

**T.C.
BAHÇEŞEHİR UNIVERSITY**



FACULTY OF ENGINEERING AND NATURAL SCIENCES

CAPSTONE FINAL REPORT

EARTHQUAKE EMERGENCY RESPONSE ROBOT

Ammar Arıfın

Coşkun Alanoğlu

Zeynep Selen Bilgiç

Kerem Bülbül

Canset Çurey

Ahmet Akif Haboğlu

Ayhan Eray Küçük

Emre Mercan

Ali Fawzi Murad

Mustafa Başar Yılmaz

Assist. Prof. Ayşe Kavuşturucu

Assist. Prof. Gökhan Gelişen

Assist. Prof. Fatih Kahraman

Assist. Prof. Ece Gelal Soyak

ISTANBUL, May 2024

STUDENT DECLARATION

By submitting this report, as partial fulfillment of the requirements of the Capstone course, the students promise on penalty of failure of the course that

- they have given credit to and declared (by citation), any work that is not their own (e.g. parts of the report that is copied/pasted from the Internet, design or construction performed by another person, etc.);
- they have not received unpermitted aid for the project design, construction, report or presentation;
- they have not falsely assigned credit for work to another student in the group, and not take credit for work done by another student in the group.

ABSTRACT

EARTHQUAKE EMERGENCY RESPONSE ROBOT

Ammar Arifin

Coşkun Alanoğlu

Zeynep Selen Bilgiç

Kerem Bülbül

Canset Çurey

Ahmet Akif Haboğlu

Ayhan Eray Küçük

Emre Mercan

Ali Fawzi Murad

Mustafa Başar Yılmaz

Faculty of Engineering and Natural Sciences

Assist. Prof. Ayşe Kavuşturucu

Assist. Prof. Gökhan Gelişen

Assist. Prof. Fatih Kahraman

Assist. Prof. Ece Gelal Soyak

May 2024

Quick and well-coordinated reaction times following an earthquake are critical for preserving human life. The goal of this project is to create an emergency response robot that can navigate through debris fields, find damage, and recognize life indicators in order to improve these efforts. With its sophisticated sensors and great mobility, the robot can visit places that are inaccessible to human teams. It can also deliver data in real time to assist the Disaster and Emergency Management Authority (AFAD) in allocating rescue efforts according to priority.

The robot fills a requirement in earthquake-affected areas: effective damage and life sign detection, particularly in areas where transportation networks are disrupted. It helps lead rescue personnel to crucial regions by providing instantaneous analysis and information sharing, increasing the likelihood of finding survivors fast and lowering hazards.

Mobility and sensors for life and damage detection are essential functional needs, whereas weight, battery life, speed, camera capabilities, and cargo capacity are performance criteria. Prototyping expenses are kept down by sponsorships and economical production techniques like 3D printing and modular design. Local supply networks and economies of scale are advantageous for mass production.

Key Words: Emergency response robot, Earthquake, Disaster response, Damage detection
Rescue operations

TABLE OF CONTENTSABSTRACT iii

TABLE OF CONTENTS	v
LIST OF TABLES	vi
LIST OF FIGURES	vi
1. OVERVIEW.....	7
1.1. Identification of the need.....	7
1.2. Definition of the problem.....	7
1.3. Conceptual solutions	9
1.4. Physical architecture	1
2. WORK PLAN	5
2.1. Work Breakdown Structure (WBS)	5
2.2. Responsibility Matrix (RM)	6
2.3. Project Network (PN).....	7
2.4. Gantt chart	8
2.5. Costs	10
2.6. Risk assessment.....	11
3. SUB-SYSTEMS.....	11
3.2. The name of the sub-system 2.....	25
4. INTEGRATION AND EVALUATION	31
4.1. Integration	31
4.2. Evaluation.....	33
5. SUMMARY AND CONCLUSION.....	35
ACKNOWLEDGEMENTS	37
REFERENCES	38
APPENDIX A	39

LIST OF TABLES

Table 2. Responsibility Matrix for the team.....	6
Table 3. Gantt chart for the materialisation phase of the project.	8
Table 4. Costs	9.
Table 5. Risk matrix	11
Table 6. Risk assessment.....	11

LIST OF FIGURES

Figure 4. Work breakdown structure for the project.	5
Figure 5. The project network.	7

1. OVERVIEW

1.1. Identification of the need

Earthquake emergency response robots are the sensors and mobility in the area where the earthquake occurs, and the subsequent damage detection and rapid determination of whether there is life or not.

Afterwards, transportation network earthquakes are often blocked. However, since these robots have the ability to move, they can play an active role in warehouses that teams cannot reach.

Since these robots can do damage assessment and life details, after the earthquake, line teams can be directed to the right points and according to visibility. This allows patients with a chance of survival to reach people as soon as possible, as well as preventing the entry of dangerous structures into damaged or slightly damaged structures.

1.2. Definition of the problem

Earthquakes can damage structures, leave people stranded and create the need for emergency aid in large areas. One of the main problems that may occur after an earthquake is the problem of communication infrastructure and transportation network. The fastest and safest way to detect a living being under the rubble after an earthquake is to instantly transfer this information to the necessary places. The problem is generally to make the right analyzes instantly and save time and eliminate the risks that aid teams will experience.

1.2.1. Functional requirements

The robot's mobility and the sensors on its body enable it to detect damage and signs of life and create an intervention sequence.

1.2.2. Performance requirements

The robot's weight, battery, maximum speed, camera and maximum payload capacity are sufficient to perform its tasks in a practical way in an emergency situation.

1.2.3. Constraints

Our designed robot, being an emergency response robot, holds significant social impact and is a critical tool. This naturally brings many challenges. For instance, designing a robot that can operate effectively in a debris area requires materials that may be economically demanding for the manufacturer. Additionally, ensuring zero defects and keeping costs low in mass production presents another challenge. However, despite these difficulties, this environmentally friendly emergency robot has the potential to significantly increase the chances of saving lives of people in need within the debris. We believe this project will be highly successful.

Currently, search and rescue operations are primarily conducted by humans and animals. Our robot, which reduces reliance on human labor, will gather necessary data from specific distances and relay it to authorized personnel. In doing so, it will help reduce chaos during disaster processes and open a new field. For our robot to operate effectively in a debris area, it must possess certain features. Mobility and maneuverability, durability and reliability, and data collection and processing speed are all factors that directly impact the robot's performance.

Furthermore, our robot must comply with the following ISO standards to ensure it operates legally and meets all necessary requirements. These standards ensure the process progresses in the most accurate and reliable manner:

1. ISO 9001 - Quality Management System: This document ensures that the design, production, and maintenance aspects of the robot are managed according to high-quality standards.
2. ISO 12100 - Machinery Safety: This standard includes general rules regarding the identification of robot hazards, risk assessment, and risk reduction measures.
3. ISO 26000 - Social Responsibility: This document indicates that the robot manufacturer complies with social responsibility principles and adopts appropriate business practices for societal cohesion.
4. ISO 18646-1 - Robots and Robotic Devices: Performance and Design Characteristics - Part 1: Industrial Robots: This standard covers the requirements for the performance and design characteristics of robots.

During the design phase, we minimized prototype costs by using open-source software and hardware, 3D printing technology, modular design, and existing components. Additionally, production partnerships and sponsorships further reduced costs. In mass production, costs will be further lowered thanks to economies of scale, automated production lines, design optimization, local supply chains, and standard components. These strategies make it feasible to produce an emergency

robot that operates efficiently and cost-effectively in debris areas.

In conclusion, our designed robot enables rapid and safe response in disaster areas, is produced with a sense of social responsibility, and is a high-quality, reliable product. By adhering to all legal requirements and standards during its development and production, we aim for our robot to fulfill its mission of saving lives in the most effective way possible.

1.3. Conceptual solutions

Solution : "Energy efficient robot capable of working long hours"
Description:

Includes an emergency response robot that can operate effectively in day and night conditions.

It offers a modular design that is equipped with sensors and can maneuver in restricted spaces.

Main Features:

- Ability to analyze its surroundings with smart sensor technology.
- Ability to reach places with different difficulties thanks to modular design.

1.3.1. Literature Review

Summary of Literature on the A* Algorithm

A-Star's basic idea is to frequently check the places that are least explored. If the location is the final destination, then the procedure is finished. In the event that it is not the intended destination, A-Star will note the location nearby and investigate alternative options. The initial point, nodes, open list, closed list, price (cost), and challenges are the fundamental terms used in the A-Star algorithm. The benefit of using heuristics as optimization is that A-Star can guide itself to the wanted solution by assigning a value to each node.

A-Star is Dijkstra's development, an algorithm that aims to process efficient path planning among several points (nodes) by using heuristics function.

Dijkstra's shortest path algorithm serves as the foundation for the A* search algorithm. Finding the shortest path between two nodes in a graph while taking barriers and other obstructions into account is its aim.

Evaluation of the A* Algorithm

The A* algorithm One of the most well-known path planning algorithms, the A* algorithm can be used in either topological or metric configuration space. The combination of heuristic and shortest path searching is used by this algorithm. Since the value of each cell in the space of configuration is determined by the A* algorithm, it is also known as the best-first algorithm:

$$F(v)=h(v)+g(v)$$

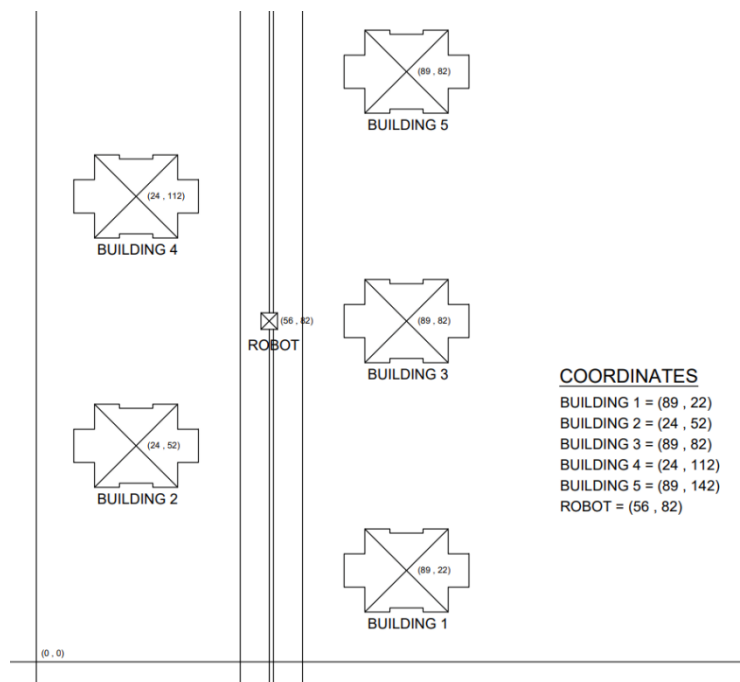
Where $g(v)$ is the measurement of the length of the route from the initial state to the final state via the chosen series of cells, and $h(v)$ is the heuristic distance (Manhattan, Euclidean, or Chebyshev) of the cell to the final state. This sequence clearly ends with the cell that is evaluated. Every neighboring cell of the cell that was truly reached is assessed using the value $f(v)$. The cell that comes after in the sequence is the one with the lowest value of $f(v)$. This algorithm's strength is that it allows for the adoption, modification, or addition of new distances in addition to the ones used as criteria. This provides a plethora of variations on this fundamental idea. For instance, function $f(v)$ can additionally incorporate time, consumption of energy, or safety. An algorithm called A* has been effectively constructed and tested to plan a mobile robot's course. These outcomes, meanwhile, were not wholly positive and gratifying. There was a significant computational delay. On an average machine, the computation for the map with 60,000 cells took more than an hour. This is really depressing for the robot's implementation of this algorithm. As a result, more research into the potential for changing the A* algorithm was done.

Implementing A* Algorithm

Using the travel salesman problem, we ran our shortest path programming through the second scenario in the first section of the project. To run the A* algorithm, we will examine the previously determined shortest path from a different angle in the sections that follow. By concentrating on the third scenario that civil engineering department offers, we hope to operate the algorithm efficiently.

Scenario 3

We made an Excel table with the distances using Scenario 3. For each of the five buildings listed as nodes in this table, we determined a new destination point to find the shortest path. The A* algorithm helped us accomplish this. For each iteration, we designated one building as the destination point. The robot's positional center is indicated by the point marked as R in the table.



	1	2	3	4	5
R	68.47	43.86	33.00	43.86	68.47
1	0.00	71.58	60.00	112.33	120.00
2	71.58	0.00	109.83	60.00	112.33
3	60.00	109.83	0.00	71.58	60.00
4	112.33	60.00	71.58	0.00	71.58
5	120.00	112.33	60.00	71.58	0.00

Below tables indicate how the algorithm works. As previously stated, final destination is a necessity for this algorithm to come up with an outcome, thus every building is identified as final destinations separately and these scenarios were conducted step by step. After final paths were defined (please see at the right-hand side of the tables), total distances of these paths were compared with each other, consequently the path with the minimum distance travelled, becomes the optimum path according to the A* algorithm.

Select Final Destination 1

R --> 2	115.44	R --> 3	3 --> 2	71.58	R --> 3 --> 2	2 --> 4	172.33	R --> 3 --> 2 --> 4 --> 5 --> 1	Total
R --> 3	93.00		3 --> 4	183.91		2 --> 5	232.33		528.49
R --> 4	156.19		3 --> 5	180.00					
R --> 5	188.47								

Select Final Destination 2

R --> 3	142.83	R --> 4	4 --> 3	181.41	R --> 4 --> 3	3 --> 5	172.33	R --> 4 --> 3 --> 1 --> 5 --> 2	Total
R --> 4	103.86		4 --> 5	183.91		3 --> 1	131.58		649.18
R --> 5	180.80		4 --> 1	183.91					
R --> 1	140.05								

Select Final Destination 3

R --> 1	128.47	R --> 4	4 --> 1	172.33	R --> 4 --> 5	5 --> 1	180.00	R --> 4 --> 5 --> 1 --> 2 --> 3	Total
R --> 2	153.69		4 --> 2	169.83		5 --> 2	222.16		608.43
R --> 4	115.44		4 --> 5	131.58					
R --> 5	128.47								

Select Final Destination 4

R --> 1	180.80	R --> 2	2 --> 1	183.91	R --> 2 --> 3	3 --> 1	172.33	R --> 2 --> 3 --> 5 --> 1 --> 4	Total
R --> 2	103.86		2 --> 3	181.41		3 --> 5	131.58		649.18
R --> 3	104.58		2 --> 5	183.91					
R --> 5	140.05								

Select Final Destination 5

R --> 1	188.47	R --> 3	3 --> 1	180.00	R --> 3 --> 1	1 --> 2	183.91	R --> 3 --> 1 --> 2 --> 4 --> 5	Total
R --> 2	156.19		3 --> 2	222.16		1 --> 4	183.91	R --> 3 --> 1 --> 4 --> 2 --> 5	588.49
R --> 3	93.00		3 --> 4	143.16					629.24
R --> 4	115.44								

Integration of Second Parameter

In this section of the report, a second parameter will be discussed to strengthen the methodology and vision of this project. In this manner the following part will focus on how to integrate a second parameter to the current A* algorithm which is executed with heuristic point of view. The heuristic side of this project is essential since this robot is meant to make reasonable decisions in extremely unsteady circumstances while at the same time playing a role in saving lives in post-earthquake situations. Thus, this project must consider the possible options for how to design a real-time decision-making robot in those circumstances.

As artificial intelligence team suggest, there are two types of heuristic parameters that can be embedded to A* algorithm, which result with reasonable outcomes. One of which is sound, and the other one is heat parameter. In this part, sound parameter will be discussed with its contribution on real-time decision-making processes.

As it is mentioned above, A* algorithm has a basis of 2 parameters which are ***h(v)*** and ***g(v)***. The first part of the function stands for the heuristic analysis and the second part of the equation mostly represents a constant variable. When this project is considered, constant variable could be taken as the ***distance, since the distance between the buildings will be the exact same with each iteration of decision making.*** (Please see the distance matrix above.) Upon that, sound parameter will be the main character in the heuristic approach of this formulation given below.

$$F(v)=h(v) + g(v)$$

However, since these two parameters are completely different than each other, adding these up would not give anything useful mathematically. Hence, an adjustment has been made with this current A* function to tune these two parameters in a meaningful way, as revised version can be found below.

$$F(v)= (w).h(v) + (1-w).g(v)$$

To obtain a useful value, these two parameters are multiplied with predetermined weights (w) *out of 1*. This derived formula makes us obtain a comparable value, which will be used, afterwards, as decision-making data. Moreover, this formulation gives the user the flexibility to identify a (w) value and optimize the importance of each parameter in different circumstances. Yet, in this project it is assumed that the sound parameter has a greater importance compared to the distance traveled by the robot, since higher sound data received could be a strong indicator of possible survivor. Keeping the importance of sound parameter in mind, this revised formula also establishes a tradeoff strategy with the distance parameter, by avoiding lasting in any distant building.

It is being said sound has a greater importance, this project identifies $w = 0.4$ as a constant weight value and the updated formulation is represented as follows.

$$F(v) = (0.4).h(v) + (0.6).g(v)$$

The reason w value is identified as 0.4 is totally related with the logic of the A* algorithm. In each iteration, algorithm seeks to have minimum $F(v)$ values, in other saying, this is a minimization equation, in which higher distance and sound values result with an increase in the solution. So, decision wise, robots will be choosing the destination with the minimum $F(v)$ value. Here is why, as the sound parameter should have a greater effect on the outcome, it has smaller multiplier, 0.4, which contributes to the outcome more by lowering $F(v)$ value proportionately.

Why use sound detection?

Embedded sensor technology serves as the primary foundation for human detection systems research. Due to challenging and changing environmental circumstances, such as hot temperatures, poor visibility, loud noises, power outages, and other scenes that occur during catastrophes, human detection-based rescue and search operations requires a broad variety of inputs.

Since lives are at risk during search and rescue efforts, time is deemed to be crucial. To deal with such situations, human detecting systems must rely on multi-sensor technology, which can quickly and accurately search a large area with improved detection and fine precision.

To construct a live algorithm, voice data detection is crucial, as those who work in the AI department mentioned. The decision-making process and the robot's path will be more functional for the project if voice input is instantly collected and included in the heuristic methodology.

Furthermore, we discovered from our review of the literature that sound is the most practical and widely used information on earthquake debris for search and rescue operations. Currently available equipment that can be employed inside the wreckage includes the ,Delsar Underwrecked Live Search Detactor, which is one of the approaches used in this context. But since these devices needed a human operator to operate them, it was only ever a notion for our project.

In addition, several devices have been designed with the express purpose of utilizing the Doppler effect. The most widely used technologies now are robotic people detection systems, FMCW, SFCW, and Continuous Wave (CW) radar. Upon closer inspection of all these equipment and systems, we find that information like temperature and sound is crucial for locating individuals buried beneath the debris.

Sound Scoring Scale

To streamline the Heuristic methods for inserting the second parameter, we developed a sound scoring scale to more readily modify the sound data. The heuristic method reached a significant step in the formula with the help of a sound scoring scale.

Our research led us to the conclusion that the best way to incorporate voice decibels into the heuristic process is by developing a scale. For the A* approach, it was not mathematically possible to determine the optimal path using only raw decibel data. In this case, establishing scale enabled us to appropriately transform the data and apply the A* algorithm. We should consider the fact that the output of the A* algorithm should be minimized for the decision-making process in order establish the discussion of the issue at hand.

In this section of the project, the sound parameter was added to the distance in order to determine the robot's destination. This allowed the robot to generate a new destination possibility at each step as the values it received changed, and it also created a heuristic perspective that could renew itself with each step. For the purposes of the scenario, the A* algorithm formula must be minimized, as we previously discussed.

The issue we have here is that the formula is maximized by decibels, while it is minimized by the distance between the two values we utilize. This consequence is in opposition to the desired scenario of having a high decibel level, even though it is also desirable for the destination the robot creates to be at a minimum distance. We developed a scale that is inversely proportional to the db magnitude in order to formulate this scenario. If any adjustments are required, the scale we have developed serves as a guide for the highest and lowest decibels that our research can offer.

The reference points they accepted were 120 dB, the volume of a human scream, and 20 dB, the typical whispering volume. This scale differs in that the lowest possible score in terms of points is 120 dB. The maximum decibel, 120, earns 0 points on this scale, where values between 0 and 1 are assigned, while our minimum decibel, 20, obtains the greatest value of 10.

<i>Desibel (db.)</i>	20	30	40	50	60	70	80	90	100	110	120
<i>Sound Score (ss.)</i>	10	9	8	7	6	5	4	3	2	1	0

i.e 66 db= 5.4 ss

Our primary goal in doing this is to minimize the formula; so, the bigger the value we obtain when we multiply our decibel magnitude by its weight in the formula, the more we want to decrease the formula. However, by applying this equation, the more decibels we increase, the more points will fall, the more the multiplication result will reduce, and finally, the lower our $F(v)$ value will be. In the following iteration, the robot will select that path.

Received Sound Data -Assumed

Building No.	Received db.	Relative Sound Score
1	62	5.8
2	83	3.7
3	73	4.7
4	97	2.3
5	112	0.8

Values in the above table represent sound data that are received by the sound sensors of the robot at its initial location. These db. data will construct the basis of the sound score logic as it is explained previously. Relative sound scores of each db are converted and indicated in the second column of the table. One should notice that these data will also establish the heuristic side of this draft model, which at the same time means, these data are meant to be updated every time robot reaches its next destination. Hence the robot will be able to recalculate its optimum path and, if necessary, regenerate a new path according to the current sound data. At the end of the day this logic provides a better flexibility where the robot could possibly react to the fluctuations in the environment and act accordingly. The scenario of robot regenerating its pre calculated optimum path will be covered in the following parts of the report as the result of changing sound data in the destination points.

First Iteration

The heuristic algorithm and, if required the developed implementation version of our heuristic from the first iteration were intended to be shared with the assistance. The project's draft report demonstrated that the same distances between five buildings were used as a point of reference. A minimal result was reached by completing the required formula for each possible location by choosing a reference point and orienting it according to the scale of the audio data in this version of the setting.

As can be seen from the formula, we multiplied the value of our heuristic function by the weight value, which is determined **0.4**. We then multiplied the distance by the weight value, which we calculated to be **0.6**. We calculated the value that minimizes our $F(V)$ function by adding all the values that were obtained. The $F(v)$ value was selected to represent the nature of the A* algorithm in the following iteration (in this case, this value was adopted as **21.68**). The same formulation was repeated in the second iteration by following the same methodology and using the db and distance values from each building from the table you see above.

As we proceed in accordance with these calculations, it is evident that following the identification of the optimal path—which we determined to be building 3—the other three decision selections also produced heuristic methodology that represent the minimum sound score and the minimum distance between the designated building 3 and the subsequent destination.

$$F(v) = (0.4).h(v) + (1-0.4).g(v)$$

R --> 1	$F(v) = [(0.4)*(5.8) + (0.6)*(68.47)] =$	43.402
R --> 2	$F(v) = [(0.4)*(3.7) + (0.6)*(43.86)] =$	27.796
R --> 3	$F(v) = [(0.4)*(4.7) + (0.6)*(33.00)] =$	21.68
R --> 4	$F(v) = [(0.4)*(2.3) + (0.6)*(43.86)] =$	27.236
R --> 5	$F(v) = [(0.4)*(0.8) + (0.6)*(68.47)] =$	41.402

$$F(v) = [(0.4).h(v) + (1-0.4).g(v)] + F(3)$$

3 --> 1	$F(v) = [(0.4*5.8) + (0.6*60)] + 21.68 =$	60
3 --> 2	$F(v) = [(0.4*3.7) + (0.6*109.83)] + 21.68 =$	89.058
3 --> 4	$F(v) = [(0.4*2.3) + (0.6*71.58)] + 21.68 =$	65.548
3 --> 5	$F(v) = [(0.4*0.8) + (0.6*60)] + 21.68 =$	58

$$F(v) = [(0.4).h(v) + (1-0.4).g(v)] + F(3) + F(5)$$

5 --> 1	$F(v) = [(0.4*5.8) + (0.6*120)] + 21.68 + 58 =$	174.88
5 --> 2	$F(v) = [(0.4*3.7) + (0.6*112.33)] + 21.68 + 58 =$	148.558
5 --> 4	$F(v) = [(0.4*2.3) + (0.6*71.58)] + 21.68 + 58 =$	123.548

$$F(v) = [(0.4).h(v) + (1-0.4).g(v)] + F(3) + F(5) + F(4)$$

4 --> 1	$F(v) = [(0.4*5.8) + (0.6*112.33)] + 21.68 + 58 + 123.548 =$	272.946
4 --> 2	$F(v) = [(0.4*3.7) + (0.6*60)] + 21.68 + 58 + 123.548 =$	240.708

$$F(v) = [(0.4).h(v) + (1-0.4).g(v)] + F(3) + F(5) + F(4) + F(2)$$

2 --> 1	$F(v) = [(0.4*5.8) + (0.6*71.58)] + 21.68 + 58 + 123.548 + 240.708 =$	486.884
---------	---	----------------

2'nd Received Sound Data -Assumed

Building No.	Received db.	Relative Sound Score
1	98	2.2
2	107	1.3
3	73	4.7
4	72	5.2
5	93	2.7

Second sound data is also received to strengthen the heuristic side of the project. It is assumed that once the robot reaches building 3, as it is concluded in the first iteration above, with the new sound data, A* algorithm works again to see if there is a better optimum path to be followed. New sound data is only taken from the rest of the 4 buildings 1,2,4 and 5. According to the new sound data, A* algorithm and its iterations are shown below.

Second Iteration

$F(v) = [(0.4).h(v) + (1-0.4).g(v)] + F(3)$				$F(v) = [(0.4).h(v) + (1-0.4).g(v)] + F(3) + F(1)$			
3 --> 1	$F(v) = [(0.4*2.2) + (0.6*60)] + 21.68 =$		58.56	1 --> 2	$F(v) = [(0.4*1.3) + (0.6*71.58)] + 21.68 + 58.56 =$		123.708
3 --> 2	$F(v) = [(0.4*1.3) + (0.6*109.83)] + 21.6$		88.098	1 --> 4	$F(v) = [(0.4*5.2) + (0.6*112.33)] + 21.68 + 58.56$		149.718
3 --> 4	$F(v) = [(0.4*5.2) + (0.6*71.58)] + 21.68$		66.708	1 --> 5	$F(v) = [(0.4*2.7) + (0.6*120)] + 21.68 + 58.56 =$		153.32
3 --> 5	$F(v) = [(0.4*2.7) + (0.6*60)] + 21.68 =$		58.76				
$F(v) = [(0.4).h(v) + (1-0.4).g(v)] + F(3) + F(1) + F(2)$				$F(v) = [(0.4).h(v) + (1-0.4).g(v)] + F(3) + F(1) + F(2)$			
2 --> 4	$F(v) = [(0.4*5.2) + (0.6*60)] + 21.68 + 58.56 + 123.708 =$		242.028	4 --> 5	$F(v) = [(0.4*2.7) + (0.6*71.58)] + 21.68 + 58.56 + 123.708 + 242.028$		490.004
2 --> 5	$F(v) = [(0.4*2.7) + (0.6*112.33)] + 21.68 + 58.56 + 123.708 =$		272.426				

This iteration is comparatively shorter than the initial one, since there are only 4 possible destinations left to be visited by the robot. Weights and sound scores work in the same manner, and in each calculation, scores of previous destinations are also included in the formula as A* algorithm is constructed in this view.

This iteration helps us realize that with the new sound data, robot was able to find a better optimum path. It successfully changes the previous path (3-5-4-2-1) to the new one as 3-1-2-4-5. This concludes that the implementation of second parameter and heuristic approach succeeded on real-time decision-making strategy, which is one of the main goals of this project.

1.3.2. Concepts

It can help to select the most suitable robotic solution to respond quickly and effectively to earthquake disasters. This selection process should include critical factors, especially cost,

maneuverability, performance and features.

In order to compare the performance of earthquake emergency response robots, Table 1 evaluates the three conceptual solutions presented below - X20, COLUSSUS, ANYMAL and our robot - in terms of cost, maneuverability, performance and features. Each solution should be considered in terms of these criteria, which are critical in earthquake disaster response, as well as specific characteristics such as resilience, responsiveness and interactivity. In this way, the selected solution can be justified to best meet earthquake emergency response requirements.

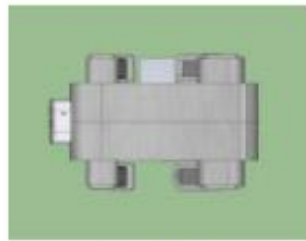
	X20	COLUSSUS	ANYMAL	OUR ROBOT
Cost	Medium	High	Low	Low
Maneuverability	Medium	Low	High	Medium
Performance	Medium	High	Low	Medium
Features	Low	High	Medium	Medium

1.4. Physical architecture

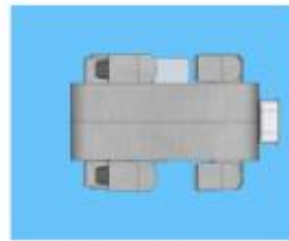
The table below compares the size, visual and thermal cameras, sound detection, gas sensors, battery and ingress protection specifications for 4 different robots.

	X20	COLOSSUS	ANYMAL	OUR ROBOT
SIZE	100*45*80	160*78*76	93*53*89	118*72*96
VISUAL AND THERMAL CAMERAS	YES	YES	YES	YES
SOUND DETECTION	YES	YES	YES	YES
GAS SENSORS	YES	YES	YES	YES
BATTERY	2-4 HOURS	12 HOURS	1-2 HOURS	2-4 HOURS
INGRESS PROTECTION	IP66 LEVEL	IP67 LEVEL	IP67 LEVEL	IP67 LEVEL

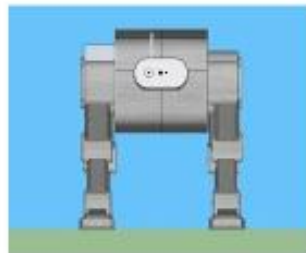
As you can see, each earthquake robot has its own advantages and disadvantages.



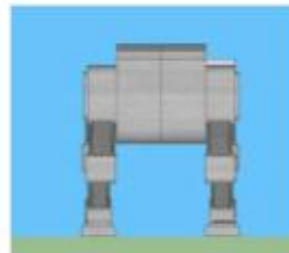
TOP VIEW



UNDER VIEW



FRONT VIEW



BACK VIEW



SECTION VIEW



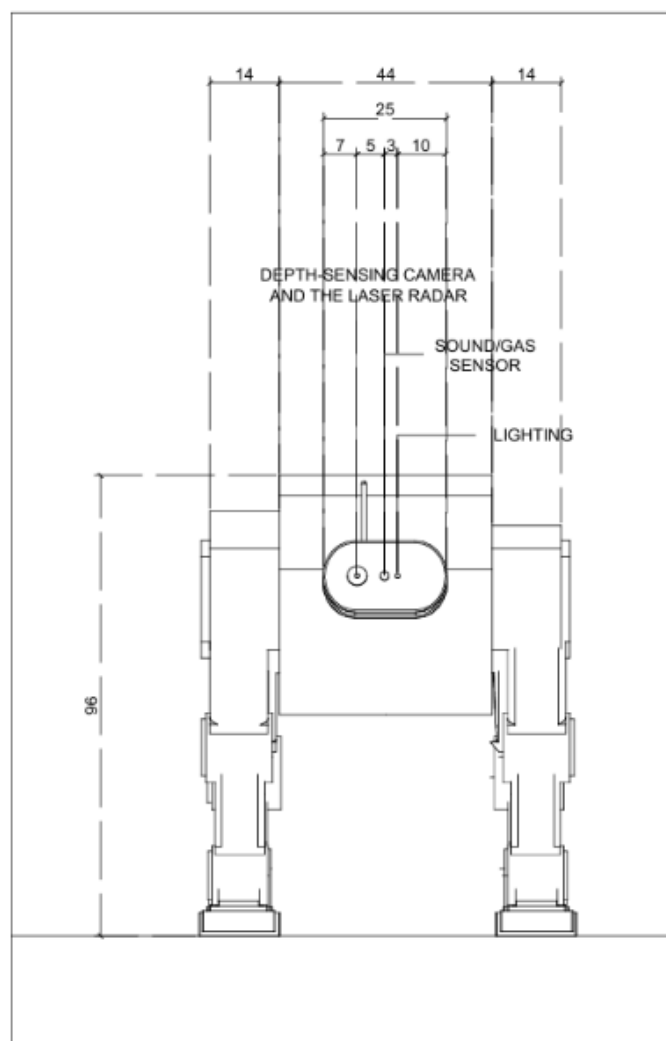
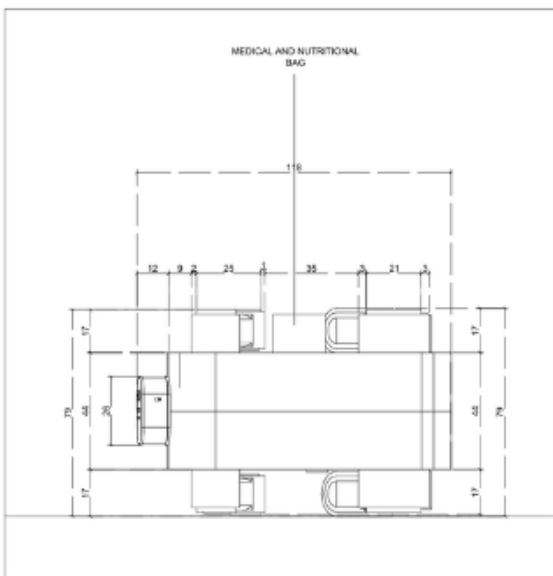
SIDE VIEW

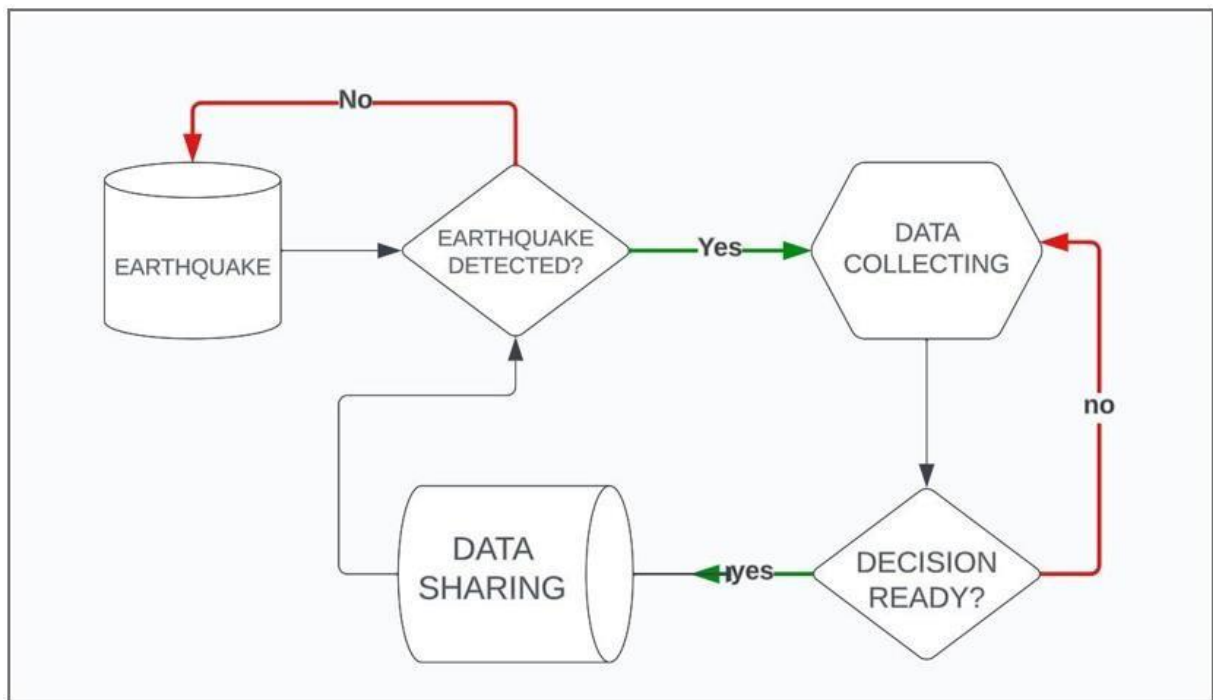


SIDE VIEW

Above are the front, back, bottom, top, side and section views of our robot.

Below are the technical drawings of our robot created with AutoCad.





2. WORK PLAN

2.1. Work Breakdown Structure (WBS)

The following Work Breakdown Structure (WBS) table presents an organized list of the tasks necessary to construct an earthquake emergency robot that can work as planned and with optimum reliability. The plan outlining the progress of the project is presented at the first stage. Following the determination of the essential stages, the requirements sections and literature study are initiated, and the hardware and software requirements for the robot's construction are identified. The project is prepared for implementation during the design phase, which also covers the software components. The project is finalized with comprehensive document preparation and reporting.

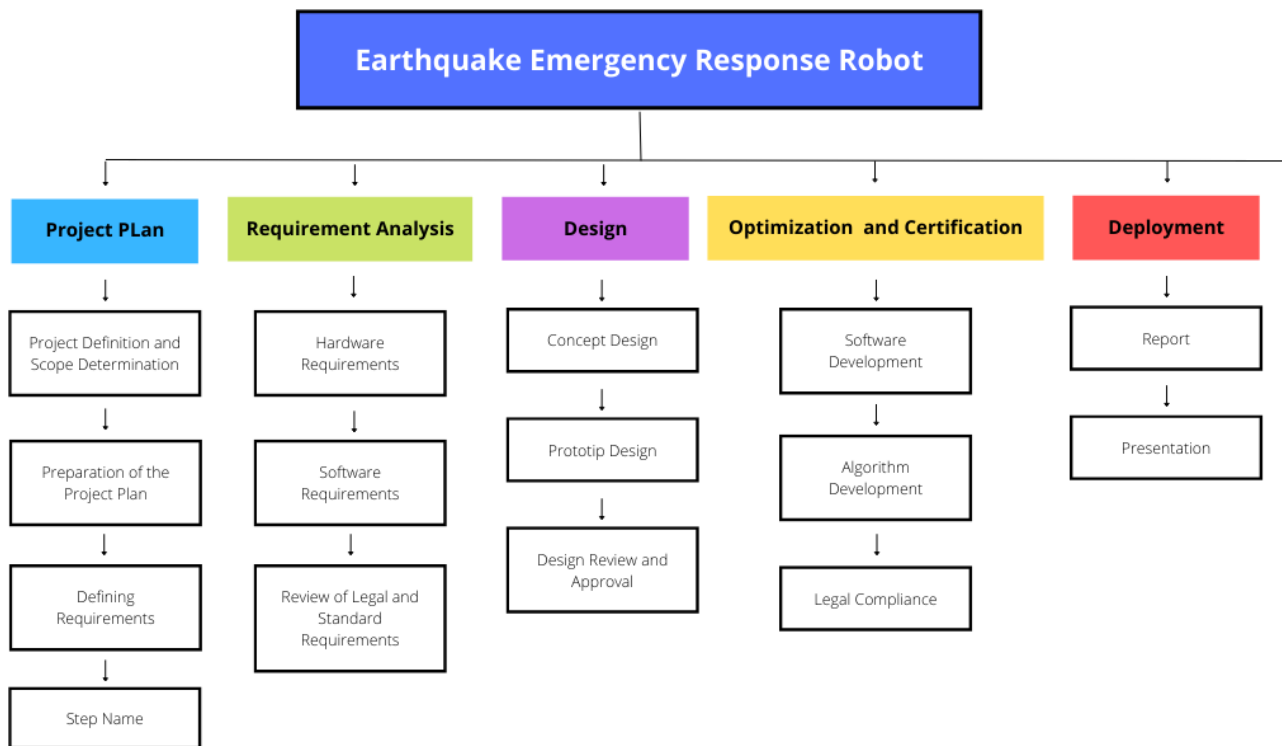


Figure 1. Work breakdown structure for the project.

2.2. Responsibility Matrix (RM)

Following project definition, a guiding responsibility matrix is created that details the allocation of responsibilities when a project is executed according to its scope, as well as which department or individual is in charge of what and who they support. Usually, the matrix is shown as a table with the roles listed on one axis and the tasks or activities listed on the other. Uncertainty, overlap, or absences in responsibilities throughout a project are avoided with the aid of the responsibility matrix. It makes it clear who is responsible for what, who must be informed at all times, who must contribute, and who is the final arbiter. This clarity helps to facilitate more efficient implementation of projects and productive team communication.

RESPONSIBILITY MATRIX

Tasks ▾	Industrial Engineerin ▾	Civil Engineerin ▾	Artificail Intelligenc ▾	Computer Engineerin ▾
Software Implementation			S	R
Path Optimization	R			
Machine Learning			R	
Creating Database		R		
Hardware Design	S	S	S	S
Decision Making	R			
Modelling		R		
Enviroment Conditions		R		
<i>Planning</i>	R	S	S	S
<i>Reporting</i>	R	S	S	S
<i>Implementing</i>	S	S	R	R
R= Responsible				S= Support

Table 1. Responsibility Matrix for the team

2.3. Project Network (PN)

The assignments, activities, and interconnections of a project are represented graphically in a project network. It's an illustration for project management that shows the connections, flow, and order of various tasks needed to finish a project. People gain from it in a variety of ways when projects are being built. As an illustration of this, it provides details specifically on monitoring the project's progress. Recognize how modifications or delays in one task may impact others. It offers details on the assignment of tasks and the distribution of resources. In the context of this project, it is critical to comprehend how each department's tasks relate to those of every other department, to support departmental work, and to monitor project progress as it moves forward. It has been prepared with the understanding that as the process moves forward, its scope may increase. Additionally, the following viewpoint needs to be taken into account in this context. A project network requires ongoing upkeep.

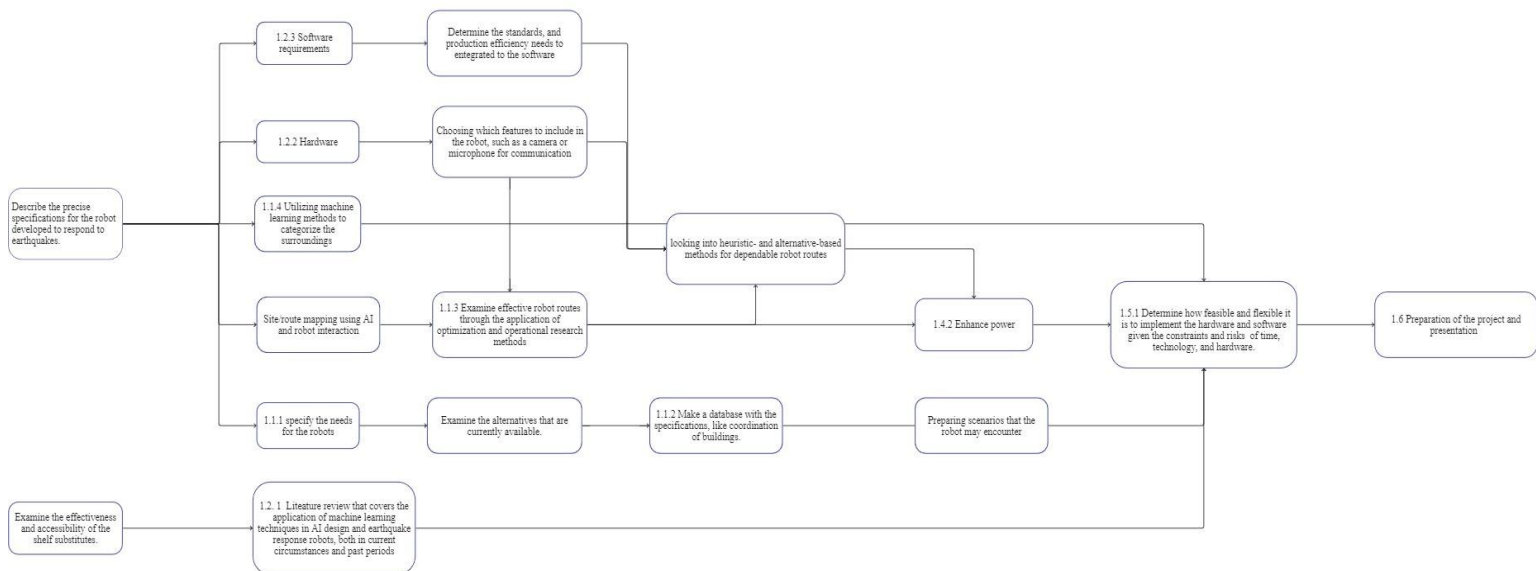
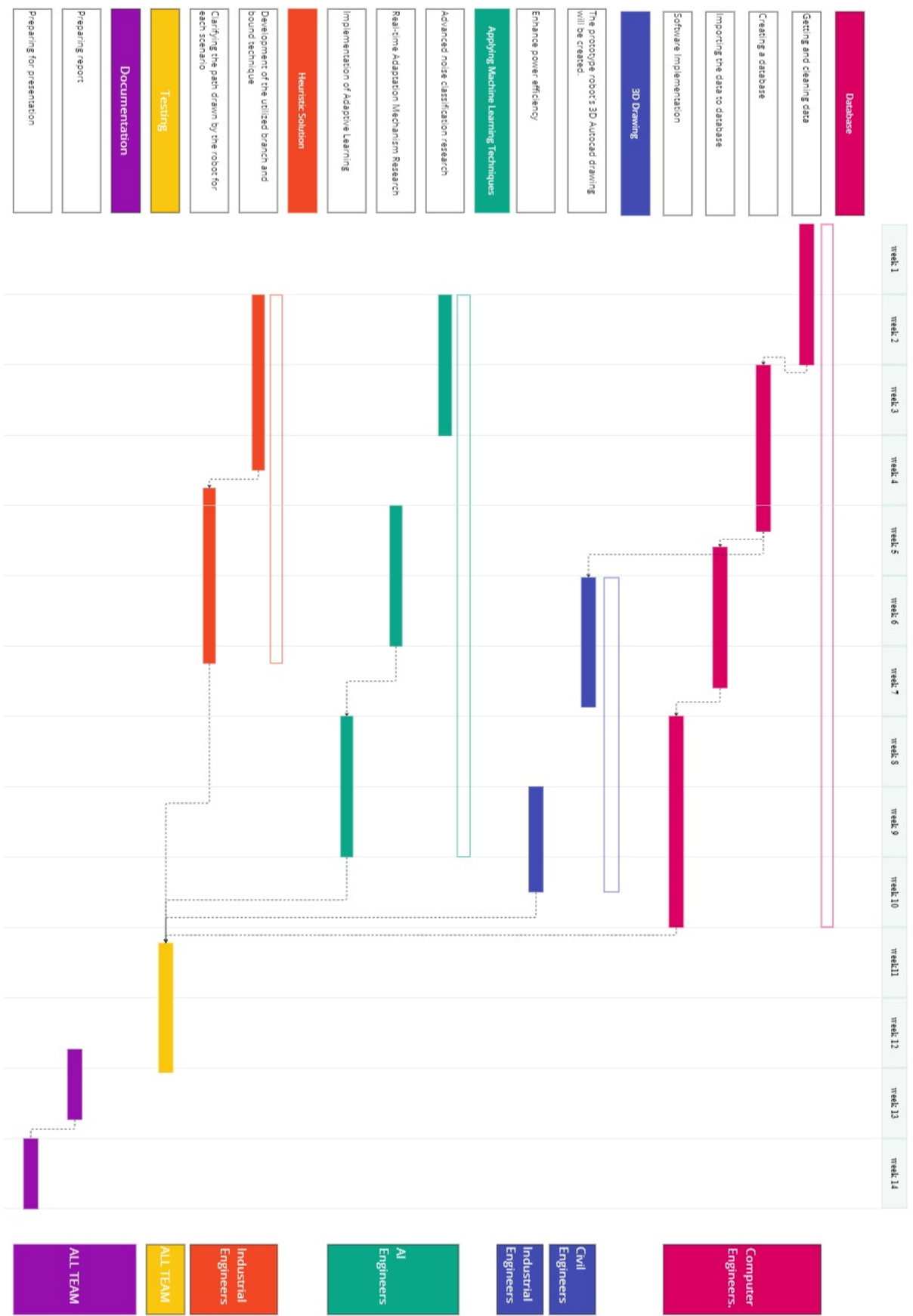


Figure 2. The project network.

2.4. Gantt chart



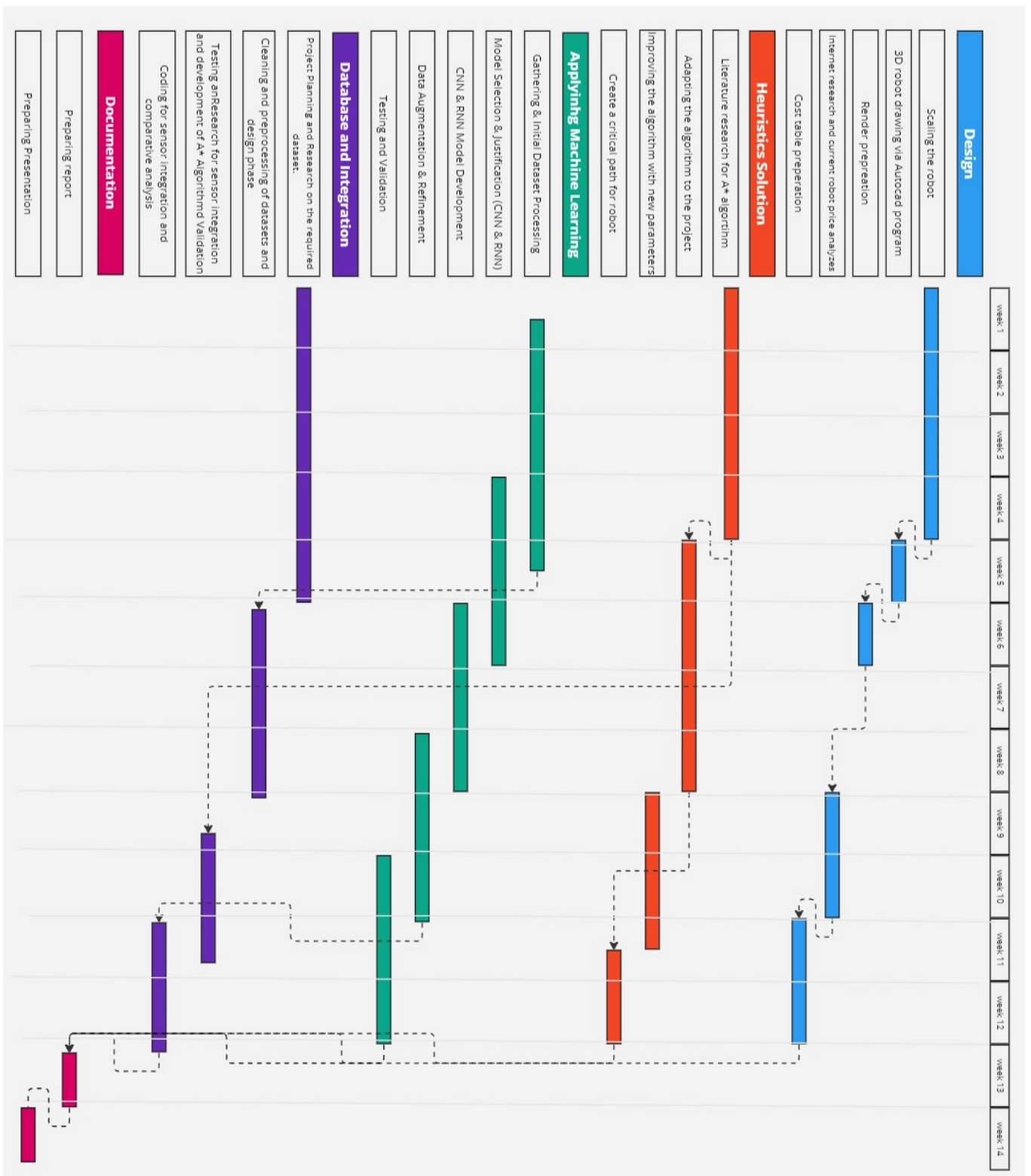


Table 3. Gantt chart for the materialisation phase of the project (whole project).

2.5. Costs

In the tables below, the features and prices of 3 alternative robots with similar features to our robot are listed.

CATEGORY	OPTION 1	PRICE
DEPTH-SENSING CAMERA	Intel RealSense D455	\$700,00
LASER RADAR	Lakibeam 1S LIDAR	\$350,00
MICROPHONE	Rode NT-USB Mini	\$100,00
GAS SENSOR	MQ-2 Gas Sensor	\$150,00
SPOTLIGHT	Olight S2R Baton II	\$70,00
5G COMMUNICATION, GPS	Quectel RM500Q-GL	\$200,00
MOVEMENT MECHANISM	X30 Quadruped Mechanism	\$2.000,00
BATTERY	Volta SE-03 60 Volt 18 Ah LiFePO4 Battery	\$350,00
SOFTWARE	ROS (Robot Operating System)	\$4.000,00
REMOTE CONTROL OF THE ROBOT	Unitree Go2 Controller	\$300,00
TOTAL PRICE:		\$8.220,00

CATEGORY	OPTION 2	PRICE
DEPTH-SENSING CAMERA	Orbbec Femto MEGA I	\$1.950,00
LASER RADAR	Robosense RS LIDAR	\$300,00
MICROPHONE	Blue Yeti	\$130,00
GAS SENSOR	Grove - Gas Sensor (O ²)	\$75,00
SPOTLIGHT	Fenix PD36R	\$100,00
5G COMMUNICATION, GPS	Telit FN980	\$250,00
MOVEMENT MECHANISM	DAGU Wild Thumper 6WD Chassis	\$1.300,00
BATTERY	LeoRover - 160Wh Super-Battery	\$1.000,00
SOFTWARE	NVIDIA Jetson Orin NX 16 GB Module	\$2.500,00
REMOTE CONTROL OF THE ROBOT	FlySky FS-i6	\$200,00
TOTAL PRICE:		\$7.805,00

CATEGORY	OPTION 3	PRICE
DEPTH-SENSING CAMERA	Microsoft Azure Kinect DK	\$400,00
LASER RADAR	G-TEK LND LIDAR	\$100,00
MICROPHONE	Audio-Technica ATR2100x-USB	\$80,00
GAS SENSOR	SPEC Sensors Carbon Monoxide	\$50,00
SPOTLIGHT	Nitecore P12	\$60,00
5G COMMUNICATION, GPS	SIMCom SIM8200EA-M2	\$180,00
MOVEMENT MECHANISM	Lynxmotion A4WD1 Rover	\$2.500,00
BATTERY	Unitree Go2 Battery	\$500,00
SOFTWARE	VPL (Visual Programming Language)	\$3.500,00
REMOTE CONTROL OF THE ROBOT	Taranis Q X7	\$120,00
TOTAL PRICE:		\$7.490,00

In the table below, the features of our robot and the prices of each part are listed and a total cost table is created.

CATEGORY	OUR ROBOT	PRICE
DEPTH-SENSING CAMERA	Intel RealSense D455	\$700,00
LASER RADAR	Robosense RS LIDAR	\$300,00
MICROPHONE	Blue Yeti	\$130,00
GAS SENSOR	MQ-2 Gas Sensor	\$150,00
SPOTLIGHT	Fenix PD36R	\$100,00
5G COMMUNICATION, GPS	Quectel RM500Q-GL	\$200,00
MOVEMENT MECHANISM	X30 Quadruped Mechanism	\$2.000,00
BATTERY	LeoRover - 160Wh Super-Battery	\$1.000,00
SOFTWARE	NVIDIA Jetson Orin NX 16 GB Module	\$2.500,00
REMOTE CONTROL OF THE ROBOT	Unitree Go2 Controller	\$300,00
TOTAL PRICE :		\$7.380,00

2.6. Risk assessment

Table 4. Risk matrix

Probability of the event occurring	RISK LEVEL	Severity of the event on the project success				
		Minor	Moderate	Major		
	Unlikely	VERY LOW	LOW	MEDIUM	VERY LOW	This event is very low risk and so does not require any plan for mitigation. In the unlikely event that it does occur there will be only a minor effect on the project.
	Possible	LOW	MEDIUM	HIGH	LOW	This event is low-risk; a preliminary study on a plan of action to recover from the event can be performed and noted.
	Likely	MEDIUM	HIGH	VERY HIGH	MEDIUM	This event presents a significant risk; a plan of action to recover from it should be made and resources sourced in advance.
					HIGH	This event presents a very significant risk. Consider changing the product design/project plan to reduce the risk; else a plan of action for recovery should be made and resources sourced in advance.
					VERY HIGH	This is an unacceptable risk. The product design/project plan must be changed to reduce the risk to an acceptable level.

Table 5. Risk assessment

3. SUB-SYSTEMS

3.1 Computer Engineering

This capstone project aims to produce an emergency robot after a possible earthquake. This robot aims to be prepared for possible emergency scenarios with smart algorithms and software solutions. As computer engineering students, we create data and share this data with artificial intelligence engineering students. Computer engineering students are also responsible for producing heuristic algorithms for the algorithm suggested by industrial

engineering students for the project.

3.1.1. Requirements

The earthquake emergency response robot, one of the most advanced experiments in its category, uses optimization algorithms. The requirements of this section, which includes the comparison of the Traveling Salesman Problem (TSP) algorithm and the A* algorithm for a robot tasked with visiting more than one building, can be detailed as follows.

A set of coordinates for the building and the robot's starting position are defined. These coordinates are entered directly into the program.

The total path length is calculated for both algorithms. It measures the computational time each algorithm takes to find paths. Situations where the A* algorithm cannot find a path are detected. Buildings may be clustered or far apart. The movement of the robot is simulated using calculated paths.

Optionally, the paths followed by the robot for both algorithms are visualized.

The code is prepared modularly to allow easy updates and maintenance. Comments are added within the code to explain complex logic and decisions. C++ is used as a programming language due to its performance advantages and widespread use in algorithmic challenges. A suitable Integrated Development Environment (IDE) such as Visual Studio, CLion, or a simple text editor with a command line interface is used. The development environment must support C++17 or later for modern language features.

For real-world testing, hardware prototypes are assembled using Raspberry Pi microcontrollers and sensors to simulate the robotic platform. Version control tools like Git are used to manage the code base, track changes, and collaborate with team members.

3.1.2. Technologies and methods

We used TSP in the first part of the project for the emergency robot. We have to say that we are stuck between TSP and A* in this process. Now let's evaluate these two algorithms with

their advantages and disadvantages.

Advantages Of Using TSP	Disadvantages Of Using TSP
Optimization of Robot Paths	Scalability
Coverage of Wreck Locations	Dynamic Environment
Simple Implementation	Complex Terrain
Guaranteed Solution	Alternative Approaches

Below are C++ implementations for comparing the TSP and A* algorithms using the provided coordinates for the robot and buildings.

```
← city.h
1 #ifndef CITY_H
2 #define CITY_H
3
4 struct Point {
5     int x, y;
6     Point(int x, int y) : x(x), y(y) {}
7 };
8
9 // Define the coordinates of the buildings and the robot
10 Point buildings[] = {
11     Point(24, 22),
12     Point(24, 22),
13     Point(24, 22),
14     Point(24, 22),
15     Point(24, 22)
16 };
17
18 Point robot(56, 82);
19
20 #endif // CITY_H
21
```

City.h File:

The city.h file defines the coordinates of the buildings and the robot. It is included in both the TSP and A* algorithm implementations to ensure the coordinates are consistently use in both algorithms.

Point Struct:

Defines a Point struct with x and y coordinates.

Provides a constructor for easy initialization.

Coordinates:

Defines an array of Point objects named buildings to represent the coordinates of the buildings. Defines a Point object named robot to represent the coordinates of the robot.

The TSP algorithm implementation reads from city.h to access the coordinates of the buildings

and the robot.

```
← tsp.cpp
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 #include <climits>
5 #include <chrono>
6 #include "city.h"
7
8 using namespace std;
9
10 #define V 6
11
12 int dist(Point a, Point b) {
13     return abs(a.x - b.x) + abs(a.y - b.y);
14 }
15
16 int travellingSalesmanProblem(Point points[], int s) {
17     vector<int> vertex;
18     for (int i = 0; i < V; i++)
19         if (i != s)
20             vertex.push_back(i);
21
22     int min_path = INT_MAX;
23     do {
24         int current_pathweight = 0;
25         int k = s;
26         for (int i = 0; i < vertex.size(); i++) {
27             current_pathweight += dist(points[k], points[vertex[i]]);
28             k = vertex[i];
29         }
30         current_pathweight += dist(points[k], points[s]);
31         min_path = min(min_path, current_pathweight);
32     } while (next_permutation(vertex.begin(), vertex.end()));
33
34     return min_path;
35 }
36
37
38 int main() {
39     Point points[V] = {robot, buildings[0], buildings[1], buildings[2], buildings[3], buildings[4]};
40     auto start = chrono::high_resolution_clock::now();
41     int result = travellingSalesmanProblem(points, 0);
42     auto end = chrono::high_resolution_clock::now();
43     chrono::duration<double> duration = end - start;
44
45     cout << "TSP total path length: " << result << endl;
46     cout << "TSP computation time: " << duration.count() << " seconds" << endl;
47     return 0;
48 }
49
```

The TSP algorithm calculates the minimum path to visit all given buildings and return to the starting point using a brute-force approach.

```
← tsp
TSP total path length: 184
TSP computation time: 0.00335083 seconds
```

Using the A* algorithm in a post-earthquake support system with AI robots can offer several advantages, but it also comes with certain limitations. Here's a breakdown of when you might consider using A* and its pros and cons:

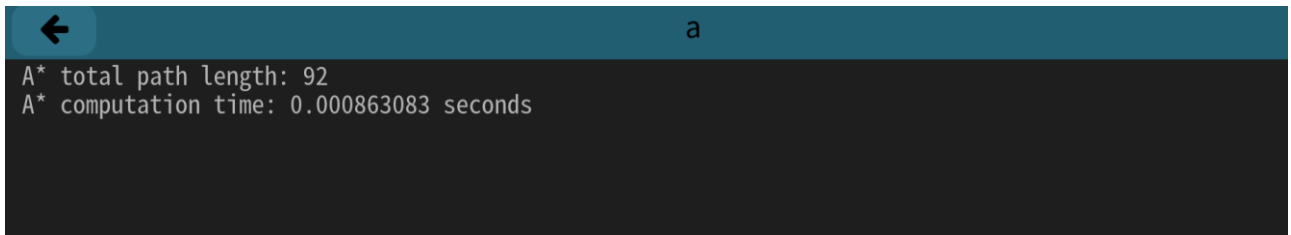
Advantages Of Using A*	Disadvantages Of Using A*
Efficient Pathfinding	Memory and Computational Requirements
Heuristic-based Approach	Heuristic Accuracy
Optimality	Dynamic Environments
Flexibility	Complexity of Implementation

The TSP algorithm is designed to find the shortest possible route that visits all given points exactly once and returns to the starting point. The TSP algorithm's higher computation time is due to its combinatorial nature. It explores all possible permutations of routes to find the optimal one, which is computationally expensive, especially as the number of points increases.

The A* algorithm finds the shortest path between the robot and each building sequentially. The A* algorithm's lower computation time indicates it is more computationally efficient for finding the shortest path in sequence. This efficiency is due to its heuristic-driven approach, which prioritizes paths that seem more promising.



```
1 #include <iostream>
2 #include <vector>
3 #include <queue>
4 #include <cmath>
5 #include <unordered_map>
6 #include <chrono>
7 #include "city.h"
8
9 using namespace std;
10
11 struct Node {
12     Point pt;
13     double cost, priority;
14     bool operator>(const Node& other) const {
15         return priority > other.priority;
16     }
17 };
18
19 double heuristic(Point a, Point b) {
20     return abs(a.x - b.x) + abs(a.y - b.y);
21 }
22
23 double aStar(Point start, Point goal, const vector<Point>& buildings) {
24     priority_queue<Node, vector<Node>, greater<Node>> openSet;
25     unordered_map<int, double> costSoFar;
26     auto hashPoint = [](const Point& pt) {
27         return pt.x * 31 + pt.y;
28     };
29
30     openSet.push({start, 0, heuristic(start, goal)});
31     costSoFar[hashPoint(start)] = 0;
32
33     while (!openSet.empty()) {
34         Node current = openSet.top();
35         openSet.pop();
36
37         if (current.pt.x == goal.x && current.pt.y == goal.y) {
38             return current.cost;
39         }
40
41         for (const Point& next : buildings) {
42             double newCost = current.cost + heuristic(current.pt, next);
43             int nextHash = hashPoint(next);
44             if (costSoFar.find(nextHash) == costSoFar.end() || newCost < costSoFar[nextHash]) {
45                 costSoFar[nextHash] = newCost;
46                 double priority = newCost + heuristic(next, goal);
47                 openSet.push({next, newCost, priority});
48             }
49         }
50     }
51
52     return numeric_limits<double>::infinity();
53 }
54
55 int main() {
56     vector<Point> points = {buildings[0], buildings[1], buildings[2], buildings[3], buildings[4]};
57     auto start = chrono::high_resolution_clock::now();
58     double totalCost = 0;
59     Point current = robot;
60
61     for (const Point& target : points) {
62         totalCost += aStar(current, target, points);
63         current = target;
64     }
65
66     auto end = chrono::high_resolution_clock::now();
67     chrono::duration<double> duration = end - start;
68
69     cout << "A* total path length: " << totalCost << endl;
70     cout << "A* computation time: " << duration.count() << " seconds" << endl;
71     return 0;
72 }
73
```

A terminal window with a dark background and light blue text. The text displays the results of an A* search: 'A* total path length: 92' and 'A* computation time: 0.000863083 seconds'. Above the text, there is a light blue header bar containing a back arrow icon on the left and the letter 'a' on the right.

```
← a
A* total path length: 92
A* computation time: 0.000863083 seconds
```

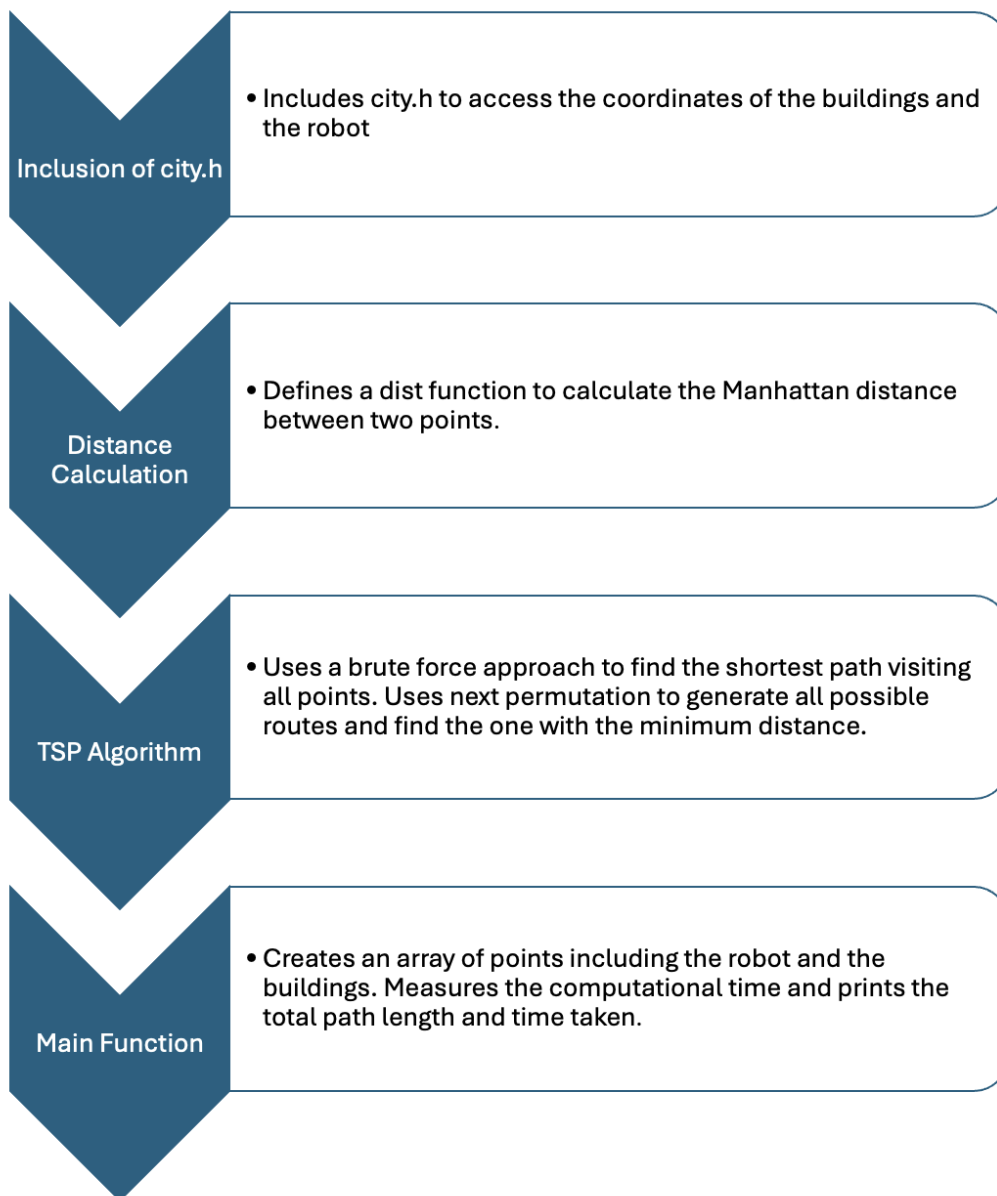
3.1.3. Conceptualization

As a computer engineering student tasked with developing algorithmic framework for earthquake emergency robot, post-earthquake support system, the conceptualization of the project revolves around leveraging the power of optimization algorithms and artificial intelligence to overcome the complex challenges of disaster response. The conceptual framework of the project is based on the fundamental principles of algorithm design, optimization theory and computational complexity. Leveraging established algorithms such as the Traveling Salesman Problem (TSP) and A*, the project aims to develop innovative solutions that optimize the deployment and navigation of AI robots in disaster-affected areas. The conceptualization process involves investigating various algorithmic approaches, testing different optimization techniques, and evaluating their performance through simulation and testing. By conceptualizing a robust algorithmic framework, the project aims to lay the foundation for the successful implementation of a post-earthquake support system and contribute to the advancement of both computer engineering and disaster response technology.

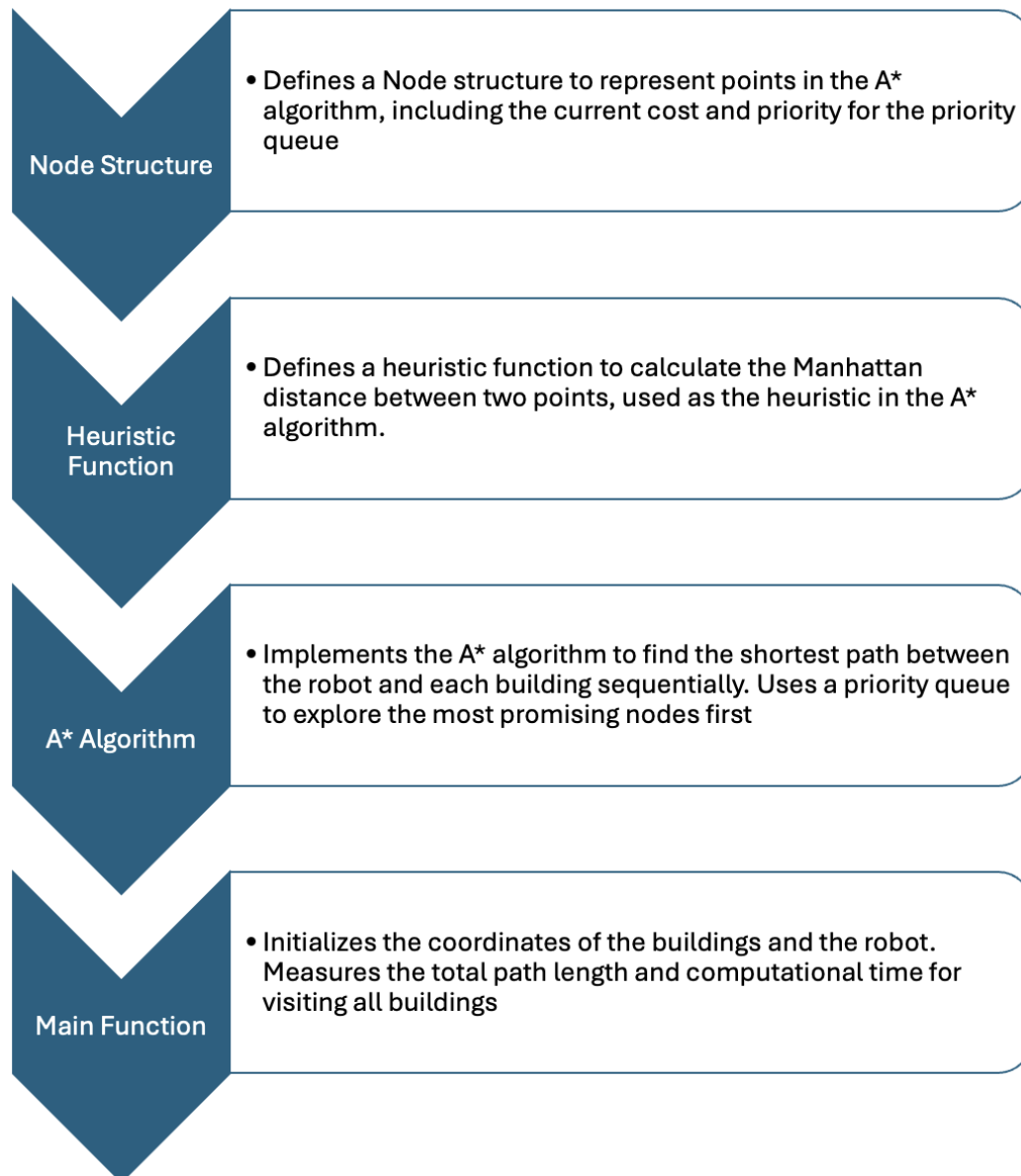
3.1.4. Physical architecture

Both implementations use the city.h file to maintain a consistent set of coordinates for the buildings and the robot. The TSP algorithm uses a brute force approach to find the shortest path visiting all points. The A* algorithm finds the shortest path between the robot and each building sequentially. Both implementations measure the total path length and computational time, allowing for a direct comparison of their performance in terms of route efficiency and computational complexity.

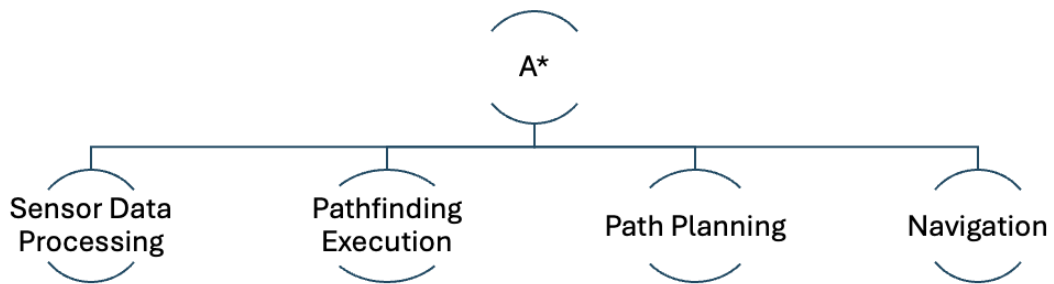
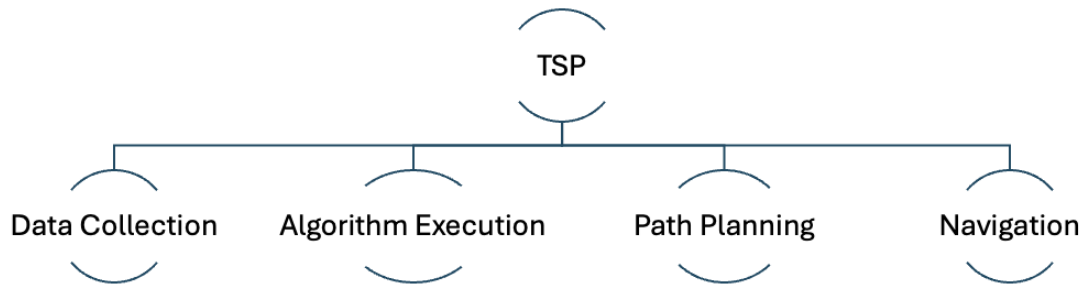
Explanation of TSP Implementation:



Explanation of A* Implementation:



A brief comparison of the physical architecture for the implementation of the Traveling Salesman Problem (TSP) and A* algorithms in the context of earthquake emergency robot, post-earthquake support system is given below:



3.1.5. Materialization

As a computer engineering student tasked with implementing the TSP and A* algorithms for the project, the implementation phase involved translating theoretical concepts into practical solutions through software development, testing, and validation. Using the C++ programming language, I worked on developing software prototypes to implement the TSP algorithm for optimizing robot routes and the A* algorithm for local path planning. Collaborative efforts with team members will enable successful integration of algorithmic solutions into the overall system architecture by facilitating information sharing and troubleshooting.

3.1.6. Evaluation

	TSP (Traveling Salesman Problem)	A* Algorithm
Total Path Length	184	92
Computation Time	0.00335083 seconds	0.000955625 seconds

If the goal is to find the shortest possible cycle that visits all points exactly once and returns to the starting point, TSP should be used despite its high computational cost. If the goal is to

visit all points sequentially, focusing on computational efficiency and shorter path lengths, A* should be used.

Looking at the results, it is seen that the A* algorithm is more efficient and practical for the earthquake emergency robot and provides shorter path lengths and faster calculation times, which are very important in emergency response situations.

Data Sources

1. Earthquake Dataset

The dataset used in this project encompasses records of 782 earthquakes occurring between 01/01/2001 and 01/01/2023, sourced from Kaggle. It comprises diverse attributes providing insights into each seismic event:

- Magnitude: Reflecting the earthquake's strength.
- Datetime: Capturing the precise date and time of occurrence.
- CDI (Community Directivity Index): Indicating the maximum reported intensity perceived by communities.
- MMI (Modified Mercalli Intensity): Estimating the maximum instrumental intensity.
- Alert: Categorizing the alert level into "green," "yellow," "orange," or "red."
- Tsunami: A binary indicator highlighting the occurrence of a tsunami.
- Sig: A numerical measure describing the event's significance, determined by various factors including magnitude, MMI, and impact.
- Net: Identifying the preferred data contributor network.
- Nst: Total seismic stations utilized for locating the earthquake.
- Dmin: Horizontal distance from the epicenter to the nearest station.
- Gap: The largest azimuthal gap between adjacent stations.
- MagType: The algorithm used for calculating magnitude.
- Depth: The depth at which the earthquake begins to rupture.
- Latitude/Longitude: Geographic coordinates of the earthquake's epicenter.
- Location: Specific site within the affected country.
- Continent/Country: Geographic context of the earthquake's impact.

Data Sources

2.Additional Datasets

- UrbanSound8K

This dataset comprises 8,732 labeled sound excerpts, each ≤ 4 seconds long, showcasing urban sounds across ten distinct classes. For this project, the focus was on the *jackhammer* class, aiding in identifying urban sounds pertinent to earthquake response scenarios. The dataset includes audio files in WAV format alongside corresponding metadata stored in UrbanSound8K.csv.

- Scream

Dataset

Designed to label urban sounds, this dataset features various audio samples enhancing the system's ability to accurately identify and classify sounds in urban environments post-earthquake. Of particular significance is the recognition of distress signals like screams, pivotal for directing emergency response efforts.

Data Analysis and Insights

Attributes Utilized

The analysis harnessed several key attributes, including magnitude, CDI, and MMI, which were crucial for assessing the immediate risk and impact of earthquakes. Additionally, attributes like alert and tsunami were instrumental in identifying higher-risk zones and potential tsunami risks. Understanding the geographical and spatial context of earthquakes was facilitated by attributes such as depth, latitude/longitude, and location. Furthermore, continent and country information provided valuable insights into the affected regions, aiding in comprehensive analysis and response planning.

Analysis Performed

In conducting the earthquake response system analysis, a comprehensive assessment was undertaken across various dimensions:

- Immediate Risk Assessment: Utilizing attributes such as magnitude, intensity, alert levels, and tsunami indicators, a thorough evaluation was conducted to gauge the immediate risk and impact of earthquakes.
- Building and Infrastructure Damage Assessment: Through a detailed examination encompassing building types, infrastructure data, and the identification of vulnerable structures,

the potential damage and collapse risks were meticulously assessed, providing crucial insights for effective response planning.

- **Healthcare and Emergency Services Analysis:** The capacity and accessibility of healthcare facilities and emergency services in affected areas were rigorously evaluated. Special attention was directed towards regions with limited access, ensuring a comprehensive understanding of the readiness and responsiveness of the healthcare and emergency service infrastructure in earthquake-affected areas.

Data Preprocessing

Cleaning

To uphold the dataset's integrity, we enacted rigorous data-cleaning procedures. This involved systematically eliminating inconsistencies and filling in missing values to bolster data accuracy. A pivotal aspect of this process was standardizing date and time formats, ensuring uniform processing, and enabling seamless integration and analysis across varying periods. Likewise, meticulous attention was devoted to standardizing geographic coordinates and location data, enhancing the accuracy of spatial analyses. Identified anomalies were promptly rectified, further enhancing the dataset's reliability. This meticulous cleaning regimen laid a sturdy foundation for subsequent analyses, yielding insightful and dependable results.

Integration with AI Models

Collaboration with AI Engineers

My collaboration with AI engineers involves integrating preprocessed datasets into machine learning models and deep learning methods. By deriving these models from audio data, we trained them to predict immediate risk factors. As a result of these efforts, we developed predictive models that provide timely forecasts for effective earthquake response.

Predictive Analytics

Through the application of algorithms, we were able to identify high-risk areas by utilizing both historical data and current updates. They improved their predictive analytics by using audio data from sources like databases of screams and jackhammers. Through the analysis of these audio signals, our technology was able to accurately determine debris hazards and the exact locations affected by the earthquakes

Example Code Integration

Here's how artificial intelligence students utilized the provided datasets, as depicted in Figure 1. They used libraries such as Pandas, NumPy, and librosa for code, data manipulation, and sound processing. The sound files were labeled, a process that transforms the audio signals into a visual representation of the frequency spectrum. Another function was created to load and preprocess the dataset, which involves normalizing the audio files to a fixed length and encoding the labels. The dataset containing labeled audio files related to seismic events such as structural damage and human activity is then loaded and preprocessed.

The resulting data is shuffled to ensure robust model training and split into training and validation sets. Artificial intelligence engineers used this preprocessed dataset to train machine learning models for the accurate prediction and classification of sounds related to seismic events.

```
# Function to extract spectrogram from audio file
def extract_spectrogram(audio_file):
    y, sr = librosa.load(audio_file, sr=22050)
    spectrogram = librosa.feature.melspectrogram(y=y, sr=sr, n_mels=128)
    log_spectrogram = librosa.power_to_db(spectrogram, ref=np.max)
    return log_spectrogram

# Function to load and preprocess dataset
def load_dataset(dataset):
    X, y = [], []
    max_columns = 128

    for audio_file, label in dataset:
        spectrogram = extract_spectrogram(audio_file)
        if spectrogram.shape[1] >= max_columns:
            spectrogram = spectrogram[:, :max_columns]
        else:
            pad_width = max_columns - spectrogram.shape[1]
            spectrogram = np.pad(spectrogram, ((0, 0), (0, pad_width)), mode='constant')

        X.append(spectrogram)
        y.append(label)

    label_encoder = LabelEncoder()
    y_encoded = label_encoder.fit_transform(y)

    return np.array(X), y_encoded

# Define the dataset
dataset = [
    ('/path/to/jackhammer.wav', 'structural_damage'),
    ('/path/to/scream1.wav', 'human_activity'),
    ('/path/to/scream2.wav', 'human_activity')
]

# Load and preprocess dataset
X, y = load_dataset(dataset)

# Shuffle and split dataset
X_shuffled, y_shuffled = shuffle(X, y, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_shuffled, y_shuffled, test_size=0.2, random_state=42, stratify=y_shuffled)
```

4.Deployment and Testing: Upon completion of the script development, the integrated system underwent rigorous testing to validate its performance and reliability under various real-world scenarios. This comprehensive testing phase included deploying the script onto the edge device and configuring the microphone for optimal audio capture. Extensive evaluations were conducted to assess the system's responsiveness and accuracy in detecting and classifying sounds related to seismic events. This rigorous testing ensured the integrated system's robustness, making it a reliable tool for providing timely and accurate information crucial for effective earthquake emergency response

efforts.

3.2. The AI department

Since earthquakes are so damaging and produce such dangerous situations, they present serious obstacles to the search and rescue efforts. Our role in this capstone project was to create a complex machine learning model for sound classification to help Earthquake Emergency Response Robots locate and identify survivors in an earthquake situation. This report goes over the research and developments of the project's goals, methods, findings, and implications for further work.

3.2.1. Requirements

The primary objectives of this project were:

- **Data Collection and Preprocessing:** gather some datasets of environmental sounds related to earthquakes, made of structural damage, human presence and emergency signal noises. Then preprocess these datasets for the model training and optimization.
- **Model Selection and Development:** Explain why we chose the Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) as the main algorithms for our sound classification model. Then develop, train and evaluate the model to achieve high accuracy.
- **Integrated System Development:** Combine the 2 models (CNN and RNN) together to make one model that is capable of classifying the environmental sounds in real-time.
- **Real-World Validation:** test the model's performance in simulated earthquake scenarios using a variety of sounds to evaluate its effectiveness and accuracy in real-world situations.

3.2.2. Technologies and methods

Data Collection and Preprocessing:

We started the project by gathering existing datasets on kaggle like ESC-50 and UrbanSound8K. These datasets includes a variety of common sounds such as screams, structure destruction and noises related to earthquakes damage. and since the data was limited,

we had to create a new dataset using audio samples from internet repositories for better model training.

Data preprocessing involved a series of crucial steps:

- **Normalization:** We standardized the audio features of all the samples we have to remove any potential biases and ensure consistent input to the models.
- **Resampling:** All audio samples were resampled to a uniform rate to ensure compatibility with our machine learning algorithms.
- **Spectrogram Generation:** Each audio sample was converted into a spectrogram representation, which captures the frequency content of the sound over time. Spectrograms serve as an effective input format for CNNs, allowing the models to learn relevant features from the audio data.
- **Data Augmentation:** To further improve model generalization and robustness, we applied various data augmentation techniques including time shifting and pitch shifting.

3.2.3. Model Selection, Justification and Development

Given that research has shown that CNNs and RNNs perform well in audio classification tasks, we chose them as our primary models (Piczak, 2015; Hershey et al., 2017). CNNs are particularly good at identifying noises connected to structural damage because they can easily extract local features and patterns from spectrograms. While RNNs do an excellent job at recognizing human vocalizations and other dynamic noises requires the ability to capture the evolution of sounds over time, because of their capacity to model temporal dependencies.

We built the CNN and RNN models for classifying structural and human sounds. The CNN model was built with numerous convolutional layers followed by fully linked layers, whereas the RNN model used a Long Short-Term Memory (LSTM) architecture to successfully capture temporal patterns. The models were trained by combining cross-entropy loss with Adam optimization.

3.2.4. Model Integration And Validation

In order to create a real-time sound classification model, we integrated the trained CNN and

RNN so that model operates as follows:

- **Audio Input:** The microphone on the robot captures environmental sounds.
- **Preprocessing:** The captured audio goes through normalization and resampling to be ready for analysis.
- **Feature Extraction:** The CNN model extracts the relevant features from the spectrogram representation of the audio files.
- **Temporal Analysis:** The RNN model analyzes the temporal patterns in the extracted features to identify sound categories.
- **Classification:** The model produces a final classification of the sound category (e.g., "human voice," "collapsing structure," "emergency signal").

The model went thorough real-world testing to make sure it is effective in earthquake scenarios. Simulated earthquake environments replicated various noise sources, including structural damage, human activity and emergency signals. Rigorous testing with real earthquake audio samples validated the model's accuracy and adaptability, confirming its suitability for deployment in Earthquake Emergency Response Robots to help rescue efforts.

3.2.5. Results And Code

Our integrated CNN-RNN sound classification system achieved high accuracy in identifying and classifying relevant sounds in simulated earthquake scenarios:

- **Locating Survivors:** The system accurately detected human voices and calls for help even in noisy environments with multiple overlapping sounds. This capability is crucial for directing rescue teams to areas where survivors are trapped.
- **Assessing Structural Damage:** The CNN model effectively classified sounds associated with collapsing structures and falling debris, providing valuable information about the extent of damage and potential danger.

below will be a general overview of the code:

1. Import and setup

We imported the necessary libraries:

- `librosa`: For audio processing and feature extraction.
- `sklearn`: For data handling and preprocessing.
- `tensorflow.keras`: For building and training the AI model.

- logging: For debugging and tracing errors.

```
import numpy as np
import pandas as pd
import librosa
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.utils import shuffle
from tensorflow.keras import models, layers
import librosa.effects as le
import cv2
import os
import tensorflow as tf
import logging
import shutil
```

2. Data augmentation

To make sure the model can handle variations.

```
# Function for data augmentation
def augment_data(spectrogram):
    augmented_spectrogram = le.time_stretch(spectrogram, rate=1.2)
    augmented_spectrogram = le.pitch_shift(augmented_spectrogram, sr=22050, n_steps=2)
    return augmented_spectrogram
```

3. Extracting spectrograms

To convert the audio signals into a visual representation that the AI can learn from.

```
# Function to extract spectrogram from audio file
def extract_spectrogram(audio_file):
    try:
        y, sr = librosa.load(audio_file, sr=22050)
        spectrogram = librosa.feature.melspectrogram(y=y, sr=sr, n_mels=128)
        log_spectrogram = librosa.power_to_db(spectrogram, ref=np.max)
        return log_spectrogram
    except Exception as e:
        print(f"Error processing {audio_file}: {e}")
        return None
```


4. Model Architecture

We created a combined CNN-RNN model to leverage both spatial and temporal features of the audio data.


```
# Model Architecture with CNN and RNN
model = models.Sequential([
    layers.InputLayer(input_shape=(128, 128, 1)),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.TimeDistributed(layers.Flatten()),
    layers.LSTM(128, return_sequences=True),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(3, activation='softmax')
])
```

5. The Model's Accuracy

Here are the training and validation accuracy of the model.

Epoch 20/20
57/57  **30s** 533ms/step - accuracy: 0.9832
 Training Accuracy: 98.62%
 Validation Accuracy: 98.68%

So, basically this summarizes the code used to build this model, below are the codes needed for the deployment of the model.

```
# Function for quantizing the model
def quantize_model(model):
    converter = tf.lite.TFLiteConverter.from_keras_model(model)
    converter.optimizations = [tf.lite.Optimize.DEFAULT]
    quantized_tflite_model = converter.convert()
    return quantized_tflite_model

# Function for loading the model
def load_model(file_path):
    interpreter = tf.lite.Interpreter(model_path=file_path)
    interpreter.allocate_tensors()
    return interpreter

# Function for running inference
def run_inference(model, audio_data):
    input_details = model.get_input_details()
    output_details = model.get_output_details()

    preprocessed_data = preprocess_audio(audio_data)
    model.set_tensor(input_details[0]['index'], preprocessed_data)
    model.invoke()
    output_data = model.get_tensor(output_details[0]['index'])

    return output_data
```

```

# Function for preprocessing audio data
def preprocess_audio(audio_data):
    spectrogram = librosa.feature.melspectrogram(y=audio_data, sr=22050, n_mels=128)
    log_spectrogram = librosa.power_to_db(spectrogram, ref=np.max)
    log_spectrogram = np.expand_dims(log_spectrogram, axis=-1)
    log_spectrogram = np.expand_dims(log_spectrogram, axis=0)
    return log_spectrogram.astype(np.float32)

# Function for deploying the model
def deploy_model(model_path, deployment_path):
    shutil.copy(model_path, deployment_path)
    print("Model deployed successfully to:", deployment_path)

```

4. INTEGRATION AND EVALUATION

The project's goal is to create a sophisticated post-earthquake response mechanism that quickly and accurately picks the best paths. In order to determine the most effective routes to damaged areas, this system combines the Traveling Salesman Problem (TSP) with the A* algorithm.

This reduces travel time and maximizes the efficacy of responses. The goal of this project is to develop a strong post-earthquake reaction system through utilizing sophisticated optimization techniques. The system guarantees that post-earthquake response robots may reach impacted areas promptly and effectively by merging the TSP and A* algorithm, eventually saving lives and lessening the impact of disasters. The system's continued effectiveness and dependability in practical situations will be guaranteed by ongoing assessment and incremental enhancements.

4.1. Integration

Sensor Integration

The integration of the seismic event detection and analysis system with real-time sound processing capabilities represents a significant advancement. By integrating the trained model into a microphone, the system can perform real-time detection and classification of earthquake-related sounds. This integration can help address the urgency associated with earthquakes by providing instant information about structural damage, human screams, and emergency signals.

Hardware

The hardware setup employed in this integration comprised microcontrollers or single-board computers such as Raspberry Pi, NVIDIA Jetson Nano, or Arduino, equipped with advanced audio processing capabilities. Additionally, high-quality microphone modules were utilized for precise audio capture, ensuring the accurate detection and analysis of seismic event-related sounds.

Software Setup

The successful integration of the earthquake response system's sound classification model with the specified hardware infrastructure. The basic structures of the software installation were as follows:

TensorFlow Lite Conversion: The trained Keras model was converted to TensorFlow Lite format, optimizing its performance for deployment on edge devices.

Librosa: Utilized for on-device audio processing and feature extraction, Librosa played a pivotal role in extracting relevant features from seismic event-related audio data.

OpenCV: Additional image processing capabilities provided by OpenCV further

enhanced the system's ability to analyze seismic event-related data, contributing to more comprehensive insights.

Python Script Execution: Python served as the primary scripting language for orchestrating the entire integration process, from audio capture and processing to model inference and result interpretation.

Steps for Integration

1. **Model Conversion to TensorFlow Lite:** The process of converting a trained Keras model to TensorFlow Lite format involves a straightforward but crucial series of steps. The provided code snippet (Figure 2) outlines this conversion.

```
import tensorflow as tf

def convert_model_to_tflite(model):
    converter = tf.lite.TFLiteConverter.from_keras_model(model)
    converter.optimizations = [tf.lite.Optimize.DEFAULT]
    tflite_model = converter.convert()
    with open('model.tflite', 'wb') as f:
        f.write(tflite_model)
    print("Model converted to TensorFlow Lite format.")

# Assuming 'model' is your trained Keras model
convert_model_to_tflite(model)
```

5.2s

First, the TensorFlow library is imported, setting the stage for the conversion process. A function named `convert_model_to_tflite` is defined to manage the conversion. Within this function, the TensorFlow Lite Converter is instantiated by calling `tf.lite.TFLiteConverter.from_keras_model(model)`, where `model` refers to the pre-trained Keras model. Optimization settings are adjusted with `converter.optimizations = [tf.lite.Optimize.DEFAULT]` to enhance performance. The conversion is executed with `tflite_model = converter.convert()`, producing the TensorFlow Lite model. Finally, the converted model is saved to a file named 'model.tflite' using a `with open('model.tflite', 'wb') as f` statement, ensuring it is written in binary format. A message "Model converted to TensorFlow Lite format." confirms the successful conversion.

2. **Deployment of Model to Edge Device:** Following the successful conversion of the trained Keras model to TensorFlow Lite format, the next crucial step was to transfer the `model.tflite` file to the designated edge device. This step facilitated seamless integration with the hardware infrastructure, ensuring that the optimized model was ready for real-time processing on devices such as Raspberry Pi, NVIDIA Jetson Nano, or Arduino. The transfer process involved securely moving the model file to the edge device's storage, setting the stage for efficient on-device inference and processing.

3. **Development of Edge Device Script:** A custom script was developed to manage the tasks of audio capture, processing, and inference using the TensorFlow Lite model (Figure 3).

```

import numpy as np
import librosa
import tensorflow as tf
import sounddevice as sd

# Load TFLite model and allocate tensors.
interpreter = tf.lite.Interpreter(model_path="model.tflite")
interpreter.allocate_tensors()

# Get input and output tensors.
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

def preprocess_audio(audio_data):
    spectrogram = librosa.feature.melspectrogram(y=audio_data, sr=22050, n_mels=128)
    log_spectrogram = librosa.power_to_db(spectrogram, ref=np.max)
    log_spectrogram = np.expand_dims(log_spectrogram, axis=-1)
    log_spectrogram = np.expand_dims(log_spectrogram, axis=0)
    return log_spectrogram.astype(np.float32)

def run_inference(audio_data):
    input_data = preprocess_audio(audio_data)
    interpreter.set_tensor(input_details[0]['index'], input_data)
    interpreter.invoke()
    output_data = interpreter.get_tensor(output_details[0]['index'])
    return output_data

def audio_callback(indata, frames, time, status):
    if status:
        print(status)
    audio_data = indata[:, 0]
    predictions = run_inference(audio_data)
    print("Predictions:", predictions)

# Set up audio stream
with sd.InputStream(callback=audio_callback, channels=1, samplerate=22050):
    sd.sleep(10000) # Keep the script running for 10 seconds to capture audio

```

Figure 1. Audio Classification Code

This script first loads the TFLite model and allocates the necessary tensors. It includes a preprocessing function to convert raw audio data into a log-scaled mel spectrogram, which serves as input for the model. The script then sets up an inference function to handle the processed audio data, utilizing the TFLite model to generate predictions. Additionally, an audio callback function is implemented to continuously capture audio data in real time, process it, and print the model's predictions. This comprehensive script orchestrates the entire workflow, from capturing and preprocessing audio to running the model inference, thereby enabling real-time detection and classification of earthquake-related sounds.

4.2. Evaluation

The development of our sound classification model for earthquake emergencies is a significant step in disaster response technology. By combining CNN and RNN, we've created a robust model capable of real-time classification of various sound categories. Integrated into Earthquake Emergency Response Robots, this model enhances the situation's awareness,

helping the responders in evaluating the area and in decision-making. Further work will ensure its effectiveness in multiple scenarios, thus saving lives and minimizing the damage.

Throughout the project, we completed the following tasks, demonstrating a comprehensive approach to developing the AI-powered sound classification model:

1. **Gathering and Initial Dataset Processing:** We collected various sound datasets, including UrbanSound8K and ESC-50, and manually added additional sounds from Freesound.org. We then applied normalization, resampling, and transformed the audio samples into spectrograms.
2. **Model Selection and Justification:** After evaluating different models, we selected CNNs and RNNs for their effectiveness in handling audio data and sequence modeling. This decision was supported by extensive research and preliminary testing.
3. **CNN and RNN Model Development:** We developed and trained separate CNN and RNN models to classify structural sounds and human presence. The CNN model utilized multiple convolutional layers, while the RNN model employed LSTM architecture to capture temporal patterns.
4. **Testing and Validation:** We conducted rigorous testing and validation in simulated earthquake scenarios, incorporating diverse noise sources and challenging conditions. The system's performance was evaluated based on its ability to accurately classify environmental sounds.
5. **Final Report and Presentation Preparation:** We compiled the findings, methodologies, and results into a comprehensive final report and prepared presentations to effectively communicate our work to stakeholders.

5. SUMMARY AND CONCLUSION

Summary

In this capstone project, we developed and compared two critical algorithms, the Traveling Salesman Problem (TSP) and A*, to optimize the route planning and pathfinding for an earthquake emergency response robot. The objective was to ensure efficient and accurate navigation to multiple buildings in a disaster-stricken area. Using C++ for implementation, we tested these algorithms with specific coordinates representing buildings and the robot's starting position. Our experimental results demonstrated the strengths and limitations of each algorithm in terms of path length and computation time.

In this capstone project also we leveraged the unique capabilities of CNNs and RNNs to develop specialized models for classifying structural and human sounds. Through iterative training and optimization, we achieved high levels of accuracy in identifying these critical sound categories.

Conclusion

The comparative analysis revealed that while the TSP algorithm is highly effective for overall route optimization with a calculated total path length of 184 units, it requires more computation time (0.00335 seconds). Conversely, the A* algorithm excels in rapid local pathfinding with a shorter path length of 92 units and significantly faster computation time (0.00096 seconds). These findings suggest that a hybrid approach, utilizing TSP for high-level route planning and A* for detailed navigation, could provide an optimal solution for emergency response scenarios. This combination leverages the strengths of both algorithms to enhance the robot's operational efficiency and responsiveness in critical situations.

The successful implementation and testing of these algorithms lay a solid foundation for future improvements and potential real-world applications. Further work may involve refining the algorithms for better scalability, integrating more sophisticated obstacle avoidance mechanisms, and enhancing the robustness of the system to handle real-time dynamic changes in an emergency environment. This project underscores the importance of algorithmic efficiency and accuracy in developing autonomous systems for disaster response and highlights the potential for significant advancements in this field.

The development of a robust sound classification model for earthquake emergency response represents a significant breakthrough in disaster response technology. By combining the strengths of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), we have created a system capable of real-time classification of diverse sound categories, including those indicative of structural damage, human presence, and infrastructure hazards.

Integrating this model into Earthquake Emergency Response Robots (EERRs) significantly enhances their situational awareness, enabling them to better assess disaster environments and inform critical decision-making by rescue teams. The system's ability to accurately identify and locate survivors, as well as assess the extent of damage, has the potential to revolutionize search and rescue operations, ultimately saving lives and minimizing the impact of earthquakes.

ACKNOWLEDGEMENTS

We wish to thank Prof. Gökhan Gelişen, Prof. Ayşe Kavuşturucu, Prof. Fatih Kahraman and Prof. Ece Gelal Soyak their invaluable guidance and help throughout every step of this project. Their expertise and encouragement have been essential in shaping our journey in this project and achieving our goal.

REFERENCES

1. T. G. Conley, and D. W. Galeson, "Nativity and wealth in mid-nineteenth century cities", *Journal of Economic History*, vol. 58, no. 2, pp. 468-493, June 1998. <http://www.jstor.org/stable/2566742>.
2. Wikipedia, "Bee", [Online]. Available: <https://en.wikipedia.org/wiki/Bee>. [Accessed: February 21, 2016].
3. *Evolution of information, communication and computing system*. (n.d.).
4. Singh, S., Liang, Q., Chen, D., & Sheng, L. (2011). Sense through wall human detection using UWB radar. *EURASIP Journal on Wireless Communications and Networking*, 2011(1).
5. Candra, A., Budiman, M. A., & Pohan, R. I. (2021). Application of A-Star algorithm on Pathfinding game. *Journal of Physics. Conference Series*, 1898(1), 012047.
6. Mavigen Digital Agency, www.mavigen.com. (n.d.). *İstanbul itfaiyesi*.
7. Eryilmaz, M., Yigitler, C., Sarp, N., Durusu, M., Levett, J., & Durmus, M. (2007). (129) If I Get Trapped under Debris after an Earthquake, What Should I do while Waiting for Help? *Balkan Military Medical Review*, 22.
8. Tosi, P., Sbarra, P., & De Rubeis, V. (2012). Earthquake sound perception. *Geophysical Research Letters*, 39(24).
9. References for the AI department below
10. McFee, B., Raffel, C., Liang, D., Ellis, D. P. W., McVicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and music signal analysis in python.
11. TensorFlow Developers. (2023). TensorFlow: An end-to-end open-source machine learning platform.
12. Schlüter, J., & Grill, T. (2015). Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks.
13. Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media.
14. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., ... & Adam, H. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference.
15. ESC-50: Dataset for Environmental Sound Classification on kaggle
16. UrbanSound8K, Scream-sound and Crack-sound Datasets on Kaggle

APPENDIX A

Code for packing (and unpacking) an occupancy value [0,1] into an unsigned char.

```
// maximum error = +-0.25% (standard dev. = 0.5/sqrt(12)
// 0% and 100% occupancies are stored exactly.

unsigned char occByte;
if      (occ==0.0) { occByte = 254; } // 0% occupancy stored exactly
else if (occ==1.0) { occByte = 255; } // 100% occupancy stored exactly
else      { occByte = int(occ*200.0); }

// unpack occupancy
float occFlot;
if      (occByte==254) { occFlot = 0.0; }
else if (occByte==255) { occFlot = 1.0; }
else      { occFlot = (occByte+0.5)/200.0; }
```