

ResQ Robot

Bin Zaprunnizam
Muhammad Amirul Hakimi
Team ResQ Robot
Hochschule Hamm Lippstadt
Lippstadt, Germany
muhammad-amirul-hakimi.bin-
zaprunnizam@stud.hshl.de

Bin Abdul Malik
Muhammad Amjad
Team ResQ Robot
Hochschule Hamm-Lippstadt
Lippstadt, Germany
muhammad-amjad-bin.abdul-
malik@stud.hshl.de

Bin Mohd Fauzi
Muhammad Iqbal
Team ResQ Robot
Hochschule Hamm-Lippstadt
Lippstadt, Germany
muhammad-iqbal.bin-mohd-
fauzi@stud.hshl.de

Bin Mohamad Shabri
Muhammad Farid Izwan
Team ResQ Robot
Hochschule Hamm-Lippstadt
Lippstadt, Germany
muhammad-farid-izwan.bin-mohamad-
shabri@stud.hshl.de

Abdul-Azeez Olanlokun
Team ResQ Robot
Hochschule Hamm-Lippstadt
Lippstadt, Germany
Abdul-Azeez.olanlokun@stud.hshl.de

Amit Chakma
Team ResQ Robot
Hochschule Hamm-Lippstadt
Lippstadt, Germany
Amit-Chakma@stud.hshl.de

Abstract—Over the last decade, the technology of robotics has been rapidly improved to serve safety and reliability in our digital world. We came a long way after developing our first robot in 1954. This paper aims to describe the design of our rescue robot system that we built, we did include software specification, a complete design and codes. Our main was to help our rescue team in extreme situations where human involvement is too risky. We considered water scenarios and land scenarios as our prime base. We gave our robot sudden goals to achieve in these scenarios like finding the target in both of the scenarios. We also gave challenges in the given territory. We ended up implementing a reliable and compact base for our rescue robot.

I. INTRODUCTION

Natural disasters like that of the Tsunami (1) are uncontrollable disasters that can occur at any time without prior notice. It is then important to have emergency rescue to help save lives and properties. Such necessity motivated us to bring about a technological innovation that can serve as an emergency rescue in this kind of situation. As mentioned earlier, the Tsunami (1) “is a series of enormous ocean wave caused by earthquakes underwater landslides, volcanic eruptions, or asteroids.”

We then thought about a rescue robot that can be used in such emergency, not only that it could move on land, but also drive on/in water and in dangerous areas, where humans cant access easily. We then designed ResQ robot that drives autonomously on land and on water, and that can detect and overcome obstacles on land, and also detect humans in water. Our robot serves as a first aid rescue support before human intervention.

During the stages of building our robot, we faced many challenges, in terms of obstacle obstructions, but with much diligent research by our team, we were able to overcome these challenges and difficulties in making our robot work perfectly as we want. Also, we learnt a lot that will surely aid us in future projects and how to successfully achieve a working project.

II. SYSTEM MODELING

A. SysML DIAGRAMS (Abdul-Azeez Olanlokun)

To design our robot to satisfy all engineering principles, we made use of SysML diagrams learned from our course of study in Systems Engineering, such as Requirement diagram, Constraint diagram, Block diagram, Use Case, Sequence/Activity diagrams and lastly system architecture, that helped us model our robot, as this kind of project is a complex system that requires a model that supports analysis, specification, design, verification and validation.

1. REQUIREMENT DIAGRAM

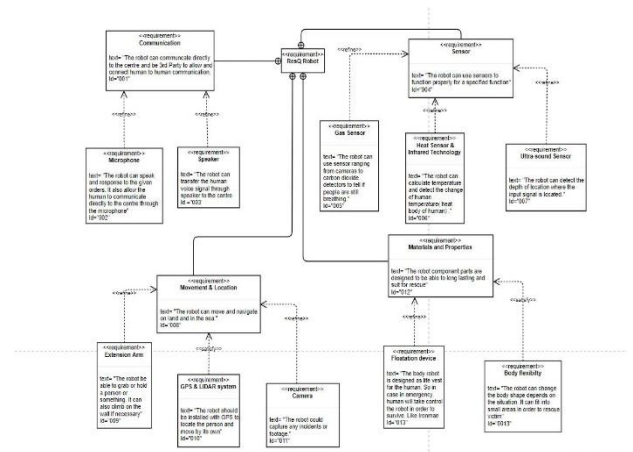


Fig. 1. Requirement Diagram

The Requirement Diagram (fig 1) shows a graphical representation of all our project requirements. With this diagram, we were able to show a textual description of our requirements with priorities in functionalities. As we know requirements takes a huge part in any system development, so it was important we analysed it for the success of our project.

In this requirement diagram, as shown in fig 1, our ResQ Robot has range of requirements connected to one another with high, medium and low priorities, which are noted with

the word <<refine>>. The must have of our robot are as follow:

Communication: which is refine with two communication channels (Microphone and Speaker), shows how the robot is able to communicate directly to the admin centre through the two-communication channels.

Movement and Location: This is the means to which our robot can navigate around, and it is also refine with camera and GPS system.

Materials and Properties: Our Robot parts are designed to withstand wear and tear, and also be aerodynamic. It is also refined with floatation device and body flexibility.

Sensor: Our robot needs sensors to perform its design purposes. It is refined with Gas, Heat & infrared and Ultra-Sonic sensors.

2. CONSTRAINT/CONTEXT DIAGRAM

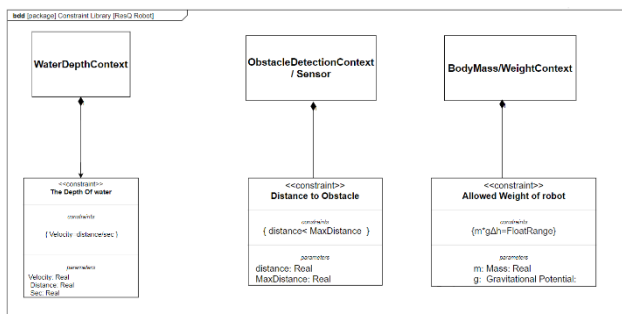


Fig. 2. Context Diagram

The constraint diagram (Fig. 2) basically specifies a network of constraints that shows mathematical expressions, these constraints are mostly from physical properties of a system. Our constraint diagram shows a graphical representation of mathematical expressions of all the constraints in our robot in a constraint Block.

The properties of our robot constraints are: The depth of water constraint, Distance to obstacle constraint and Allowed weight of robot constraint.

3. BLOCK DIAGRAM

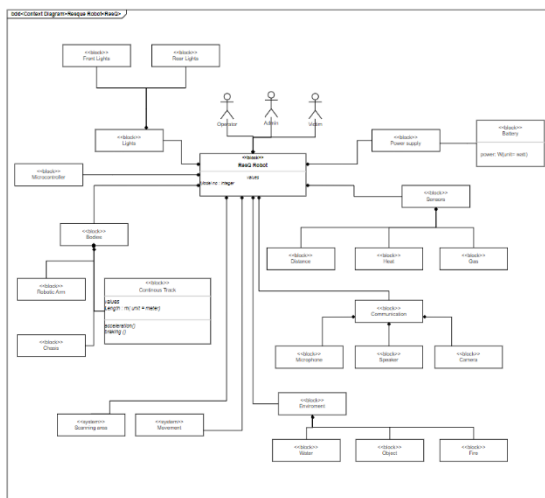


Fig. 3. Block Diagram

The Block Diagram (Fig 3) shows the modular unit of our system, which abridges the contents such as the operations, attributes and constraints of our system. The block diagram shows the structure of our robotic system, from the physical to the logical attributes.

4. INTERNAL BLOCK DIAGRAM

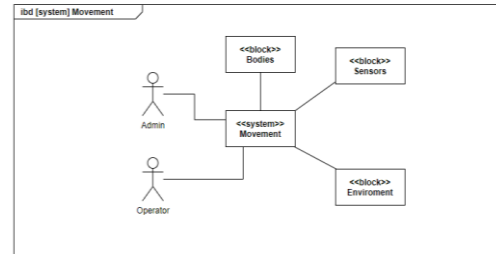


Fig. 4. Internal Block Diagram(Movement)

The Internal Block Diagram (Fig. 4) shows a graphical view of the internal structure of a block in our system. This shows all the main decomposition of our robot in one single internal block diagram. This decomposition shows the <<system>> Movement which other external features act on it. The external features are: The Admin/Operator, bodies, Sensors and Environment.

The Scanning Area (IBD):

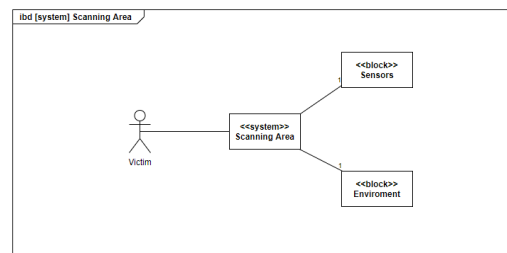


Fig. 5. Internal Block Diagram(Scanning Area)

The Scanning area (Fig.5) is a segment of the internal block diagram, showing how our robot navigates around the environment and searches for the victim.

5. ACTIVITY DIAGRAM (Amit Chakma):

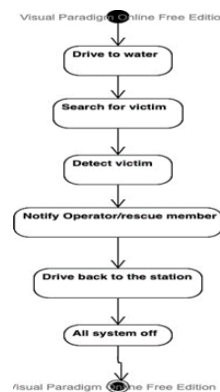


Fig. 6. Activity Diagram

7. SYSTEM ARCHITECHTURE DIAGRAM

In the activity diagram [Fig 6] we can see the activities of the robot in water. Here the robot is activated by the rescue members and the robot dives in to search for the victim. When the robot detects the dummy, it sends the exact location to the safety guard. After the robot finishes the mission, it returns to its charging station for the next rescue.

6. USE CASE DIAGRAM

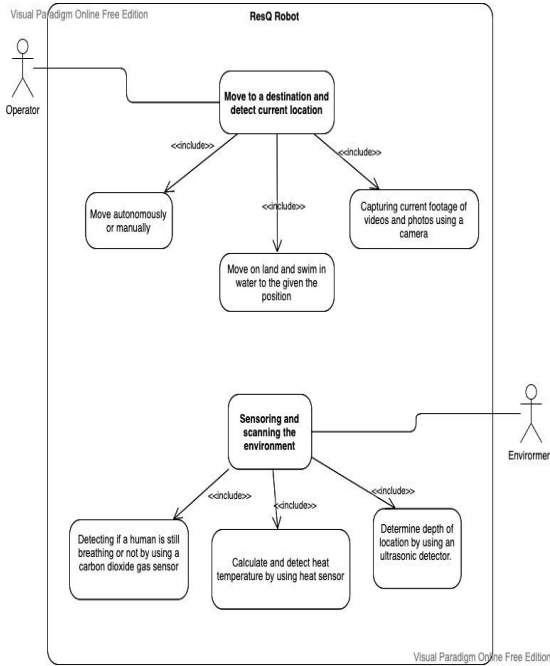


Fig. 7. Use Case Diagram

In the use case diagram [Fig 7] can see the potential scenarios of our rescue robot. We have an operator who is responsible for controlling the robot.

The robot can move autonomously and can also be controlled by the operator. We will also get continuous GPS location from the robot in any scenarios. We included a camera which is capable of taking videos and photos in extreme conditions.

We also have an environment that is responsible for feeding data to our sensors. By using our sensors, we can check the body temperature of the victim. We will be also able to detect that our victim is breathing using a CO2 detector and at last we also have an ultrasonic sensor to measure the distance or depth.

For the system architecture [Fig 8] we have used Five layer architecture which is very common in real time systems. In a layered pattern we can work on each of the layers independently and update as required.

Application layer contains the application-level classes . In the application layer we can see the common tasks handled by the robot from things like diving on water , driving on land and water , swimming , object detection.

User interface layer contains classes specific to the user interface.I/O operations help transfer information between computer main memory and the outside world. Context switching used to switch between processes to handle multiple tasks .

Communication domain contains classes necessary to transport data ,commands and events among the object. Here we have Buses which help to transfer digital signals to transfer data rapidly .Interprocess communication used for exchanging data between multiple threads in one or more programs. The Processes may be running on multiple computers connected by a WLAN.

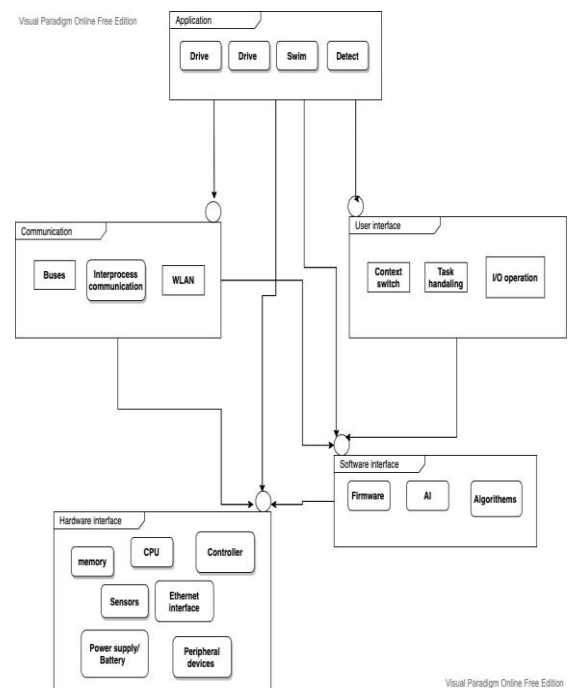


Fig. 8. System Architecture Diagram

Hardware interface layer provides classes that represent devices and interfaces. We have our memory , CPU , peripheral devices , sensors , controller and ethernet interface.

Software interface layer provides the classes to manage threads and memory and other system services. Here we have firmware , AI and algorithms which will provide instructions for our hardware devices.

III. CODING

We have 4 different tasks that we need to solve. Each task has different maps and different condition to fulfill. Our main purpose throughout this project is that our ResQ robot (R) would be able to reach the target whether the target is on the water(t) or on land mode(T). To reach the target, the robot must be able to destroy obstacles (*), avoid from hitting the walls(#) and changing the mode. For example, by changing the mode from water mode to land mode. In order to fulfill the task requirement, we have decided to use the Breadth First Search algorithm (BFS).

A. Breadth First Search Algorithm

Breadth first search is an algorithm for finding shortest path to reach the target or solve puzzle. BFS works best when there is a concept of layers or levels of neighborhoods in the graph we are dealing with. The BFS algorithm starts from a root vertex and explores the vertices in the neighborhood vertices. It then moves to the next neighborhood level and repeats the process [3]. For breadth-first search (BFS), the set Open is realized as a first-in first-out queue (FIFO). The Insert operation is called Enqueue and adds an element to the end of the list; the Dequeue operation selects and removes its first element. As a result, the neighbors of the source node are generated layer by layer (one edge apart, two edges apart, and so on) then, BFS stops as soon as it generates the goal [2].

1) Initialization

We will use two data structures:

- visited: This contains all the vertices that have been visited. Initially, it will be empty [3].
- queue: This contains all the vertices that we have want to visit in next iterations [3].

2) The main loop

Next, we will implement the main loop. It will keep on looping until there is not even a single element in the queue. For each node in the queue, if it has already been visited, then it visits its neighbor [3].

B. Task 1

At the beginning, we only need to move the robot to find target without any obstacles. The robot only requires avoiding from hitting the walls (#) and reach the target which located on land. In this part, we only have one map that need to be tested. The map is shown in figure 9.

```
char world1[200] = (
'#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '\n',
'#', 'T', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', '\n',
'#', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', '\n',
'#', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', '\n',
'#', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', '\n',
'#', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', '\n',
'#', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', '\n',
'#', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', '\n',
'#', 'O', 'R', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', '\n',
'#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '\n'
);
```

Fig. 9. Map for Task 1

1) Pseudocode

- Find robot position.
- Find target position.
- Find row and column of robot in the map.
- Find row and column of target in the map.
- Find column difference between robot and target.
- Find row difference between robot and target.
- If column difference greater than 0, move south.
- If column difference less than 0, move north.
- If row difference greater than 0, move east.
- If row difference less than 0, move west.

2) Code

Based on pseudocode, we implement the code as shown in figure 2.

```
//find robot's location
for (Robot = 0; world[Robot] != 'R'; Robot++)
{
}

//find target's location
for (Target = 0; world[Target] != 'T'; Target++)
{
}

//find target column location
int Horizontal_T;
Horizontal_T = Target / 21;
//find robot column location
int Horizontal_R;
Horizontal_R = Robot / 21;
//column difference
int difference_H = Horizontal_T - Horizontal_R;
//find target row location
int Vertical_T = Target % 21;
//find robot row location
int Vertical_R = Robot % 21;
//Row difference
int difference_V = Vertical_T - Vertical_R;

if (difference_H < 0)
{
return 1;
}

else if (difference_H > 0)
{
return 3;
}

else if (difference_V < 0)
{
return 4;
}

else if (difference_V > 0)
{
return 2;
}
```

Fig. 10. Code for Task 1

C. Task 2

Now the task is quite challenging as the walls are located randomly, not just on the border. The maps are shown in figure 3. Furthermore, robot will be able to move both on water and on land by changing its mode either land mode or water mode. To reach the target without any crash or failure, the robot requires to find the shortest path to the target by avoiding the walls and switch the mode correctly.

[illegible]

Fig. 11. Maps for Task 2

1) Pseudocode

- Find target and robot position.
- Make array for “queue”.
- Put initial robot position in the first “queue”.
- Make array for “visited” and fill it with null.
- Mark current robot position as visited.
- Make “previous” array for previous node and fill it with null.
- Make loop to check neighbor of nodes until target is found.
- If neighbor nodes is not wall and obstacle (for task 3), then put node in “queue” array.
- Mark the node as visited from avoiding checking the same node.
- Keep track of node in array “previous”.

- Call “previous” array to construct reverse path (from target to robot).
- Use reverse path to construct correct path (from robot to target).
- If there is water in path, change the robot mode from land to water mode.
- Move robot to target according to the correct path.

2) Code

Based on pseudocode, we implement the code as shown in figure 2.

```
//Put initial robot position in the first "queue
queue[0] = current_position;

//Make array for "visited "and fill it with null.
for (int v = 0; v < 200; v++)
{
    visited[v] = 0;
}

visited[current_position] = 1;
//Make "previous" array for previous node and fill it with null
for (int prv = 0; prv < 200; prv++)
{
    previous[prv] = 0;
}
```

Fig. 12. Construct visited array and previous node array.

```
// loop to check neighbour and put inside array queue
int* pointer;
pointer = &queue[0];
for (int y = 0; queue[y] != position_target;)
{
    current_position = *pointer; // pointer to check array neighbour
    //loop to check neighbour for every direction
    for (int d = 0; d < 4; d++) // d for direction
    {
        neighbour = current_position + direction[d];
        if (neighbour < 0 || neighbour > 200)
        {continue;
        }
        if (world[neighbour] == 'T')
        { queue[neighbour];
          d = 5;
        }
        if (world[neighbour] == '#' || world[neighbour] == '*')
        { continue;
        }
        if (visited[neighbour] == 1)
        { continue;
        }
        else
        { queue[y + 1] = neighbour;
          visited[neighbour] = 1;
          previous[neighbour] = current_position;
          printf("Queue = %d ", queue[y + 1]);
          y++;
        }
    }
    pointer++;
}
```

Fig. 13. Fill up queue array and check neighbor.


```

/*to find reverse path from
: target to robot*/
int find = position_target;
reversePath[0] = position_target;
for (int z = 1; find != Robot; z++)
{
    reversePath[z] = previous[find];
    find = reversePath[z];
    m = z;
}
int new = m;
/*find correct path from robot
: to target*/
int try = current_position;
path[m] = current_position;
for (int k = 0; try != Target; k++)
{
    path[k] = reversePath[m];
    try = reversePath[m];
    m--;
}
path[new] = position_target;
for (int k = 0; k < new + 1; k++)
{
    if (path[k] == position_target)
    {status = 1;
    }
}
}

```

Fig. 14. Array for reverse path and correct path

```
/*Movement when no Water
[west] is just example to move west
change west to direction north,south and east
to move every direction in map*/
if ((path[forward + 1] - path[forward]) == -1 && world[west] != '~')
{
    forward++;
    return 4;
}
```

Fig. 15. Movement of robot on land in west direction as example.

```
/* example : change into toggling mode if water(~) at west
change[west] to every direction[south, north, east]*/
if ((path[forward + 1] - path[forward]) == -1 && world[west] == '~')
{
    if (mode_a == 0 && world[west] == '~')
    {
        mode_a = 1;
        return 5; //change mode
    }
}
```

Fig. 16. Change mode from water mode to land mode (west as example)

```
/* example : movement in water(~) at west
:change[west] to every direction[south, north, east]*/
if (mode_a == 1 && world[west] == '~')
{
    forward++;
    ...
    return 4;
}
```

Fig. 17. Movement of robot in water in direction west as example.

```
/* example : change into land mode if space (0) at west
change[west] to every direction[south, north, east]*/
if (((path[forward+1]-path[forward])==-1) && (world[west]=='0'&&(mode_a == 1))
    mode_a = 0;
    return 5;
```

Fig. 18. Change back water mode to land mode (west as example)

D. Task 3

After finding the shortest path in task 2, our robot can manage to return to its original position where it is located at first for task 3. Instead of only to keep away from hitting the walls, robot has included with new features which is avoiding the obstacles (*) and use less energy. In this task also, we have extra two maps as shown in figure...

```
char world7[200] = {  
    '#','#','#','#','#','#','#','#','#','#','#','#','#','#','#','#','\n'  
    '#','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','\n'  
    '#','0','0','R','0','0','0','0','0','0','0','0','0','0','0','0','0','\n'  
    '#',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','\n'  
    '#',' ','T',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','\n'  
    '#',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '\n'  
    '#','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','\n'  
    '#','#','#','#','#','#','#','#','#','#','#','#','#','#','#','\n'  
};  
  
char world8[200] = {  
    '#','#','#','#','#','#','#','#','#','#','#','#','#','#','#','\n'  
    '#','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','\n'  
    '#','0','0','R',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','\n'  
    '#','0',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '\n'  
    '#','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','\n'  
    '#','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','\n'  
    '#','#','#','#','#','#','#','#','#','#','#','#','#','#','#','\n'  
};
```

Fig. 19. Maps for Task 3

1) *Pseudocode*

- For the first step, follow pseudocode task 2.
- Move the robot back to its initial position using reverse path.
- If in reverse path has water or land, then change robot mode from land mode to water mode and vice versa.

2) Code

```
//reverse movement when no Water
/* example : reverse movement on land at west
            change[Rwest] to every direction[Rsouth, Rnorth, Reast]*/
if ((reversePath[backward + 1] - reversePath[backward]) == -1 && world[Rwest] != '~')
{
    backward++;
    return 4;
}
```

Fig. 20. Reverse movement of robot on land in west direction as example

```

/* example : change into toggling mode if water(~) at west
change[4west] to every direction[Rsouth, Rnorth, Reast]*/

else if ((reversePath[backward + 1] - reversePath[backward]) == -1 && world[Rwest] == '~')
{
    if (mode_b == 0 && world[Rwest] == '~')
    {
        mode_b = 1;
        return 5;//change mode
    }
}

```

Fig. 21. Change mode from water mode to land mode (west as example) in reverse path.

```
/* example : reverse movement in water(~) at west
change[Rwest] to every direction[Rsouth, Rnorth, Reast]*/
if (mode_b == 1 && world[Rwest] == '~')
{
    backward++;
    return 4;
}
```

Fig. 22. Reverse movement of robot in water in direction west as example.

```

/* example : change into land mode if space (0) at west
: change [west] to every direction [rsouth, rnorth, Reast]*/
else if (((reversePath[backward + 1] - reversePath[backward]) == -1) && (world[Rwest] == '0') && (mode_b == 1))
{
    mode_b = 0;
    return 5;
}

```

Fig. 23. Change back water mode to land mode (west as example) in reverse path.

is necessary to have a clear goal, while keeping the objectives that we want to achieve.

2) Control Design

In the second step it is necessary to know how the control systems will work. For this reason, it is necessary. We define control variables, select the best sensors to measure each control variable, analyze the variables to define the features of the controller, as well as how many inputs and outputs, (digital or analogue), the controller needs.

3) Electronic Systems Design

This step aims to define the circuits for the robot operation, which calls for design of the following elements: signal conditioners, motors power circuits, control circuits, if the controller will be analogue, or circuits like microcontroller or microprocessor, if the circuits will be digital.

4) Electrical Design

This stage defines whether the robot will be battery-powered, or wire-powered. With all the steps done it is possible to proceed with the final step of the design process. This final step consists of the algorithms of control.

5) Algorithms of Control

In this step, the algorithms that are in charge of the behavior of the robot are designed. The algorithms must be able to make the synchronization of all the signals received from the sensors and the motion of the motors. Finally, it is very important to know that these are the general steps for the design process of a climbing robot, and if they are followed, good results will ensue.

B. Material Used

The material used to build a robot might come as an afterthought to some robotics developers, but of course the choice of materials will affect its safety, durability, and even aesthetics. Any design project should include considerations of how a robot will move, whether it will operate around people, what tasks it will perform, and the anticipated environment. Since our robot will most likely operate around people in rescuing patients no matter whether it is on land or water, so we came out with some ideas on what materials should we use and consider, to build our robot. The materials chosen for each subsequent part are as listed below:

1. The chassis is made of light steel. The reason behind this is to increase its durability so that it can withstand a compromised weight and pressure both when operating and moving on land and water. The weight should also be a bit light to ease the robot to float on water but should not be too light so that it can move smoothly on land.
2. The robot's interior should be water and salt-resistant so that it can be driven both in the sea or ocean.
3. The compartments inside the robot should be fitted well on onto another to avoid any leaking or any water to fill into the robot and later making it heavy enough to be submerged in the water whenever the robot enter the water.

4. As for the safety precaution, the high voltage electrics and the electronic parts in the robot should be well shielded, so that there is no risk of electrocution. We should give serious attention on this part as it is a really important and essential element on our robot. Therefore, this matter should be taken into consideration as the Murphy's law stated that "Anything that can go wrong will go wrong".

C. Early Sketch

In this fragment of project, which is in the design part, we decided to go for the Paper Prototyping approach in order to realize our real prototype for our ResQ Robot. Paper Prototyping is a throwaway prototyping that involves creating a rough, hand sketch, drawings of an interface to use as prototypes, models or designs. It is also an original type, form or an instance that serves as a model on which later stages are based on and judged. So, we came out with a rough hand sketch and measurement as an early sketch for our robot as shown in the figure below.

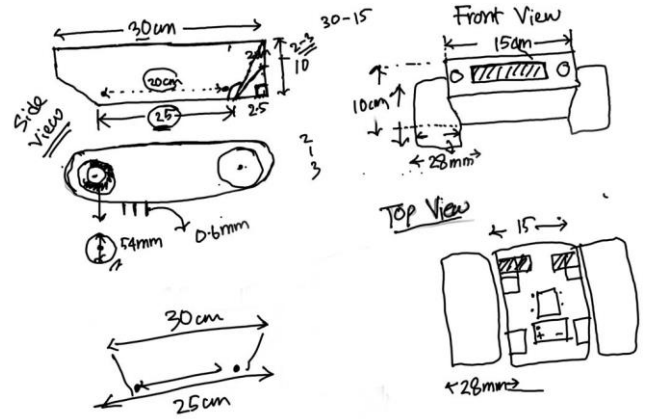


Fig. 29. Early sketch of ResQ Robot

In this section, we also include the measurements of our robot in real-life size and also how big it will be in the prototype. We were given a task to create the prototype of our robot with the dimension not more than 30cm x 20cm x 20cm.

TABLE I. MEASUREMENT OF THE EACH ROBOT PARTS IN REAL LIFE

	Tyre	Shaft	Chassis	Wheel
Length/cm	45.0	181.0	300.0	30.0
Width/cm	-	-	150.0	-
Height/cm	-	-	100.0	-
Radius/cm	25	5.9	-	80

TABLE II. MEASUREMENT OF THE EACH ROBOT PARTS IN PROTOTYPE

	Tyre	Shaft	Chassis	Wheel
Length/cm	4.5	18.1	30	3
Width/cm	-	-	15	-
Height/cm	-	-	10	-
Radius/cm	2.5	0.59	-	8

So, in the prototyping part, we have decided to create our robot's prototype with the scale of 1:10 to meet one of the requirements in our task.

D. 3D Modelling

3D modeling has changed the way we design; for the better. Not only does 3D modeling help the designers and end users visualize space requirements, but also improves drawing efficiency and accuracy. 3D modeling for design allows the designer to see what they would not see when designing in 2D. It gives the designer the ability to physically see how much real estate an object takes from all perspectives. When designing in 2D, the designer needs to create a separate plan and elevation view to see the space requirements of an object, which takes longer to do.

When designing in 3D, the design is done in one model. Whereas when a design is done in 2D, it is typically done in multiple models, one for each view. By doing a design in multiple models it creates an atmosphere where more mistakes can occur by having information duplicated. When a design is done in 3D, it assists designers with coordination. The designer can walk through a 3D model with specialized software and see the actual size and space of the design. It also allows the designer to see if their designs conflict with other disciplines or existing conditions they may not readily see in 2D. The 3D walkthrough software also allows the designer to run interference checks to see if the design clashes with other items in the 3D model. By using the 3D walkthrough software, the designer can easily see whether the design allows for equipment maintenance access and operational access, and addresses safety concerns. This allows the designer to create a more user-friendly design for the end user.

By designing in 3D, the designer can also review a design using the 3D walkthrough software with the end user. This is particularly helpful for end users who have a hard time to visualize designs from 2D drawings. This allows them to see how much clearance and access they will have around a design before it is physically built.

The advantages of 3D modeling for designers is not limited to productivity and coordination, it is an excellent communication tool for both the designer and end user. 3D models can help spark important conversations during the design phase and potentially avoid costly construction mishaps.

In this project, we decided to use the SOLIDWORKS software to realize all of the parts separately first and then combined it altogether later on to have the whole view of the robot's prototype. Subsequently, each of the design of our robot's parts will be explained in more detail to give the reader a better understanding of our projects and how it will work. We will also explain the purpose of each feature included in our design in each part.

Tyre & Shaft

Figures below show the tyre that we will use in our robot to move on the land. We decided to connect both front tyres directly to the motor to control and maneuver the direction employed to the tyres in order to steer it according to the desired direction. The tyres are also connected to caps to avoid water from entering the body parts when moving in water. Other than that, we also include the features of the edges on the tyres to increase friction when it moves on the land. For example, increasing the grip when the robot is moving on the sand or muddy areas. The edges also will help to overcome some obstacles on the ground such as pebbles and stones while moving to the target or destination.

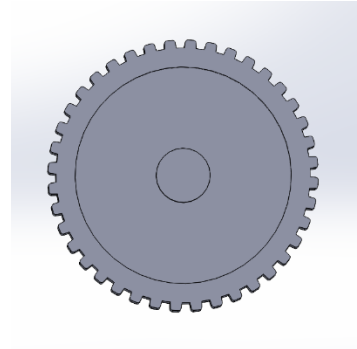


Fig. 30.1 Front view of tyre

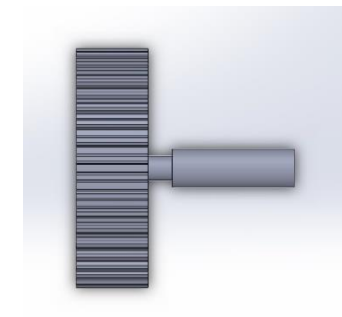


Fig. 30.2 Top view of tyre

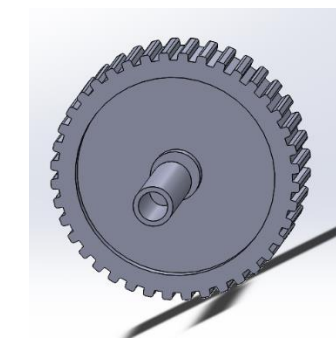


Fig. 30.3 Isometric (I) view of tyre

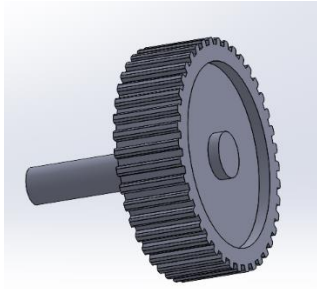


Fig. 30.4 Isometric (II) view of tyre

Meanwhile for the rear tyres, we designed as if it is not connected to any motor at all. Instead, we connected both tyres on the rear part with a shaft. Basically, they will just follow the movements of front tyres and just rotating by the help of the shaft connecting both tyres left and right.

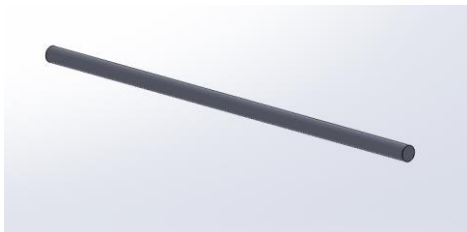


Fig. 30.5 Isometric view of shaft

Water Wheel

Next, we come to the most important components that will play a big role when the robot is moving on water. We designed two water wheels that will be placed on the center of the robot, which is between the front and rear tyres. The wheels are placed a bit higher compared to the tyres because we designed it only to move whenever our robot is floating on the water. The reason behind this is to reduce the amount of energy used when the robot is moving on land since both water wheels are connected directly to the motor as well.

Aside from that, we also decided to design the water wheels with some blades to mimic a propeller. As we know, propeller is a device with a rotating hub and radiating blades that are set at a pitch to form a helical spiral, that, when rotated, exerts linear thrust upon a working fluid, such as water or air. Propellers are used to pump fluid through a pipe or duct, or to create thrust to propel a boat through water. The blades are specially shaped so that their rotational motion through the fluid causes a pressure difference between the two surfaces of the blade by Bernoulli's principle by exerts force on the fluid, which then will allow to move the robot forward or backward. Not to forget, since both wheels are connected directly to the motor, the robot will also be able to move to the left or right.

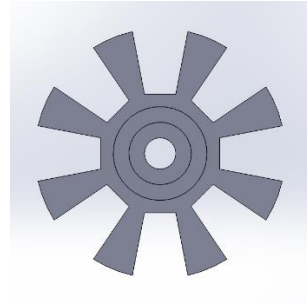


Fig. 31.1 Front view of wheel

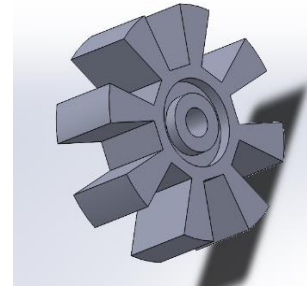


Fig. 31.2 Isometric view of wheel

Chassis

Here we come to the strongest part of our robot which is chassis. The chassis will act like a body, it needs to be strongly built and well design to hold other parts or components at the robot. As we mentioned earlier, the material used for chassis is light steel which to increase its durability so that it can withstand a compromised weight and pressure both when operating and moving on land and water. For the design of our rescue robot, we inspired from the design of a tank as shown in the figure below.

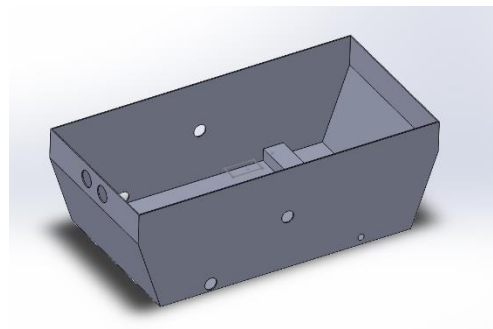


Fig. 32.1 Isometric view of chassis

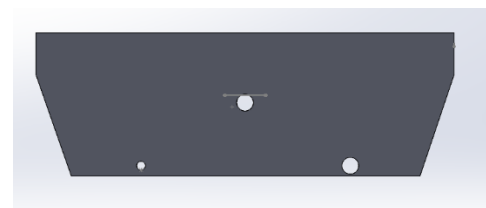


Fig. 32.2 Side view of chassis

In order to make our robot able to move or float on water, we also consider buoyancy factor. We have increased the volume of the robot to increase the buoyancy. The model of our robot consists of aerodynamics to ensure our robot can move through air or surroundings easier and faster.

Electronics

These are the electronics components that been used in our robot. Firstly, we are going to attach 4 motors with front tyres and wheels. They are important to certify all of the tyre and wheels keep rotating synchronous and simultaneously.

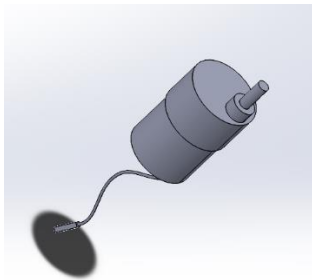


Fig. 33.1 Motor

Second is Arduino. Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards can read inputs such as light on a sensor or a finger on a button and then turn it into an output such as activating a motor or turning on an LED. Here, on this single board, all the electronics components are connected.

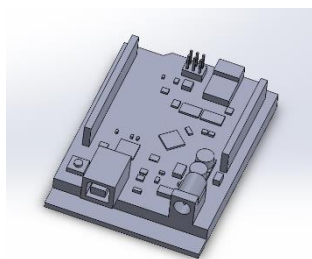


Fig. 33.2 Arduino UNO

Lastly is ultrasonic sensors which work by emitting sound waves at a frequency too high for humans to hear. They then wait for the sound to be reflected back, calculating distance based on the time required. This is similar to how radar measures the time it takes a radio wave to return after hitting an object. The purpose we use ultrasonic sensor is to calculate better distance of obstacles with our robot so it can avoid or overcome it.

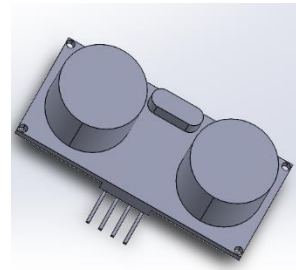


Fig. 33.3 Ultrasonic Sensor

Final Prototype

After combining all the parts stated before, we finally come to the final prototype of our robot which portrays the complete side of the robot. We also provided some figures as an aid for a better understanding on our robot.

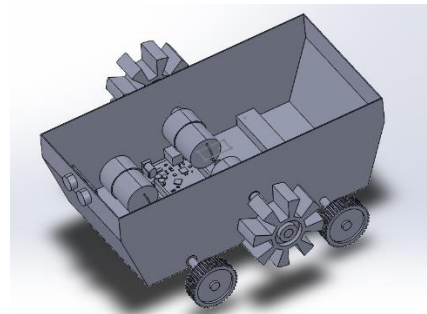


Fig. 34.1 Isometric view of the robot

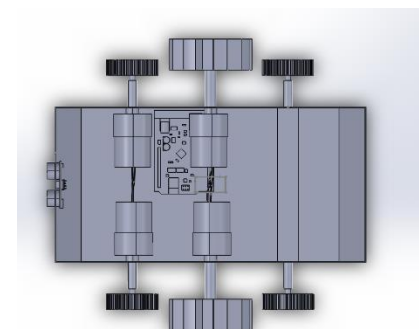


Fig. 34.2 Top view of the robot

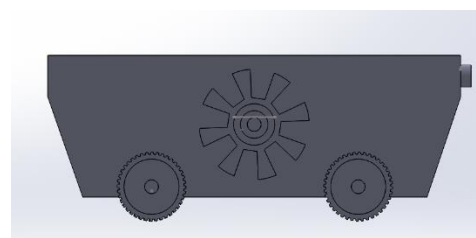


Fig. 34.3 Side view of the robot

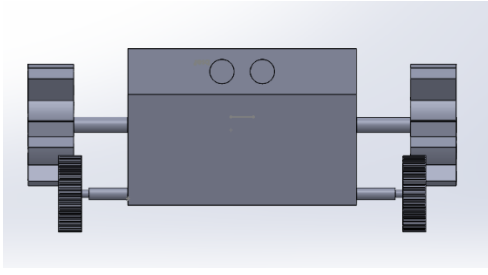


Fig. 34.4 Front view of the robot

E. Printing

As what has been suggested by our professor, we have used a software called Ultimaker Cura to calculate the total estimation time to 3D print our robot. After inserting each parts of our combined prototype in the software space, we have calculated the total time of 3D printing of ResQ Robot and the result shows that the total time taken to print each single part of robot is 33 hours and 26 minutes as shown in the figures below.

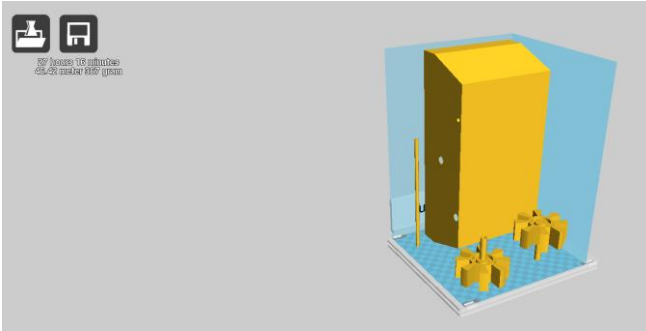


Fig. 35.1 3D printing (I)

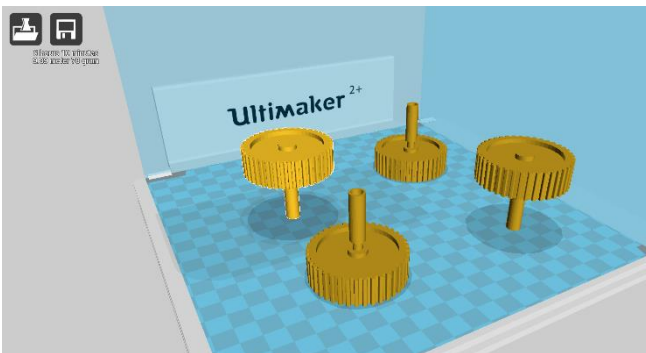


Fig. 35.2 3D printing (II)

CONCLUSION

A rescue robot is designed in this paper, which can help rescue workers in complex environments and complete search-and-rescue missions. To address the requirements of disaster relief, the robot was designed as four-tyres-with-two-water-wheel structure with rescue function modules. The tyres structure has the advantages of high carrying capacity, easy control, and simple structure to adapt to a complex disaster scene. The water wheel function modules can help rescue workers to complete the rescue work by

moving both on land and water. To determine the most comprehensive and best performance of the robot, an optimization objective function was proposed according to the motion characteristics of robot, which include mechanical property and movement stability. There may be cases in which the current level of our robot technology cannot ensure adequate functions, depending on conditions in disaster-stricken areas. Nevertheless, throughout this project, we hope that more rescue robots can be expected to achieve progress through their actual use. Henceforth, great expectations are being placed on international cooperation in the development and operation of rescue robots, such as for the creation of a “Rescue Team without Frontiers.”[4]

V. CONTRIBUTION

A. Documentation

Member	Contribution	
	Topics	Pages
Amit Chakma	System Modelling	1-3
Abdul Azeez	System Modelling	1-3
Iqbal	Coding	4-7
Farid	Coding	4-7
Amjad	Design	7-12
Amirul	Design	7-12

B. Project Development

1) System Modelling

Member	Contribution Percentage (%)
Amit	System Architecture (100%)
Azeez	Context Diagram (100%)
Iqbal	Activity Diagram (100%)
Farid	Use Case Diagram (100%)
Amjad	Sequence Diagram (100%)
Amirul	Block Diagram (100%)

2) Design

Member	Contribution Percentage (%)
Amit	16%
Azeez	16%
Iqbal	16%
Farid	16%
Amjad	16%
Amirul	16%

3) Coding

Member	Contribution Percentage (%)
Amit	10%
Azeez	10%
Iqbal	20%
Farid	20%
Amjad	20%
Amirul	20%

REFERENCES

- [1] Dominey-Howes D. (2007) Geological and historical records of tsunami in Australia. *Marine Geology* 239: 99-123 doi:10.1016/j.margeo.2007.01.010
- [2] S. Schrödl and S. Edelkamp, *Heuristic Search: Theory and Applications*. Morgan Kaufmann Publishers, 2011
- [3] I. Ahmad, *40 Algorithms Every Programmer Should Know*. [S.l.]: Packt Publishing, 2020.
- [4] E. Magid, T. Tsubouchi, E. Koyanagi and T. Yoshida, "Building a Search Tree for a Pilot System of a Rescue Search Robot in a Discretized Random Step Environment", *Journal of Robotics and Mechatronics*, vol. 23, no. 4, pp. 567-581, 2011. Available: 10.20965/jrm.2011.p0567.

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.

bin Zaprunnizam,

Muhammad Amirul Hakimi

Soest, 15.07.2021



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.

bin Abdul Malik,
Muhammad Amjad

Soest, 15.07.2021



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.

bin Mohamad Shabri,
Muhammad Farid Izwan

Soest, 15.07.2021



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.

bin Mohd Fauzi,
Muhammad Iqbal

Soest ,15.07.2021



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.

Olanlokun, Abdul-Azeez	Hamm, 15.07.2021	
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.

Chakma, Amit

Lippstadt 15.07.2021



Name

Ort, Datum

Unterschrift

Last Name

Location, Date

Signature