T.C.
IŞIK UNIVERSTIY
SCHOOL OF GRADUATE STUDIES

MASTER THESIS
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
ELECTRONICS ENGINEERING PROGRAM

FAISAL ALI HASAN BAQUTYAN

DESIGN OF QUADRUPED ROBOT FOR AUTONOMOUS EXPLORATION AND EARTHQUAKE RESCUE

SUPERVISOR
Assoc. Prof. RAMAZAN KÖPRÜ

İSTANBUL, September 2025

T.C.
IŞIK UNIVERSITY
SCHOOL OF GRADUATE STUDIES

MASTER THESIS
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
ELECTRONICS ENGINEERING PROGRAM

FAISAL ALI HASAN BAQUTYAN

(23ELEC5005)

DESIGN OF QUADRUPED ROBOT FOR AUTONOMOUS EXPLORATION AND EARTHQUAKE RESCUE

SUPERVISOR

Assoc. Prof. RAMAZAN KÖPRÜ

İSTANBUL, September 2025

T.C.
IŞIK UNIVERSITY
SCHOOL OF GRADUATE STUDIES

MASTER'S THESIS
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
ELECTRONICS ENGINEERING PROGRAM

FAISAL ALI HASAN BAQUTYAN

(23ELEC5005)

DESIGN OF QUADRUPED ROBOT FOR AUTONOMOUS EXPLORATION AND EARTHQUAKE RESCUE

Date: 17/09/2025

Thesis Supervisor: Assoc. Prof. Ramazan Köprü/ Işık University

Thesis Co-Advisor: Assoc. Prof. Erkin Dinçmen/ Isik University

Jury Members:

Prof. Ümit Güz/ Isik University

Assoc. Prof. Umut Karagüzel/ Yildiz Technical University

Assoc. Prof. Lida Kouhalvandi/ Dogus University

İSTANBUL, September 2025

ÖZET

OTONOM KEŞİF VE DEPREM KURTARMA İÇİN DÖRT AYAKLI ROBOT TASARIMI

Son yıllarda, tekerlekli robotlar için sorun olabilecek zorlu arazilerde yürüme kabiliyetlerine sahip oldukları için dört ayaklı robotlar büyük ilgi görmüştür. Bu da onları afet ortamlarında keşif ve Arama-Kurtarma (SAR) operasyonları için oldukça uygun hale getirmektedir. Bu tez, algılama görevlerini yerine getirirken yürüyebilen, uygun maliyetli, otonom bir dört ayaklı robotun tasarımını sunmaktadır. Araştırmanın iki ana odağı mekanik tasarım ve elektronik tasarımdır. Mekanik yön, yapısal çerçeveyi, çalışma mekanizmasını, ters kinematiği ve özel bir yürüyüş modelini içerir. Diğer yandan, elektronik sistem tasarımı, otonom navigasyon, algılama ve kontrolü sağlamak için bir ESP32 mikrodenetleyici ve ESP32-CAM'i çeşitli sensörler ve aktüatörlerle entegre etmektedir. Yazılım sistemi, ultrasonik sensörler kullanarak gerçek zamanlı engel tespiti, acil durum sesli uyarısı olan bir MQ sensörü kullanarak gaz izleme ve video akışı sunmaktadır. Ayrıca, haritalama, yerelleştirme ve yol planlama dahil olmak üzere gelişmiş otonom keşif için bir 2D RPLIDAR A1 sensörü entegre edilmiştir. Bu yetenekler, görselleştirme için RViz ve gerçekçi simülasyon için Gazebo dahil olmak üzere Robot İşletim Sistemi (ROS) kullanılarak uygulanmış ve test edilmiştir. Bu iş birliği çalışmaları, düşük maliyetli dört ayaklı robotların gerçek dünya acil durum senaryolarında konuşlandırılma olasılığını kanıtlamaktadır. Donanım, yazılım ve simülasyonun entegrasyonu, tasarımı doğrulamakla kalmayıp aynı zamanda arama kurtarma görevlerinde daha fazla özerklik ve sağlamlığın sürekli geliştirilmesi için de olanaklar sunmaktadır.

Anahtar Kelimeler: Dört ayaklı, Otonom, ROS, Haritalama, Yerelleştirme

ABSTRACT

DESIGN OF QUADRUPED ROBOT FOR AUTONOMOUS EXPLORATION AND EARTHQUAKE RESCUE

Recently, quadruped robots have gained high attention because they have the ability to walk through difficult terrain that might be a problem for wheeled robots. This makes them highly suitable for exploration and Search-and-Rescue (SAR) operations in disaster environments. The thesis presents the design of a cost-effective autonomous quadruped robot capable of walking while performing sensing tasks. The research has two main focuses, which are mechanical design and electronics design. The mechanical aspect includes the structural framework, working mechanism, inverse kinematics, and a custom gait pattern. On the other hand, the electronics system design integrates an ESP32 microcontroller and ESP32-CAM, along with several sensors and actuators, to enable autonomous navigation, perception, and control. The software system offers real-time obstacle detection using ultrasonic sensors, gas monitoring using an MQ sensor with an emergency buzzer alert, and video streaming. In addition, a 2D RPLIDAR A1 sensor is integrated for advanced autonomous exploration, including mapping, localization, and path planning. These capabilities are implemented and tested using the Robot Operating System (ROS), including RViz for visualization and Gazebo for realistic simulation. These collaborative efforts prove the possibility of deploying low-cost quadruped robots in real-world emergency scenarios. The integration of hardware, software, and simulation not only validates the design but also opens possibilities for continued development of more autonomy and robustness in SAR missions.

Keywords: Quadruped, Autonomous, ROS, Mapping, Localization

ACKNOWLEDGEMENT

First, I want to sincerely thank Assoc. Prof. Ramazan KOPRU, my supervisor, for the invaluable guidance he gave me throughout my master's program. His deep expertise and and insightful sugestion allowed me to present the best version of my work. I have gained a great knowledge and experience woking under his supervision.

Additionally, I want to thank my co-advisor, Assoc. Prof. Erkin DINCMEN, whose counsel helped me to shape my thesis. I sincerely appreciate the time and effort he invested in looking over my work.

I would like to extend my special gratitude to all faculty and staff members of the Electrical and Electronics department of Isik University, who assisted me during my studies. Wheteher through academic advise or friendly conversation.

Most importantly, I want to thank my family from the bottom of my heart. Especially to my parents, grandfather, and grandmother, thank you for your unconditional love sacrifices, and endless encouragement through every step of my academic path. This achievement would not have been possible without your constant support.

Faisal BAQUTYAN

TABLE OF CONTENTS

PAGE NO:

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS LIST

SAR: Search-and-Rescue

DOF: Degree of Freedom

LIDAR: Light Detection and Ranging

IMU: Inertial Measurement Unit

RGB: Red, Green, and Blue

DC: Direct Current

AC: Alternating Current

PWM: Pulse Width Modulation

BLDC: Brushless DC

SLAM: Simultaneous Localization and Mapping

DRL: Deep Reinforcement Learning

PLA: Polylactic acid

ABS: Acrylonitrile Butadiene Styrene

PETG: Polyethylene terephthalate glycol

GPIO: General Purpose Input/Output

RAM: Random-Access Memory

MHz: Megahertz

KB: Kilobyte

MB: Megabyte

SRAM: Static RAM

PSRAM: Pseudo SRAM

ROS: Robot Operating System

NH3: Ammonia

NOx: Nitrogen Oxides

CO2: Carbon Dioxide

Kg: Kilogram

$: Dollar

N: Newton

mm: Millimetre

g: Gram

cm: Centimetre

v: Voltage

K: kiloohms

LiPo: Lithium Polymer

NiMH: Nickel-Metal Hydride

SBEC: Switching Battery Eliminator Circuit

AMCL: Adaptive Monte Carlo Localization

URDF: Unified Robot Description Format

CHAPTER 1

1. INTRODUCTION

1.1. BACKGROUND OF QUADRUPED ROBOTS

The development of legged robots continues to rise due to their high maneuverability in complex environments and excellent navigation over obstacles compared to other mobile robot types, such as wheeled and tracked robots. They are inspired by animals that use their legs to move fast on different terrains with perfect locomotion and agility. Quadruped robots, which can be defined as four-legged robots that mimic the locomotion of a four-legged animal, have advantages among all legged robot types since they're less complicated than hexapod robots and more stable than biped robots (He et al., 2019). Table 1.1 shows a comparison between quadruped robots and hexapod robots.

Table 1.1 Comparison between Quadruped and Hexapod robot

Feature	Quadruped robot	Hexapod robot
Number of legs	4	6
Stability	Less stable	More stable
Complexity	Simpler	More complex
Energy efficiency	More energy efficient	Less energy efficient
Maneuverability	More agile	Less agile
Redundancy	Less redundancy	High redundancy
Cost	Lower cost	Higher cost

It can be noticed from the comparison that the quadruped robot has an advantage in all criteria compared to the hexapod, except for stability, since the

quadruped stands on 4 legs and the hexapod stands on 6 legs. However, good stability can still be achieved in quadruped robots by applying an advanced control system design.

The quadruped robot is divided into two main types, which are mammal robots and sprawling robots. The mammal robots, as shown in Figure 1.1 a, have their legs below their bodies like dogs or horses. They can withstand more payload, as well as having a good walking speed, which makes them capable of climbing slopes and stairs more easily (Li et al., 2011). On the other hand, sprawling robots, as shown in Figure 1.1 b, have their legs splayed outward like crocodiles or lizards. Their walking gait consists of lateral bending of their body with leg movement (Suzuki et al., 2021). This design is more stable since the body is close to the ground, which enables the robot to navigate tight spaces more effectively.

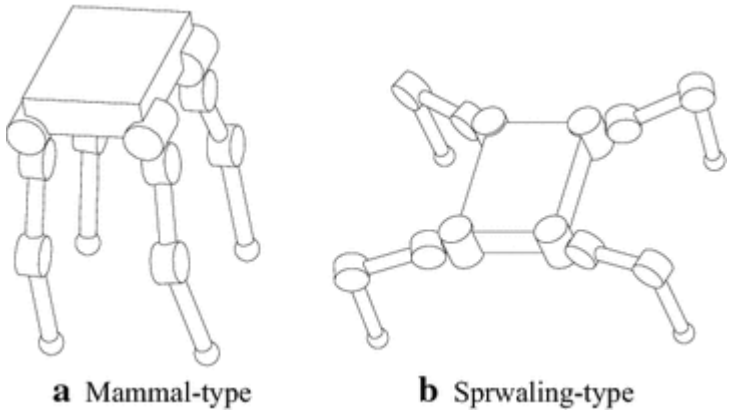


Figure 1.1 a Mammal robot and b Sprawling robot (Kitano et al., 2016)

1.2. IMPORTANCE OF QUADRUPED ROBOTS IN DISASTER RESPONSE

Natural disasters such as earthquakes, storms, and floods, as well as man-made disasters such as attacks and chemical threats, are a serious global problem. The effect of these disasters frequently leads to the death of many people, with many of them were unable to be assisted in time since they were trapped in

inaccessible locations where it is hard for the rescue teams to reach. Also, environments after these disasters are marked by significant debris and buildings that may collapse, which complicate the efforts of rescue teams in their assessments and movements. The use of quadruped robots in these scenarios is a perfect choice to increase the number of victims who can be rescued or helped faster. Such quadruped robots can navigate rough terrain and pass through tight places (Hashimoto et al., 2017).

1.3. PURPOSE OF THE THESIS

Turkey has suffered from destructive earthquakes in recent years, which have been a significant threat to Turkish citizens and residents. Many of them were trapped under rubble in hazardous and unstable environments for days. The urgency of SAR operations requires advanced technological solutions that can navigate complex terrains and detect survivors. However, existing quadruped robots that can be used for earthquake rescue are often expensive, which makes large-scale deployment difficult. There is a crucial need for a cost-effective and autonomous quadruped robot that can help and assist the rescue teams after an earthquake, as well as maintain reliable performance in harsh environments. The main purpose of this thesis is to bridge the gap between affordability and functionality by designing a cost-effective quadruped robot equipped with an embedded system able to explore autonomously and detect survivors after an earthquake. It mainly focuses on reducing costs without giving up essential features such as obstacle avoidance and real-time environmental sensing.

CHAPTER 2

2. GENERAL INFORMATION

2.1. DESIGN PRINCIPLES

2.1.1. Biomimetic Design Principle

Quadruped robots draw inspiration from the locomotion of four-legged animals, enabling them to pass through complex environments with agility, stability, and efficiency. One of the most essential aspects of quadruped mobility is the gait selection, which determines how the quadruped coordinates its legs to achieve efficient movement. By mimicking natural movement patterns, quadruped robots can be taught to adjust to different environments, maintain stability, and reduce energy consumption. Below is some different gait patterns found in nature that are used in quadruped robots.

Walking gait: It can be defined as a slow and stable gait pattern in which two or three legs always remain in contact with the ground. This gait pattern is mostly used in dangerous places in which an accurate foot placement is important to achieve the task (Pongas, Mistry, and Schaal, 2007). In earthquake rescue operations, an autonomous quadruped robot with a walking gait can find the survivors more efficiently, but not faster than the other gait patterns.

Trotting gait: This can be defined as a medium-speed gait in which diagonal pairs of legs move together. For example, the front right and back left legs move together. This gait pattern provides a balance between speed and stability (RunBin et al., 2013). In earthquake rescue operations, an autonomous quadruped robot with a trotting gait can find the survivors faster than a walking pattern, but with less efficiency.

Pacing gait: It can be defined as a high-speed gait in which two legs on the same side of the body move together. Even though this gait pattern offers a

high speed, it may not be stable, and the quadruped will fall (Majithia et al., 2024). In earthquake rescue operations, it is not suitable to use an autonomous quadruped robot with a pacing gait since it is not stable and cannot operate in complex environments.

2.1.2. Mechanical Design Principle

The mechanical design of quadruped robots has an important effect on their mobility and stability. One of the key design considerations is the degree of freedom (DOF), described as the quantity of independent joint variables needed to determine each link's position (Briot & Khalil, 2015). The DOF can determine how flexibly the robot can move its legs, which directly impacts the quadruped's ability to navigate obstacles and optimize energy efficiency

In general, quadruped robots feature 8 DOF, 12 DOF, or 16 DOF, each of these offers a variety of levels in terms of complexity, maneuverability, and energy consumption. The selection of DOF is a trade-off between mechanical complexity and functional performance, which means that a higher DOF could provide great flexibility, but it will increase complexity and the power requirements.

8 DOF: It is considered the simplest quadruped design, with each leg having only two joints, which are the hip joints and the knee joints (Atique et al., 2018). It has basic walking and trotting gaits and lower power consumption. However, it is not suitable for earthquake rescue operations due to its limited flexibility, which makes it less efficient on highly uneven surfaces.

12 DOF: It develops the movement to another level by adding an extra joint per leg (Shi et al., 2021). It has better terrain adaptability than 8 DOF, which allows the quadruped robot to adjust its legs in complex environments. Also, it has smoother and more natural gait patterns that improve locomotion efficiency. Finally, it strikes a balance between cost and functionality, which makes it a perfect choice for earthquake rescue operations.

16 DOF: It provides a high level of flexibility. With four independent joints in each leg, the 16 DOF supplies highly dynamic gaits as well as an

advanced posture adjustment (Bhattacharya et al., 2019). Even though a quadruped robot with 16 DOF has superior stability in earthquake rescue operations, its complex control algorithms and high cost restrict its use.

2.2. EMBEDDED SYSTEMS IN QUADRUPED ROBOTS

An embedded system can be defined as a programmable device that drives some specific set to the system. It connects to sensors that collect data from the environment, has its own computing devices to process and analyze this data, and sends commands to the actuators. Based on this command, the actuators provide an output that performs the task (Dey & Mukherjee, 2016). The main purpose of embedded systems in quadruped robots is to provide real-time control, efficient and reliable data processing, and autonomous decision-making.

2.2.1. Sensors

It is essential for the quadruped robot to recognize the environment around it. This can be achieved by using sensors to collect data. For instance, we can use sensors for the path planning of the quadruped robot, which can realize autonomous walking (Liu et al., 2020). Generally, the sensors is divided into two main groups, which are internal sensors and external sensors. The quadruped robot's internal sensors measure its own characteristics, including joint angles and motor speed, while its external sensors are used to collect information from the surroundings, such as distance and light intensity (Ma, 2020). Here are some sensors that are used in quadruped robots.

LIDAR: It stands for light detection and ranging and is one of the most important sensors in quadruped robots. The main purpose of using LIDAR is to facilitate mapping and obstacle avoidance. LIDAR system is capable of gathering spatial information in three varieties, which are one-dimensional (1D), two-dimensional (2D), and three-dimensional (3D), and this spatial information is important for creating an accurate 3D map of the environment (Raj et al., 2020).

RGB Depth Camera: Detecting and tracking humans is an essential feature in quadruped robots. One of the most used sensors in this case is an RGB Depth camera due to its availability, affordability, and efficiency (Liciotti et al., 2017). It consists of two main sensors, which are the RGB sensor and the Depth sensor. The RGB sensor captures standard color images, which can help with object classification. On the other hand, the Depth sensor measures the distance of objects, which allows the quadruped robot to detect obstacles.

IMU: One of the problems that the quadruped robot can face is stability on a complex floor; the movement of the quadruped can be late, and the center of gravity is not balanced. An IMU, which stands for Inertial Measurement Unit, is a crucial sensor for maintaining balance, estimating orientation, and improving locomotion stability. It consists of two sensors, which are an Accelerometer and a Gyroscope. To make the quadruped balanced on uneven flooring, the IMU will supply a proportional controller and an input of the quadruped's tilting degree that can be processed in stabilization algorithms. (Prayogo et al., 2018).

Force sensors: They are used in quadruped robots to measure the contact forces between the feet and the ground. These sensors help to improve balance and locomotion and ensure proper weight distribution. They can be used in a running quadruped robot (Ananthanarayanan et al., 2012).

2.2.2. Actuator

The actuators enable the robot to move freely in different directions as well as determine its performance, such as speed, torque, and jumping agility (Hussain et al., 2021). Generally, actuators are divided to 3 main types, which are electric actuators, hydraulic actuators (Suzumori & Faudzi, 2018), and pneumatic actuators (Kim et al., 2021). Electric actuators convert electrical energy into either mechanical torque or linear force. Hydraulic actuators use pressurized fluid to generate force. Pneumatic actuators use compressed air to move. Here are some actuators that are used in quadruped robots.

Servo motor: A type of motor that turns the control signal to rotational angular displacement or angular velocity. It includes direct current motors (DC) and alternating current motors (AC) (Firoozian, 2014). Servos are used in quadruped robots due to their precise position control, lightweight, cost-effectiveness, and ease of control since they work with pulse width modulation (PWM) signals.

BLDC motor: It stands for brushless DC motors, which is widely used in high-performance quadruped robots due to their high torque-to-weight ratio, high speed, and low power consumption. A permanent magnet serves as the brushless DC motor's rotor, and it is powered by the magnetic force of the stator's winding circuit (Visconti & Primiceri, 2017). The main disadvantage of BLDC motors is their expensive price.

Series elastic actuators: They are advanced actuators that have an elastic element between the motor and the load. These actuators provide many benefits in force control of quadruped in an uneven environment, such as high force, low friction, shock absorption, and good force control bandwidth (Pratt & Krupp, 2004). They are a perfect option for quadruped robots, but their expensive price and complexity limit their use.

2.2.3. Microcontrollers

Microcontrollers are a part of embedded systems in quadruped robots, which can be defined as an integrated chip that contains a programmable input and output ports named general-purpose input/output (GPIO), processor, and memory. The real power of microcontrollers is their ability to perform a real-time task (Dey & Mukherjee, 2016). They are essential in quadruped robots since they can handle real-time control tasks such as motor actuation, sensor data processing, and communication. The quadruped robots require precise and synchronized movement for walking and balancing, which can be provided by the microcontroller. Here are some important microcontrollers used in quadruped robots.

ESP32 microcontrollers: ESP32 is a high-powered System-on-chip (SoC) microcontroller with integrated WIFI, dual Bluetooth version 4.2, and a variety of peripherals (Babiuch et al., 2019). It is a good choice for quadruped robots due to their wireless communication and remote-control capabilities that allow the user to control the quadruped robot through a smartphone or PC. It is cost-effective as well.

STM32 microcontrollers: STM32 uses an ARM Cortex-M3 core and offers a framework for creating a wide variety of embedded systems, ranging from basic dongles that run on batteries to intricate real-time systems. such as quadruped robots (Brown, 2012). Even though STM32 does not have a wireless capability, it is used a lot in quadruped robots due to its faster real-time processing for advanced gaits.

2.2.4. Microprocessors

A microprocessor is a type of digital electronics device that has several data and address buses, registers, a control unit, and an arithmetic logic unit (ALU). It uses a set of instructions in its ALU that are managed by a timer clock produced by the microprocessor's control unit. (Dey & Mukherjee, 2016). In quadruped robots, microprocessors are essential for high-level computing tasks such as AI-based navigation and object recognition, complex motion planning, Simultaneous Localization and Mapping (SLAM), and high-resolution sensor data processing. Here is an example of a microprocessor that is widely used in quadruped robots.

Raspberry Pi: It can be considered a smaller version of a modern-day computer, capable of accomplishing tasks effectively. It can enable the installation of open-source operating systems and apps and support different programming languages such as Python, C, and C++. It uses a Broadcom BCM2711 and Quad-core Cortex-A72 processor. It supports Bluetooth and Wi-Fi (Ghael et al., 2021). An essential reason why Raspberry Pi is used in quadruped robots is that it can handle advanced image processing as well as Robot Operating System (ROS) control.

2.3. LITERATURE REVIEW

2.3.1. Existing Quadruped Robots Used in Disaster Response

In recent years, the use of quadruped robots in the field of SAR has largely increased efficiency and safety. One of the most advanced quadruped robots used in this field is the Jueying X20 (DeepRobotics Co., Ltd, 2022), which is a quadruped robot designed and developed by DEEP Robotics to navigate complex and hazardous environments.

The Jueying X20 is designed to be used in SAR operations such as post-earthquake landscapes and inside vulnerable buildings. It has the ability to operate in toxic and high-density smoke environments. Its robust design allows the quadruped robot to avoid obstacles up to 20 cm high, climb on a 35-degree slope, and operate in extreme weather conditions such as heavy rain and sandstorms very well. The Jueying X20 is integrated with advanced sensors such as high-resolution panoramic cameras and LIDAR, which enables it to have long-distance communication, track the heat source, and detect hazardous gases (Robotics 24/7 staff, 2022).

Figure 2.1 Jueying X20 in harsh conditions (Robotics 24/7 staff, 2022)

Another quadruped robot used in SAR operations is called Spot from Boston Dynamics, which was released in 2020. It has a speed of 5.76 km/h, its total weight is around 30 kg and can sustain up to 14 kg. All joints are electrically driven and battery powered. The robot uses 3D vision (SLAM), which stands for simultaneous localization and mapping, giving depth information to avoid obstacles. The robot has two modes: it can operate remotely by a human or autonomously. It can climb stairs and balance itself (Boston Dynamics, 2020).

In April 2023, there was a parking garage which has been collapsed in Manhattan, which led to the death of the garage's manager as well as injuring five others. Later, the New York City firefighters could take the victims out of the debris, but the department stated that the place was not stable enough to search for any remaining people. That's why they decided to send a Spot quadruped robot to the place and record a video that was streamed to the fire officials in real time (Boston Dynamics, 2023). This situation proved the efficiency of using a Spot quadruped robot in SAR operations.

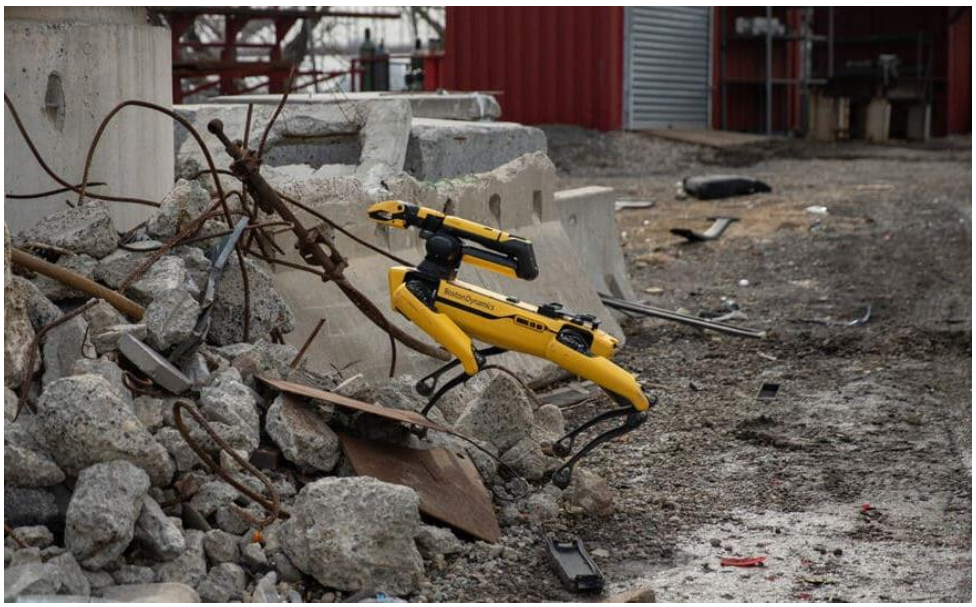


Figure 2.2 Spot in rubble (Boston Dynamics, 2023)

Another example of a quadruped robot made for search and rescue missions is the ANYmal quadruped robot, created by ANYbotics. It can sense its surroundings, create maps, and locate itself since it has advanced sensors, which include laser sensors and cameras. ANYmal is designed to be easily maintained and handled by anyone thanks to its modular design. The quadruped robot's unique ability to run and jump can be useful for avoiding large obstacles. ANYmal was primarily designed to function autonomously for longer than two hours in challenging conditions. (Hutter et al., 2017). ANYmal's huge capabilities have been used in real-world scenarios by the Defense Advanced Research Projects Agency (DARPA) in subterranean missions due to its high endurance and ability to traverse challenging environments (Tranzatto et al., 2022).

Figure 2.3 ANYmal in rubble (Robotsguide, n.d.)

To conclude, the development of quadruped robots for search and rescue operations indicates their growing role in hazardous environments. The Jueying quadruped robot from Deep Robotics focuses on the importance of hazard

detection since it is developed to operate in disaster-stricken areas and identify environmental risks at the same time. Inspired by this, my quadruped robot uses a gas sensor to detect harmful gases, which increases its functionality in earthquake rescue missions. The spot from Boston Dynamics has featured autonomous exploration and video streaming in SAR applications, which allows the rescue team to assess dangerous areas without direct exposure. Motivated by this, my quadruped robot also features video streaming and autonomous navigation, providing the rescue team with live visual feedback. Finally, the ANYmal from ANYbotics highlights the importance of a long-time operation in SAR applications, which ensures continuous mission execution. Also, it is built in a modular way. Inspired by this, my quadruped robot concentrated on efficient power management in order to get the longest operation time, and it is designed modularity.

2.4.2 Autonomous Exploration Algorithms

For a quadruped robot to autonomously explore in SAR missions, it must be able to build a map of the environment, localize its position on the map, and make decisions about where to move next. These features are possible using two main algorithmic systems which are SLAM and autonomous navigation strategies. Both works together to enable the quadruped robot to explore unfamiliar environments, avoid obstacles, and search for the survivors after the earthquake.

One of the most robust SLAM approaches for quadruped robots is the LIDAR-IMU tightly coupled SLAM, which consists of laser scan data along with inertial measurements to accurately estimate the quadruped pose in space (Zhou et al, 2024). A graph-optimization-based SLAM method was developed specifically for quadruped robots to reduce trajectory estimation error, and the proposed algorithm is shown in Figure 2.4.

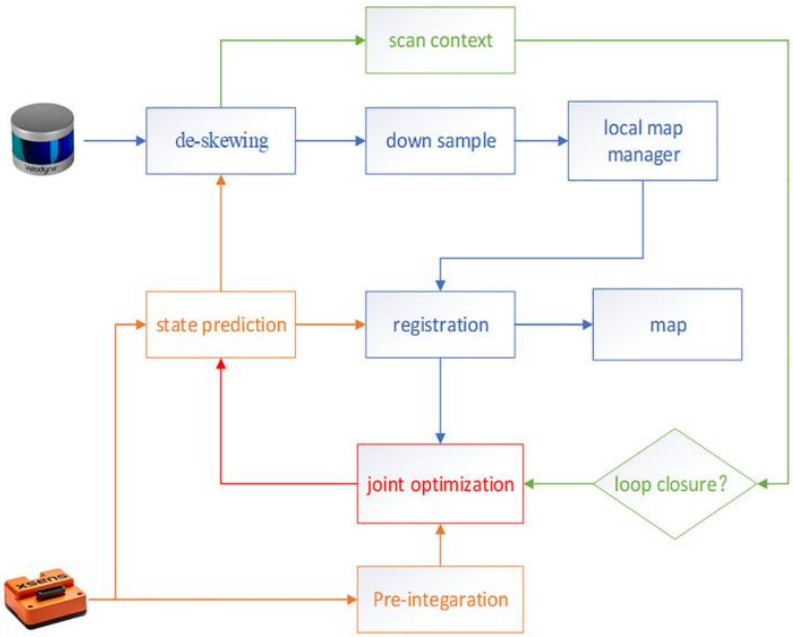


Figure 2.4 Framework design of graph-optimization-based SLAM (Zhou et al, 2024)

Another good method is visual-inertial SLAM, which combines camera data with IMU readings to create a real-time map. Chen and Dellaert introduced "A1 SLAM", which is a SLAM algorithm designed for Unitree A1 robot using ROS2, in which a visual-inertial odometry enables the quadruped robot to operate in fast, and dynamic movements without losing localization (Chen and Dellaert, 2022).

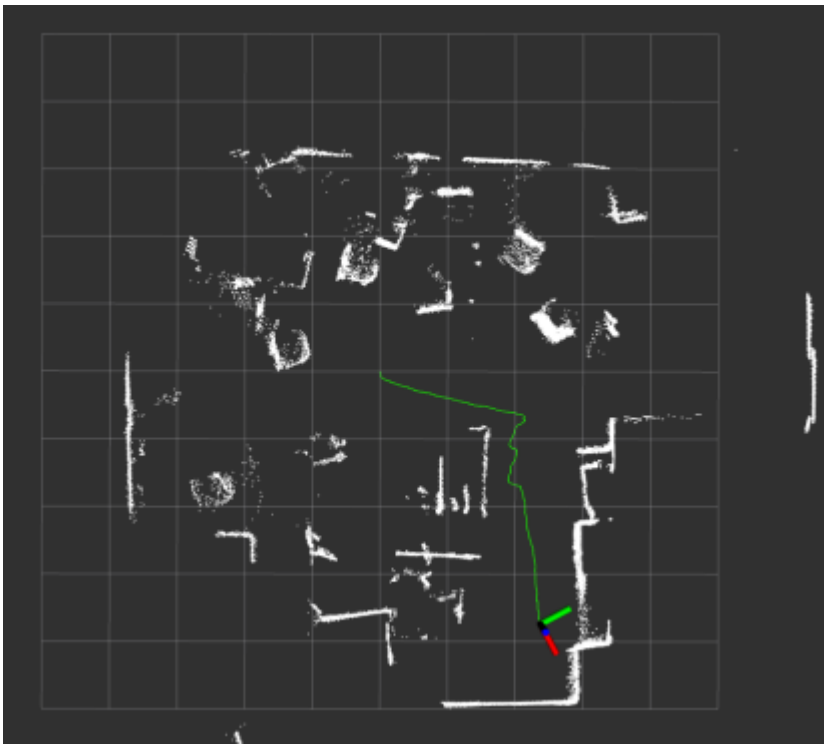


Figure 2.5 The resulting map produced from the A1 SLAM algorithm (Chen and Dellaert, 2022)

This approach is beneficial for search missions in collapsed buildings where visual landmarks are usually the only available references.

On the other hand, navigation strategies are essential in determining how a quadruped robot explores an unknown environment. One of the most widely used strategies is called frontier-based exploration, in which the quadruped robot frontiers exist between known and unknown areas. This strategy was effectively implemented by Zhang et al (2020) who proposed a graph-search-based frontier exploration algorithm using a data called “frontier-graph”. This method allowed the quadruped robot to prioritize unexplored zones while considering a safety path. The proposed algorithm is seen in Figure 2.6.

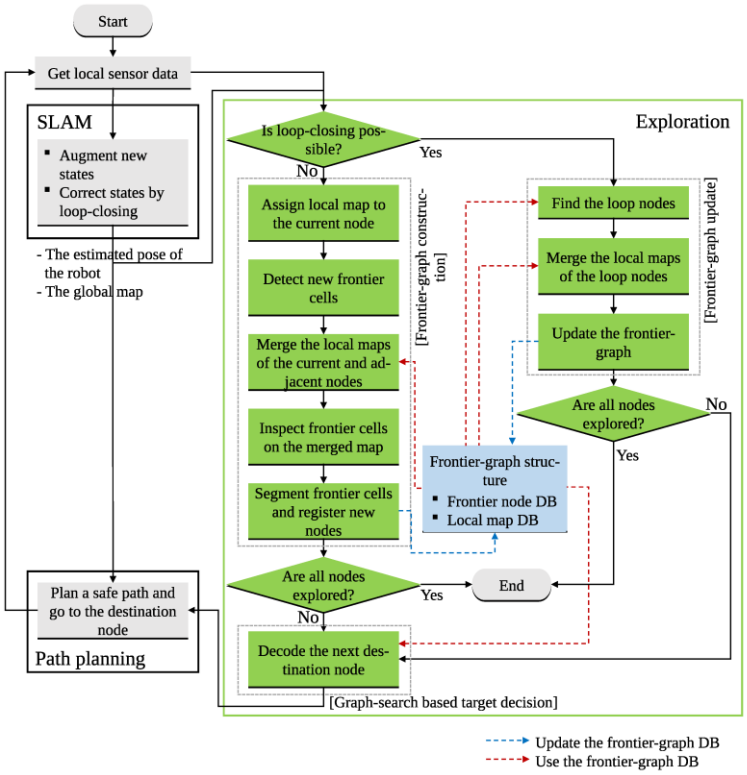


Figure 2.6 Proposed algorithm using graph-search-based frontier exploration (Zhang et al 2020)

Recently, Deep Reinforcement Learning (DRL) can be used for autonomous navigation. These systems allow the quadruped robots to learn exploration policies from simulated environments and generalize them to the real world. DRL-trained agents learn to avoid obstacles, find passageways, and optimize movement based on trial and error (Lee and Yusuf, 2020).

To conclude, the integration of advanced SLAM algorithms and autonomous navigation strategies enables the quadruped to perform sophisticated exploration tasks in unknown environments. Inspired by these approaches, my quadruped robot features SLAM localization and autonomous navigation, which allows the quadruped robot to build a map and make safe exploration.

CHAPTER 3

3. THESIS STUDY

3.1. REAEARCH METHODOLOGY

To design a cost-effective quadruped robot for autonomous exploration and earthquake rescue, a structured process is required to ensure efficiency, reliability, and affordability. In this section we will expand on the processes used in the thesis, starting from defining primary design requirements to simulation validation using ROS.

The process began clearly defining the quadruped's functional and performance requirements. The key criteria were low cost compared to the other quadrupeds in the market, lightweight construction, and navigating complex environments. Also, the quadruped was expected to operate autonomously, detect obstacles, sense hazardous gases, and detect survivors during video streaming.

After defining the requirements, an extensive review of an open-source quadruped platform was conducted. A proper base design was selected and modified to meet the specific demands of search and rescue operations. The modification focused on optimizing the mechanical structure as well as the mobility strategy using custom gait patterns. At the same time, the electronics system was chosen with a strong emphasis on affordability and functionality. Detailed discussions of component selection and system integration are provided in section 3.3 Electronics system design.

To validate the full autonomous behavior of the quadruped, simulation testing was conducted using the Robot Operating System (ROS). A complete 3D model of the quadruped robot was created using URDF and simulated in Gazebo supported by realistic physics and terrain interaction. The robot's sensors such as LiDAR and camera were simulated as well. LiDAR functionality was utilized

for more advanced autonomous exploration activity, and SLAM-based mapping, localization, and path planning, while the camera functionality was used for video streaming. The results were visualized and validated using RViz, allowing for an accurate representation of the environment and success of the navigation task.

In summary, this methodology shows a great approach for designing a functional and affordable quadruped. It uses open-source tools, real-world sensing concepts, ROS-based simulation, and mapping algorithms to achieve a robust prototype suitable for future improvement and deployment in SAR missions.

3.2. MECHANICAL DESIGN

The mobility and stability of a quadruped robot in SAR activities, particularly following an earthquake, are significantly influenced by its mechanical design. Since the quadruped robot is meant for SAR missions, the mechanical structure must be lightweight, able to walk and balance on complex environments, and detect and avoid obstacles.

In addition to enabling motion, the mechanical framework of the quadruped robot supports and carries the electronic components. This structural integrity is essential for keeping them safe.

The initial mechanical design of the quadruped robot was inspired by an open-source Kangal robot designed by Baris ALP (Baris ALP, 2021). Later, an original new design using Autodesk Inventor Professional software was designed to provide better support for integration of sensors and to specifically address the urgent needs for autonomous operation during earthquake rescue.

This section explains the mechanical aspects of the quadruped robot including structural framework design, working mechanism, inverse kinematics and gait pattern which enable coordinated and stable movement on rough terrain.

3.2.1 Structural Framework Design

Nowadays, the open-source quadruped robots have been increasing around the internet, which allowed us to explore different structural framework designs, and the one that inspired me was the Kangal robot designed by Baris ALP. Based on his design, my quadruped robot was designed which has extra features to make it suitable for earthquake rescue, as seen in Figure 3.1.

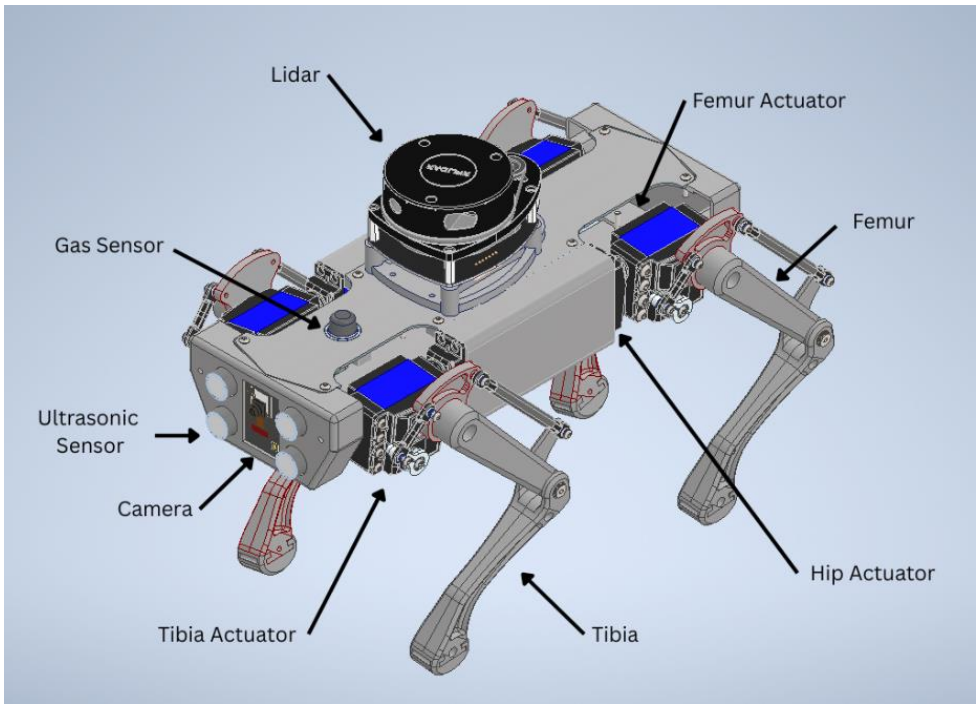


Figure 3.1 The designed Quadruped robot

The quadruped robot contains 12 DOFs, so that three rotatable joints in each leg can be achieved. The three basic bones that form one leg are the hip bone, the femur bone, and the tibia bone. 12 servos are utilized in this design (three per leg) for movement control of the links and joints.

The structural framework of the quadruped robot is created using a 3D-printed part, which perfectly meets the criteria of designing a lightweight and cost-effective quadruped robot.

To select the most appropriate material for 3D printing, a Pugh matrix was used in Table 3.1 to evaluate multiple materials based on weight, strength, cost, availability, and ease of printing. Three candidate materials were considered which are PLA, ABS, and PETG.

Table 3.1 Pugh matrix for selecting material

Criteria	Weight	PLA	ABS	PETG
Weight (Lightness)	3	+1	+1	0
Mechanical Strength	4	+2	0	+1
Cost	3	+1	+2	0
Availability	2	+1	+1	+1
Ease of 3D printing	2	+2	0	+1
Total score		20	11	8

The results of Pugh matrix clearly favor PLA since it has the highest total score. Therefore, it was selected as the most suitable material for the quadruped's structural framework.

A gas sensor is placed on the top of the quadruped robot for several reasons. It allows the sensor to detect rising gases efficiently. Also, if the sensor is placed at the ground level, it may be exposed to dust or debris, which may affect the sensor reading. That's why the top of the quadruped robot is the perfect place for the gas sensor. Also, a Lidar is placed at the top of the quadruped robot to enable an advanced autonomous exploration.

The ultrasonic sensors are placed on the front of the quadruped robot to ensure they have a clear view of the object in the robot's path, as well as provide immediate distance measurement, which allows quick obstacle avoidance decisions.

The camera is placed between the two ultrasonic sensors on the front of the quadruped robot to stream a video, which shows what the quadruped robot sees that can help in detecting humans and scanning the environment.

The structural framework of the quadruped robot follows the modular design principle that subdivides the structural framework into smaller parts in which we can change any parts if they are broken instead of changing the whole structural framework. As a result, the maintenance cost is minimized.

3.2.2 Working Mechanism

The working mechanism of the quadruped robot shows how the quadruped robot moves and maintains stability. Efficient working mechanisms are important for exploring complex environments; that's why the design of the leg structure, DOF, joint actuation, and gait patterns are crucial to ensure smooth operation as well as stable movement. Comprehensive research was conducted into existing open-source quadruped mechanisms to achieve reliable quadruped motion. The main purpose was to identify a working mechanism that provides a good balance and could be applied to the quadruped design. After evaluating multiple options, the Watt six-bar linkage mechanism implemented in the Iron Dog mini was selected (Rahman et al., 2023). In this working mechanism, the servos are placed as shown in Figure 3.2.

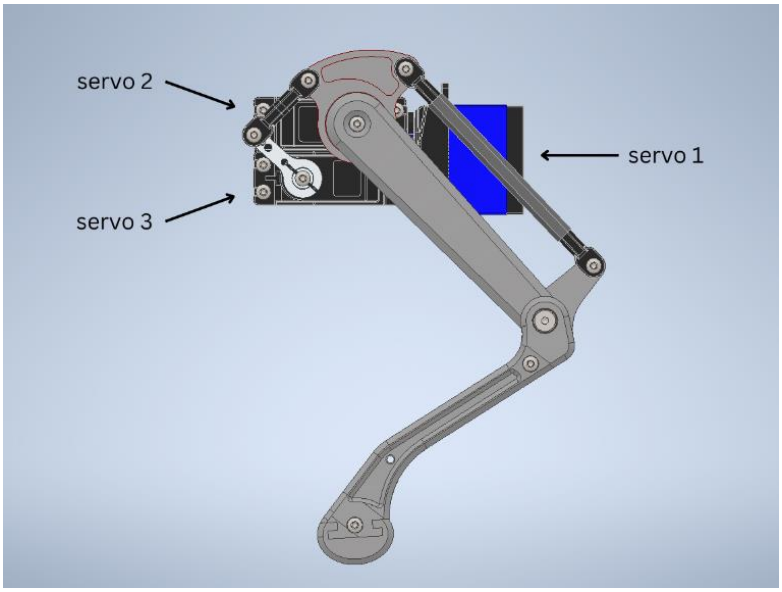


Figure 3.2 Single leg configuration

Servo 1 is responsible for controlling the joint angle of the coxa. Here, the servo is directly connected to the hip joint. Servo 2 is responsible for controlling the femur. Here, the servo motor is directly connected to the femur joint. Finally, servo 3 is responsible for controlling the tibia. Here, the servo motor is not directly connected to the tibia joint, but it uses the Watt six-bar linkage mechanism to control the tibia joint as seen in Figure 3.3.

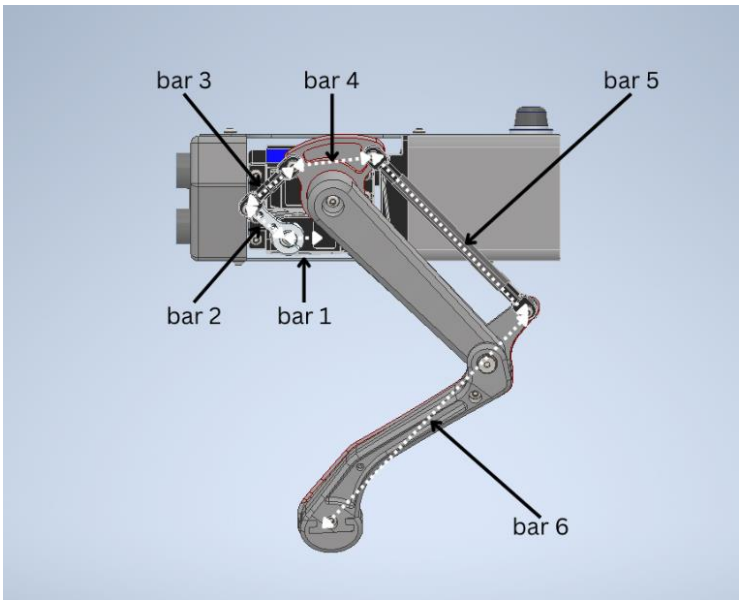


Figure 3.3 Watt six-bar mechanism

3.2.3 Inverse Kinematics

A fundamental concept in quadruped robots is inverse kinematics, which describes the joint angles needed to position the quadruped's legs. For stable movement, seamless gait transitions, and accurate foot placement, inverse kinematics is crucial. By calculating the necessary joint angles using inverse kinematics, we can easily specify the exact servo angles needed to move the quadruped robot. The inverse kinematics was derived from the Iron Dog mini (Rahman et al., 2023). First, based on Figure 3.4, the angle θ1 for the coxa is calculated.

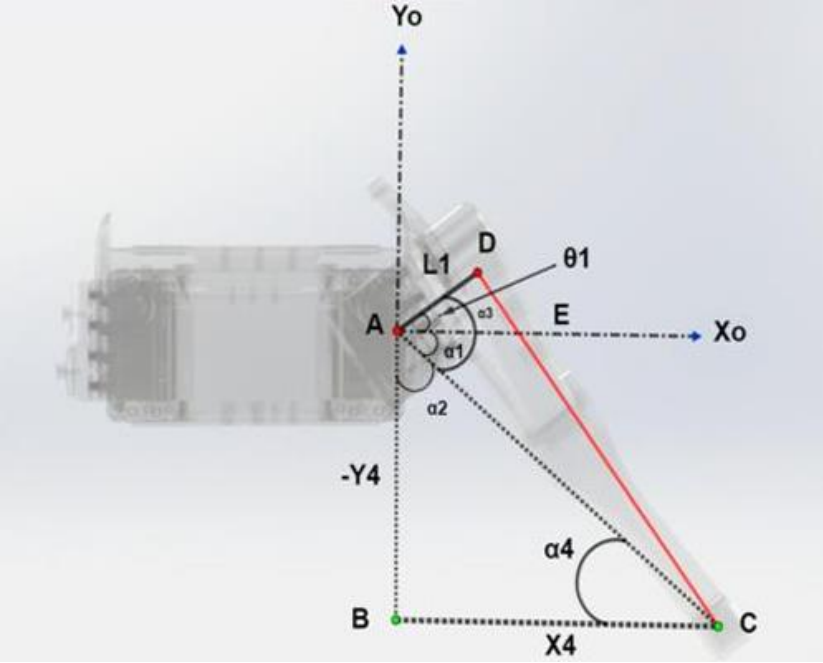


Figure 3.4 Front view of the robot to determine θ1 (Rahman et al., 2023)

$$\theta 1 = \alpha 3 - \alpha 1 \tag{3.1}$$

$$in\ \Delta\ ADC,\ \alpha 3 = \arctan\left(\frac{\sqrt{x4^2 + y4^2 - L1^2}}{L1}\right) \tag{3.2}$$

$$in\ \square\ ABCE\ \ \alpha 1 + \alpha 2 = 90 \rightarrow \alpha 1 = 90 - \alpha 2 \tag{3.3}$$

$$in\ \Delta\ ABC\ \ \alpha 2 + \alpha 4 + \angle B = 180\circ \tag{3.4}$$

$$\alpha 2 + \alpha 4 = 180 - 90\ [\angle B = 90\circ\ ],\ \ \alpha 2 = 90\circ - \alpha 4 \tag{3.5}$$

$$\alpha 1 = 90\circ - (90\circ - \alpha 4) \rightarrow \alpha 1 = \alpha 4 \tag{3.6}$$

$$in\ ABC,\ \alpha 4 = \arctan\left(\frac{-y4}{x4}\right) \tag{3.7}$$

$$\theta 1 = \alpha 3 - \alpha 1 \tag{3.8}$$

$$\theta 1 = \arctan\left(\frac{\sqrt{x4^2 + y4^2 - L1^2}}{L1}\right) - \arctan\left(\frac{-y4}{x4}\right) \tag{3.9}$$

Next, based on Figure 3.5, the angle θ2 for the femur is calculated. This calculation takes into account the relative positioning of the joints to achieve the correct orientation of the leg

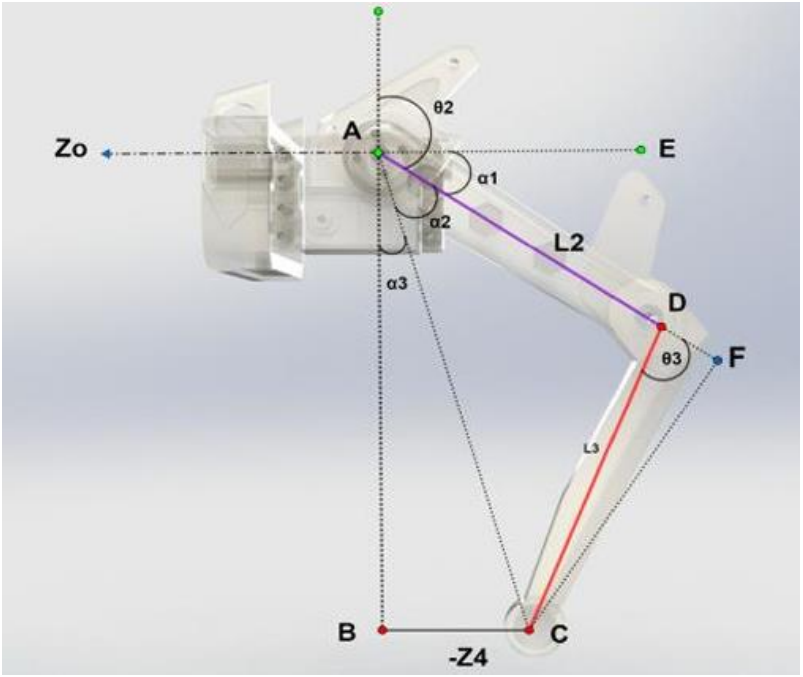


Figure 3.5 Left view of the robot to determine θ2 (Rahman et al., 2023)

$$\theta 2 = -90 + \alpha 1 \tag{3.10}$$

$$ABFE \;\; \alpha 1 + \alpha 2 + \alpha 3 = 90 \;\rightarrow \alpha 1 = 90 - \alpha 2 - \alpha 3 \tag{3.11}$$

$$\Delta ABC \;\; \alpha 3 = \arctan\left(\frac{-z4}{\sqrt{x4^2 + y4^2 - L1^2}}\right) \tag{3.12}$$

$$\Delta ACF \;\; \alpha 2 = \arctan\left(\frac{CF}{AF}\right) \tag{3.13}$$

$$\Delta CDF \;\; sin\theta 3 = \frac{CF}{CD} \tag{3.14}$$

$$CF = CDsin\theta 3 = L3sin\theta 3 \tag{3.15}$$

$$cos\theta 3 = \frac{DF}{CD} \tag{3.16}$$

$$DF = CDcos\theta 3 = L3cos\theta 3 \tag{3.17}$$

$$AF = AD + DF = L2 + L3cos\theta 3 \tag{3.18}$$

$$\alpha 2 = \arctan\left(\frac{L3cos\theta 3}{L2 + L3cos\theta 3}\right) \tag{3.19}$$

$$\alpha 1 = 90 - \arctan\left(\frac{L3cos\theta 3}{L2 + L3cos\theta 3}\right) - \arctan\left(\frac{-z4}{\sqrt{x4^2 + y4^2 - L1^2}}\right) \tag{3.20}$$

$$\theta 2 = -\ \arctan\left(\frac{L3cos\theta 3}{L2 + L3cos\theta 3}\right) - \arctan\left(\frac{-z4}{\sqrt{x4^2 + y4^2 - L1^2}}\right) \tag{3.21}$$

Finally, based on Figure 3.6, the angle θ3 is calculated by applying the trigonometric relations derived from the leg's geometry. This step provides the final joint value needed to fully define the leg configuration.

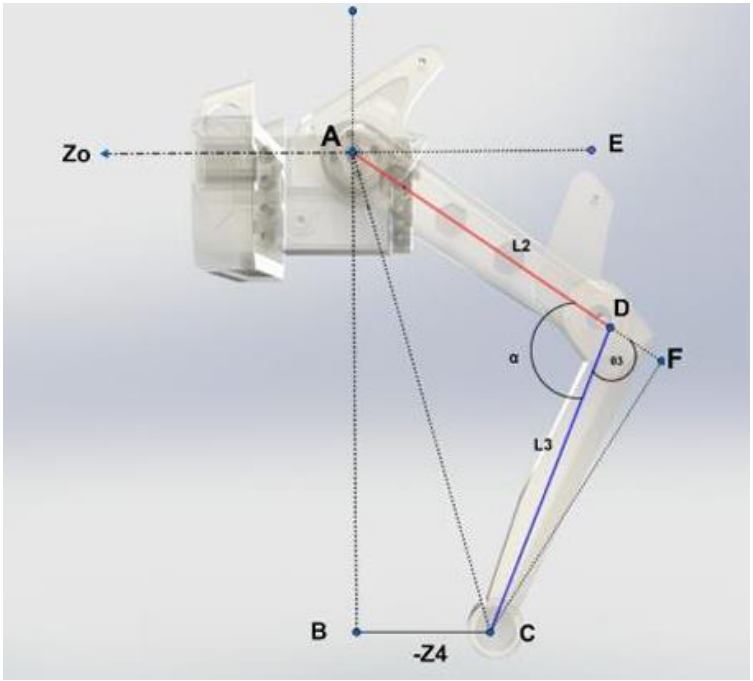


Figure 3.6 Left view of the robot to determine θ3 (Rahman et al., 2023)

$$\theta 3 = 180 - \alpha \tag{3.22}$$

$$\Delta ABD,\ AC^2 = AD^2 + CD^2 - 2.AD.CDcos(\alpha) \tag{3.23}$$

$$\alpha = \arccos\left(\frac{AD^2 + CD^2 - AC^2}{2.AD.CD}\right) \tag{3.24}$$

$$\Delta ABC,\ AC^2 = AB^2 + BC^2 \tag{3.25}$$

$$AC^2 = \left(\sqrt{x4^2 + y4^2 + L1^2}\right)^2 + (-z4)^2 \tag{3.26}$$

$$AC^2 = x4^2 + y4^2 + L1^2 + z4^2 \tag{3.27}$$

$$\alpha = \arccos\left(\frac{L2^2 + L3^2 - x4^2 - y4^2 - L1^2 - z4^2}{2.L2.L3}\right) \tag{3.28}$$

$$[AD = L2,\ CD = L3] \tag{3.29}$$

$$\theta 3 = 180 - \arccos\left(\frac{L2^2 + L3^2 - x4^2 - y4^2 - L1^2 - z4^2}{2.L2.L3}\right) \tag{3.30}$$

3.2.4 Gait Pattern

To achieve a quadruped robot to walk efficiently and smoothly, one needs to consider the synchronization of legs to move. This synchronization is what we refer to as the gait pattern. Similar to animals employing different gaits based on the speed and ground like walking, trotting, and pacing, quadruped robots can be programmed to have several gait patterns based on the task

In this project, the trot gait was implemented, which is one of the most balanced and stable gaits for quadruped locomotion. Trot gait is generally explained in “2.2.1 Biomimetic design principles” which is a symmetrical gait where both the pairs of diagonal legs move together.

For instance, the front-right leg moves with the rear-left leg, and the front-left leg moves simultaneously with the rear-right leg.

Each leg consists of two consecutive phases:

- The stance phase, in which the leg is on the ground supports the robot's weight
- The swing phase, where the leg is lifted and moved forward for the next step

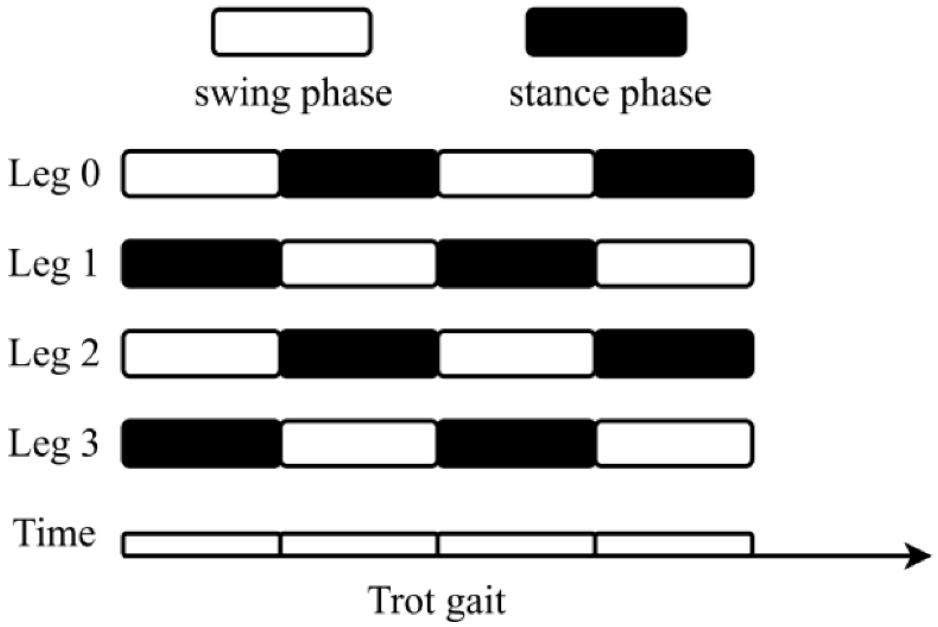


Figure 3.7 Stance and Swing phase of trot gait (Meng et al., 2023)

As shown in Figure 3.7, Leg 0 and Leg2, and similarly Leg 1 and Leg 3, move synchronously in pairs. If one diagonal pair is in the swing phase, the other will be in the stance phase. This alternating gait pattern allows the quadruped to maintain dynamic balance during its movement cycle. The black bars represent the stance phase, and the white bars represent the swing phase. The timing is carefully coordinated, so that the quadruped never has any fewer than two feet on the ground.

3.3. ELECTRONICS SYSTEM DESIGN

The electronics system design of the quadruped robot is an essential approach to achieve autonomous movement, perception, and control. This system consists of different electronics components such as computing devices, sensors, actuators, and a power management system to guarantee efficient performance in exploration and earthquake rescue.

One of the main purposes of the electronics system is the smooth interaction between sensors and actuators that allows the quadruped robot to sense its environment and respond accordingly. Different types of sensors provide real-time feedback on environmental conditions as well as obstacles to ensure adaptive movement and obstacle avoidance. On the other hand, actuators are responsible for locomotion and require precise control to achieve stability.

Power management is important in design. The quadruped robot has a battery-powered setup, that's why it must make sure that each component receives a proper voltage and current to work properly in changing environments. Otherwise, the system might be damaged.

This section presents detailed information about the electronics system, including component selection, power management, and schematic design. Each subsection explores the role of these elements in enabling the quadruped to operate reliably and efficiently.

3.3.1. Component Selection

The selection of electronics components helps in determining the overall performance, reliability, and efficiency of the quadruped robot. That's why each component must be carefully chosen based on factors such as functionality, cost, and power consumption. The component must provide stable and efficient operation in a dynamic and potentially hazardous environment since the quadruped is used for autonomous exploration and earthquake rescue.

Before diving to the component selection, a block diagram of the quadruped robot must be designed to choose the component based on it. The designed block diagram is seen in Figure 3.8.

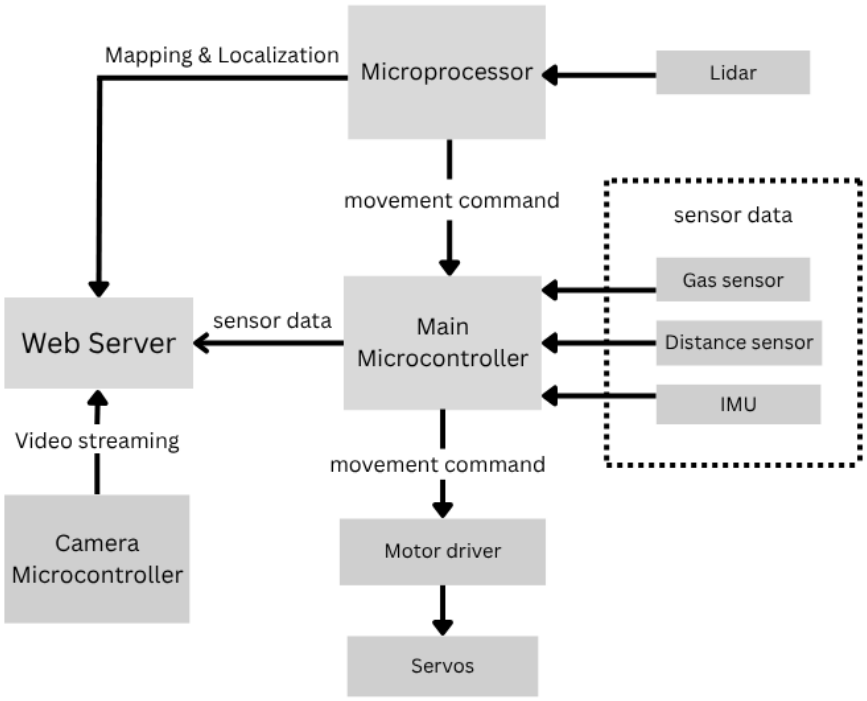


Figure 3.8 Block diagram of the quadruped robot.

3.3.1.1. Computing devices

The computing units of the quadruped robot were selected to include both microcontrollers and a microprocessor to effectively handle various tasks such as motor control, sensor data processing, communication, and autonomous exploration. The microcontroller was specified to handle low-level control tasks, while the microprocessor was specified to handle high-level tasks such as SLAM, and path planning

Microcontrollers: The main microcontroller is used for handling motor control, sensor data processing, and communication. The chosen microcontroller must provide sufficient processing power, connectivity options, and energy efficiency to support real-time operations. After

comprehensive research related to microcontrollers used in quadruped robots, three widely used families of microcontrollers were noticed, which are Arduino, esp32, and stm32. Three good, small-sized microcontroller development boards from each family were selected and compared to choose the best for the designed quadruped robot.

STM32F411 Black Pill: It was chosen to represent the STM32 family due to its small size, high-performance ARM Cortex M4 processor, and low power consumption, as well as its support of Arduino, MicroPython, and C programming (Mouser Electronics, n.d.). It provides a good balance between computational power and efficiency, which makes it suitable for embedded control applications. However, its main problem is that it lacks built-in Wireless connectivity. More information is seen in Table 3.2.

Arduino Nano: It was selected to represent the Arduino family due to its small size, ease of use, and strong community support. It is widely used in robotics because of its simplicity and compatibility with different sensors and actuators. However, its main problems are that it has only an 8-bit processor and lacks built-in wireless connectivity (Arduino, n.d.). More information is seen in Table 3.2.

ESP32 DevKitC: It was preferred to represent the ESP32 family due to its high processing power, built-in Wi-Fi, and Bluetooth (Espressif, n.d.). It provides an excellent balance between performance and connectivity, which makes it suitable for real-time control and wireless communication. Table 3.2 provides more information.

Table 3.2 Comparation between three microcontrollers.

Feature	STM32F411 Black Pill	Arduino Nano	ESP32 DevKitC-32E
Processor	ARM Cortex-M4 (100MHz)	ATmega328P (16MHz)	Dual-core Xtensa LX6 (240MHz)
Flash Memory	512KB	32KB	4MB
RAM	128KB	2KB	520KB SRAM + 2MB PSRAM
Wireless Connectivity	No WiFi/Bluetooth	No WiFi/Bluetooth	Built-in WiFi/Bluetooth
Power consumption	low	Very low	Moderate
GPIO pins	33	22	25
Cost	14$	4.5$	13$

First, the Arduino Nano was excluded due to its limited processing capabilities, which were found to be insufficient for real-time control and sensor data handling in a quadruped robot. Between the remaining candidates, the STM32F411 Black Pill was found to offer lower power consumption and more GPIO pins, while the ESP32 DevKitC-32E provided greater memory and built-in wireless communication. The ESP32 DevKitC-32E was finally selected, as its onboard communication features and memory capacity enabled it to manage multiple tasks simultaneously while allowing for potential remote control and web-based data visualization.

Another criterion in the quadruped robot is video streaming, which helps in detecting survivors or hazards in the disaster zones. A single ESP32 DevKitC-32E is not capable of handling video streaming, besides motor control and sensor data processing. That is why another microcontroller was selected for this purpose, which is the ES32-CAM module.

ESP32-CAM: It is a development board with many GPIOs, an OV2640 camera, a microSD card slot, and an ESP32-S chip.

In addition to handling video streaming, it enables users to create a web server for video streaming (DFROBOT, n.d.).

Microprocessor: To support autonomous exploration, which includes SLAM and path planning, a microprocessor is required. The microprocessor should have enough computing power, software compatibility, and connectivity to allow autonomous exploration. Several widely used single-board computers were evaluated to determine the most suitable and cost-effective solution. The selection process was based on various components including processing power, usability with ROS, community support, wireless connectivity, and cost. The candidate boards included Raspberry Pi 4 Model B, Raspberry Pi 5, and NVIDIA Jetson Nano.

NVIDIA Jetson Nano: It was selected for the performance of its GPU, capacity to use AI processing, and ease of use with ROS. It is widely used in robotics applications such as machine learning and computer vision. However, its main problems are its high cost, higher power consumption, and lack of built-in wireless connectivity. More information is seen in Table 3.3.

Raspberry Pi 5 (4GB RAM): It was selected for its powerful processor, faster RAM, and upgraded GPU. It supports both desktop and embedded robotics applications. The main disadvantage is that it is more expensive than previous models, making it less desirable for cost-effective design ideas. More information is present in Table 3.3.

Raspberry Pi 4 Model B (4GB RAM): It was selected due to its balanced performance, quad-core processor, 4GB RAM, and wide community support. It offers an excellent balance of computing power, cost, and compatibility with SLAM and navigation software, which makes it suitable for autonomous robotic applications. Table 3.3 provides more information.

Table 3.3 Comparation between three microprocessors.

Feature	Raspberry Pi 4 Model B	Raspberry Pi 5	NVIDIA Jetson Nano
Processor	Cortex-A72 (1.5GHz)	Cortex-A76 (2.4GHz)	Cortex-A57 (1.43GHz)
RAM	4GB LPDDR4	4GB / 8GB LPDDR4X	4GB LPDDR4
Wireless connectivity	Built-in WiFi/Bluetooth	Built-in WiFi/Bluetooth	No WiFi/Bluetooth
Power consumption	Low	Moderate	High
Community support	Extensive	Growing	Moderate
Cost	55$	82$	99$

The NVIDIA Jetson Nano was the first candidate removed because it has high cost and power demands, which were found to be unnecessary for autonomous exploration tasks. Among the remaining candidates, Raspberry Pi 5 appeared to provide a greater processor. On the other hand, Raspberry Pi 4 Model B (4GB) provided a better balance between cost and performance. The Raspberry Pi 4 Model B (4GB) was finally selected because its computing resources and compatibility with ROS allowed it to support autonomous exploration while maintaining the low-cost design goal of the thesis.

3.3.1.2. Sensors

For the quadruped robot to navigate its environment autonomously, detect hazardous gases, and maintain stability, a set of integrated sensors must be carefully chosen. The sensors can provide critical data that enables autonomous motion, obstacle detection and avoidance, mapping, and environmental awareness, which make them important for the quadruped robot's overall

functionality. For the designed quadruped robot, four main features were targeted which are distance measurement for obstacle detection, monitoring hazardous gases, balancing the robot, and mapping. Selecting the appropriate sensors follows some criteria, which are accuracy, power efficiency, availability, and cost.

Distance Measurement Sensors: For distance measurement, two different types of sensors were found to be widely used in quadruped robots and meet all the criteria, which are ultrasonic sensors and time-of-flight sensors. One sensor from each type was selected and compared to choose the most suitable for the designed quadruped. The chosen sensors were VL53L0X time of flight and HC-SR04 ultrasonic sensors.

HC-SR04 ultrasonic sensor: It consists of a transmitter and a receiver. The transmitter generates an ultrasonic wave, and when the ultrasonic wave hits an object, it will bounce back, and the receiver will receive it. Finally, the time will be measured during which that ultrasonic wave propagates at a distance from the transmitter and receiver (Zhmud et al., 2018). More information is shown in Table 3.2

VL53L0X time of flight: Similar to the ultrasonic sensor, it employs the time-of-flight technology to estimate distances by sending a brief light pulse and timing the interval between the signals that are sent and received (Komarizadehasl et al., 2022). More information is available on Table 3.4

Table 3.4 Comparison between distance measurement sensors.

Feature	VL53L0X	HC-SR04
Measurement principle	Time-of-flight (Laser)	Ultrasonic sound waves
Range	2m	4m
accuracy	± 3mm	± 3mm
Field of View	Narrow	Wide
Cost	15$	2.75$

HC-SR04 was preferred over VL53L0X since it has a higher range and a wider Field of View, which can cover more angles. Alao, it is way cheaper. Two HC-SR04 were selected for the quadruped robot to allow obstacle detection from multiple directions, reducing blind spot, and providing redundancy in case one sensor becomes unreliable due to environmental interference.

Hazardous Gas Sensor: For monitoring hazardous gases, the MQ-series gas sensors are perfect candidates for detecting different types of harmful gases due to their high sensitivity and low cost (Ajiboye et al., 2021). For instance, MQ-2 can detect methane and propane (Pololu, n.d.-a), and MQ-3 can detect benzene and Hexane (Pololu, n.d.-b). After discovering different types of MQ-series gas, the MQ-135 gas sensor was selected for the quadruped robot.

MQ135 gas sensor: It is used for air quality monitoring applications due to its capability to detect various types of gases, including Ammonia (NH3), Nitrogen Oxides (NOx), alcohol, benzene, smoke, and Carbon Dioxide (CO2). It has a high sensitivity and long life (Olimex, n.d.).

The MQ-135 was preferred due to its multi-gas detection capability, which is essential for identifying areas with poor air quality or hazardous gases in disaster environments. Its sensitivity, affordability, and suitability for earthquake rescue operations made it an ideal choice for the quadruped robot.

Orientation Sensor: For maintaining balance and stability, a sensor capable of tracking angular and linear motion was required. The MPU6050 was found to be commonly used in various quadruped robot projects due to its performance and compatibility with embedded systems (Prayogo et al., 2021; Ranjane et al., 2021; Knälmann & Saläng, 2023).

MPU6050: It is a widely used type of IMU that combines a 3-axis accelerometer, 3-axis gyroscope, and a Digital Motion Processor (DMP) all in a small package. It is designed to measure angular velocity as well as linear acceleration, which makes it suitable for applications requiring motion tracking and balance control (InvenSense, 2013).

The MPU6050 was selected because of its dependability, accessibility, and simplicity of integration into the quadruped robot's embedded control system.

Mapping Sensor: To support SLAM-based autonomous exploration, a 2D LiDAR sensor was required. After evaluating cost-effective options, the RPLIDAR A1 was selected for its performance and affordability.

RPLIDAR A1: It is a 360-degree 2D laser scanning device and operates through triangulation-based ranging. It provides up to 8000 samples per second and supports real-time environment mapping. Moreover, its compact size, low power consumption, and ROS compatibility make it suitable for quadruped robots.

The RPLIDAR A1 was chosen due to its ability to provide accurate 2D mapping at a relatively low cost, which supports the robot's autonomous navigation capabilities while maintaining the cost-effectiveness of the overall design.

3.3.1.3. Servos

For the quadruped robot to move in complex environments and provide precise control over its legs for walking, turning, and stabilization, the servos must be carefully selected. Before choosing the servos, the method of simultaneously controlling 12 motors was determined. The ESP32 DevKitC-32E microcontroller was found to have an insufficient number of PWM-capable GPIO pins to drive all servos directly. Therefore, an efficient servo driver is required. The PCA9685 PWM driver was selected due to its capability to control up to 16 servos simultaneously and its cost-effectiveness.

PCA9685 PWM driver: it is an I2C bus-controlled 16-channel PWM driver designed to control multiple LEDs or servos using a single I2C communication interface (I2C BUS, 2015). It is widely used in robotics applications, especially for projects that require precise and simultaneous control of multiple servos, like the quadruped robot.

To select suitable servos for the quadruped robot, first a torque calculation is required. It was assumed that the robot is in a static pose and the body's weight is distributed equally in all four legs. The required parameters are shown on table 3.5.

Table 3.5 Required parameters for torque calculation.

parameter	value
Total robot mass	M = 2.2kg
Weight per leg	FR = 5.4 N
Friction on rough surface	μ = 0.65
Femur	X2 = 100 mm
Tibia	X3 = 115 mm

A Free body diagram is designed for a single leg as seen in Figure 3.9

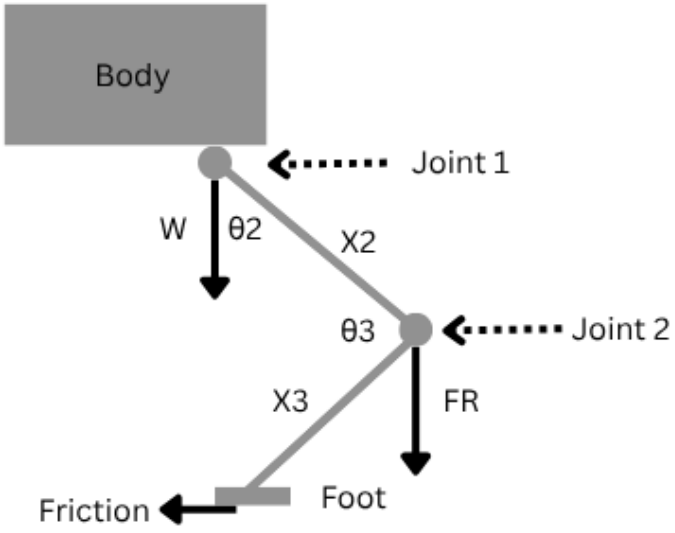


Figure 3.9 Free body diagram for torque calculation

First, torque at joint 1 is calculated

$$\tau 1 = \mu \times FR \times X2 \tag{3.31}$$

$$\tau 1 = 0.65 \times 5.4 \times 0.1 = 0.351\, Nm = 3.58 \frac{kg}{cm} \tag{3.32}$$

Next, torque at joint 2 is calculated using two cases. Case A: **θ3** = 45

$$\tau 2 = FR \times \sin(45) \times X3 \tag{3.33}$$

$$\tau 2 = 5.4 \times 0.707 \times 0.115 = 0.44\, Nm = 4.48 \frac{kg}{cm} \tag{3.34}$$

Finally, the torque at joint 2 when **θ3** = 90

$$\tau 2 = FR \times \sin(90) \times X3 \tag{3.35}$$

$$\tau 2 = 5.4 \times 1 \times 0.115 = 0.62\, Nm = 6.32 \frac{kg}{cm} \tag{3.36}$$

Using a safety margin of 2X for dynamic movement the selected servo motor should have at least 13 kg/cm torque.

Although servos with higher torque rating can increase the quadruped robot's load-carrying capacity, other important factors such as cost, power consumption, and weight are considered. Three servos were selected for comparison, which are FEETECH FS5115M 15kg/cm (FEETECH, 2021), TD 8325MG 25kg/cm (Howes Models, n.d.), and SPT SPT5435LV-180W 35kg/cm (Reacher Technology Co., Ltd, n.d.)

FEETECH FS5115M 15kg/cm: It represents the minimum torque threshold required for the quadruped robot to move. It is a cheap and lightweight option. Even though it can handle the quadruped's movement, it may face some problems that affect performance under high stress since it operates closer to the quadruped's maximum capacity. More information is seen in Table 3.3.

TD 8325MG 25kg/cm: It offers a mid-range torque of 25kg/cm, which achieves a balance between strength and cost. It can provide sufficient power and smooth movement for the quadruped robot in different environments. It is cheap and lightweight as well. More information is available on Table 3.3.

SPT SPT5435LV-180W 35kg/cm: It is a high-performance servo used in applications that require high torque. With its 35kg/cm torque rating, it can ensure stability under heavy loads. It is expensive and has higher power

consumption, and weight than the other options. More information is seen in Table 3.6.

Table 3.6 Comparison between three servo motors

Feature	FEETECH FS5115M	TD 8325MG	SPT SPT5435LV
Torque	15 kg/cm	25 kg/cm	35 kg/cm
Operating voltage	4.8v – 6v	4.8v – 6.8v	4.8v – 6v
weight	60.7g	58g	70g
Bearing type	Two ball bearing	Two ball bearing	Two ball bearing
Power consumption	low	Medium	high
Cost	13$	10$	17$

The FEETECH FS5115M was the first servo to be excluded from the consideration, since it may struggle with high load making it insufficient option for earthquake rescue operations. A comparison between the TD 8325MG and the SPT SPT5435LV-180W was then made. Although the SPT5435LV-180W offers the highest performance, its weight, power consumption, and cost were not aligned with the project's cost-effective and energy-efficient design goals. The TD 8325MG was finally selected as the most suitable option. It was found to provide sufficient torque, reliability, and balance between cost and performance, making it a practical and efficient choice for earthquake rescue operations.

3.3.1.4. Power management

Power management is a critical aspect in the design of quadruped robots. It makes all components in the quadruped robot receive the appropriate voltage and current for stable movement. The system must deliver power effectively from the primary power source. At the same time, it must protect sensitive electronics and maintain efficiency. A 7.4V LiPo rechargeable battery was selected as a primary power supply. However, different components require different operating voltages, which require a structured power distribution system. To achieve this, the power management system was designed to include a Switching Battery Eliminator Circuit (SBEC), a DC-DC voltage regulator, a power distribution bus, and a main power switch. Each of these elements was selected to ensure reliable voltage regulation, current delivery, and power safety.

Main power source: The quadruped robot is powered by a 7.4V LiPo rechargeable battery, which is selected due to its high energy density, lightweight, and ability to supply high discharge current. They have higher energy density and higher discharge current compared to NiMH batteries, which were also a choice for the quadruped robot. Also, LiPo batteries ensure longer working time, which is one of the criteria of a quadruped robot that is used for autonomous exploration and earthquake rescue

Voltage regulator: The ESP32, ESP32 CAM, PCA9685 logic, and sensors should not exceed 5V. That's why LM2596 is used to regulate voltage from 7.4V down to 5V, which ensures safe and stable operation. The LM2596 is capable of driving 3A (Texas Instruments, 2023), which is enough for the components since they all require less than 2.5A. This converter provides a constant supply voltage to all components that require a 5V so they do not face any fluctuations that could lead to erratic sensor behavior.

Switching Battery Eliminator Circuit (SBEC): To safely convert the 7.4V output of the LiPo battery to the voltage levels required by the servos and the motor driver, a 6V SBEC was included in the power management system, which is usually used in drones. Since the PCA9685 motor driver can withstand

a maximum of 6V for servo motor supply, the SBEC can efficiently step down the 7.4V output from the LiPo battery to a constant 6V output, capable of delivering a high current, which is needed by the servo motor.

Power distribution bus: A power distribution bus was selected to efficiently supply 5V to multiple components such as ESP32, raspberry Pi, ESP32 CAM, ultrasonic sensors, lidar, and buzzers. The power disruption bus simplifies wiring by being the center connection point for multiple power lines instead of running separate wires from the LM2596 to each component. Also, using power distribution reduces the voltage drops caused by redundant wiring length, and ensures that each 5V component receives stable voltage.

Power control: A main switch is used between the battery and the power distribution system to easily control the power. This switch allows the user to completely cut off the power supply if the quadruped robot is not used, which prevents unnecessary battery drain and protects the component from potential electrical damage. The switch is placed before the power divides, which guarantees that all power lines are disconnected when the switch is turned off. A high-quality switch that can withstand high voltage and current is selected for a reliable operation.

3.3.2. Schematics

The schematic diagram is the design that shows the wiring connection of the electronics component for proper functionality between the microcontrollers, sensors, actuators, and power management. The design is based on modularity for troubleshooting and future upgrades. EasyEDA is the software that is being used to create schematic diagrams. Using the Net Label principle allows connecting points in the circuit without physical wiring, the schematic is simpler and more understandable.

ESP32 DevKitC-32D: The ESP32 DevKitC-32D is the primary control unit in the quadruped robot. It is responsible for managing sensor input and executing the control algorithm. The schematic diagram in Figure 3.10 shows the main connection elements associated with ESP32 DevKitC-32D.

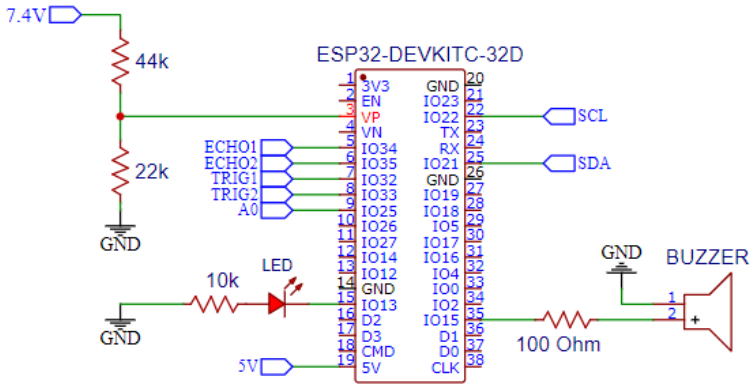


Figure 3.10 Schematic diagram of the ESP32 DEVKITC.

There are two power input options provided in the schematic. The first one uses the voltage divider principle formed by 44K and 22K resistors to step down the 7.4V to 2.5V in the analog input pin VP, which is used to monitor the main supply voltage level

$$Vout = Vin\frac{R2}{R1 + R2} = 7.4\frac{22}{44 + 22} \tag{3.37}$$

The second option is using the regulated 5V directly to the 5V pin, which enables stable operation in the board. The microcontroller interfaces with two ultrasonic sensor through digital I/O pins, which are the ECHO pins that are connected to IO34 and IO35 and the TRIG pins that are connected to IO32 and IO33, an MQ-135 gas analoge input that is connected to IO25, a power LED that is connected IO13 through a 10k resistor, and a buzzer that is connected to IO15 through a 100ohm resistor. Additionally, the SCL and SDA pins, which are used by the MPU6050 and the PCA9685 for I2C communication, are connected to IO22 and IO21. The ESP32 and the Raspberry Pi 4 are connecting wirelessly. This communication allows the Raspberry Pi to send high-level navigation commands to the ESP32, which then executes corresponding leg movements and handles low-level sensor data processing.

ESP32-CAM: The ESP32-CAM module is used in the quadruped robot for real-time video streaming. It operates independently from the main ESP32 microcontroller. The schematic shown in Figure 3.11 illustrates the essential wiring for programming and powering the module

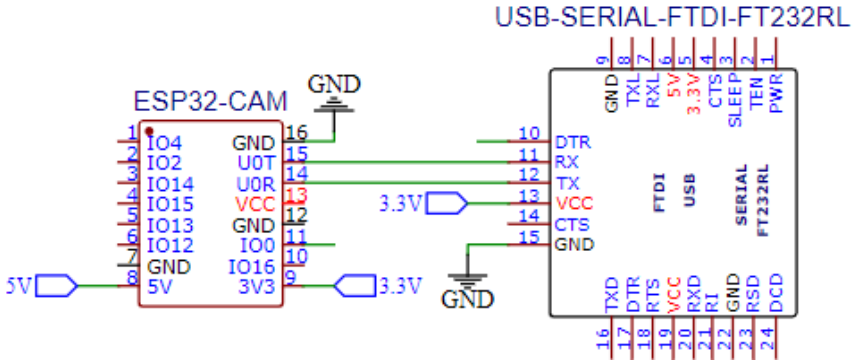


Figure 3.11 Schematic diagram of the ESP32-CAM.

Power is supplied to the ESP32-CAM through its 5V pin, which is regulated internally. That's why a stable 5V input will ensure a reliable operation. An FT232RL-based USB-to-Serial converter is used to program the ESP32-CAM. The U0R and the U0T pins of the ESP32-CAM are connected to the TX and the RX pins of the FT232RL for serial communication, while the VCC pin of the FT23RL is powered by the 3.3V of the ESP32-CAM.

Raspberry Pi 4B: The quadruped robot's high-level processor is the Raspberry Pi 4B. It is in charge of managing complex computing activities including path planning, SLAM, and LIDAR data processing. The schematic diagram of the Raspberry Pi 4B is seen in Figure 3.12

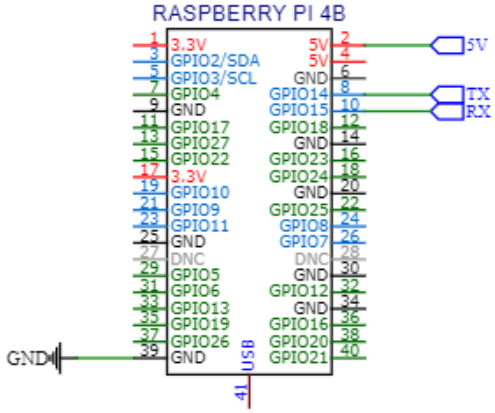


Figure 3.12 Schematic diagram of the Raspberry Pi 4B.

A regulated 5V is used to power the Raspberry Pi 4B. A serial communication link between Lidar and Raspberry Pi 4B is established via the UART port. The Raspberry Pi's TX (GPIO14) pin is linked to Lidar's RX. The Raspberry Pi's RX (GPIO15) pin is linked to Lidar's TX. Both devices share a common GND to ensure proper voltage reference. The Raspberry Pi communicates with the ESP32 wirelessly.

HC-SR04 ultrasonic sensors: The quadruped robot uses two HC-SR04 ultrasonic sensors for obstacle detection and distance measurement. The schematic in Figure 3.13 shows the wiring for both ultrasonic sensors, named ultrasonic1 and ultrasonic2.

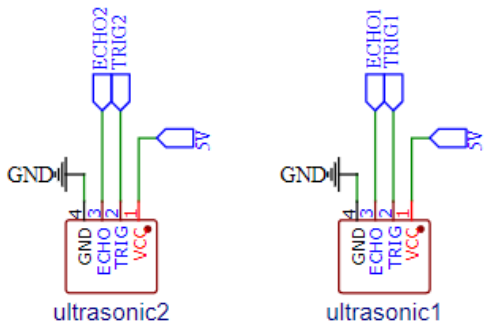


Figure 3.13 Schematic diagram of the ultrasonic sensors.

Each sensor has four pins, which are VCC, GND, TRIG, and ECHO. Both sensors are powered by a stable and regulated 5V and grounded to the system's common GND. The TRIG and ECHO pins are connected directly to the ESP32's digital I/O pins.

MQ135 gas sensor: The MQ135 gas sensor is used in the quadruped robot to monitor air quality. The schematic in Figure 3.14 shows the wire connection for the MQ135 gas sensor.

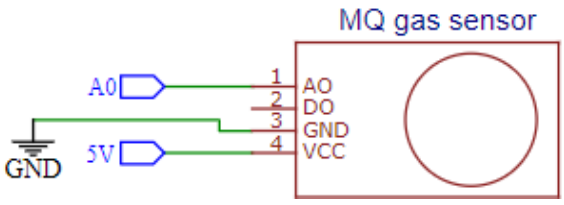


Figure 3.14 Schematic diagram of MQ135.

The A0 analog output of the MQ135 is connected directly to the main ESP32. The sensor is powered using a regulated 5V and grounded the same as the ultrasonic sensor.

MPU6050: The MPU6050 is essential in the quadruped robot for estimating the orientation and movement of the quadruped robot. Figure 3.15 shows the schematic of the MPU6050.

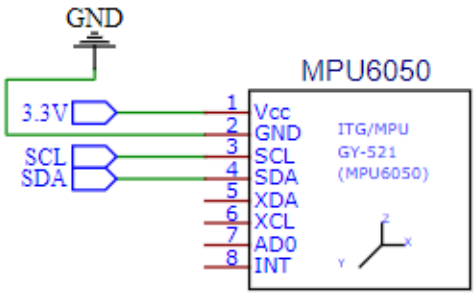


Figure 3.15 Schematic diagram of the MPU6050.

In the schematic, the SDA and SCL lines of the MPU6050 are connected to the corresponding I2C pins in the main ESP32, which are IO21 for SDA and IO22 for SCL. The MPU6050 is powered directly from the ESP32's regulated 3.3V output. GND is connected to the shared system ground.

RPLIDAR A1: The RPLIDAR A1 is used for 2D laser scanning and is responsible for providing environmental mapping data to Raspberry Pi 4. The schematic of the RPLIDAR A1 is shown in Figure 3.16.

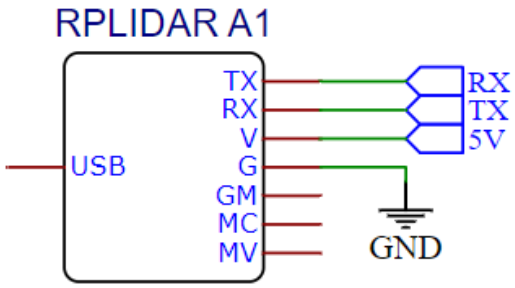


Figure 3.16 Schematic diagram of the RPLIDAR A1.

The RPLIDAR A1 is connected to the Raspberry Pi by the UART interface, enabling serial communication. The scanner is powered by a regulated 5V. The lidar continuously transmits range data to the Raspberry Pi, which is processed in real-time for SLAM, localization, and path planning.

PCA9685 PWM driver: The PCA9685 is a 16-channel PWM driver used in the quadruped robot to control the servo motor that moves the robot's legs. The schematic seen in Figure 3.17 shows the wire connection of the PCA9685.

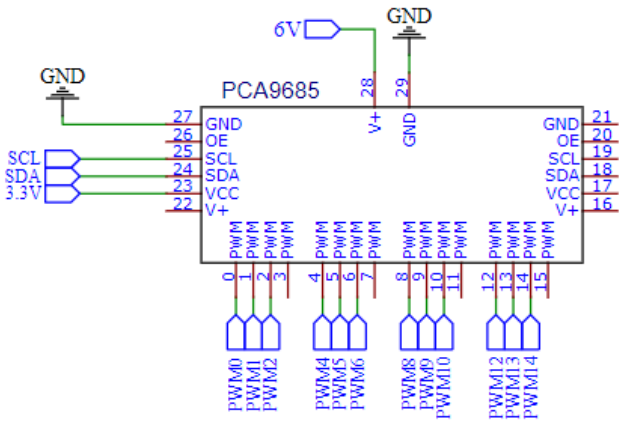


Figure 3.17 Schematic diagram of the PCA9685.

The SDA and SCL pins of the PCA9685 are connected to the same I2C bus used by the MPU6050 in the main ESP32. But it has a different I2C address from the MPU6050, which allows both devices to coexist on the same communication lines without conflict. The module has two separate power inputs which are the VCC pin that is connected to the ESP32's regulated 3.3V output to power the chip of the PCA9685, and the V+ pin that is connected to a regulated 6V to power the servos. In the schematic, 12 PWM outputs of the PCA9685 are connected to 12 servo motors PWM.

TD 8325MG: The quadruped robot uses 12 servo TD 8325MG servo motors named servo1 to servo12 to control its four legs, each leg is driven by three servos for femur, tibia, and hips. Figure 3.18 shows the schematic of the TD 8325MG.

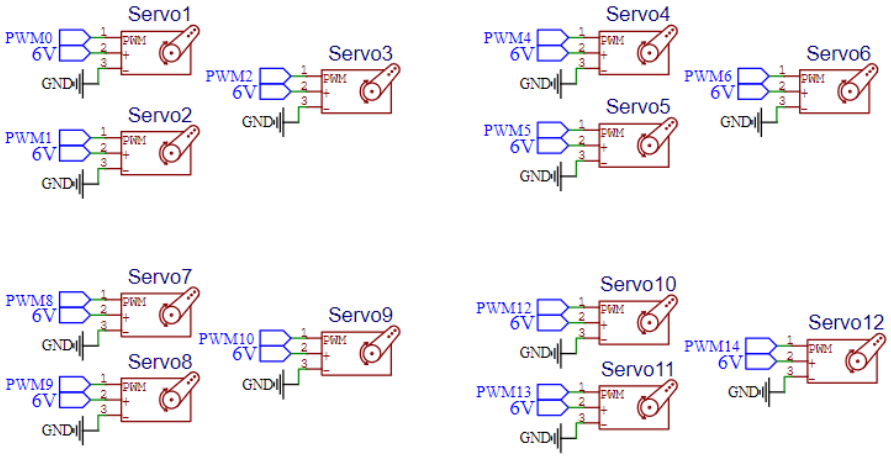


Figure 3.18 Schematic diagram of the servo motors.

Each servo's PWM control signal is connected to one of the 12 output channels seen in the PCA9685 PWM motor driver. The power supply for all 12 servos is delivered from the V+ pin of the PCA9685 module, which is regulated to 6V. This voltage is within the operating range of the TD 8325MG servos. The GND pins of the servos are connected to PCA9685 ground.

Power Management: Efficient and reliable power management is important for the quadruped robot's performance. The schematic shown in Figure 3.19 illustrates the power management wiring.

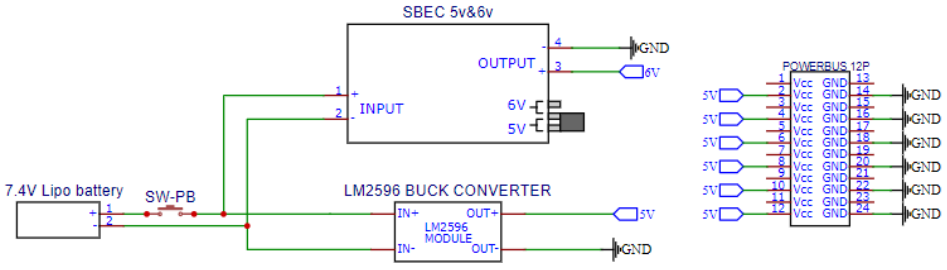


Figure 3.19 Schematic of power management.

A 7.4V LiPo battery, chosen for its excellent energy density, powers the complete system. Immediately after the battery, there is a power switch used to manually control the robot's power state. After the switch, the power line is divided into two lines. The first line goes to the LM2596 buck converter, which steps down the 7.4V input to a regulated 5V output, then the 5V output is connected to a power distribution bus that supplies the ESP32-DEVKITC-32D, the ESP32-CAM, the ultrasonic sensors, and the MQ135 gas sensor. The second line goes to an SBEC, which converts the 7.4V battery voltage to a steady 6V, and this 6V is connected to the V+ terminal of the PCA9685 to power the 12 servo motors.

3.4. AUTONOMOUS EXPLORATION

3.4.1. System Design

To achieve full autonomous exploration in unknown and earthquake affected environments, it is essential to integrate both hardware and software systems that enable perception, mapping, localization, path execution and responding to hazardous gases. The autonomous exploration system is equipped with a Lidar and gas sensor. The core functionality of the autonomous system depends on six fundamental components which are Global Map Construction, Self-Localization, Path Planning, Motion Control, Environment Perception (Zhang et al., 2024), and newly added Gas Hazard Heatmap Layer. These components enabled the quadruped robot to make informed decisions based not only on terrain and obstacle, but also on the presence of hazardous gases in the environment. The designed flowchart is seen in Figure 3.20.

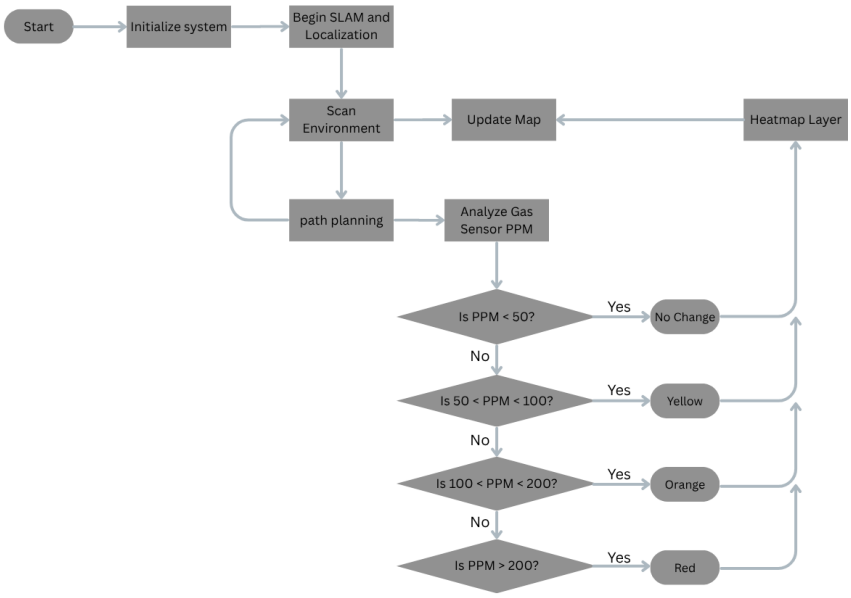


Figure 3.20 Designed autonomous exploration flowchart.

Global Map: built using a SLAM algorithm, and it is a complete map of the explored environment. This map is static and updates only when the robot discovers new or unmapped regions. The global map plays a critical role in long-range path planning and high-level navigation.

Self-Positioning (Localization): Precise localization is needed for the robot to interpret its location on the global map. It is used to Estimate the current pose using odometry or inertial data and match the LiDAR scan data to the known features of the existing global map using scan-matching algorithms.

Path Planning: Autonomous navigation integrates both the global map and a local map which is a high-frequency scan that reflects the robot's immediate surroundings. The local map is used to detect close obstacles and adjust the robot's route accordingly. Path planning combines global guidance with local corrections, allowing the robot to follow long-term goals while avoiding obstacles in real time.

Map Scope and Frequency: The global map is suitable for strategic navigation and is updated less frequently. On the other hand, the local map operates at high frequency, capturing any sudden changes such as moving

obstacles, debris, or terrain variation. These two maps ensure that the robot can react to its environment without losing its long-term navigation plans

Motion Control: This module is responsible for translating the outputs of the path planning algorithm into low-level commands that drive the servos. Motion control ensures that the robot's leg movements stay stable while maintaining coordination with the navigation goals

Environmental Perception: The system's sensory module continuously perceives the environment, and provides the required data used by both SLAM and real-time obstacle avoidance. Accurate perception supports the reliability of mapping, localization, and control.

Gas Hazard Heatmap Layer: A novel addition to this architecture used to enhance situational awareness by mapping hazardous gas concentrations across the environment. The robot uses a gas sensor to sample gas density at each explored location. These readings are geotagged and layered on top of the existing occupancy grid.

3.4.2. GMapping SLAM

In autonomous exploration tasks, accurate real time mapping is essential for ensuring that the quadruped robot can localize itself and explore the environment safely. Gmapping SLAM is one of the most efficient 2D SLAM algorithms for this task and is appropriate for the quadruped robot because of its robustness and real-time abilities. Gmapping is based on the Rao-Blackwellized Particle Filter, which separates the problem of pose estimation from map estimation. As a result, computational efficiency is improving without sacrificing accuracy (Su et al., 2020). The flowchart of the Gmapping algorithm is shown in Figure 3.21.

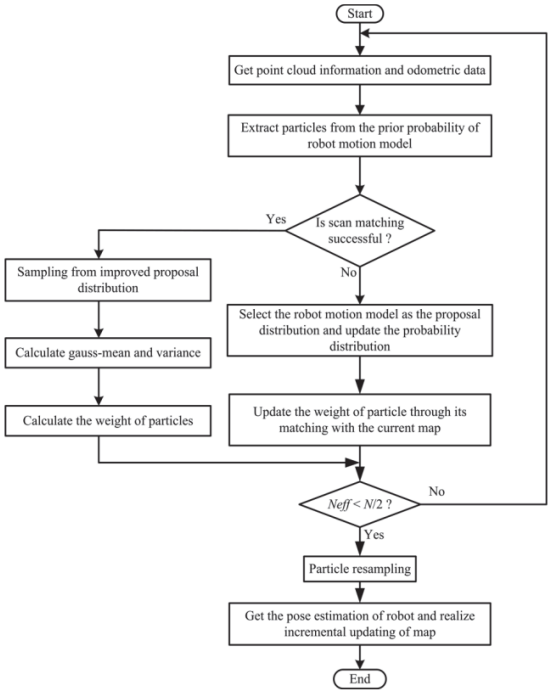


Figure 3.21 Flowchart of Gmapping algorithm (Su et al., 2020).

Gmapping's primary concept is to use a set of weighted particles to estimate the posterior distribution of the map and the robot's trajectory. Every particle carries a corresponding map created from the trajectory's history and represents a potential robot trajectory. The posterior distribution of the complete SLAM problem is factorized as follows

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) = p(m \mid x_{1:t}, z_{1:t})\, p(x_{1:t} \mid z_{1:t}, u_{1:t}) \tag{3.38}$$

where $x_{1:t}$ is the robot's trajectory over time, m is the map, $z_{1:t}$ is the set of observations (laser scans), and $u_{1:t}$ is the set of control inputs (odometry or IMU data). To improve sampling efficiency and reduce the number of required particles, Gmapping replaces the conventional motion-model-based proposal distribution with an observation-based proposal distribution that incorporates laser scan information. This significantly concentrates the sampling around more likely robot poses, especially in environments where motion noise is high. The improved proposal distribution is given by

$$p(x_k \mid m^i{}_{k-1}, x^i_{k-1}, z_k, u_{k-1}) = \frac{p(z_k|m^i_{k-1}, x^i_t)\, p(x_k \mid x^i_{k-1}, u_{k-1})}{p(z_k|\, m^i_{k-1}, x^i_{k-1}, u_{k-1})} \quad (3.39)$$

This formulation uses current observation z_k, the previous map $_{k-1}$, the control input u_{k-1}, and the previous pose x^i_{k-1} to generate a new sample x^i_k for each particle. To combat particle depletion, which occurs when a small number of particles dominate the weight distribution, Adaptive resampling is used in Gmapping according to the N_{eff} effective sample size. This metric is calculated as

$$N_{eff} = \frac{1}{\sum_{i=1}^{N}(w^i_t)^2} \quad (3.40)$$

Where w^i_t is the weight of the i-th particle at time t, and N is the total number of particles. Resampling is triggered only when N_{eff} falls below a certain threshold (typically N/2), which prevents unnecessary resampling and preserves particle diversity. Another critical feature of the Gmapping algorithm is scan matching, which aligns the current laser scan with the existing map to refine pose estimation. This process is carried out in two steps. The first is estimating the robot's pose using gradient descent based on the scan-to-map correlation, and the second updating the map using the refined pose. A minimum matching score threshold is applied to determine the quality of scan alignment. If the matching score falls below this threshold, the algorithm defau.lts to using the motion model alone for pose prediction, which reduces mapping accuracy.

3.3.3. Adaptive Monte Carlo Localization (AMCL)

When a quadruped robot operates autonomously in unknown environments, the ability to determine its location accurately on a map is essential. To overcome this difficulty, the probabilistic technique Adaptive Monte Carlo Localization (AMCL) employs a particle filter to estimate the robot's pose from motion models and sensor data. AMCL forms part of the

autonomous navigation of the robot, which allows the robot to navigate towards targets and avoid previously mapped obstacles in real-time (Zhang et al., 2024). The particle filter has 4 main steps which are initialization, prediction, measurement update, and resampling. In the initialization step, the particles are randomly spread across the map to represent possible poses. In the prediction step, each particle's pose is updated based on control input and a motion model. The measurement update step uses an actual sensor to determine from the particles how closely each particle's predicted observation matches the sensor input by adjusting particles weights. In the resampling step, the particles are selected based on probabilities proportional to their weights to create a new set of particles that focuses on the highest likelihood places. Based on the prior state, control input, and process noise, the system dynamic model predicts the quadrupled robot's state at time t using the equation

$$x(t) = f(t)\big(x(t-1), u(t), w(t)\big) \tag{3.41}$$

Where x(t) is the quadruped robot's state at time t, f(t) is the dynamic model function, u(t) is the control input, and w(t) is the process noise. The observation model predicts the expected sensor measurement at time t, based on the current state and observation noise according to this equation

$$z(t) = h(t)\big(x(t), v(t)\big) \tag{3.42}$$

Where z(t) is the observed sensor data from the laser scan, h(t) is the observation model function, and v(t) is the observation noise.

3.3.4 A-star Path Planning

To achieve efficient global navigation over the 2D occupancy grid map created by the SLAM, the A-star algorithm was chosen. For grid-based robotic navigation, the method is effective because it seeks to decrease the number of investigated nodes while ensuring optimality (Xu, 2024). The A-star algorithm gives each candidate node a score based on a cost function f(n), defined as

$$f(n) = g(n) + h(n) \tag{3.43}$$

Where h(n) is the expected cost from node n to the target node and g(n) is the actual cost from the start node to the current node n. The function h(n) can be computed using either the Manhattan distance or the Euclidean distance

$$h(n) = |\, dx - nx \,| + |dy - ny\,| \tag{3.44}$$

$$h(n) = \sqrt{(dx - nx)^2 + (dy - ny)^2} \tag{3.45}$$

Here, (dx, dy) are the coordinates of the destination node and (nx, ny) are the coordinates of the current node. Equation (3.44) is more suitable for grid-based planning when diagonal movement is not allowed, while equation (3.45) provides a more accurate metric when diagonal moves are permitted. To improve the planning accuracy in environments with obstacles, the traditional heuristic function h(n) is extended by introducing a penalty term l(r) to assess potential unknown path costs due to nearby obstacles. The new evaluation function is now

$$h'(n, r) = \sqrt{(dx - nx)^2 + (dy - ny)^2} + l(r) \tag{3.46}$$

The obstacle-included cost is defined as l(r), where r is the number of obstacle points along the extension line between the current node and its parent node.

$$l(r) = r.\omega \tag{3.47}$$

In this formula, ω is a tunable parameter that indicates how sensitive the cost is when considering obstacles. If there is no obstacles, where r = 0, the cost function reduces to the standard Euclidean heuristic. However, if obstacles exist near the path, the added term l(r) increases the predicted cost, which results in a deviation from potentially unsafe areas. By adding environmental awareness into the heuristic, the improved A-star algorithm results in safer and more realistic path planning. This approach ensures that the planned trajectory avoids

high-risk areas while still guiding the quadruped robot efficiently toward its goal. When integrated with the SLAM-generated map, the A-star planner dynamically adapts its cost evaluations based on updated environmental data during exploration.

CHAPTER 4

4. RESULTS AND DISCUSSION

4.1. SIMULATION SETUP

In order to test the performance of the quadruped robot in a realistic virtual environment, a comprehensive simulation setup was developed in the robot operating system (ROS). The simulation process began with creating a detailed URDF model that accurately represents the mechanical design of the robot, including joint placements, link dimensions, and inertial properties. This URDF was then used to test visualization and control testing. The urdf_tutorial package was used for initial visualization and joint control. This package enabled a clear and interactive display of the robot model in Rviz, allowing for real-time monitoring and manual adjustment of joint angles. Also, simulated PWM signals were used to control the servo motors virtually, mimicking the response of the physical servos attached to each leg. This step was essential for ensuring that the robot's kinematics matched its intended physical behavior. The URDF visualization is seen in Figure 4.1.

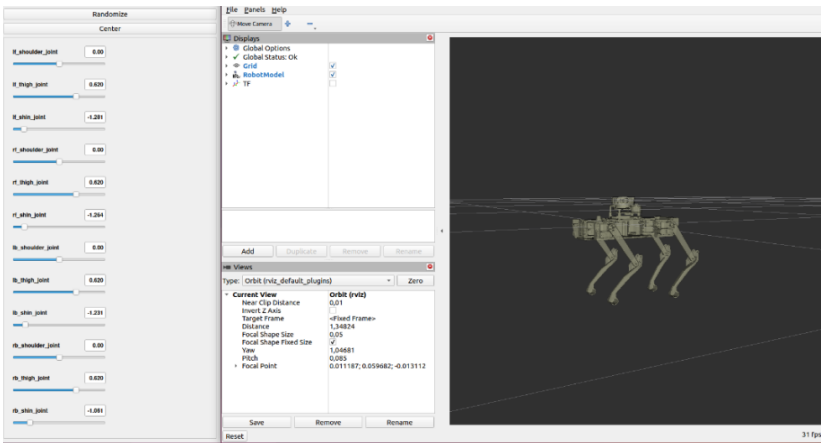


Figure 4.1 The URDF visualization of the quadruped robot.

To simulate a post-disaster environment, a custom Gazebo world was used. The virtual environment was designed to resemble a terrain affected by an earthquake, with a collapsed building, and various obstacles. This world provided a realistic testing ground for evaluating the quadruped's ability to navigate complex and unpredictable terrain as seen in Figure 4.2.

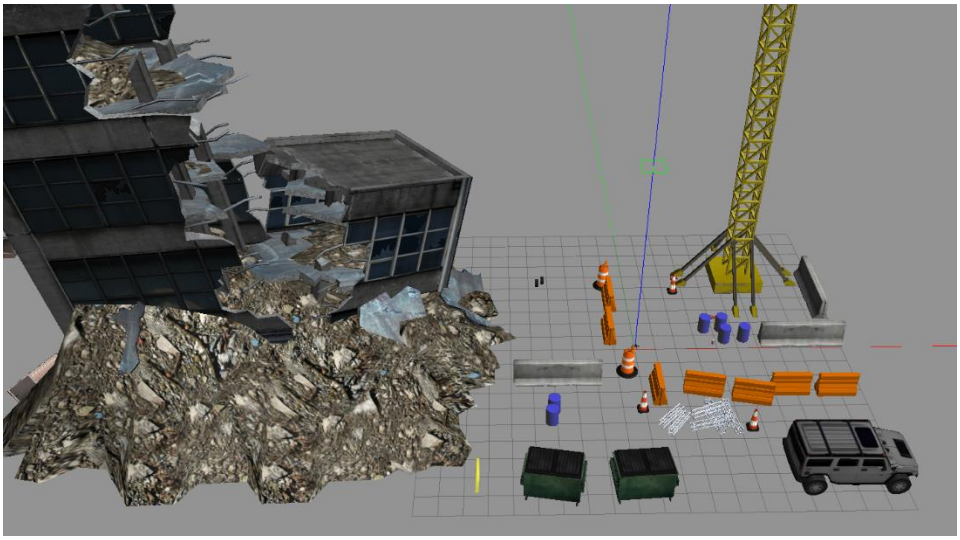


Figure 4.2 Gazebo world affected by an earthquake.

Furthermore, the CHAMP framework was integrated into the simulation. CHAMP is an open-source development tool, which enables the implementation and testing of different gait patterns, mappings, localizations, path planning strategies, and control algorithms in a ROS. In general, the simulation setup allowed the quadruped to test the critical features of its functional aspects. It was a safe and flexible way to test locomotion strategies, sensor integration, and test autonomy without the need of a physical robot. The simulation environment provided a way to speed up development and provide a basis for more complex applications in the real world.

4.2. MAPPING

For the quadruped robot to understand and navigate its environment, a SLAM technique was implemented using the Gmapping package. Gmapping is a SLAM algorithm available in ROS that builds a 2D occupancy grid map using laser scan data from the Lidar and odometry. The robot was equipped with the RPLIDAR A1 and simulated tested in a Gazebo simulation world and validated Gmapping under various conditions. The mapping tests conducted initially produced inaccurate maps and inconsistent mapping accuracy. That's why multiple parameters were iteratively adjusted to improve performance, stability, and map quality. Through experimental tuning and visual inspection, a set of optimized parameters were selected that consistently produced accurate and complete 2D maps. These parameters influenced aspects such as sensor range interpretation, map update rates, scan matching resolution, and particle filtering behavior. Table 4.1 shows essential Gmapping parameters that were adjusted.

Table 4.1 Gmapping parameters.

Parameter	Value	Description
maxUrange	13.0	Maximum usable range of laser scan
map_update_interval	15.0	Time interval (s) between map updates
linearUpdate	0.3	Distance threshold (meters) before triggering a map update
angularUpdate	0.3	Angular threshold (radians) before triggering a map update
particles	50	Number of particles used in the filter
minimumScore	50	Minimum score to consider a scan matching successful
delta	0.05	Resolution (meters/cell) of the occupancy grid map
transform_publish_period	0.1	Period (seconds) to publish the map-to-odom transform

After applying these parameters, the resulting maps showed significant improvement in clarity, consistency, and coverage of the outdoor environment. Figure 4.3 shows the mapping progress across six key stages, which are the initial scan, detection of nearby objects and obstacles, gradual map expansion as the robot explores further, refinement of structural boundaries, stabilization of mapped areas, and finally, the complete reconstruction of the 2D outdoor environment.

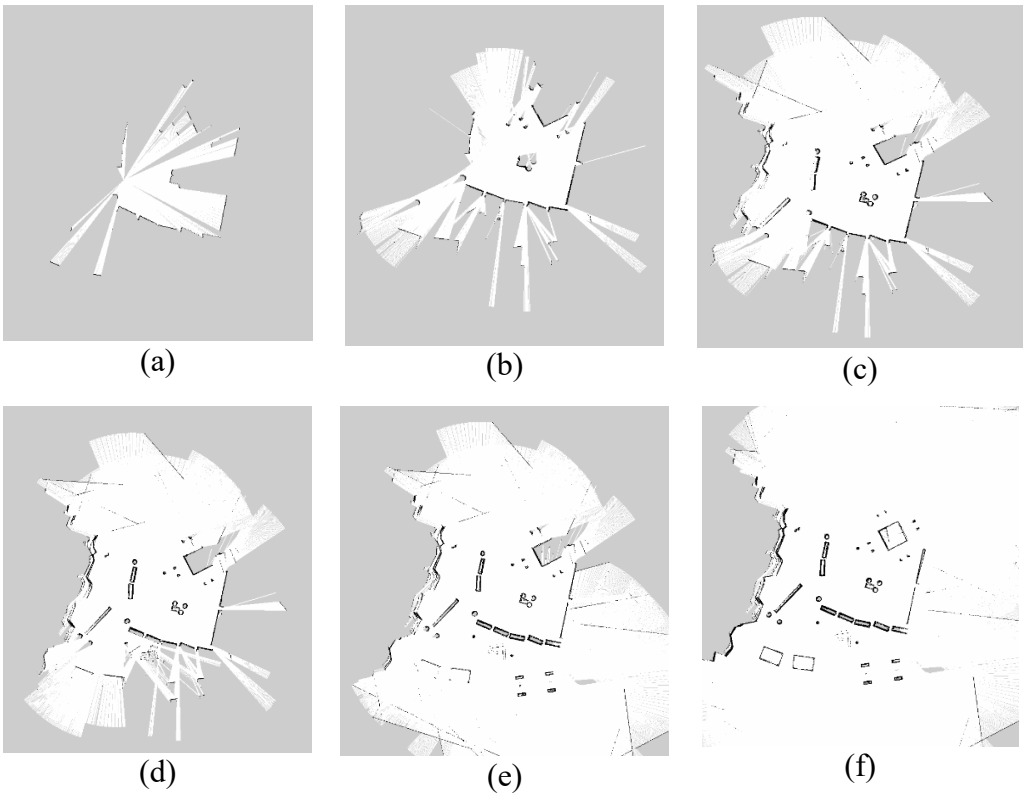
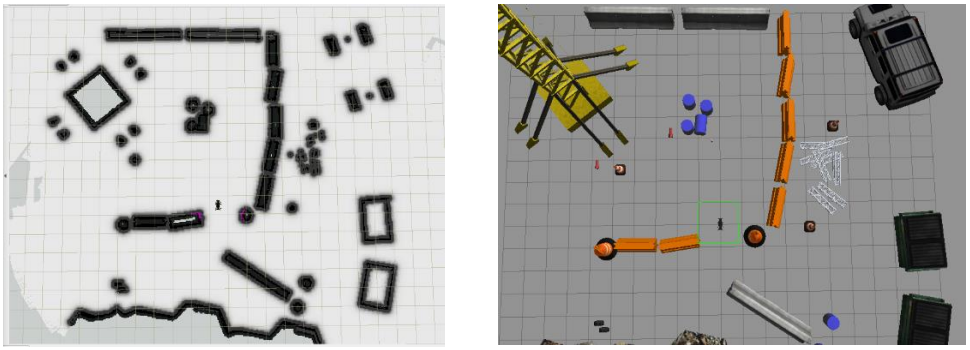


Figure 4.3 (a) Initial scan, (b) Detection of nearby objects, (c) Gradual map expansion, (d) Refinement of structural boundaries, (e) Stabilization of mapped areas, and (f) The complete reconstruction of the 2D outdoor environment.

This successful mapping phase provided a reliable foundation for subsequent localization and navigation algorithms, proving that Gmapping was an effective SLAM solution for the intended rescue exploration scenario.

4.3. LOCALIZATION AND PATH PLANNING

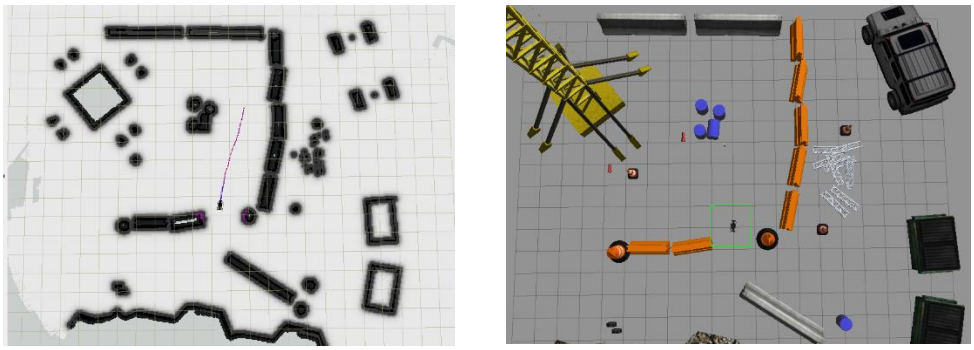
To enable precise navigation in an unknown outdoor environment, the quadruped robot was equipped with an Adaptive Monte Carlo Localization (AMCL) package for position estimation and the A* algorithm for optimal path generation. The localization process began with the robot positioned at its initial starting point in the simulated Gazebo environment. AMCL precisely matched the robot's current scan to the pre-made map by using odometry and Lidar sensor data. As a result, the robot was able to precisely determine its position within the global frame. In the first stage of the experiment, the robot localized itself correctly at the starting position without any drift or misalignment. This is shown in Figure 4.4a (Rviz) and Figure 4.4b (Gazebo top-down view), where the position estimate and actual physical location are in perfect correspondence.



(a) (b)

Figure 4.4 Initial position in (a) Rviz and (b) Gazebo.

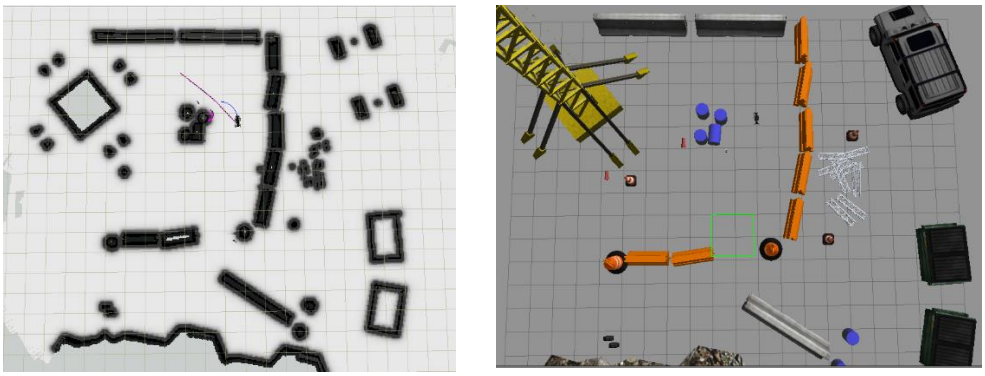
Once localization was achieved, the A* path planning algorithm was initiated to generate an optimal route to the target destination. At this stage, the path was calculated and visualized in RViz as a smooth trajectory avoiding known obstacles. The robot, however, had not yet started moving, allowing for inspection of the computed path. This is illustrated in Figure 4.5a (Rviz) and its corresponding physical layout in Figure 4.5b (Gazebo).



(a) (b)

Figure 4.5 Starting path planning in (a) Rviz and (b) Gazebo.

In the final stage, the robot executed the planned path, successfully reaching the designated goal point while maintaining accurate localization throughout its movement. Upon arrival, a new target location was assigned, and a fresh path planning process was triggered. The updated path is shown in Figure 4.6a (Rviz) alongside the real-time top-down simulation in Figure 4.6b (Gazebo).



(a) (b)

Figure 4.6 New localization and path planning in (a) Rviz and (b) gazebo.

Since the gas hazard detection system was not active during these tests, the environment was assumed to be safe (0 ppm). That's why the heatmap was not updated. Overall, the combination of AMCL and A* path planning demonstrated

reliable localization and efficient navigation. The close correspondence between RViz and Gazebo visualizations confirmed that both the simulated localization system and the physical movement model were aligned, ensuring that the robot could successfully navigate to multiple destinations in a post-disaster environment.

4.4. GAS HAZARD HEATMAP LAYER

A Gas Hazard Heatmap Layer was integrated into the mapping and navigation system to enhance the quadruped robot's capabilities for post-earthquake rescue missions. The purpose of this layer is to visualize areas with hazardous gas concentrations directly on the robot's map, allowing rescue teams to make informed decisions about their movement and safety measures. The system design follows a straightforward yet effective approach. As the robot explores the environment, gas concentration data from the onboard sensor is continuously read and georeferenced using the robot's current position from AMCL. Each data point is then stored as a hazard cell in a cost map structure. The intensity of the hazard is visually represented on the map using a color gradient

- 50–100 ppm (Yellow): Caution zone – rescue teams should proceed carefully
- 100–200 ppm (Orange): Safety zone required – personal protective equipment recommended
- Above 200 ppm (Red): Danger zone – rescue teams should avoid entry.

In the theoretical implementation, a ROS node subscribes to both the gas sensor data topic and the robot's position topic (/amcl_pose). The node processes the data in real time, assigns a hazard level, and publishes a visual marker array for RViz. This allows hazardous areas to be overlaid directly on the SLAM-generated map while the quadruped continues its exploration and path planning. Although no real gas sensor simulation was available in Gazebo for this

experiment, the logic was validated by assuming a constant gas level of 0 ppm during testing. As a result, no hazard zones appeared on the map in the present results. The proposed Gas Hazard Heatmap Layer represents a significant step towards creating a multi-layered situational awareness tool for rescue operations. By combining environmental hazard mapping with autonomous localization and navigation, the quadruped robot can serve as both an explorer and a safety advisor in dangerous post-disaster zones.

4.5. VIDEO STREAMING

One of the key components of the quadruped robot's functionality during search and rescue missions is its ability to stream a video of the environment in real time. For this purpose, a simulated onboard camera was integrated into the robot and tested in the Gazebo simulation environment. The main purpose of this camera system is to stream live video during the exploration process, allowing human operators to remotely observe the surroundings and identify signs of life or objects that may indicate the presence of survivors after an earthquake. During the simulation, the robot navigated through a virtual affected place by an earthquake. The camera provided a first-person visual stream from the robot's perspective, which helps in detecting relevant environmental features. Figure 4.7 shows four key frames captured during the exploration which are a soft drink can, a garbage bin, a parked vehicle, and a partially collapsed building. These pictures show the robot's ability to recognize items that could be associated with the presence of humans or hazards in the area.

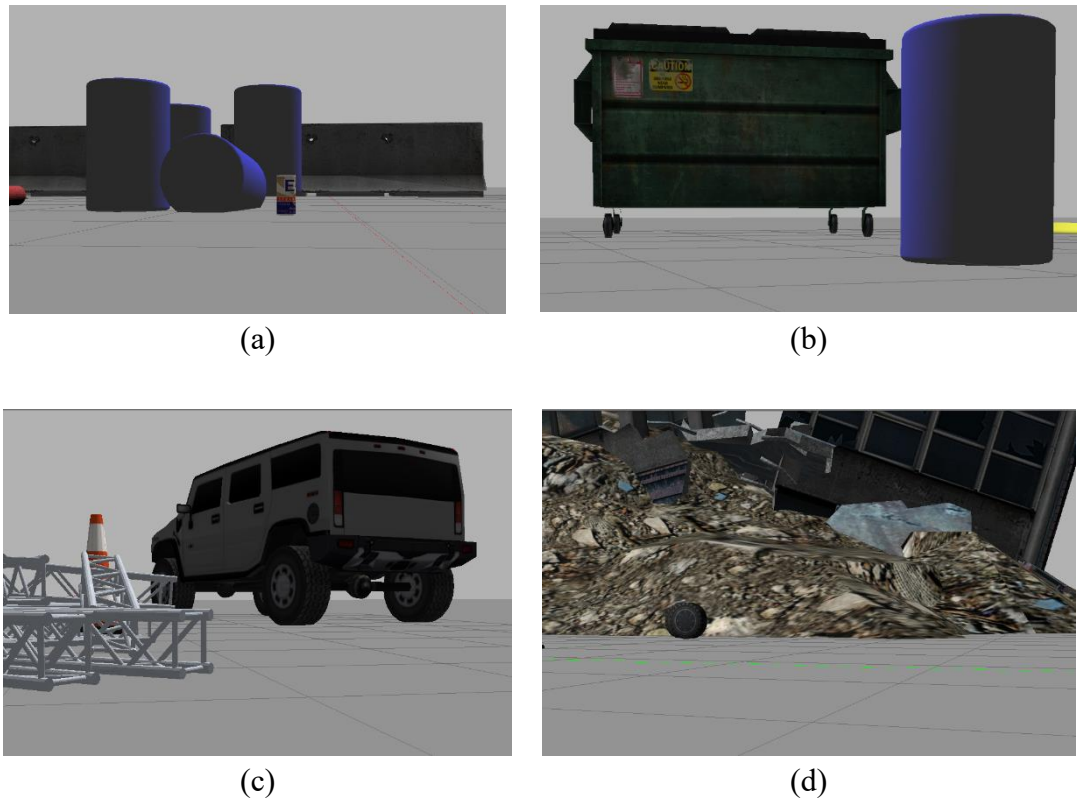


Figure 4.7 (a) Soft drink can, (b) Garbage bin, (c) Vehicle, and (d) Collapsed building.

These examples demonstrate the camera's effectiveness in identifying possible traces of human presence, debris, and obstacles that may require navigation adjustments or human action. The implementation of video streaming complements the mapping and localization systems, offering another layer of sensory input that supports the robot's role in Search and Rescue (SAR) operations.

4.6 RUNNING TIME

In order to evaluate the running time of the quadruped robot, an estimation of its running time was calculated based on the power consumption of all components and the capacity of the LiPo battery used. This estimation provides valuable insight into mission feasibility, especially in post-earthquake search

and rescue scenarios. The calculation process began by identifying the total average current drawing of the system during normal walking and mapping operation. The non-actuator components, including the ESP32 microcontroller, Raspberry Pi 4, RPLIDAR A1, camera, gas sensor, IMU, and ultrasonic sensors were measured to consume approximately 2.5 A. Also, the quadruped uses 12 servo motors movement, each drawing an average of 0.3 A during normal gait cycles. This results in a total of servo current consumption of 3.6 A. The overall system current draw was therefore calculated as 6.1 A during operation. The running time T in minutes was calculated using the equation.

$$T = \frac{Battery\ Capacity\ (mAh)}{Average\ current\ Draw\ (mA)} \times 60 \quad (4.1)$$

To compare different power configurations, four LiPo battery capacities were evaluated, and the results are summarized in Table 4.5.

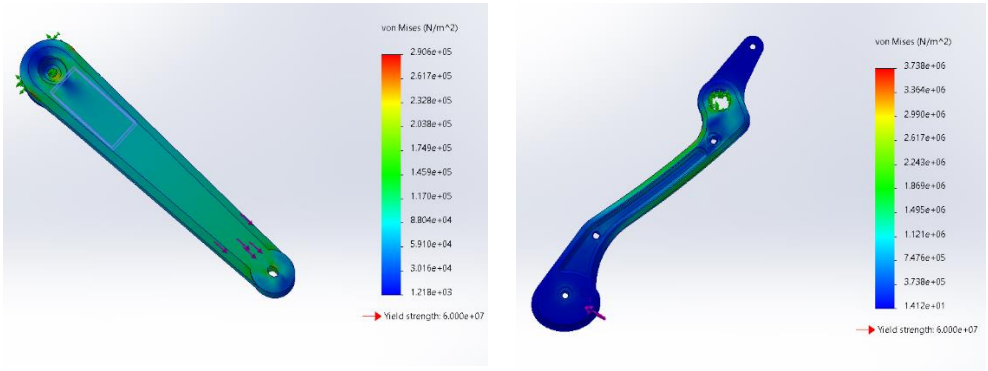
Table 4.2 Running time for four LiPo battery capacities.

Battery capacity (mAh)	Estimated running time (m)
2200	21.63
3300	32.45
4000	39.34
5000	49.18

Overall, the results demonstrate that the quadruped robot's endurance has a strong dependence on battery capacity and servo operating conditions. While larger capacity batteries extend mission time, they also increase weight, which can affect mobility and energy consumption. Therefore, an optimal balance between capacity and weight must be considered to maximize operational efficiency in the field.

4.7 MATERIAL ANALYSIS

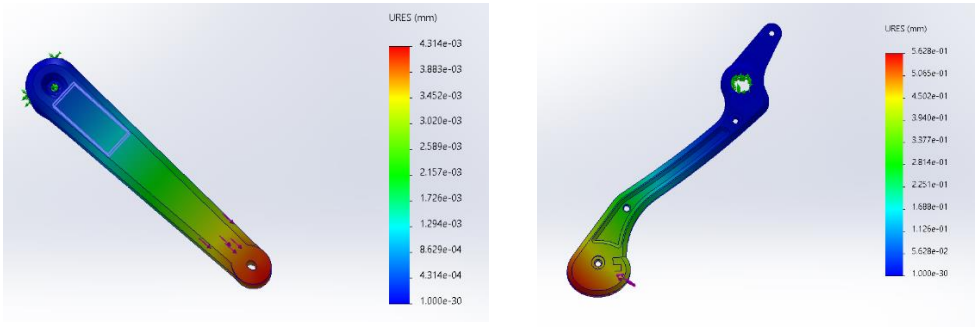
A material analysis was conducted using SolidWorks SimulationXpress to ensure the structural reliability of the quadruped robot's legs. The analysis focused on the femur and the tibia since they carry the whole body, with two main objectives, which are evaluating strength using the Von Mises stress and determining the expected deformation underload. The force acting on one leg during operation was calculated as 5.4 N. To account for unexpected loads and uncertainties, a factor of safety of 3 was applied, resulting in a design load of 16.2 N. For both the femur and tibia, the material used has a yield strength of 6.000×10^7 N/m². The strength analysis presents that the femur experiences a maximum Von Mises stress of 2.906×10^5 N/m², while the tibia experiences 3.738×10^6 N/m². Both values are significantly less than the yield strength, confirming that the components will remain within safe operating limits under the applied load as seen in Figure 4.8.



(a) (b)

Figure 4.8 Strength analysis for (a) Femur and (b) Tibia.

Next, the deformation analysis showed a maximum displacement of 4.3×10^{-3} mm for the femur and 5.6×10^{-1} mm for the tibia as shown in Figure 4.9.



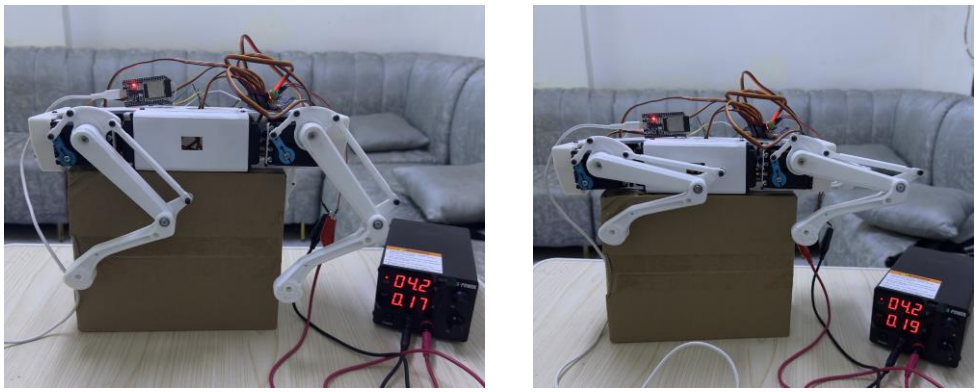
(a) (b)

Figure 4.9 Deformation analysis for (a) Femur and (b) Tibia.

These values are within acceptable limits for the intended application, ensuring that the leg structure will not undergo excessive bending or instability during movement.

4.8 PRELIMINARY PHYSICAL PROTOTYPE TESTING

Before transitioning fully to simulation-based development, an initial physical prototype of the quadruped robot was printed using 3D PLA material as it is the selected material for the quadruped robot. The aim of this stage was to validate the basic mechanical structure, servo integration, and control logic for the robot's fundamental poses and gait. First, two poses were tested which are the standing pose and the rest pose. In both cases, the robot was positioned above the ground to eliminate load stress on the servos during initial trials. The standing pose successfully raised all four legs into the intended upright configuration, while the rest pose lowered the legs into a stable, compact form as seen in Figure 4.10.



(a) (b)

Figure 4.10 The prototype doing (a) standing pose and (b) rest pose.

Following the pose validation, a basic locomotion test was performed using a trot gait. The prototype was able to execute coordinated, alternating leg movements, producing small and fast steps. These initial movements proved that the gait generation logic and servo control system functioned as expected, even though the robot remained above ground for these trials. Finally, the prototype was placed directly on the ground to evaluate its static posture from the front view. The robot maintained its standing pose without structural deformation, confirming that the assembled frame and servo mounts could support its weight under stationary conditions as seen in Figure 4.11.

Figure 4.11 Front view of the prototype standing on the ground.

This early stage of physical testing provided valuable confirmation of the core mechanical design and movement algorithms, laying the groundwork for the more advanced, sensor-integrated simulation work described in the later sections of this thesis.

4.9 COST EVALUATION

One of the essential aims of this thesis is to design a cost-effective quadruped robot that could be used in SAR scenarios, especially in the earthquake affected areas. During the design process, efforts were made to select components that provide a good balance between performance, cost and availability. Table 4.2 shows the key components used in the unfinished prototype as well as the components required to complete the quadruped robot.

Table 4.3 Cost table.

Component	Quantity	Total cost ($)
Ultrasonic sensor	2	5.5
MQ135	1	4.58
MPU6050	1	4.78
ESP32-DevkitC	1	13
Raspberry Pi 4B	1	54.33
RPLIDAR A1	1	84.15
PCA9685	1	8.25
LM2596	1	4.65
ESP32-CAM	1	11.75
684Z bearing	8	8.82
Sleeve bearing 3x5x5	16	3.8
3300mah 7.4v LiPo	1	12.86
SBEC 6A	1	11.13
M3 screw	80	4.5
TD 8325MG servo	12	113.5
ON-OFF switch	1	3
3D printed parts	62	26
Total cost		375.2

Overall, the total cost of the quadruped robot was 375.2$ which is a cost-effective price compared to the other quadruped robot in the market, demonstrating that it is possible to build a functional and autonomous quadruped robot with limited budget.

CONCLUSION AND SUGGESTIONS

To conclude, this thesis sought to design and simulate a cost-effective quadruped robot that is able to assist in exploration and earthquake rescue missions. From the beginning, the focus was to design and build a lightweight, affordable, and autonomous quadruped. The design process began with a comprehensive review of existing quadruped designs, and then there was the step of designing the quadruped and selecting and integrating reliable electronics components such as the ESP32 microcontroller, Raspberry Pi 4B microprocessor, ESP32-CAM, several sensors, and actuators.

A significant advancement in this work was the migration to the Robot Operating System (ROS) for simulation, testing, and algorithm integration. A detailed URDF model was created to accurately represent the robot's mechanical structure and kinematics. Using ROS visualization tools and the CHAMP framework, the quadruped was tested in a realistic Gazebo world resembling earthquake-affected terrain. This allowed for controlled experimentation with mapping using GMapping, localization with AMCL, path planning by A* algorithm, and live video streaming from the simulated onboard camera. These experiments demonstrated the robot's ability to navigate complex terrain, maintain accurate localization, and execute planned paths effectively.

Furthermore, the concept of a Gas Hazard Heatmap Layer was introduced to enhance rescue mission safety. By simulating gas sensor readings, the system proved how hazardous gas zones could be represented on a map, enabling informed decision-making for rescue teams. Although the sensor readings in the simulation were set to zero for testing, the integration process established a foundation for incorporating real gas sensing hardware in the future.

Overall, the results validate the feasibility of building a cost-effective quadruped robot with a budget of $380 that can meet essential functional requirements for search and rescue operations, which is way less than the other quadruped robots in the market such as Spot that costs 75000$. The work

provides a robust platform for future enhancements, such as deploying advanced navigation algorithms, integrating real-time hazard detection, and transitioning from simulation to extended real-world testing.

While this thesis has successfully demonstrated the development and simulation of a functional quadruped robot with autonomous navigation, mapping, localization, path planning, and environmental hazard detection, there remain several opportunities for further enhancement.

One important next step is to transition the tested ROS-based algorithms into extended **real-world trials** with the **physical prototype**, particularly in outdoor and post-disaster-like environments. This would allow for the evaluation of system robustness under real lighting conditions, irregular terrain, and physical obstacles.

To improve stability, a **balancing mechanism** using a **PID (Proportional-Integral-Derivative)** controller can be implemented using the IMU sensor data. This would help maintain body posture during the movement in uneven environments.

Another promising area for expansion is the full integration of the **Gas Hazard Heatmap Layer with real gas sensor data** in the physical robot. This would transform the hazard mapping concept tested in simulation into a deployable safety tool for rescue teams.

REFERENCES

Ajiboye A. T., Opadiji J. F., Yusuf A. O., Popoola J. O. (2021). Analytical determination of load resistance value for MQ-series gas sensors: MQ-6 as case study. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, *19*(2), 575. https://doi.org/10.12928/telkomnika.v19i2.17427

Ananthanarayanan, A., Foong, S., & Kim, S. (2012). A compact two DOF magneto-elastomeric force sensor for a running quadruped. *2012 IEEE International Conference on Robotics and Automation*. https://ieeexplore.ieee.org/document/6225201

Arduino. (n.d.). Arduino Nano datasheet. https://docs.arduino.cc/resources/datasheets/A000005-datasheet.pdf

Atique, M., Sarker, R., & Ahad, A. (2018). Development of an 8DOF quadruped robot and implementation of Inverse Kinematics using Denavit-Hartenberg convention. *Heliyon*. https://doi.org/10.1016/j.heliyon.2018.e01053

Babiuch, M., Foltýnek, P., & Smutný, P.(2019). Using the ESP32 microcontroller for data processing. *International Carpathian Control Conference (ICCC).* https://ieeexplore.ieee.org/document/8765944

Baris ALP. (2021, March 15). *Diy quadruped robot*. https://grabcad.com/library/diy-quadruped-robot-1

Bhattacharya, S., Singla, A., Abhimanyu, Dholakiya, D., Bhatnagar, S., Amrutur, B., Ghosal, A., & Kolathaya, S. (2019). Learning active spine behaviors for dynamic and efficient locomotion in quadruped robots. *IEEE International Workshop on Robot and Human Communication (ROMAN)*. https://ieeexplore.ieee.org/document/8956332

Biswal, P., & Mohanty, P. (2020). Development of quadruped walking robots: A review. *Ain Shams Engineering Journal*. https://doi.org/10.1016/j.asej.2020.11.005

Boston Dynamics. (2020). *Spot® - The Agile Mobile Robot.* https://bostondynamics.com/products/spot/

Boston Dynamics. (2023). *Spot to the Rescue*. https://bostondynamics.com/blog/spot-to-the-rescue/

Briot, S., & Khalil, W. (2015). Calculation of the Number of Degrees of Freedom of Robots with Closed Chains. *Dynamics of Parallel Robots, Mechanisms and Machine Science*. https://doi.org/10.1007/978-3-319-19788-3

Brown, G. (2012). Discovering the STM32 microcontroller.

Chen, J., & Dellaert, F. (2022, November 26). *A1 SLAM: Quadruped SLAM using the A1's Onboard Sensors*. arXiv.org. https://arxiv.org/abs/2211.14432

De Santos, P., Garcia, E., & Estremera, J. (2006). *Quadrupedal locomotion*. Springer-Verlag London Limited.

DeepRobotics Co., Ltd. (2022). *Jueying X20*. https://www.deeprobotics.cn/en/index/product.html

Dey, N., & Mukherjee, A. (2016). *Embedded Systems and Robotics with Open Source Tools*. Taylor & Francis Group, LLC

DFROBOT. (n.d.) ESP-32CAM Development Board. https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0602_Web.pdf

Espressif. (n.d.). ESP32-WROOM-32E ESP32-WROOM-32UE Datasheet. https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf

FEETECH. (2021). SHENZHEN FEETECH RC MODEL CO.,LTD. PRODUCT SPECIFICATION. https://www.pololu.com/file/0J1270/FS5115M-specs.pdf

Firoozian, R. (2014). *Servo Motors and Industrial Control Theory* (2nd edition). In Mechanical engineering series. https://doi.org/10.1007/978-3-319-07275-3

Fukuoka, Y., Kimura, H., Hada, Y., & Takase, K. (2003). Adaptive dynamic walking of a quadruped robot "Tekken" on irregular terrain using a neural system model. *International Conference on Robotics & Automation*.

Ghael, H. D., Solanki, Dr. L., & Sahu, G. (2021). A Review Paper on Raspberry Pi and its Applications. *International Journal of Advances in Engineering and Management (IJAEM)*, 2(12), 225–227.

Hashimoto, K., Matsuzawa, T., Teramachi, T., Uryu, K., Sun, X., Hamamoto, S., Koizumi, A., & Takanishi, A. (2017) A Four-Limbed Disaster-Response robot having high mobility capabilities in extreme environments. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. https://ieeexplore.ieee.org/document/8206436

He, J., Shao, J., Sun, G., & Shao, X. (2019). Survey of Quadruped Robots Coping Strategies in Complex situations. *Electronics*, 8(12) 1414. https://doi.org/10.3390/electronics8121414

Howes Models. (n.d.). TD-8325MG 25KG Metal Gear Waterproof High Torque Digital Servo for RC Hobby Crawler, Truck, Car, Boats, Robots. https://howesmodels.co.uk/product/td-8325mg-25kg-metal-gear-waterproof-high-torque-digital-servo-rc-hobby-crawler-truck-robotic-education/

Hussain, K., Omar, Z., Wang, X., Adewale, O., & Elnour, M. (2021). Analysis and Research of Quadruped Robot's Actuators: A review. *International Journal of Mechanical Engineering and Robotics Research*, 436–442. https://doi.org/10.18178/ijmerr.10.8.436-442

Hutter, M., Gehring, C., Lauber, A., Gunther, F., Bellicoso, C. D., Tsounis, V., Fankhauser, P., Diethelm, R., Bachmann, S., Bloesch, M., Kolvenbach, H., Bjelonic, M., Isler, L., & Meyer, K. (2017). ANYmal - toward legged robots for harsh environments. *Advanced Robotics*, *31*(17), 918–931. https://doi.org/10.1080/01691864.2017.1378591

I2C BUS. (2015). PCA9685 16-channel, 12-bit PWM Fm+ I2C-bus LED controller. https://cdn-shop.adafruit.com/datasheets/PCA9685.pdf

InvenSense. (2013). MPU-6000 and MPU-6050 Product Specification Revision 3.4. https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

Jerome Graves. (2020). *GitHub - Jerome-Graves/yertle: A 3D Printed Quadrupedal Robot for Locomotion Research.* GitHub. https://github.com/Jerome-Graves/yertle

Jimeno, J. M. (2019). *CHAMP: An open-source framework for quadruped robots* [Computer software]. GitHub. https://github.com/chvmp/champ

Kim, Y., Lee, Y., & Cha, Y. (2021). Origami Pump Actuator Based Pneumatic Quadruped Robot (OPARO). *IEEE Access*, *9*. https://doi.org/10.1109/access.2021.3065402

Kitano, S., Hirose, S., Horigome, A., & Endo, G. (2016). TITAN-XIII: sprawling-type quadruped robot with ability of fast and energy-efficient walking. *ROBOMECH Journal*, *3*. https://doi.org/10.1186/s40648-016-0047-1

Knälmann, J., & Saläng, M. (2023). *A study on selfbalancing for a quadruped robot* [Thesis, KTH, Royal institute of technology].

Komarizadehasl, S., Mobaraki, B., Ma, H., Lozano-Galant, J., & Turmo, J. (2022). Low-Cost Sensors accuracy study and enhancement strategy. *Applied Sciences*, *12*(6), 3186. https://doi.org/10.3390/app12063186

Li, Y., Li, B., Ruan, J., & Rong, X. (2011). Research of Mammal Bionic Quadruped Robots: a Review. *IEEE Conference Publication | IEEE Xplore*. https://ieeexplore.ieee.org/document/6070476

Liciotti, D., Paolanti, M., Frontoni, E., & Zingaretti, P. (2017). People Detection and Tracking from an RGB-D Camera in Top-View Configuration: Review of Challenges and Applications. *Springer International Publishing AG*, 208–209. https://doi.org/10.1007/978-3-319-70742-6_20

Liu, Y. Jiang, L. Zou, F. Xing, B. Wang, Z. & Su, B. (2020). Research on path planning of quadruped robot based on globally mapping localization. *IEEE Conference Publication | IEEE Xplore*. https://ieeexplore.ieee.org/document/9275012

Ma, C. (2020). *Embedded system design and joint motion control of a quadruped robot*. [Master thesis, University of Victoria]. https://dspace.library.uvic.ca/items/1d130071-cbe3-4e4e-9a75-f415cb8a1bd3

Majithia, A., Shah, D., Dave, J., Kumar, A., Rathee, S., Dogra, N., M, V. H., Chiniwar, D. S., & Hiremath, S. (2024). Design, motions, capabilities, and applications of quadruped robots: a comprehensive review. *Frontiers in Mechanical Engineering*, *10*. https://doi.org/10.3389/fmech.2024.1448681

Meng, X., Liu, W., Tang, L., Lu, Z., Lin, H., & Fang, J. (2023). Trot Gait Stability control of small quadruped robot based on MPC and ZMP methods. *Processes*, *11*(1), 252. https://doi.org/10.3390/pr11010252

Mouser Electronics. (n.d.). DFRobot STM32F411 BlackPill Development Board. https://www.mouser.com.tr/new/dfrobot/dfrobot-stm32f411-blackpill-board/?srsltid=AfmBOoq8jNxYGJijYBG7gST8p3u1_5vh9Qqdh-wLoVvqEGohzVYbe-RV

Olimex. (n.d.). MQ-135 GAS SENSOR. https://www.olimex.com/Products/Components/Sensors/Gas/SNS-MQ135/resources/SNS-MQ135.pdf

Pololu. (n.d.). MQ-2 Semiconductor Sensor for Combustible Gas. https://www.pololu.com/file/0j309/mq2.pdf

Pololu. (n.d.). MQ-3 Semiconductor Sensor for Alcohol. https://www.pololu.com/file/0j310/mq3.pdf

Pongas, D., Mistry, M., & Schaal, S. (2007) A robust quadruped walking gait for traversing rough terrain. *2007 IEEE International Conference on Robotics and Automation*

Pratt, J., & Krupp, B. (2004). Series Elastic Actuators for legged robots. *Unmanned Ground Vehicle Technology VI*, 135. https://doi.org/10.1117/12.548000

Prayogo, R. C., Triwiyatno, A., & Sumardi. (2018). Quadruped Robot with Stabilization Algorithm on Uneven Floor using 6 DOF IMU based Inverse Kinematic. *IEEE Conference Publication | IEEE Xplore*. https://ieeexplore.ieee.org/document/8576969

Prayogo, R. C., Triwiyatno, A., & Sumardi. (2021). Quadruped Robot with Stabilization Algorithm on Uneven Floor using 6 DOF IMU based Inverse Kinematic. *Diponegoro University*.

Rahman, M. H., Alam, S. B., Mou, T. D., Uddin, M. F., & Hasan, M. (2023). A dynamic approach to Low-Cost design, development, and computational simulation of a 12DOF quadruped robot. *Robotics*, *12*(1), 28. https://doi.org/10.3390/robotics12010028

Raj, T., Hashim, F. H., Huddin, A. B., Ibrahim, M. F., & Hussain, A. (2020). A survey on LiDAR scanning mechanisms. *Electronics* 9(1), 741. https://doi.org/10.3390/electronics9050741

Ranjane, R., Chavan, R., Mahajan, R. (2021). Quadruped Robot with Speed and Direction control. In *International Research Journal of Engineering and Technology (IRJET)* (p. 1965).

Reacher Technology Co.,Ltd. (n.d.). SPT5435LV - 180W. https://reacherbrushless.com/product/product-19-462.html

Robotics 24/7 staff (2022, July 26). DEEP Robotics releases Jueying X20 quadruped robot for hazard detection and rescue. *Robotics 24/7*. https://www.robotics247.com/article/deep_robotics_releases_jueying_x20_quadruped_robot_for_hazard_detection_and_rescue#:~:text=DEEP%20Robotics%20says%20its%20Jueying,search%2Dand%2Drescue%20efforts.&text=DEEP%20Robotics%20Co.%20has%20announced,and%20emergency%20search%20and%20rescue.

Robotlocomotion. (2000). Quadruped Robot "Tekken". http://www.robotlocomotion.kit.ac.jp/research/Quadruped/photo-movie-tekken-e.html

Robotsguide. (n.d.). *ANYmal*. https://robotsguide.com/robots/anymal

RunBin, C., YangZhen, C., WenQi, H., Jiang, W., & HongXu, M. (2013). Trotting Gait of a quadruped robot based on the Time-Pose control method. *International Journal of Advanced Robotic Systems*. https://doi.org/10.5772/50979

Ryu, H. (2020). Graph Search-Based exploration Method using a Frontier-Graph structure for mobile robots. *Sensors*, *20*(21), 6270. https://doi.org/10.3390/s20216270

Shi, Y., Li, S., Guo, M., Yang, Y., Xia, D., & Luo, X. (2021). Structural design, simulation and experiment of quadruped robot. *Applied Sciences*, *11*(22), 10705. https://doi.org/10.3390/app112210705

Su, Z., Zhou, J., Dai, J., & Yongguo Zhu. (2020). Optimization Design and Experimental Study of Gmapping Algorithm [Optimization Design and Experimental Study]. *IEEE*, 4894. https://ieeexplore.ieee.org/document/9122020

Suzuki, S., Kano, T., Ijspeert, A. J., & Ishiguro, A. (2021). Sprawling quadruped robot driven by decentralized control with Cross-Coupled sensory feedback between legs and trunk. *Frontiers in Neurorobotics*, 14. https://doi.org/10.3389/fnbot.2020.607455

Suzumori, K., & Faudzi, A. (2018). Trends in hydraulic actuators and components in legged and tough robots: a review. *Advanced Robotics*, *32*(9), 458–476. https://doi.org/10.1080/01691864.2018.1455606

Texas Instruments. (2023). LM2596 SIMPLE SWITCHER® Power Converter 150-kHz 3-A Step-Down Voltage Regulator. https://www.ti.com/lit/ds/symlink/lm2596.pdf

Tranzatto, M., Mascarich, F., Bernreiter, L., Godinho, C., Camurri, M., Khattak, S., Dang, T., Reijgwart, V., Loeje, J., Wisth, D., Zimmermann, S., Nguyen, H., Fehr, M., Solanka, L., Buchanan, R., Bjelonic, M., Khedekar, N., Valceschini, M., Jenelten, F., Alexis, K. (2022, January 18). CERBERUS: Autonomous legged and aerial robotic exploration in the tunnel and urban circuits of the DARPA Subterranean Challenge. *arXiv*. https://arxiv.org/abs/2201.07067

Visconti, P., & Primiceri, P. (2017). An overview on state-of-art and future application fields of BLDC motors: Design and characterization of a PC-interfaced driving and motion control system. *ARPN Journal of Engineering and Applied Sciences*. https://www.researchgate.net/publication/319873677

Xu, B. (2024). Precise path planning and trajectory tracking based on improved A-star algorithm. *Measurement and Control*, *57*(8), 1025–1037. https://doi.org/10.1177/00202940241228725

Zhang, J., Wang, X., & Zheng, L. (2024). Research on Autonomous Navigation system of agricultural quadruped Robot. *2022 IEEE International Conference on Mechatronics and Automation (ICMA)*, 1286–1290. https://doi.org/10.1109/icma61710.2024.10633192

Zhmud, V., Kondratiev, N., Kuznetsov, K., Trubin, V., & Dimitrov, L. (2018). Application of ultrasonic sensor for measuring distances in robotics. *Journal of Physics Conference Series*, *1015*, 032189. https://doi.org/10.1088/1742-6596/1015/3/032189

Zhou, Z., Zhang, C., Li, C., Zhang, Y., Shi, Y., & Zhang, W. (2024). A tightly-coupled LIDAR-IMU SLAM method for quadruped robots. *Measurement and Control, 57(7)*, 1004–1013. https://doi.org/10.1177/00202940231224593

APPENDICES

APPENDIX A. GAS HAZARD HEATMAP PUBLISHER CODE

```python
import rospy
import numpy as np
from nav_msgs.msg import OccupancyGrid
from geometry_msgs.msg import PoseWithCovarianceStamped
from std_msgs.msg import Float32

class GasHazardMap:
    def __init__(self):
        rospy.init_node("gas_hazard_map_node")

        # Map settings (should match your SLAM map resolution & size)
        self.map_resolution = 0.05  # meters per cell
        self.map_width = 1000       # number of cells
        self.map_height = 1000
        self.origin_x = -25.0       # meters
        self.origin_y = -25.0

        # Initialize map (all 0 = safe)
        self.hazard_map = np.zeros((self.map_height, self.map_width), dtype=np.int8)

        # Subscribers
        rospy.Subscriber("/gas_sensor_ppm", Float32, self.gas_callback)
        rospy.Subscriber("/amcl_pose", PoseWithCovarianceStamped, self.pose_callback)

        # Publisher
        self.map_pub = rospy.Publisher("/gas_hazard_map", OccupancyGrid, queue_size=1)

        # Store latest readings
        self.latest_ppm = 0
        self.robot_x = 0
        self.robot_y = 0

        rospy.Timer(rospy.Duration(1.0), self.publish_map)
        rospy.loginfo("Gas Hazard Map Node Started")
        rospy.spin()

    def gas_callback(self, msg):
        self.latest_ppm = msg.data

    def pose_callback(self, msg):
        self.robot_x = msg.pose.pose.position.x
        self.robot_y = msg.pose.pose.position.y

        # Update hazard map cell for robot's position
        mx = int((self.robot_x - self.origin_x) / self.map_resolution)
        my = int((self.robot_y - self.origin_y) / self.map_resolution)

        if 0 <= mx < self.map_width and 0 <= my < self.map_height:
            self.hazard_map[my, mx] = self.ppm_to_hazard_level(self.latest_ppm)

    def ppm_to_hazard_level(self, ppm):
        if ppm < 50:
            return 0     # Safe (no color)
        elif 50 <= ppm < 100:
            return 85    # Yellow (light caution)
        elif 100 <= ppm < 200:
            return 170   # Orange (moderate hazard)
        else:
            return 255   # Red (severe hazard)

    def publish_map(self, event):
        grid = OccupancyGrid()
        grid.header.stamp = rospy.Time.now()
        grid.header.frame_id = "map"

        grid.info.resolution = self.map_resolution
        grid.info.width = self.map_width
        grid.info.height = self.map_height
        grid.info.origin.position.x = self.origin_x
        grid.info.origin.position.y = self.origin_y

        # Flatten map array
        grid.data = self.hazard_map.flatten().tolist()

        self.map_pub.publish(grid)

if __name__ == "__main__":
    try:
        GasHazardMap()
    except rospy.ROSInterruptException:
        pass
```

APPENDIX B: URDF CODE FOR THE BODY AND ONE LEG

```xml
<?xml version="1.0"?>
<robot name="quadruped">

  <material name="black">
    <color rgba="0.3 0.3 0.3 1" />
  </material>
  <material name="blue">
    <color rgba="0 0 0.8 1" />
  </material>
  <material name="grey">
    <color rgba="0.6 0.6 0.5 1" />
  </material>

  <link name="base_link">
    <visual>
      <geometry>
        <mesh filename="package://quadruped_description/models/frame222.stl" scale="1.7 1.7 1.7"/>
      </geometry>
      <origin xyz="0.155 -0.323 0.272" rpy="1.5708 0 1.5708" />
      <material name="grey" />
    </visual>
    <collision>
      <geometry>
        <mesh filename="package://quadruped_description/models/frame222.stl" scale="1.7 1.7 1.7"/>
      </geometry>
      <origin xyz="0.155 -0.323 0.272" rpy="1.5708 0 1.5708" />
    </collision>
  </link>

  <link name="base_inertia">
    <inertial>
      <mass value="0.5"/>
      <inertia ixx="0.01" ixy="0" ixz="0" iyy="0.01" iyz="0" izz="0.01"/>
    </inertial>
  </link>
  <joint name="base_link_to_base_inertia" type="fixed">
    <parent link="base_link"/>
    <child link="base_inertia"/>
    <origin rpy="0 0 0" xyz="0 0 0"/>
  </joint>


  <gazebo reference="base_link">
    <material>Gazebo/Grey</material>
  </gazebo>

  <joint name="hokuyo_joint" type="fixed">
    <origin rpy="0 0 0" xyz="0.0 0.0 0.08"/>
    <parent link="base_link"/>
    <child link="hokuyo_frame"/>
  </joint>
  <link name="hokuyo_frame">
    <inertial>
      <mass value="0.0270"/>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <inertia ixx="2.632e-4" ixy="0" ixz="0" iyy="2.632e-4" iyz="0" izz="1.62e-4"/>
    </inertial>
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://hector_sensors_description/meshes/hokuyo_utm30lx/hokuyo_utm_30lx.dae"/>
      </geometry>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 -0.0115"/>
      <geometry>
        <box size="0.058 0.058 0.087"/>
      </geometry>
    </collision>
  </link>
  <gazebo reference="hokuyo_frame">
    <sensor name="hokuyo" type="ray">
      <always_on>true</always_on>
      <update_rate>30</update_rate>
      <pose>0 0 0 0 0 0</pose>
      <visualize>false</visualize>
      <ray>
        <scan>
          <horizontal>
            <samples>1040</samples>
            <resolution>1</resolution>
```

```
            <min_angle>-2.26892802759</min_angle>
            <max_angle>2.26892802759</max_angle>
          </horizontal>
        </scan>
        <range>
          <min>0.2</min>
          <max>30.0</max>
          <resolution>0.01</resolution>
        </range>
        <noise>
          <type>gaussian</type>
          <mean>0.0</mean>
          <stddev>0.004</stddev>
        </noise>
      </ray>
      <plugin filename="libgazebo_ros_laser.so" name="gazebo_ros_hokuyo_controller">
        <topicName>scan</topicName>
        <frameName>hokuyo_frame</frameName>
      </plugin>
    </sensor>
  </gazebo>

  <link name="imu_link">
    <inertial>
      <mass value="0.001"/>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <inertia ixx="1e-09" ixy="0.0" ixz="0.0" iyy="1e-09" iyz="0.0" izz="1e-09"/>
    </inertial>
  </link>
  <joint name="imu_joint" type="fixed">
    <parent link="base_link"/>
    <child link="imu_link"/>
  </joint>
  <gazebo>
    <plugin filename="libhector_gazebo_ros_imu.so" name="imu_controller">
      <updateRate>50.0</updateRate>
      <bodyName>imu_link</bodyName>
      <topicName>imu/data</topicName>
      <accelDrift>0.005 0.005 0.005</accelDrift>
      <accelGaussianNoise>0.005 0.005 0.005</accelGaussianNoise>
      <rateDrift>0.005 0.005 0.005 </rateDrift>
      <rateGaussianNoise>0.005 0.005 0.005 </rateGaussianNoise>
      <headingDrift>0.005</headingDrift>
      <headingGaussianNoise>0.005</headingGaussianNoise>
    </plugin>
  </gazebo>

  <!-- Left Front Leg -->
  <link name="lf_shoulder">
    <visual>
      <geometry>
        <mesh filename="package://quadruped_description/models/lf_shoulder2.stl" scale="1.7 1.7 1.7"/>
      </geometry>
      <origin xyz="0 0 0" rpy="3.141 3.141 0"/>
      <material name="grey"/>
    </visual>
    <collision>
      <geometry>
        <mesh filename="package://quadruped_description/models/lf_shoulder2.stl" scale="1.7 1.7 1.7"/>
      </geometry>
      <origin xyz="0 0 0" rpy="3.141 3.141 0"/>
    </collision>
    <inertial>
      <mass value="0.25"/>
      <inertia ixx="0.0001" ixy="0" ixz="0" iyy="0.0001" iyz="0" izz="0.0001"/>
    </inertial>
  </link>
  <gazebo reference="lf_shoulder">
    <material>Gazebo/Grey</material>
  </gazebo>

  <joint name="lf_shoulder_joint" type="revolute">
    <parent link="base_link"/>
    <child link="lf_shoulder"/>
    <origin xyz="0.088 0.0714 0" rpy="0 0 0"/>
    <axis xyz="1 0 0">
      <dynamics damping="2.5" friction="0.1"/>
    </axis>
    <limit lower="-1.57" upper="1.57" effort="5.0" velocity="1.0"/>
  </joint>

  <transmission name="lf_shoulder_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <actuator name="lf_shoulder_motor">
      <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
    <joint name="lf_shoulder_joint">
      <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
    </joint>
  </transmission>


  <link name="lf_thigh">
    <visual>
      <geometry>
        <mesh filename="package://quadruped_description/models/upper_leg.stl"  scale="0.001 0.001 0.001"/>
```

```
    </geometry>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <material name="grey"/>
  </visual>
  <collision>
    <geometry>
      <mesh filename="package://quadruped_description/models/upper_leg.stl"  scale="0.001 0.001 0.001"/>
    </geometry>
    <origin xyz="0 0 0" rpy="0 0 0"/>
  </collision>
  <inertial>
    <origin xyz="0 0 -0.0705"/>
    <mass value="0.125"/>
    <inertia ixx="0.00038739" ixy="0.0" ixz="0.0" iyy="0.00040406" iyz="0.0" izz="3.54166e-05"/>
  </inertial>
</link>
<gazebo reference="lf_thigh">
  <material>Gazebo/Grey</material>
</gazebo>

<joint name="lf_thigh_joint" type="revolute">
  <parent link="lf_shoulder"/>
  <child link="lf_thigh"/>
  <origin xyz="0.04 0.058 0" rpy="0 0 0"/>
  <axis xyz="0 1 0">
    <dynamics damping="2.5" friction="0.1"/>
  </axis>
  <limit lower="-1.57" upper="1.57" effort="5.0" velocity="1.0"/>
</joint>

<transmission name="lf_thigh_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <actuator name="lf_thigh_motor">
    <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
  <joint name="lf_thigh_joint">
    <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
  </joint>
</transmission>


<link name="lf_shin">
  <visual>
    <geometry>
      <mesh filename="package://quadruped_description/models/lower_leg.stl"  scale="0.001 0.001 0.001"/>
    </geometry>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <material name="grey"/>
  </visual>
  <collision>
    <geometry>
      <mesh filename="package://quadruped_description/models/lower_leg.stl"  scale="0.001 0.001 0.001"/>
    </geometry>
    <origin xyz="0 0 0" rpy="0 0 0"/>
  </collision>
  <inertial>
    <origin xyz="0 0 -0.063"/>
    <mass value="0.125"/>
    <inertia ixx="0.00025854" ixy="0.0" ixz="0.0" iyy="0.00026934" iyz="0.0" izz="2.08854e-05"/>
  </inertial>
</link>
<gazebo reference="lf_shin">
  <material>Gazebo/Grey</material>
  <kp>1000000.0</kp>
  <kd>1.0</kd>
  <mu1>0.8</mu1>
  <mu2>0.8</mu2>
  <maxVel>0.0</maxVel>
  <minDepth>0.001</minDepth>
</gazebo>

<joint name="lf_shin_joint" type="revolute">
  <parent link="lf_thigh"/>
  <child link="lf_shin"/>
  <origin xyz="0 0 -0.141" rpy="0 0 0"/>
  <axis xyz="0 1 0">
    <dynamics damping="2.5" friction="0.1"/>
  </axis>
  <limit lower="-1.57" upper="1.57" effort="5.0" velocity="1.0"/>
</joint>

<transmission name="lf_shin_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <actuator name="lf_shin_motor">
    <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
  <joint name="lf_shin_joint">
    <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
  </joint>
</transmission>


<link name="lf_foot_link"/>

<joint name="lf_foot_joint" type="fixed">
  <parent link="lf_shin"/>
  <child link="lf_foot_link"/>
  <origin xyz="0 0 -0.141" rpy="0 0 0"/>
</joint>
```

APPENDIX C: FULL SCHEMATIC DIAGRAM

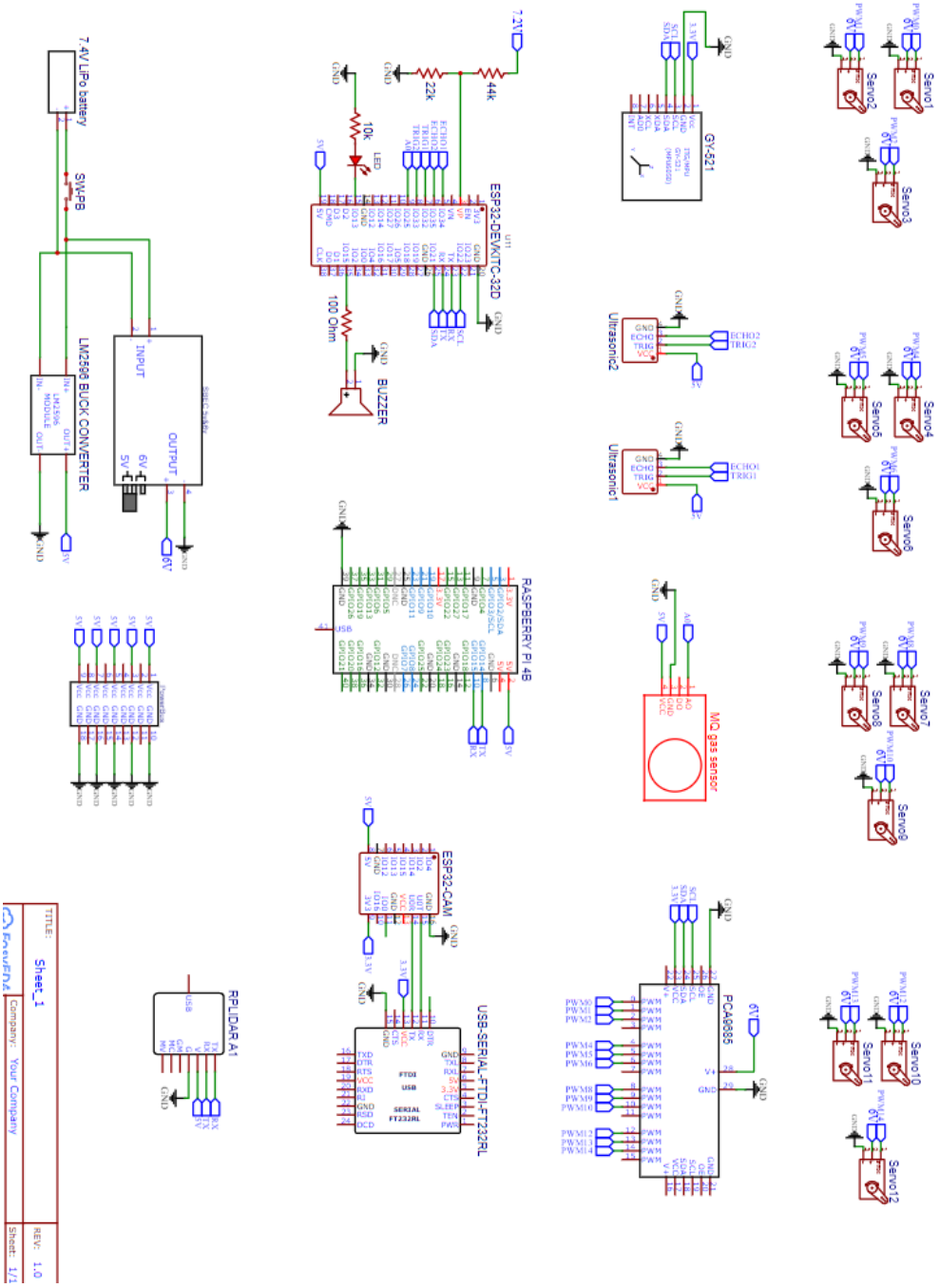


Figure C.1 Full Schematic Diagram

CURRICULUM VITAE