cdc.gov.2021.FastStats.[online]Availableat: <<https://www.cdc.gov/nchs/fastats/leading-causes-of-death.htm>> [Accessed 16 July 2021].

Eidesstattliche Erklärung

Hiermit bestätige ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen sowie Hilfsmittel genutzt habe. Alle Ausführungen, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind deutlich kenntlich gemacht. Außerdem versichere ich, dass die vorliegende Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Affidavit

I hereby confirm that I have written this paper independently and have not used any sources or aids other than those indicated. All statements taken from other sources in wording or sense are clearly marked. Furthermore, I assure that this paper has not been part of a course or examination in the same or a similar version.

Ali Hisham Omer Ahmed	16. Jul 2021	
Name, Vorname	Ort, Datum	Unterschrift
Last Name, First Name	Location, Date	Signature

Eidesstattliche Erklärung

Hiermit bestätige ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen sowie Hilfsmittel genutzt habe. Alle Ausführungen, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind deutlich kenntlich gemacht. Außerdem versichere ich, dass die vorliegende Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Affidavit

I hereby confirm that I have written this paper independently and have not used any sources or aids other than those indicated. All statements taken from other sources in wording or sense are clearly marked. Furthermore, I assure that this paper has not been part of a course or examination in the same or a similar version.

Awosuru, Damilola	Lippstadt, 16.07.2021	Damilola
Name, Vorname	Ort, Datum	Unterschrift
Last Name, First Name	Location, Date	Signature

Eidesstattliche Erklärung

Hiermit bestätige ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen sowie Hilfsmittel genutzt habe. Alle Ausführungen, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind deutlich kenntlich gemacht. Außerdem versichere ich, dass die vorliegende Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Affidavit

I hereby confirm that I have written this paper independently and have not used any sources or aids other than those indicated. All statements taken from other sources in wording or sense are clearly marked. Furthermore, I assure that this paper has not been part of a course or examination in the same or a similar version.

Darko, Johnson Kwabena	Neuss,15-07-2021	JKD
Name, Vorname	Ort, Datum	Unterschrift
Last Name, First Name	Location, Date	Signature