ವಿಶ್ವೇಶ್ವರಯ್ಯ ತಾಂತ್ರಿಕ ವಿಶ್ವವಿದ್ಯಾಲಯ

("ವಿ ಟಿ ಯು ಅಧಿನಿಯಮ ೧೯೯೪" ರ ಅಡಿಯಲ್ಲಿ ಕರ್ನಾಟಕ ಸರ್ಕಾರದಿಂದ ಸ್ಥಾಪಿತವಾದ ರಾಜ್ಯ ವಿಶ್ವವಿದ್ಯಾಲಯ)

"ಜ್ಞಾನ ಸಂಗಮ", ಬೆಳಗಾವಿ-590018, ಕರ್ನಾಟಕ, ಭಾರತ

Visvesvaraya Technological University

(State University of Government of Karnataka. Established as per the VTU Act, 1994)

"Jnana Sangama" Belagavi-590018, Karnataka, India.

A PROJECT REPORT ON

"Design and Development of a Hybrid Quadruped Robot with Wheeled Locomotion for Terrain Navigation"

Submitted in the partial fulfillment for the award of

BACHELOR OF TECHNOLOGY

in

ROBOTICS AND AUTOMATION

Submitted by

Mr. OM PATIL	**2VX22RA010**
Mr. OMKAR PUJAR	**2VX22RA011**
Mr. OMKAR PATIL	**2VX22RA012**
Mr. PREETAM PATIL	**2VX23RA403**
Mr. RAGHAV DALAWAI	**2VX23RA404**

Under the Guidance of

Dr. Babu Reddy

Chairperson

Department of Mechanical Engineering, VTU, Belagavi-590018

Co-Guidance of

Mr. Ravichandra S. D

Assistant Professor, (Ad-hoc)

Department of Mechanical Engineering, VTU, Belagavi-590018

2025-2026

ವಿಶ್ವೇಶ್ವರಯ್ಯ ತಾಂತ್ರಿಕ ವಿಶ್ವವಿದ್ಯಾಲಯ

("ವಿ ಟಿ ಯು ಅಧಿನಿಯಮ ೧೯೯೪" ರ ಅಡಿಯಲ್ಲಿ ಕರ್ನಾಟಕ ಸರ್ಕಾರದಿಂದ ಸ್ಥಾಪಿತವಾದ ರಾಜ್ಯ ವಿಶ್ವವಿದ್ಯಾಲಯ)
"ಜ್ಞಾನ ಸಂಗಮ", ಬೆಳಗಾವಿ-590018, ಕರ್ನಾಟಕ, ಭಾರತ

Visvesvaraya Technological University

(State University of Government of Karnataka. Established as per the VTU Act, 1994)
"Jnana Sangama" Belagavi-590018, Karnataka, India.

DEPARTMENT OF MECHANICAL ENGINEERING

CERTIFICATE

This is to certify that the Project (BRA786) work, title **"Design and Development of a Hybrid Quadruped Robot with Wheeled Locomotion for Terrain Navigation"** is a Bonafide work completed by the students of the Department of Mechanical Engineering, Visvesvaraya Technological University, "Jnana Sangama," Belagavi, in partial fulfillment for the award of Bachelor of Technology in Robotics and Automation of Visvesvaraya Technological University, Belagavi, during the academic year 2025–26. It is certified that all the corrections or suggestions have been approved as they satisfy the academic requirements concerning the project work prescribed by VTU, Belagavi, for the said degree.

Mr. OM PATIL	**2VX22RA010**
Mr. OMKAR PUJAR	**2VX22RA011**
Mr. OMKAR PATIL	**2VX22RA012**
Mr. PREETAM PATIL	**2VX23RA403**
Mr. RAGHAV DALAWAI	**2VX23RA404**

Co Guide | Chairperson

(Mr.Ravichandra S. D) | (Dr. Babu Reddy)

Name of the Examiners | Signature with date

1)____

2)____

ವಿಶ್ವೇಶ್ವರಯ್ಯ ತಾಂತ್ರಿಕ ವಿಶ್ವವಿದ್ಯಾಲಯ

("ವಿ ಟಿ ಯು ಅಧಿನಿಯಮ ೧೯೯೪" ರ ಅಡಿಯಲ್ಲಿ ಕರ್ನಾಟಕ ಸರ್ಕಾರದಿಂದ ಸ್ಥಾಪಿತವಾದ ರಾಜ್ಯ ವಿಶ್ವವಿದ್ಯಾಲಯ)

"ಜ್ಞಾನ ಸಂಗಮ", ಬೆಳಗಾವಿ-590018, ಕರ್ನಾಟಕ, ಭಾರತ

Visvesvaraya Technological University

(State University of Government of Karnataka. Established as per the VTU Act, 1994)

"Jnana Sangama" Belagavi-590018, Karnataka, India.

DECLARATION

We, the students of seventh semester B-Tech Robotics and Automation, Visvesvaraya Technological University, Belagavi, hereby declare that this project report entitled **"Design and Development of a Hybrid Quadruped Robot with Wheeled Locomotion for Terrain Navigation"** embodies the details of our project completed at the Department of Mechanical Engineering, Visvesvaraya Technological University, "Jnana Sangama," Belagavi, under the guidance of **Dr. Babu Reddy, with Chairperson of the Department of Mechanical Engineering**, **Visvesvaraya Technological University, Belagavi.** This report is submitted in partial fulfillment of the award of Bachelor of Technology in Robotics and Automation at Visvesvaraya Technological University, Belagavi, during the academic year 2025–26.

Mr. OM PATIL	**2VX22RA010**
Mr. OMKAR PUJAR	**2VX22RA011**
Mr. OMKAR PATIL	**2VX22RA012**
Mr. PREETAM PATIL	**2VX23RA403**
Mr. RAGHAV DALAWAI	**2VX23RA404**

Place: Belagavi

Date:

ACKNOWLEDGEMENT

I would like to express my gratefulness to almighty God for giving me strength and wisdom in my work. I have achieved a good amount of knowledge through this work and also got help from many others. I owe my gratitude to all those who made this mini project work possible and made my experience cherishable forever.

I express my deep sense of gratitude and indebtedness to my Project Guide **Dr. Babu Reddy**, Chairperson, Department of Mechanical Engineering, V.T.U. Belagavi, for providing precious guidance, inspiring discussion and constant supervision throughout the course of this work. His timely help, constructive criticism, and conscientious efforts made it possible to present the work contained in this thesis.

I'm thankful to **Mr. Ravichandra S. D**, Professor, Department of Mechanical Engineering,V.T.U. Belagavi, for contributing his valuable time and efforts throughout the course of this work.

I am also thankful to all the staff members of Department of Mechanical Engineering and to all my well-wishers for their inspiration and help. Also, I would thank my friends and family who supported me emotionally as well as academically to complete this project without any obstacles.

ABSTRACT

Hybrid robotic locomotion systems are emerging as a transformative solution for navigating complex and unstructured terrains, combining the agility of legged systems with the efficiency of wheeled mechanisms. This project focuses on the design and development of a hybrid quadruped robot integrated with wheeled locomotion to achieve superior adaptability in terrain navigation scenarios. The hybrid design enables the robot to switch seamlessly between rolling on smooth surfaces and walking over irregular, rugged terrains where wheels alone would fail.

The development process involves the mechanical design of leg-wheel assemblies, employing lightweight and durable materials to optimize both load capacity and energy efficiency. Advanced kinematic and dynamic models are formulated to govern the coordinated movement of the legs and wheels, ensuring stability, balance, and smooth transition between locomotion modes. Control algorithms are developed using PID controllers and gait planning strategies, supported by real-time terrain sensing.

Simulation environments such as ROS and Gazebo are utilized to validate the robot's performance under varied conditions, allowing for iterative refinement of its locomotion strategies. The final prototype is tested across multiple terrains including gravel, sand, uneven surfaces, and inclined planes to assess its real-world performance.

The outcomes of this research demonstrate that hybrid quadruped robots can significantly enhance the versatility and operational range of mobile robots in challenging environments. This work paves the way for the deployment of such systems in applications ranging from search and rescue operations to industrial inspections, offering a scalable, adaptive solution to modern robotic mobility challenges.

TABLE OF CONTENTS

LIST OF FIGURES:

LIST OF TABLES:

CHAPTER 1

INTRODUCTION

1.1 Overview

With the advancement of robotics and mobility technologies, there has been an increasing interest in developing mobile robots capable of traversing diverse and challenging terrains. Traditional mobile robots often rely solely on wheels or legs, each having its own limitations: wheeled robots struggle on rough, uneven terrains while legged robots tend to be slower, less energy-efficient, and more complex to control.

A hybrid locomotion system, combining the advantages of both legged and wheeled mechanisms, provides a promising solution for terrain navigation. The hybrid quadruped robot integrates wheels into its leg structure, allowing it to quickly roll over smooth surfaces and efficiently step over obstacles or traverse complex terrains when necessary. This dual capability expands the robot's operational range and versatility across various real-world applications including disaster response, search and rescue, military reconnaissance, industrial inspection, and planetary exploration.

Over the past decade, many industries — such as defence, logistics, construction, mining, and space exploration — have shown growing interest in such hybrid systems due to their unmatched adaptability, efficiency, and ability to operate in unpredictable environments. This has led to extensive research and development efforts to design robots that can intelligently switch between different modes of locomotion depending on the terrain and task requirements.

The complexity of hybrid quadruped robots arises from their unique combination of mechanical design, control algorithms, kinematic analysis, and real-time terrain sensing. Unlike purely wheeled or legged systems, hybrid quadrupeds require sophisticated motion planning, balancing algorithms, and actuation strategies to ensure stable and efficient operation across different terrains.

This project focuses on the design and development of a hybrid quadruped robot with wheeled locomotion for terrain navigation. The robot is designed to operate efficiently on both flat and irregular surfaces by employing wheels for high-speed movement on even terrain and legs for obstacle negotiation and stability on rough surfaces. The integration of these two systems poses significant challenges in terms of mechanical integration, control system design, sensor fusion, and locomotion planning.

Advanced simulation tools such as Robot Operating System (ROS), Gazebo, and RViz are employed to model, test, and refine the robot's locomotion strategies before hardware implementation. These tools allow detailed analysis of joint kinematics, stability, and energy consumption across multiple terrains, ensuring optimal system performance.

Several path-planning and motion control algorithms — including gait planning, terrain-adaptive walking, and wheel-leg mode switching — are explored to achieve smooth and efficient locomotion. Real-time feedback from onboard sensors (IMU, LIDAR, force sensors, cameras) allows the robot to dynamically adapt its locomotion strategy based on environmental conditions.

In conclusion, this project aims to bridge the gap between the limitations of traditional locomotion systems by developing a highly adaptive hybrid quadruped robot. The outcome of this work will contribute to the ongoing advancements in the field of mobile robotics and enable practical deployment of hybrid locomotion robots in real-world complex and dynamic environments.

1.2Background

Mobile robots have become increasingly important in a wide range of applications, including exploration, search and rescue, agriculture, and military operations. Traditional wheeled robots offer high speed and energy efficiency on flat, even surfaces but struggle to navigate rough, uneven, or obstacle-filled terrains. On the other hand, legged robots, such as quadrupeds, provide superior adaptability and stability over challenging terrain but are often limited by slower speeds and higher energy consumption.

Hybrid locomotion systems, combining wheeled and legged mobility, have emerged as a promising solution to leverage the strengths of both approaches. Hybrid quadruped robots integrate wheels with legged mechanisms, enabling them to efficiently traverse flat surfaces using wheels while retaining the ability to walk and negotiate obstacles when required. This combination expands the operational capability and versatility of mobile robots for terrain navigation.

The development of hybrid quadruped robots requires advances in mechanical design, control algorithms, and locomotion planning to ensure seamless transitions between walking and rolling modes. This background sets the foundation for designing novel hybrid robots that can effectively address diverse environmental challenges, enhancing robotic mobility in complex real-world scenarios.

1.3 Literature Review

The increasing demand for versatile robotic platforms capable of navigating diverse and complex terrains has driven research towards hybrid locomotion systems combining legged and wheeled mobility. Hybrid quadruped robots leverage the stability and adaptability of legged locomotion while incorporating wheels to improve speed and energy efficiency on relatively flat surfaces. This review critically examines key contributions relevant to the design, control, and development of hybrid quadruped robots with wheeled locomotion for terrain navigation.

Traditional Quadruped Robots

Quadruped robots have long been studied for their ability to traverse rough, uneven terrain with stability and agility. Early works such as Boston Dynamics' Spot robot and MIT's Cheetah demonstrated advanced legged locomotion using complex kinematics and control algorithms (Raibert et al., 2008; Kim et al., 2019). These robots excel in obstacle negotiation and uneven ground but often suffer from limited speed and high energy consumption due to complex gait cycles and actuator demands.

Wheeled Locomotion and Hybrid Approaches

Wheeled robots offer advantages in speed and efficiency on smooth surfaces but struggle with uneven or soft terrain. To overcome the limitations of pure wheeled or legged systems, researchers have developed hybrid locomotion platforms integrating wheels with legs. Hybrid robots can switch or combine locomotion modes depending on terrain conditions, balancing speed, stability, and energy efficiency.

For instance, researchers like Focchi et al. (2017) explored wheeled-legged robots with independently actuated wheels embedded in legs, allowing smooth transitions between rolling and stepping. Similarly, Kim and Oh (2020) proposed a hybrid quadruped robot design with actuated wheels at the feet, enabling seamless switching between walking and driving modes. These studies highlighted the importance of adaptable locomotion strategies and advanced control algorithms for effective terrain navigation.

Control Strategies and Navigation for Hybrid Robots

Effective navigation of terrain environments requires sophisticated control frameworks capable of real-time gait adaptation, balance maintenance, and path planning. Lee et al. (2019) developed a hybrid locomotion control system employing sensor fusion and model predictive control to dynamically switch between wheeled and legged gaits based on terrain classification.

Furthermore, research by Wu et al. (2021) emphasized the integration of perception systems and terrain mapping to enable autonomous decision-making for locomotion mode selection. They demonstrated that hybrid quadruped robots could efficiently traverse complex environments by leveraging wheeled locomotion on flat areas and legged locomotion on obstacles or rough ground.

Mechanical Design Considerations

Designing a hybrid quadruped robot with wheeled locomotion poses mechanical challenges, such as integrating wheels without compromising leg dexterity, ensuring structural robustness, and maintaining low weight. Works by Ma et al. (2018) investigated the mechanical design trade-offs, focusing on compact wheel actuators embedded in leg joints and suspension systems to absorb shocks.

The use of lightweight materials and modular designs has also been explored to enhance maneuverability and ease of maintenance (Zhao et al., 2020).

Simulation and Prototyping Tools

Simulation environments like Gazebo, ROS, and MATLAB/Simulink are widely used for validating hybrid locomotion algorithms before hardware implementation. These platforms allow testing of gait transitions, wheel-leg coordination, and sensor feedback integration under various virtual terrains (Chen et al., 2020).

Rapid prototyping with 3D printing and modular electronics has accelerated development cycles, enabling iterative design improvements based on experimental feedback (Singh et al., 2021).

Summary of Gaps and Contributions

Despite significant advances, several challenges remain in the design and development of hybrid quadruped robots with wheeled locomotion:

- Seamless integration of wheel and leg actuation while maintaining stability during mode transitions requires further research.
- Real-time adaptive control algorithms that robustly handle unpredictable terrain variations are still under development.
- Comprehensive workflows combining mechanical design, control, simulation, and field testing are limited.
- Energy efficiency optimization for prolonged operations in mixed-terrain environments needs more attention.

This study addresses these gaps by proposing a novel hybrid quadruped robot design incorporating wheeled locomotion optimized for terrain navigation. It develops an integrated control framework and validates the system through simulation and experimental trials, contributing towards practical, versatile robotic platforms for challenging environments.

1.4 Problem Statement

In traditional mobile robotics, locomotion planning often relies on simplified coordinate systems and linear motion assumptions, which work well for wheeled robots operating on flat terrain. However, hybrid quadruped robots with wheeled locomotion utilize complex joint-based movements across multiple degrees of freedom (DOF) to achieve versatile terrain navigation. Directly applying conventional path-planning methods designed for wheeled-only robots to such hybrid systems often leads to significant challenges, including joint-limit violations, unreachable configurations, abrupt gait transitions, and instability during mode switching.

These issues can cause inefficient locomotion, decreased stability, and failure to navigate challenging environments effectively. Moreover, current locomotion planning algorithms and control software frequently do not account for the unique kinematics and mechanical constraints inherent to hybrid quadruped robots with wheels. Addressing these limitations is critical for enabling smooth transitions between walking and rolling, improving navigation performance, and realizing the full potential of hybrid terrain robotic platforms.

1.5 Objectives

- To design and develop a hybrid quadruped robot integrating wheeled locomotion for enhanced terrain navigation. To generate path points using surface sampling on 3D CAD models
- To develop effective path planning algorithms that enable smooth transitions between walking and rolling modes. To use inverse kinematics to assign pathways to robot joints
- To generate optimized locomotion trajectories using surface sampling and terrain mapping techniques
- To apply inverse kinematics for accurate joint and wheel control in complex terrains.

- To utilize simulation tools such as ROS, Gazebo, and RViz for validating robot motion and navigation strategies in diverse environments.

1.6 Scope of the Project

- The project includes the complete workflow from mechanical design, modeling, and simulation to the physical implementation of the hybrid quadruped robot with wheeled locomotion.
- It is physically implemented using a quadruped robotic platform equipped with wheels integrated into its legs for hybrid mobility.
- It incorporates path planning, gait generation, and terrain analysis using simulation environments like ROS, Gazebo, and Python-based scripting.
- Using inverse kinematics customized for hybrid locomotion, it transforms terrain-based navigation paths into joint-space and wheel-space trajectories.
- The system utilizes hardware components such as motorized wheels, articulated legs, sensors for terrain mapping, and an onboard control unit.
- Robot motion is planned, simulated, and validated in virtual environments using ROS, MoveIt, RViz, and Gazebo before physical execution.
- With optimal motion control algorithms, the locomotion system is tested in real-time on the hybrid robotic platform, ensuring stable gait transitions, smooth mobility across diverse terrains, and avoidance of kinematic singularities.

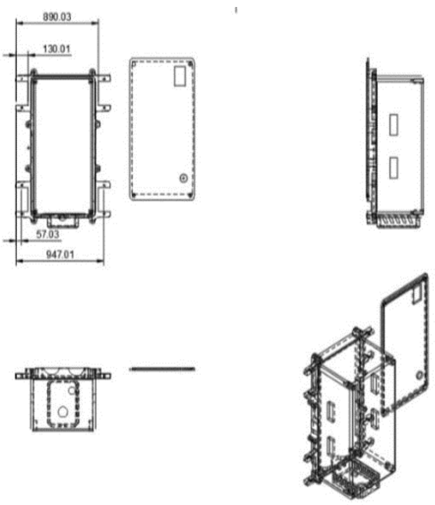
CHAPTER 2

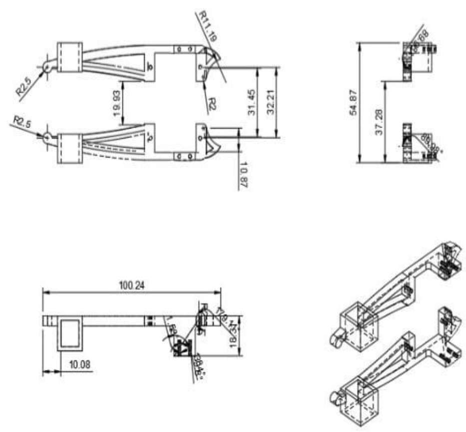
METHODOLOGY

This project's methodology focuses on developing a hybrid quadruped robot that integrates both legged and wheeled locomotion to achieve efficient and stable navigation across diverse terrains. The system is designed to autonomously adapt to different ground conditions by switching between wheels and legs depending on terrain complexity, obstacles, and stability requirements. The approach is broken down into several stages, each critical to ensuring that the robot achieves seamless and robust navigation capabilities suitable for real-world deployment.

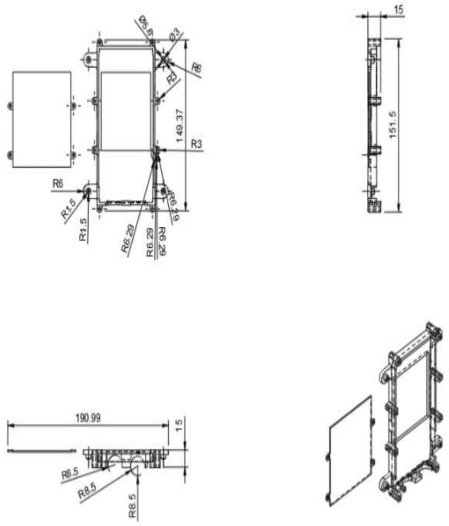
2.1 Mechanical Design and Modelling

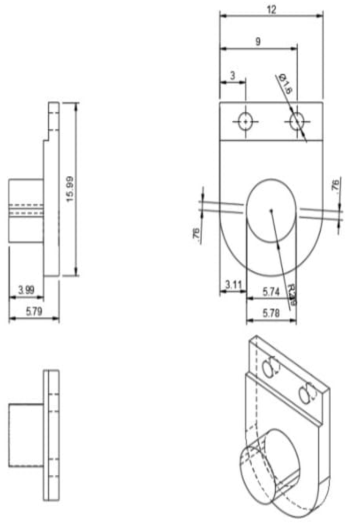
The first step in the process involves designing the robot's mechanical structure. Using CAD software like Fusion 360 or SolidWorks, the hybrid quadruped is modelled to feature four articulated legs with integrated wheels at their end-effectors. Each leg is equipped with multiple degrees of freedom for articulation, while the wheels provide high-speed locomotion on flat surfaces. The robot body houses essential electronic components, control systems, power supply units, and sensors. The CAD model is exported in STEP or STL format for further simulation and analysis. STEP files are particularly preferred as they preserve detailed geometry necessary for accurate simulation and control.



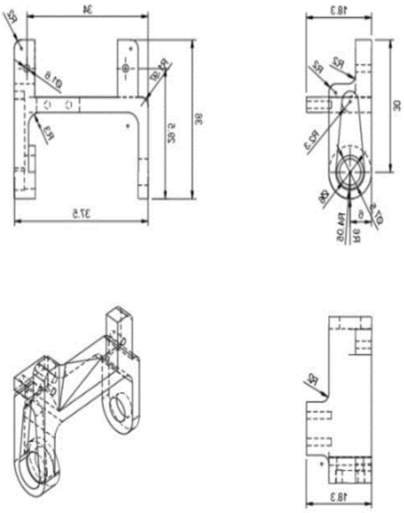


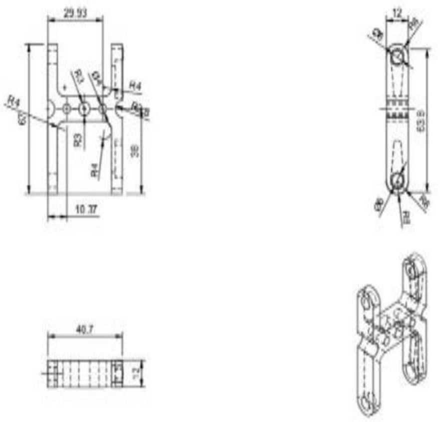
1. Body Top **2. Legs**





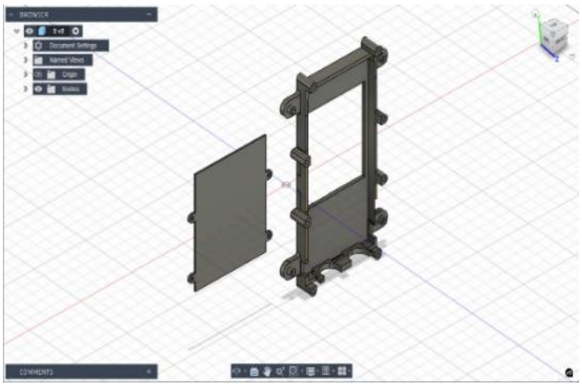
3. Body Bottom **4. Hinge**



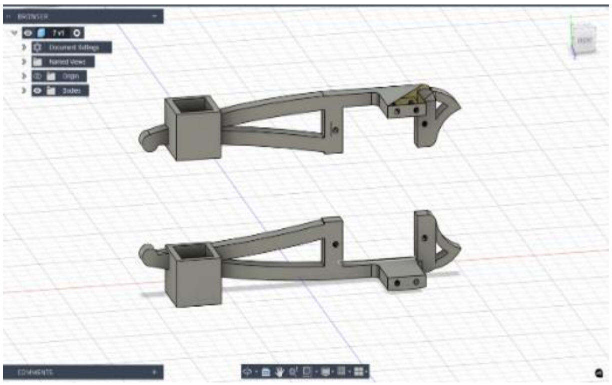


5. Backward & Forward Shoulder **6. Femur**

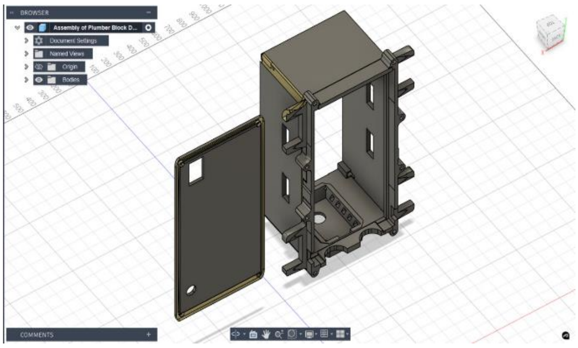
2.1.1:3D Modal:



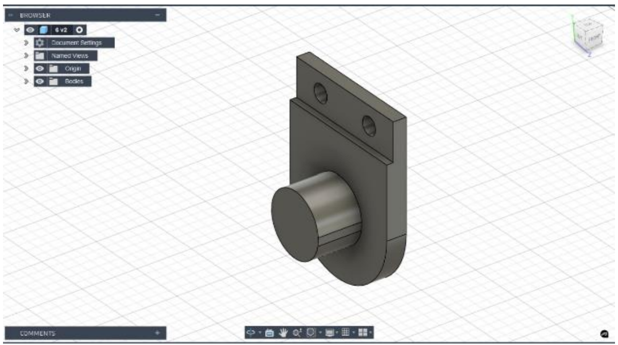
1.Body Top



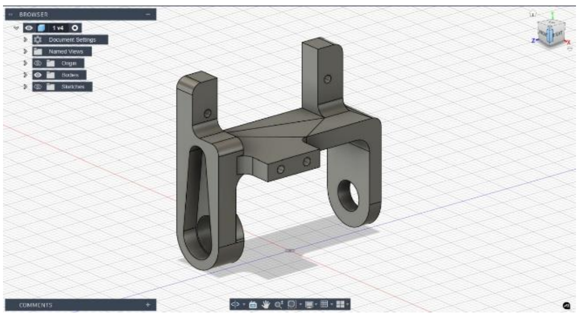
2.Legs



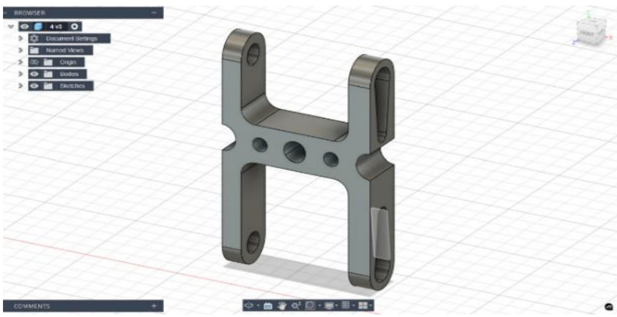
3. Body Bottom



4. Hinge



5.Backward & Forward Shoulder



6.Femur

2.2 Terrain Sensing and Surface Sampling

Unlike conventional robots that operate on predefined surfaces, this hybrid robot is designed to continuously scan and analyze the terrain it traverses. Using sensors such as LiDAR, stereo cameras, ultrasonic sensors, and IMUs, the robot collects real-time environmental data. Python-based processing scripts convert the raw sensor data into 3D point clouds or elevation maps. Sampling of surface data is performed along curvilinear coordinates (u, v), which are translated into Cartesian coordinates (x, y, z) to accurately map the ground profile. This real-time terrain mapping allows the robot to plan its locomotion mode effectively, adjusting its strategy based on slope angles, surface roughness, and obstacles detected.

2.3 Locomotion Mode Planning Algorithms

Path planning algorithms are used to select the optimal locomotion mode for the robot depending on the terrain. Three primary algorithms are evaluated:

- **Greedy Algorithm:** Quickly selects the nearest feasible terrain region but may result in inefficient or unstable transitions.
- **Weighted Greedy Algorithm:** Improves upon the basic greedy method by incorporating directional bias, reducing abrupt mode changes, and generating smoother locomotion sequences.
- **TSP Solver:** Solves locomotion transitions as a traveling salesman problem to find globally optimal paths, though it is computationally intensive for real-time applications.

The Weighted Greedy Algorithm is selected for this project as it offers a balanced approach between computational efficiency and locomotion smoothness, enabling stable movement across mixed terrains with minimal delays.

2.4 Kinematic Mapping and Gait Generation

Once the path is planned, kinematic mapping translates the robot's global motion plan into joint-level commands for both wheeled and legged operations. Inverse kinematics algorithms are applied to calculate joint angles for each leg, ensuring correct foot placements and balance during walking. For wheeled locomotion, wheel speed, torque, and steering angles are calculated. The robot’s kinematic model accounts for joint limits, link lengths, wheelbase dimensions, and leg extension ranges to ensure smooth transitions between rolling and walking modes while avoiding singularities and unstable configurations.

2.5 Path Optimization and Refinement

After kinematic mapping, the generated trajectories are further optimized for energy efficiency, stability, and mechanical safety. High-acceleration zones, sharp turns, and unnecessary joint movements are smoothed out to prevent wear and tear on motors and mechanical components. This optimization process also ensures consistent speed and orientation control, contributing to reliable material deposition or payload stability during mission execution.

2.6 Simulation and Validation

The optimized trajectories are tested in simulation environments before hardware implementation. Using platforms such as ROS, MoveIt, RViz, and Gazebo, the robot's motion is visualized and analyzed in a virtual world that includes various simulated terrains. The simulations verify:

- Collision avoidance with terrain obstacles.
- Feasibility of joint motions within mechanical limits.
- Smoothness and continuity of transitions between locomotion modes.

This stage plays a crucial role in validating the control algorithms and ensuring safe operation before real-world deployment.

2.7 Hardware Integration and Deployment

Following successful simulation, the control algorithms and optimized trajectories are transferred to the physical robot. Motor drivers, sensors, controllers, and embedded processors are integrated into the hardware platform. The robot is programmed to autonomously adjust its locomotion strategy based on live sensor feedback. Wireless communication modules allow for remote monitoring and system updates during field testing.

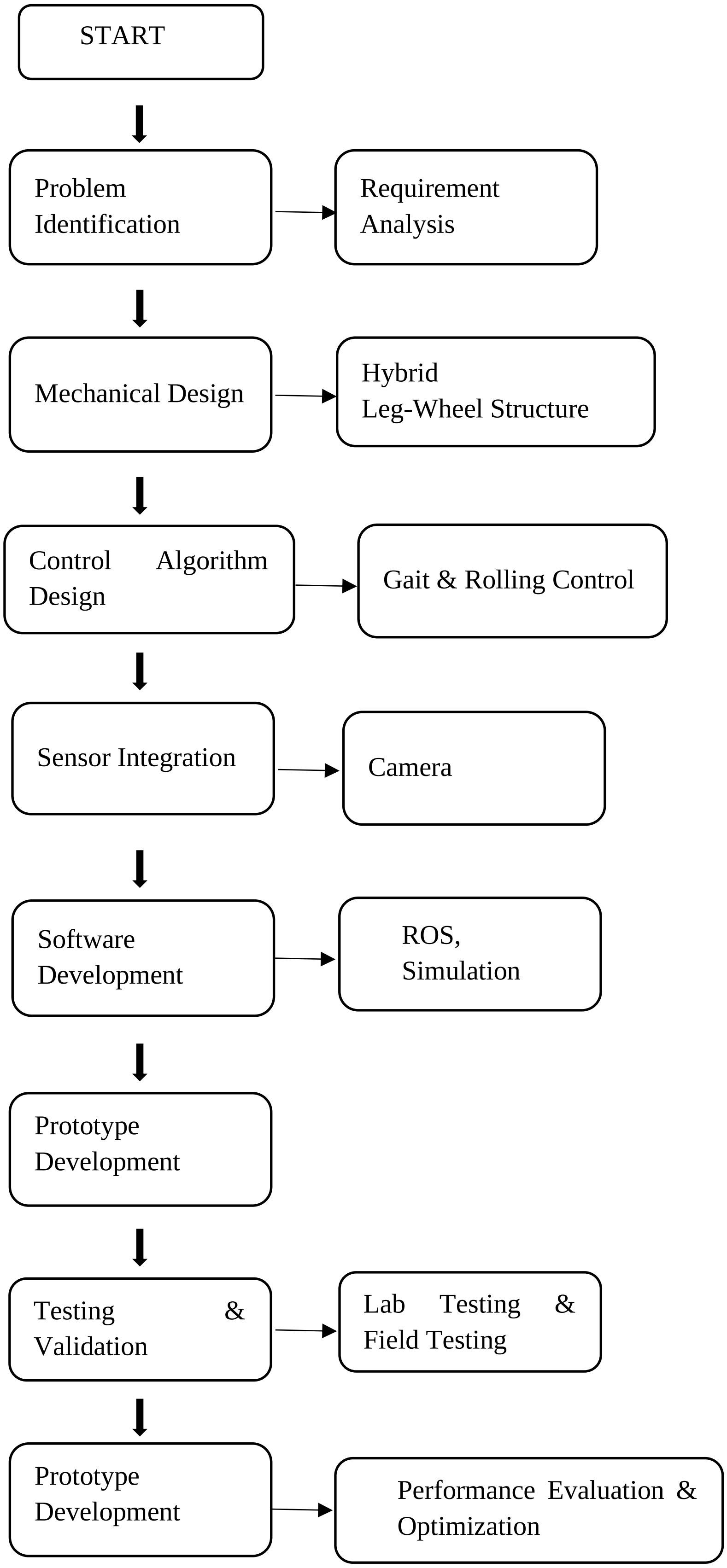
2.8 Testing and Field Trials

The final stage involves extensive real-world testing on various terrains such as flat surfaces, inclined planes, rocky areas, and stairs. The hybrid quadruped is evaluated for its ability to:

- Seamlessly transition between wheeled and legged locomotion.
- Maintain stability and balance under diverse terrain conditions.
- Optimize energy consumption while maximizing mobility.

The successful tests confirm the robustness and adaptability of the system for practical applications in search and rescue, industrial inspection, and unstructured outdoor environments.

Flowchart


START
Problem Identification
Requirement Analysis
Mechanical Design
Hybrid Leg-Wheel Structure
Control Algorithm Design
Gait & Rolling Control
Sensor Integration
Camera
Software Development
ROS, Simulation
Prototype Development
Testing & Validation
Lab Testing & Field Testing
Prototype Development
Performance Evaluation & Optimization

Start

The project begins with a clear initiation phase to set the direction and scope of the hybrid quadruped robot development.

Problem Identification → Requirement Analysis

In this first step, the core challenges of terrain navigation are identified, such as the need to combine legged mobility for rough, uneven surfaces with wheeled locomotion for speed and efficiency on flat terrain. The specific requirements are then analyzed, including mechanical design criteria, sensor needs, control system specifications, and performance targets necessary for reliable hybrid locomotion.

Mechanical Design → Hybrid Leg-Wheel Structure

Based on the requirements, the mechanical design phase focuses on creating a hybrid locomotion structure. This involves designing four articulated legs integrated with wheels at their ends, providing both walking and rolling capabilities. CAD tools are used to model the robot ensuring it can physically support and execute the intended hybrid motions.

Control Algorithm Design → Gait & Rolling Control

Next, control algorithms are developed to manage the robot's movement. This includes gait generation for walking with stability, balance control during legged locomotion, and control of wheel speed and steering for rolling. These algorithms enable seamless switching and coordination between walking and rolling modes depending on the terrain.

Sensor Integration → Camera

To enable autonomous navigation, various sensors are integrated into the robot. Sensor data is crucial for terrain analysis and adaptive locomotion mode selection.

Software Development → ROS, Simulation

The control and navigation software is developed using Robot Operating System (ROS) to manage communication between sensors, actuators, and algorithms. Simulation environments such as Gazebo and RViz are used to test robot behaviors virtually, enabling early validation of mechanical design and control strategies before hardware implementation.

Prototype Development

With the design and software in place, a physical prototype of the hybrid quadruped robot is constructed. This phase involves assembling mechanical components, installing sensors, and integrating control hardware to create a functional system ready for testing.

Testing & Validation → Lab Testing & Field Testing

The prototype undergoes rigorous testing starting in controlled lab environments to verify basic locomotion, sensor functionality, and control algorithm performance. Subsequently, field tests are conducted on various terrains (e.g., flat ground, rocky surfaces, inclines) to validate the robot's ability to navigate terrain environments and switch effectively between walking and rolling.

Performance Evaluation & Optimization

Data from testing is analyzed to evaluate key performance metrics such as stability, speed, energy efficiency, and adaptability. Based on these results, further optimization is performed on mechanical design, control algorithms, and sensor fusion techniques to enhance overall robot performance and reliability.

END

The project concludes with a robust hybrid quadruped robot capable of efficient terrain navigation, validated through simulation and real-world testing, ready for potential practical applications like search and rescue, inspection, or exploration missions.

CHAPTER 3

HARDWARE AND SOFTWARE COMPONENTS

3.1 Hardware Components

The project contains various electromechanical components. The components used and their quantities are mentioned in the below table

List of Components Required:

SL.No	Components	Quantity
1	Arduino Board UNO	1
2	Servo Motor	12
3	Gear Motor	4
4	Wheel	4
5	Rechargeable Cell 3.7v	3
6	Regulator	1
7	ON/OFF Switch	1
8	Jumper Wire Male/Female	As Required
9	Material use PLA+ filament	1

The Following Paragraph Explained in Detail The Various Components Used

3.1.1: Arduino Board UNO:

Fig3.1.1 Arduino uno

Building a **quadruped robot** using an **Arduino UNO** is an exciting robotics project that helps you understand servo control, balance, and gait coordination. The Arduino UNO serves as the robot's main controller — it is compact, lightweight, and powerful enough to handle multiple servos at once, making it ideal for small robotic systems. A typical quadruped robot has four legs, each with two or three servo motors that control the hip and knee joints, allowing the robot to walk, turn, and lift its legs in a coordinated way.

3.1.2: Servo Motor:

In a **quadruped robot**, **servo motors** are used as the main actuators that control the movement of each leg. Each leg typically has two or three servo motors — one for the hip joint, one for the knee joint, and sometimes one for the ankle — allowing the robot to move forward, backward, lift, and balance itself. The servos receive control signals from the **Arduino Nano**, which sends angle commands to adjust their positions precisely.By coordinating the motion of all servos through programmed sequences, the robot can perform walking gaits, turning motions, and even climbing small obstacles. Servo motors are ideal for such robots because they provide accurate position control, are easy to interface with microcontrollers, and can move smoothly within a defined angle range (usually 0°–180°). Proper power management is essential since servos require more current than the Arduino can supply directly, so they are powered from an external 5V source with a common ground. Overall, servo motors give the quadruped robot its flexibility and lifelike motion, making them essential for legged locomotion.

Figure 3.2: Servo Motor

3.1.3: Gear Motor:

In a **quadruped robot** that uses **gear motors**, the motors serve as the main driving components for the movement of each leg. Unlike servo motors, gear motors provide continuous rotation and higher torque, making them suitable for carrying heavier loads or providing stronger, smoother motion. Each leg can be driven by one or more gear motors connected through mechanical linkages or gears to control the hip and knee joints. The **Arduino Nano** controls these motors using **motor driver modules** such as the L298N or L293D, which manage the direction and speed of the motors through pulse-width modulation (PWM) signals. By programming the Arduino to coordinate the rotation of each gear motor, the robot can perform walking, turning, and balancing movements. A separate power supply is used for the motors since they draw more current than the Arduino can provide. Overall, gear motors give the quadruped robot robust strength and stability, making them ideal for larger or more powerful legged robots that need more torque than servo-based designs.

Figure 3.3: Gear Motor

3.1.4: Wheels:

In a **quadruped robot** that uses **wheels** instead of articulated legs, the wheels serve as the robot's primary means of movement, simplifying the mechanical design while maintaining a four-point structure for stability. Each of the four legs or corners of the robot's frame is fitted with a **DC gear motor with a wheel**, allowing it to move forward, backward, or turn by varying the speed and direction of the wheels. The **Arduino Nano** acts as the central controller, sending control signals through a **motor driver** (such as L298N or L293D) to manage each wheel's motion. By programming the Arduino to coordinate the motors, the robot can perform differential steering — for example, rotating one side's wheels forward and the other side's backward to turn in place. Power for the motors is supplied by a separate battery pack, while the Arduino is powered by a regulated 5V source. Using wheels in a quadruped robot makes it easier to build and control compared to leg-based robots, providing smoother motion, lower power consumption, and better efficiency on flat surfaces.

Figure 3.4 : Wheels

3.1.5 : Rechargeable Cell 3.7v:

In a **quadruped robot**, a **3.7V rechargeable cell** (such as a lithium-ion or lithium polymer battery) is used as the main **power source** to supply energy to the motors, sensors, and the Arduino Nano. Since most components in the robot require 5V or higher to operate efficiently, the 3.7V from the cell is typically **boosted using a DC-DC step-up converter** to provide a stable 5V output. This regulated voltage is then distributed to power the Arduino Nano and other modules like motor drivers, servo motors, and sensors. The rechargeable cell offers advantages such as **lightweight design, high energy density, and reusability**, making it ideal for mobile robots. To ensure safe operation, the battery is connected through a **battery management or charging module** (such as TP4056) that allows safe recharging via USB. Using a 3.7V rechargeable cell makes the quadruped robot **portable and energy-efficient**, allowing it to operate independently without being tethered to a power source.

Figure 3.5: Rechargeable Cell 3.7v

3.1.6: UBEC Voltage Regulator:

In a **quadruped robot**, a **voltage regulator** is used to provide a stable and consistent power supply to the various electronic components, such as the **Arduino Nano, sensors, and motors**. Since different parts of the robot may require different operating voltages—for example, the Arduino runs on 5V while some sensors or servos might need 3.3V—a voltage regulator ensures that each component receives the correct voltage level. It takes input from a higher-voltage source, such as a **battery pack or rechargeable cell**, and converts it to a steady, lower voltage suitable for the electronics. Common regulators used in such robots include **LM7805** for 5V output or **AMS1117** for 3.3V output. By preventing voltage fluctuations and overvoltage, the regulator protects sensitive components from damage and ensures smooth, reliable performance of the robot. In short, the voltage regulator acts as a **power stabilizer**, maintaining consistent operation and extending the overall lifespan of the quadruped robot's electronic system.

Figure 3.6 : UBEC Voltage Regulator

3.1.7: ON/OFF Switch:

An **ON/OFF switch** is a fundamental electrical component used in a **quadruped robot** to control the power supply to its electronic and mechanical systems. It acts as the main control point for starting and shutting down the robot safely. When the switch is in the **ON** position, it completes the electrical circuit, allowing current to flow from the power source—such as a battery—to the robot's components, including the microcontroller, sensors, and motors. This enables the robot to perform its functions, such as walking, balancing, and responding to commands. Conversely, when the switch is in the **OFF** position, it breaks the circuit, cutting off the power supply and stopping all operations immediately. This is crucial for preventing electrical damage, conserving battery power, and ensuring safety during maintenance or when the robot is not in use. In a quadruped robot, the ON/OFF switch is often strategically placed for easy access, allowing users to quickly power the system without unplugging cables or interfering with internal electronics. Overall, it serves as a simple yet vital component for **power management and operational safety**.

Figure 3.7 : ON/OFF Switch

3.1.8 : Jumper Wire Male/Female:

Jumper wires (male/female) play a crucial role in the electrical connections of a quadruped robot. These wires are used to establish flexible and reliable links between various electronic components, such as sensors, servos, the microcontroller (like Arduino or Raspberry Pi), and the power supply. **Male-to-female jumper wires** are particularly useful for connecting modules or sensors that have male header pins to breadboards or control boards with female connectors. For example, when connecting servo motors to a motor driver or control board, male-to-female jumper wires allow easy and secure plug-and-play connections without the need for soldering. Their color-coded insulation helps in identifying signal, power, and ground lines, reducing wiring errors and improving maintenance efficiency. Because quadruped robots require multiple servo connections for leg movements, jumper wires simplify the wiring process, ensure stable signal transmission, and allow quick modifications during testing and troubleshooting. In short, male/female jumper wires are essential for building and prototyping a quadruped robot, providing both flexibility and ease of assembly in its electronic circuitry.

Figure 3.8 : Jumper Wire Male/Female

3.1.9 : Jumper Wire Female/Male:

Jumper wires (female-to-male type) play a crucial role in building and connecting different electronic components in a quadruped robot. These wires are used to establish flexible, temporary, and reliable connections between modules such as sensors, servo motors, microcontrollers (like Arduino or Raspberry Pi), and the power supply. The **male end** of the jumper wire typically plugs into the **female header pins** of the microcontroller board, while the **female end** connects to the **male pins** of components such as servo motors or sensors. This type of connection allows for easy assembly, testing, and modification of the robot's circuitry without the need for soldering. In a quadruped robot, jumper wires transmit control signals from the microcontroller to the servo motors that move the legs, as well as carry feedback signals from sensors used for balance and obstacle detection. Their color-coded insulation helps in identifying connections easily and preventing wiring errors. Overall, female-to-male jumper wires provide a simple and efficient means of prototyping and maintaining electrical connectivity in the quadruped robot's electronic system.

Figure 3.9 : Jumper Wire Female/Male

3.1.10 : PLA+ filament :

The design and development of the hybrid quadruped robot with wheeled locomotion incorporates **PLA-based 3D-printed components** as the primary material due to their low cost, light weight, and excellent dimensional accuracy. PLA and PLA+ filaments are used for most structural elements such as the chassis, leg segments, motor mounts, and sensor housings because they offer good stiffness and are easy to manufacture with FDM printing. For parts subjected to higher stress or temperature such as wheel hubs, joints, and battery or motor-mounting areas stronger filaments like **PETG or PLA+** are included to improve impact resistance and durability.

Figure 3.10 : PLA+ filament

3.1 Software Components

: SolidWorks / Fusion 360 / FreeCAD

3.2.1:CAD software for mechanical design of leg-wheel system and chassis.

SolidWorks, Fusion 360, or FreeCAD are used because they provide powerful tools for accurately designing and developing the mechanical components of the hybrid quadruped robot with wheeled locomotion. These CAD softwares allow precise 3D modeling of the leg-wheel system, joints, and chassis, ensuring correct dimensions, alignment, and smooth integration of moving parts. They enable assembly modeling and motion simulation to analyze leg articulation, wheel rotation, and walking–rolling transitions, which is essential for terrain navigation. Additionally, built-in analysis features help evaluate structural strength, load distribution, and stability, reducing the risk of mechanical failure. The software also supports design optimization, easy modifications, and generation of manufacturing files for 3D printing or machining, making the development process efficient, cost-effective.

3.2.2: Introduction to Arduino Software:

Arduino is an open-source electronics platform based on easy-to-use hardware and software.The Arduino Software (IDE), also known as the Arduino Integrated Development Environment(IDE), is a platform for writing, compiling, and uploading code to Arduino boards. It allows users to create programs, also called "sketches," that control physical devices like sensors, motors, lights, and other components.

The Arduino IDE is compatible with a variety of Arduino boards, such as the Arduino Uno, Arduino Nano, Arduino Mega, and more. The software is designed to be beginner-friendly, making it accessible for those new to programming and electronics, while still powerful enough for advanced users to create complex projects.

- **User-Friendly Interface:**

  The Arduino IDE features a simple, clean interface with basic functions that make it easy to write and upload code. It provides a text editor for writing code, a message area for displaying errors or status, and buttons for compiling and uploading the code to the Arduino board.

- **Cross-Platform Compatibility:**

  The Arduino IDE is available for Windows, Mac OS X, and Linux, making it accessible to a wide range of users.

- **Sketches and Libraries:**

  Programs written for Arduino are called sketches. These sketches are written in a simplified version of C++, a widely-used programming language. The IDE also provides a wide range of libraries that allow users to easily interface with sensors, motors, displays, and other components.

- **Code Structure:**

  Arduino sketches are structured with two main functions:

  • setup: This function runs once when the Arduino is powered on or reset. It is used to initialize variables, settings, and configurations.

  • loop(: This function runs repeatedly after the setup function. It contains the code that will be executed continuously, such as reading sensors, controlling actuators, or handling inputs and outputs.

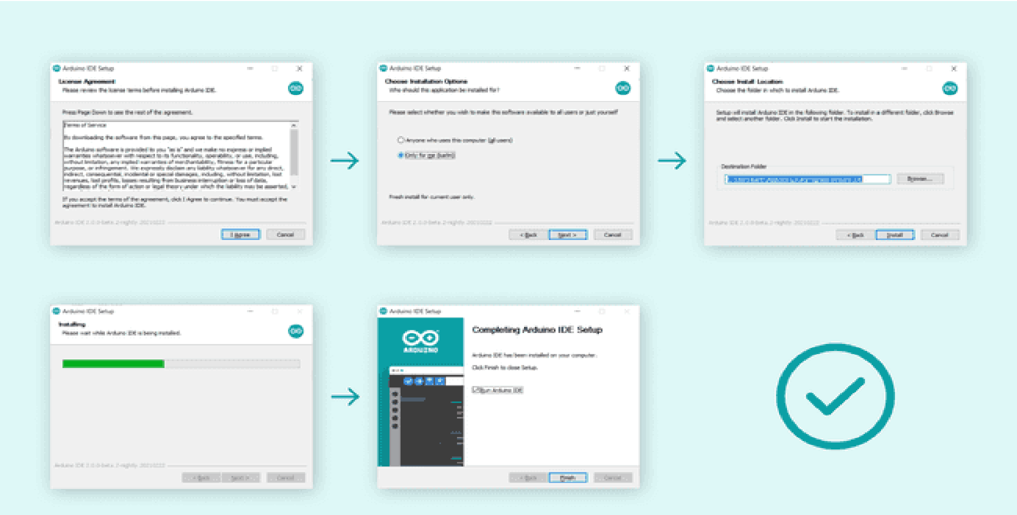


Figure: 3.2.1 Installation of Arduino Software

The Arduino IDE (Integrated Development Environment) is a software application that allows users to write code and upload it to an Arduino microcontroller board, essentially acting as the platform to program and interact with Arduino hardware; it provides a user-friendly interface with a text editor to write code, a compiler to translate it into machine code, and the ability to send that code directly to the Arduino board to run.

• Open-source: The Arduino IDE is freely available to download and use.

• Cross-platform: It works on various operating systems like Windows, macOS, and Linux.

• Simple programming language: The code written in Arduino IDE is based on C++ but simplified for easier learning.

• Features: Includes a text editor, code verification, upload functionality, serial monitor for debugging, and access to various libraries for common functions.

3.2.3: How The Arduino Works:

- **Write code:** Open the Arduino IDE and write your program in the text editor using the Arduino language
- **Select board and port:** Choose the specific Arduino board you are using and the communication port it's connected to on your computer
- **Verify code:** Click "Verify" to check for syntax errors in your code

- **Upload code:** If no errors are found, click "Upload" to send the compiled code to the Arduino board.

3.2.4: Program

This code is used in a hybrid quadruped robot to enable wireless control, live video streaming, and mode switching using an ESP32-CAM. It allows the robot to transmit real-time camera footage to a web interface, helping the user monitor the robot's surroundings. Through the same interface, movement commands, walking/driving mode selection, speed control, and axis adjustments are sent to the robot. This makes the robot smarter, easier to control remotely, and more suitable for tasks like navigation, inspection, and surveillance where both legged motion and visual feedback are essential.

```
// ESP32-CAM Robot Control with Camera Stream

#include "esp_camera.h"

#include <WiFi.h>

#include "esp_timer.h"

#include "img_converters.h"

#include "fb_gfx.h"

#include "soc/soc.h"

#include "soc/rtc_cntl_reg.h"

#include "esp_http_server.h"


const char* ssid = "LegionEYE";

const char* password = "123456789";


// Camera pins for AI-THINKER ESP32-CAM
```

```
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22
#define LED_GPIO_NUM       4

#define PART_BOUNDARY "123456789000000000000987654321"

static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary=" PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n";
```

```
httpd_handle_t stream_httpd = NULL;
httpd_handle_t camera_httpd = NULL;

const char INDEX_HTML[] = R"rawliteral(
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,initial-scale=1">
  <title>LEGION MARK-2</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      background: #1a1a2e;
      color: #e6e6e6;
      margin: 0;
      padding: 20px;
    }
    .container {
      max-width: 1200px;
      margin: 0 auto;
    }
```

```
.header {
    margin-bottom: 20px;
}
h1 {
    color: #4cc9f0;
    font-size: 2.5em;
    margin-bottom: 10px;
}
.status {
    padding: 10px;
    border-radius: 5px;
    margin: 10px auto;
    max-width: 500px;
    font-weight: bold;
}
.online {
    background: rgba(46, 213, 115, 0.2);
    color: #2ed573;
    border: 1px solid #2ed573;
}
.offline {
    background: rgba(255, 71, 87, 0.2);
    color: #ff4757;
    border: 1px solid #ff4757;
```

```
}
.camera-container {
    background: #2d2d44;
    padding: 20px;
    border-radius: 15px;
    margin: 20px auto;
    max-width: 800px;
    border: 2px solid #4cc9f0;
}
.camera-feed {
    width: 100%;
    max-width: 640px;
    height: 480px;
    background: #000;
    border: 2px solid #4cc9f0;
    border-radius: 10px;
    margin: 0 auto;
    display: flex;
    align-items: center;
    justify-content: center;
    color: #4cc9f0;
    font-size: 1.2em;
}
.camera-feed img {
```

```
    max-width: 100%;
    max-height: 100%;
    border-radius: 8px;
}
.control-panel {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
    gap: 20px;
    margin: 20px 0;
}
.control-group {
    background: #2d2d44;
    padding: 20px;
    border-radius: 15px;
    border: 1px solid #3a3a5e;
}
.control-group h3 {
    color: #4cc9f0;
    margin-bottom: 15px;
}
.button {
    background: #4cc9f0;
    color: #16213e;
    border: none;
```

```
    padding: 12px 20px;
    margin: 5px;
    border-radius: 8px;
    cursor: pointer;
    font-size: 16px;
    font-weight: bold;
    transition: all 0.3s;
}
.button:hover {
    background: #3ab0d9;
    transform: translateY(-2px);
}
.movement-grid {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 10px;
    max-width: 200px;
    margin: 0 auto;
}
.forward { grid-column: 2; grid-row: 1; }
.backward { grid-column: 2; grid-row: 3; }
.left { grid-column: 1; grid-row: 2; }
.right { grid-column: 3; grid-row: 2; }
.stop {
```

```
    grid-column: 2; grid-row: 2;
    background: #ff4757;
}
.stop:hover { background: #e94560; }
.slider-container {
    margin: 15px 0;
}
.slider-label {
    display: block;
    margin-bottom: 8px;
    color: #b8b8d0;
    font-weight: bold;
}
.slider {
    width: 100%;
    margin: 10px 0;
}
.slider-value {
    color: #4cc9f0;
    font-weight: bold;
}
.mode-display {
    font-size: 1.2em;
    font-weight: bold;
```

```
            color: #4cc9f0;
            margin-top: 10px;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="header">
            <h1>🤖 LEGION MARK-2</h1>
            <div class="status offline" id="status">📷 Camera: Connecting to stream...</div>
        </div>

        <div class="camera-container">
            <h3>📹 Live Camera Feed</h3>
            <div class="camera-feed">
                <div id="streamStatus">Loading camera stream...</div>
                <img id="streamImage" style="display:none" alt="Live Camera Feed" />
            </div>
            <div style="margin-top: 15px;">
                <button class="button" onclick="reloadStream()">🔄 Restart Stream</button>
                <button class="button" onclick="toggleLED()">💡 Toggle Flash LED</button>
            </div>
        </div>
```

```html
    <div class="control-panel">
      <div class="control-group">
        <h3>🎮 Movement Controls</h3>
        <div class="movement-grid">
          <button class="button forward" onclick="sendMove('f')">⬆</button>
          <button class="button backward" onclick="sendMove('b')">⬇</button>
          <button class="button left" onclick="sendMove('l')">⬅</button>
          <button class="button right" onclick="sendMove('r')">➡</button>
          <button class="button stop" onclick="sendMove('s')">⏹</button>
        </div>
      </div>

      <div class="control-group">
        <h3>🔧 Operation Mode</h3>
        <button class="button" onclick="setMode('walk')">🚶 Walk Mode</button>
        <button class="button" onclick="setMode('drive')">🚗 Drive Mode</button>
        <div class="mode-display">Current Mode: <span
id="currentMode">walk</span></div>
      </div>

      <div class="control-group">
        <h3>⚙ Axis Controls</h3>
```

```
        <div class="slider-container">
          <label class="slider-label">X-axis: <span class="slider-value"
id="xdVal">62</span></label>
          <input type="range" min="10" max="100" value="62" class="slider" id="xd"
oninput="updateSlider('xd', this.value)">
        </div>
        <div class="slider-container">
          <label class="slider-label">Y-axis: <span class="slider-value"
id="ydVal">62</span></label>
          <input type="range" min="10" max="100" value="62" class="slider" id="yd"
oninput="updateSlider('yd', this.value)">
        </div>
        <div class="slider-container">
          <label class="slider-label">Z-axis: <span class="slider-value" id="zdVal">-
50</span></label>
          <input type="range" min="-100" max="0" value="-50" class="slider" id="zd"
oninput="updateSlider('zd', this.value)">
        </div>
      </div>

      <div class="control-group">
        <h3> Speed Controls</h3>
        <div class="slider-container">
          <label class="slider-label">Leg Speed: <span class="slider-value"
id="lsVal">8</span></label>
```

```
                <input type="range" min="4" max="12" value="8" class="slider" id="ls"
oninput="updateSlider('ls', this.value)">
            </div>
            <div class="slider-container">
                <label class="slider-label">Movement Speed: <span class="slider-value"
id="ssVal">60</span></label>
                <input type="range" min="30" max="150" value="60" class="slider" id="ss"
oninput="updateSlider('ss', this.value)">
            </div>
        </div>
    </div>
  </div>

  <script>
    let streamUrl = '';

    function initializeInterface() {
      const baseUrl = window.location.origin;
      streamUrl = baseUrl.replace(':80', ':81') + '/stream';
      console.log('Stream URL:', streamUrl);
      startStream();
    }

    function startStream() {
      const streamImg = document.getElementById('streamImage');
```

```
        const streamStatus = document.getElementById('streamStatus');
        const statusDiv = document.getElementById('status');

        streamImg.onload = function() {
          streamStatus.style.display = 'none';
          streamImg.style.display = 'block';
          statusDiv.textContent = '📷 Camera: Live stream active';
          statusDiv.className = 'status online';
        };

        streamImg.onerror = function() {
          streamImg.style.display = 'none';
          streamStatus.style.display = 'block';
          streamStatus.textContent = '❌ Stream connection failed';
          statusDiv.textContent = '📷 Camera: Connection failed';
          statusDiv.className = 'status offline';
        };

        streamImg.src = streamUrl + '?t=' + new Date().getTime();
      }

      function reloadStream() {
        console.log('Reloading stream...');
        startStream();
```

```
}

function sendMove(direction) {
    fetch('/action?go=move&value=' + direction)
        .catch(error => console.error('Move error:', error));
}

function setMode(mode) {
    document.getElementById('currentMode').textContent = mode;
    fetch('/action?go=mode&value=' + mode)
        .catch(error => console.error('Mode error:', error));
}

function updateSlider(type, value) {
    document.getElementById(type + 'Val').textContent = value;
    fetch('/action?go=' + type + '&value=' + value)
        .catch(error => console.error('Slider error:', error));
}

function toggleLED() {
    fetch('/action?go=led&value=toggle')
        .catch(error => console.error('LED error:', error));
}
```

```
        setInterval(() => {
            const streamImg = document.getElementById('streamImage');
            if (streamImg.style.display === 'none') {
                startStream();
            }
        }, 5000);

        window.onload = initializeInterface;
    </script>
</body>
</html>
)rawliteral";

static esp_err_t index_handler(httpd_req_t *req) {
    httpd_resp_set_type(req, "text/html");
    return httpd_resp_send(req, INDEX_HTML, strlen(INDEX_HTML));
}

static esp_err_t stream_handler(httpd_req_t *req) {
    camera_fb_t * fb = NULL;
    esp_err_t res = ESP_OK;
    size_t _jpg_buf_len = 0;
    uint8_t * _jpg_buf = NULL;
    char * part_buf[64];
```

```
res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
if(res != ESP_OK){
  return res;
}

while(true){
  fb = esp_camera_fb_get();
  if (!fb) {
    Serial.println("Camera capture failed");
    res = ESP_FAIL;
  } else {
    if(fb->format != PIXFORMAT_JPEG){
      bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
      esp_camera_fb_return(fb);
      fb = NULL;
      if(!jpeg_converted){
        Serial.println("JPEG compression failed");
        res = ESP_FAIL;
      }
    } else {
      _jpg_buf_len = fb->len;
      _jpg_buf = fb->buf;
    }
```

```
        }
        if(res == ESP_OK){
            size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
            res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
        }
        if(res == ESP_OK){
            res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
        }
        if(res == ESP_OK){
            res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY,
strlen(_STREAM_BOUNDARY));
        }
        if(fb){
            esp_camera_fb_return(fb);
            fb = NULL;
            _jpg_buf = NULL;
        } else if(_jpg_buf){
            free(_jpg_buf);
            _jpg_buf = NULL;
        }
        if(res != ESP_OK){
            break;
        }
    }
```

```
    return res;
}

static esp_err_t cmd_handler(httpd_req_t *req) {
    char*  buf;
    size_t buf_len;
    char variable[32] = {0,};
    char value[32] = {0,};

    buf_len = httpd_req_get_url_query_len(req) + 1;
    if (buf_len > 1) {
        buf = (char*)malloc(buf_len);
        if(!buf){
            httpd_resp_send_500(req);
            return ESP_FAIL;
        }
        if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK) {
            if (httpd_query_key_value(buf, "go", variable, sizeof(variable)) == ESP_OK &&
                httpd_query_key_value(buf, "value", value, sizeof(value)) == ESP_OK) {
            } else {
                free(buf);
                httpd_resp_send_404(req);
                return ESP_FAIL;
            }
```

```
    } else {
        free(buf);
        httpd_resp_send_404(req);
        return ESP_FAIL;
    }
    free(buf);
} else {
    httpd_resp_send_404(req);
    return ESP_FAIL;
}

// Handle commands
if(strcmp(variable, "move") == 0) {
    Serial.println(value); // f, b, l, r, s
}
else if(strcmp(variable, "mode") == 0) {
    if(strcmp(value, "drive") == 0) Serial.println("h");
    else if(strcmp(value, "walk") == 0) Serial.println("s");
}
else if(strcmp(variable, "xd") == 0 || strcmp(variable, "yd") == 0 ||
        strcmp(variable, "zd") == 0 || strcmp(variable, "ls") == 0) {
    Serial.print(variable);
    Serial.print(",");
    Serial.println(value);
```

```
    }
    else if(strcmp(variable, "ss") == 0) {
        Serial.print("ss,");
        Serial.println(value);
    }
    else if(strcmp(variable, "led") == 0) {
        digitalWrite(LED_GPIO_NUM, !digitalRead(LED_GPIO_NUM));
        Serial.println("LED toggled");
    }

    httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
    return httpd_resp_send(req, NULL, 0);
}

void startCameraServer(){
    httpd_config_t config = HTTPD_DEFAULT_CONFIG();
    config.server_port = 80;

    httpd_uri_t index_uri = {
        .uri       = "/",
        .method    = HTTP_GET,
        .handler   = index_handler,
        .user_ctx  = NULL
    };
```

```
httpd_uri_t cmd_uri = {
    .uri       = "/action",
    .method    = HTTP_GET,
    .handler   = cmd_handler,
    .user_ctx  = NULL
};

httpd_uri_t stream_uri = {
    .uri       = "/stream",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
};

Serial.println("Starting web server on port 80");
if (httpd_start(&camera_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(camera_httpd, &index_uri);
    httpd_register_uri_handler(camera_httpd, &cmd_uri);
}

config.server_port = 81;
config.ctrl_port += 1;
Serial.println("Starting stream server on port 81");
```

```
    if (httpd_start(&stream_httpd, &config) == ESP_OK) {
        httpd_register_uri_handler(stream_httpd, &stream_uri);
    }
}

void setup() {
    WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector

    Serial.begin(115200);
    Serial.setDebugOutput(false);

    // Initialize LED
    pinMode(LED_GPIO_NUM, OUTPUT);
    digitalWrite(LED_GPIO_NUM, LOW);

    // Camera configuration
    camera_config_t config;
    config.ledc_channel = LEDC_CHANNEL_0;
    config.ledc_timer = LEDC_TIMER_0;
    config.pin_d0 = Y2_GPIO_NUM;
    config.pin_d1 = Y3_GPIO_NUM;
    config.pin_d2 = Y4_GPIO_NUM;
    config.pin_d3 = Y5_GPIO_NUM;
    config.pin_d4 = Y6_GPIO_NUM;
```

```
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;

// Use lower resolution for better stability
if(psramFound()){
  config.frame_size = FRAMESIZE_VGA;
  config.jpeg_quality = 12;
  config.fb_count = 2;
} else {
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;
}
```

```
    // Camera init
    esp_err_t err = esp_camera_init(&config);
    if (err != ESP_OK) {
      Serial.printf("Camera init failed with error 0x%x", err);
      return;
    }

    // Camera settings
    sensor_t * s = esp_camera_sensor_get();
    if (s != NULL) {
      s->set_vflip(s, 1);
      s->set_hmirror(s, 1);
    }

    // WiFi AP
    WiFi.softAP(ssid, password);
    Serial.print("Camera Ready! Use 'http://");
    Serial.print(WiFi.softAPIP());
    Serial.println("' to connect");

    startCameraServer();
  }
```

```
void loop() {
    delay(10000);
}
```

CHAPTER 4

IMPLEMENTATION

4.1 Model Implementation:

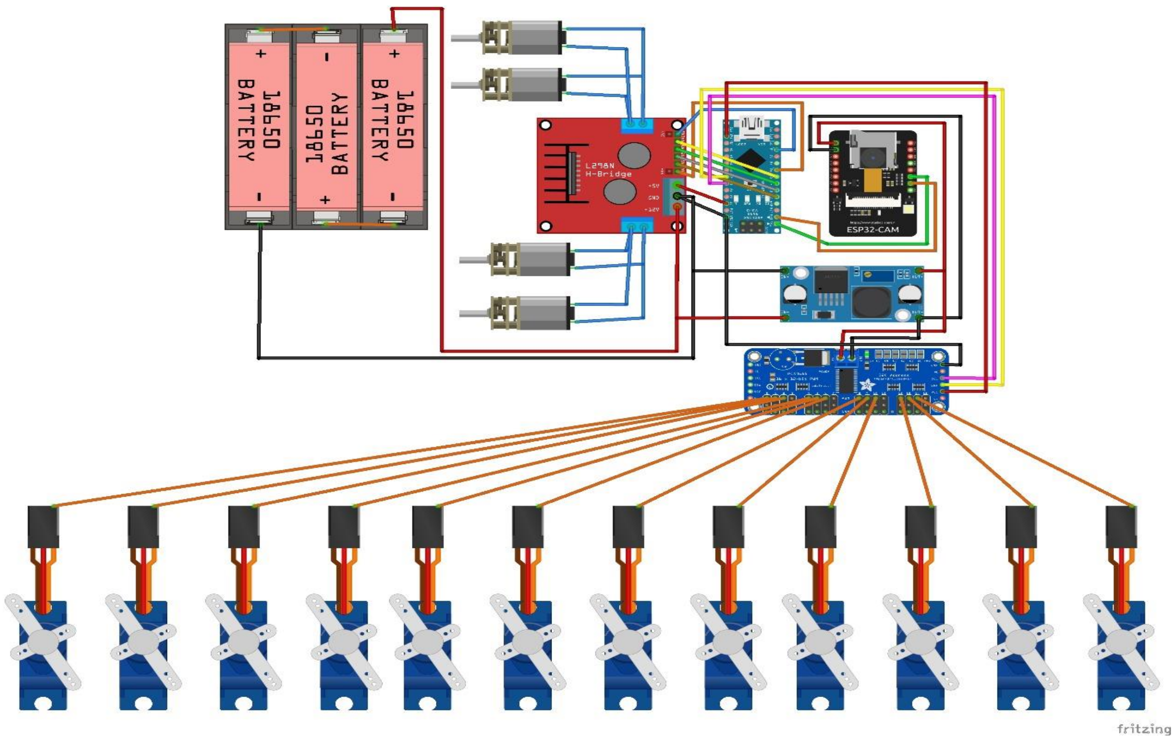


Figure: 4.1.1 Implementation

The model implementation for the design and development of a hybrid quadruped robot with wheeled locomotion integrates mechanical, electronic, and software subsystems to produce a robust platform for challenging-terrain navigation. Mechanically, each limb is realized as a modular leg–wheel module combining a revolute hip and knee driven by brushless DC motors and harmonic-drive reducers for high torque, with a small omnidirectional wheel embedded at the foot to enable low-resistance rolling. The electrical architecture centers on a distributed motor driver network and a real-time embedded controller (e.g., an ARM Cortex-M / single-board computer pair) that fuses data from IMU, wheel encoders, lidar, and depth cameras.

Control is implemented with a layered software stack: low-level joint-space torque/position controllers ensure precise limb actuation; a mid-level locomotion manager switches between walking, wheeled, and hybrid modes using gait libraries (trotting, crawl) and kinematic transformations for wheel-surface contact; and a high-level planner performs obstacle avoidance and path planning using SLAM and model-predictive control (MPC) to anticipate slippage and uneven terrain. Stability is enforced through a real-time ZMP/COM estimator and reflexive foothold replanning, while energy efficiency is improved by opportunistic rolling (using wheels on flat segments) and regenerative braking.

Development relies on physics-based simulation (Gazebo/ROS or MuJoCo) for controller tuning, followed by incremental hardware-in-the-loop testing and field trials to validate gait transitions, payload handling, and fault tolerance. The proposed implementation emphasizes modularity, maintainability, and scalability so the platform can be adapted for varied mission profiles—from rapid corridor traversal to delicate manipulation over rough ground.

4.2 Model

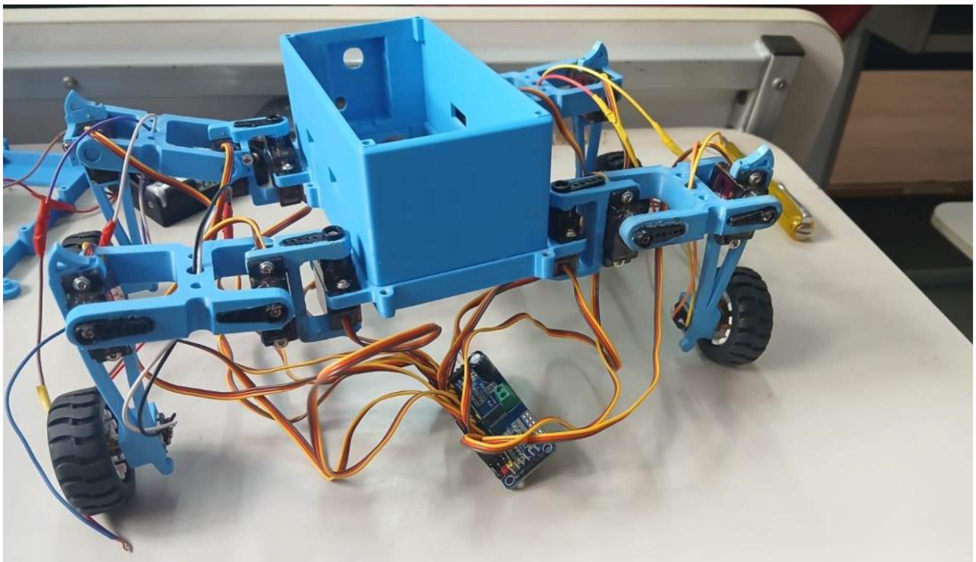


Figure: 4.1.2 Assembly Intermediate Model

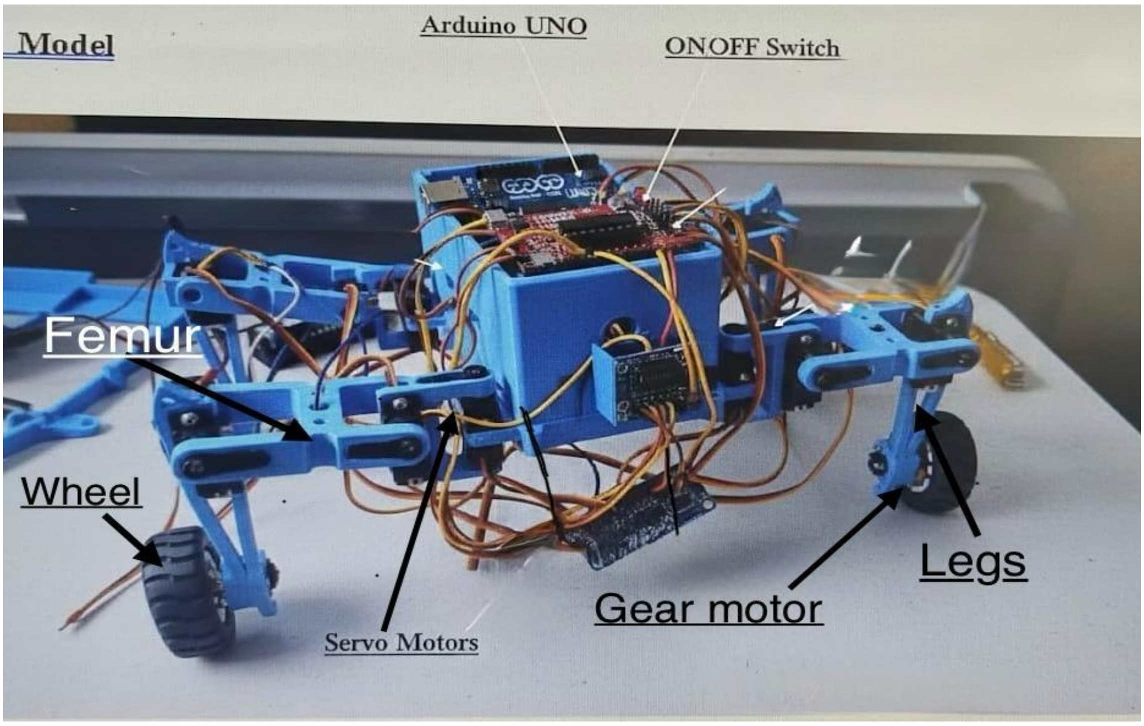


Figure: 4.2.2 Assembly Model

CHAPTER 5

5.1 Learning Objectives

- **Understand hybrid locomotion concepts**
  Explain the principles of combining legged (quadruped) and wheeled locomotion systems and their advantages for navigating varied terrains.
- **Apply mechanical design principles**
  Design and model a quadruped robot structure integrating wheels, considering stability, load distribution, degrees of freedom, and material selection.
- **Develop kinematic and dynamic models**
  Analyze the kinematics and basic dynamics of a quadruped robot to enable coordinated leg–wheel motion.
- **Implement embedded control systems**
  Design and program microcontroller-based control systems to coordinate actuators, sensors, and locomotion modes.
- **Design locomotion and mode-switching strategies**
  Develop control algorithms that enable smooth transitions between walking, rolling, and hybrid movement modes.
- **Integrate sensors for terrain awareness**
  Utilize sensors (e.g., IMU, ultrasonic, encoders) to detect terrain conditions and support adaptive navigation.
- **Evaluate terrain navigation performance**
  Test and assess robot performance across different terrains, analyzing speed, stability, energy efficiency, and maneuverability.
- **Apply system integration skills**
  Integrate mechanical, electrical, and software subsystems into a functional robotic platform.
- **Demonstrate problem-solving and optimization**
  Identify design challenges and implement improvements to enhance locomotion efficiency and robustness.

- **Document and present engineering work**
  Prepare technical documentation and presentations that clearly communicate the design process, implementation, and experimental results.

5.2 Learning Outcomes

The outcomes achieved from the project are summarized as:

- Successfully designed and developed a functional hybrid quadruped robot prototype with integrated leg-wheel mechanisms.
- Gained practical knowledge in modeling robot joints, linkages, and wheel actuation.
- Developed gait algorithms enabling the robot to switch between rolling and walking modes.
- Applied inverse kinematics and trajectory planning to ensure stable locomotion over varied terrains.
- Integrated sensor fusion algorithms to enhance navigation, stability, and obstacle detection.
- Implemented ROS-based control architecture, using MoveIt and Gazebo for motion planning, visualization, and simulation validation.
- Addressed real-world challenges like joint limit avoidance, slip prevention, and terrain

5.3 Applications

The design and development of hybrid quadruped robots with wheeled locomotion have transformed mobile robotics by combining the advantages of legged and wheeled systems into a single platform capable of handling diverse and challenging environments. Traditional robots are often limited by either their inability to handle rough terrains (in the case of wheeled robots) or their slow and energy-intensive operation (in the case of legged robots). Hybrid quadruped robots overcome these limitations, offering superior adaptability, stability, and efficiency for a wide range of applications across various industries. These robots have found significant relevance and impact in multiple sectors due to their unique ability to switch between different locomotion modes depending on terrain requirements.

1. Search and Rescue Operations

In search and rescue scenarios, hybrid quadruped robots prove invaluable due to their capability to traverse complex, hazardous terrains that are typically inaccessible to conventional robots or even humans. Their ability to roll quickly across flat areas and then switch to legged locomotion to climb over debris, rubble, or collapsed structures makes them highly effective in disaster zones such as earthquake aftermaths, landslides, or collapsed buildings. Equipped with cameras, thermal sensors, and communication devices, they can assist in locating trapped victims, monitoring hazardous areas, and relaying real-time data back to rescue teams, thereby minimizing human risk in dangerous conditions.

2. Military and Defense Applications

The defense sector benefits significantly from hybrid quadruped robots, particularly for reconnaissance, surveillance, and logistics in difficult terrains. These robots can perform autonomous patrols, navigate rough or urban environments, and deliver critical payloads across challenging landscapes. Their silent rolling capabilities allow for stealth operations, while their legged mode enables them to overcome obstacles such as stairs, ditches, or rocky surfaces. Additionally, hybrid quadrupeds can be outfitted with advanced sensors for mine detection, hazardous material inspection, and perimeter security, supporting a wide range of military missions with enhanced safety and operational efficiency.

3. Industrial Inspection and Maintenance

In industrial environments, especially in oil refineries, power plants, factories, and warehouses, hybrid quadruped robots can efficiently navigate narrow spaces, stairs, and obstacles that typically hinder wheeled inspection robots. They can autonomously perform routine inspections, maintenance checks, and equipment monitoring while minimizing downtime and risk to human workers. Their ability to operate in confined and hazardous spaces makes them ideal for high-risk industrial operations, ensuring safety compliance and operational continuity.

4. Agriculture and Forestry

Agricultural fields and forests present highly irregular and constantly changing terrain conditions. Hybrid quadruped robots are particularly suited for precision agriculture applications such as crop monitoring, soil analysis, automated planting, pest control, and selective harvesting, even on uneven and sloped ground. In forestry, these robots can assist in forest monitoring, fire surveillance, biodiversity studies, and resource management. Their lightweight design and adaptable locomotion minimize soil compaction, preserving soil health while efficiently managing field operations.

5. Construction and Infrastructure Monitoring

Construction sites are dynamic, unstructured environments where hybrid quadruped robots offer significant advantages. They can perform inspections, transport materials, monitor structural integrity, and assist in quality control across uneven surfaces and partially constructed pathways. These robots enable detailed inspection of bridges, tunnels, buildings, and other critical infrastructure components while navigating debris, scaffolding, and temporary structures that would otherwise hinder traditional robotic systems. Their ability to collect high-fidelity visual and sensor data supports better project management and worker safety.

6. Space Exploration

Hybrid quadruped robots are highly promising for planetary exploration missions, where surface conditions are unpredictable and highly variable. Their ability to roll efficiently over flat plains and then switch to legged motion to climb craters, rocks, or steep slopes makes them ideal for extraterrestrial exploration on planets like Mars or the Moon. These robots can carry scientific instruments, collect geological samples, and perform critical tasks autonomously in harsh and remote environments while reducing risks to astronauts and human explorers.

7. Medical and Assistive Applications

Looking into future applications, hybrid quadruped robots have potential roles in healthcare and assistive robotics. They can serve as mobility aids for elderly or disabled individuals, capable of safely navigating both indoor and outdoor environments with varying surfaces such as stairs, ramps, and uneven flooring. These robots can be adapted to provide patient transportation, support home-based care, and assist healthcare workers by delivering medical supplies or equipment, particularly in complex environments where conventional wheelchairs or walking aids may be ineffective.

References

Research Papers

1. Raibert, M., Blankespoor, K., Nelson, G., & Playter, R. (2008). *BigDog, the rough-terrain quadruped robot*. Proceedings of the 17th World Congress of the International Federation of Automatic Control (IFAC), 10822–10825.

2. Kim, S., Park, H. W., Lee, D., & Seo, K. (2019). *Highly dynamic quadruped robot with improved agility and robustness*. IEEE Robotics & Automation Magazine, 26(3), 21–30.

3. Focchi, M., Barasuol, V., Havoutis, I., et al. (2017). *Heuristic planning for rough terrain locomotion in wheeled–legged robots*. IEEE Robotics and Automation Letters, 2(3), 1345–1352.

4. Kim, J., & Oh, J. H. (2020). *Design and control of a hybrid quadruped robot with wheel-actuated feet*. IEEE Transactions on Robotics, 36(5), 1457–1472.

5. Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., & Hutter, M. (2019). *Learning quadrupedal locomotion over challenging terrain*. Science Robotics, 4(26), eaau5872.

6. Wu, Y., Ding, X., Yang, J., & Zhao, Y. (2021). *Terrain-aware locomotion mode switching for hybrid wheeled–legged robots*. IEEE/ASME Transactions on Mechatronics, 26(4), 2105–2116.

7. Ma, J., Zhao, N., Zhang, C., & Chen, Q. (2018). *Mechanical design and analysis of a wheeled–legged quadruped robot*. Journal of Mechanical Design, 140(9), 093301.

8. Zhao, X., Liu, Y., Wang, Z., & Chen, B. (2020). *Lightweight modular design of hybrid quadruped robots for enhanced mobility*. Robotics and Autonomous Systems, 124, 103394.