LOCOMOTION OF A QUADRUPED ROBOT

A MINI-PROJECT REPORT

submitted by

SANJITH SAJI (MGP22URB085)
SEFIN SURESH (MGP22URB086)
P ARJUN (MGP22URB077)

in partial fulfilment of the requirements
for the award of the degree of

Third Year B.Tech

(ROBOTICS AND AUTOMATION ENGINEERING)

2022-2026

DEPARTMENT OF ELECTRONICS ENGINEERING
SAINTGITS COLLEGE OF ENGINEERING (AUTONOMOUS)
KOTTAYAM

APRIL 2025

CERTIFICATE

This is to certify that the mini project report entitled **LOCOMOTION OF A QUAD RUPED ROBOT**,submitted by **SANJITH SAJI (MGP22URB085)**,**SEFIN SURESH (MGP22URB086)**, **P ARJUN (MGP22URB077)**, to the A P J Abdul Kalam Technological University, in partial fulfillment of the requirements for the award of the Degree of **Bachelor of Technology in Robotics And Automation** from **Saintgits College of Engineering**, is a bonafide record of the project work carried out. This report in any form has not been submitted to any other University or Institute for any purpose.

Er. Chinn Mohanan
Project Guide
Assistant Professor
Dept. of Electronics Engineering

Er.Bini Thomas
Project Co-ordinator
Assistant Professor
Dept. of Electronics Engineering

External Examiner

Er.Rizwana Akbar
Program Co-ordinator
Assistant Professor
Dept. of Electronics Engineering

Place: Kottayam
Date:

Department of Electronics
Robotics and Automation

Vision

To achieve academic and professional excellence in emerging areas of Electronics engineering, fostering a value-based, sustainable smart living society.

Mission

- Deliver quality technical education in Electronics and allied engineering streams.
- Develop a robust research culture, promoting interdisciplinary engineering solutions.
- Share state-of-the-art knowledge and facilities to mentor institutes in quality improvement.
- Inspire innovation and entrepreneurial spirit to build a sustainable and connected society.

Program Specific Outcomes (PSO)

- **PSO1:** Ability to apply the fundamental concepts of robotics and automation to develop solutions for simple problems.
- **PSO2:** Ability to design and implement intelligent systems using the skills and expertise acquired.

SAINTGITS COLLEGE OF ENGINEERING(AUTONOMOUS)

DEPARTMENT OF ELECTRONICS ENGINEERING

KOTTAYAM, KERALA (INDIA)

DECLARATION

We, **SANJITH SAJI (MGP22URB085),SEFIN SURESH (MGP22URB086), P ARJUN (MGP22URB077)** hereby declare that this thesis entitled "**LOCOMOTION OF A QUADRUPED ROBOT**" is the record of the original work done by us under the guidance of **Er. Chinn Mohanan**, Assistant Professor, Department of Electronic Engineering, Saintgits College of Engineering. To the best of my knowledge, this work has not formed the basis for the award of any degree/diploma/fellowship or a similar award to any candidate in any University.

Place: **Signature of the Student**

Date:

Er. Chinn Mohanan
Assistant Professor
(Project Advisor)

Er.Bini Thomas
Assistant Professor
(Project Coordinator)

Contents

Acknowledgements

We express our deepest gratitude to our Principal, **Dr. Sudha T.**, Professor and Principal of Saintgits College of Engineering (Autonomous), for providing a supportive environment that has greatly contributed to the successful completion of this project.

I also thank **Dr. Sreekala K S**, Professor and Head of the Department of Electronics Engineering for her advice and encouragement.

Our sincere thanks go to **Er.Rizwana Akbar** Assistant Professor in the Department of Electronics Engineering at Saintgits College of Engineering (Autonomous), for providing all the departmental facilities

Our sincere thanks go to **Er.Bini Thomas** Assistant Professor in the Department of Electronics Engineering at Saintgits College of Engineering (Autonomous), for her continuous guidance, encouragement, and valuable insights throughout the duration of this project..

We are especially grateful to **Er. Chinn Mohanan** Assistant Professor in the Department of Electronics Engineering, for his constant supervision, timely advice, and valuable support in every stage of this project.

Additionally, we are thankful to all the faculty members of the department for their guidance, help, and support throughout this project. We also express our deep gratitude to our parents, friends, and well-wishers for their unwavering support, encouragement, and timely help.
Finally, we thank Almighty God for providing us with the strength, patience, and guidance necessary to complete this project successfully.

List of Figures

Abstract

This paper presents the design and development of a quadruped robot aimed at achieving stable locomotion on smooth surfaces. The primary objective of this project is to develop a four-legged robotic system capable of executing a trot gait efficiently while maintaining balance and coordination.

The quadruped robot utilizes an ESP32 microcontroller as the central control unit, which processes data and commands for movement. The robot is equipped with PCA9685 motor drivers and high-torque servo motors to control limb movements. The gait mechanism is implemented through precise servo control, ensuring synchronized leg motion for efficient mobility.

To enhance its autonomous capabilities, the robot incorporates an MPU6050 IMU sensor for real-time orientation feedback and an ultrasonic sensor for basic obstacle detection. A 12v 200watt SMPS powers the system, with a buck converter ensuring proper voltage regulation.

The quadruped robot offers several advantages, including improved mobility over conventional wheeled robots in specific environments. Its legged locomotion makes it suitable for applications such as search-and-rescue, exploration, and surveillance, particularly on terrain where wheels struggle. The modular design of the robot also allows for future enhancements, including AI-based path planning and terrain adaptation. This project contributes to the field of robotics and automation by demonstrating an efficient approach to quadruped locomotion, serving as a foundation for further advancements in autonomous robotic mobility.

KEYWORDS : Quadruped Robot, Gait Mechanism, ESP32, Autonomous Navigation.

Chapter 1

Introduction

Quadruped robots are advanced, four-legged machines designed to operate effectively in environments where wheeled or tracked robots struggle. Their ability to navigate debris, unstable surfaces, and confined spaces makes them essential in search and rescue (SAR) missions, especially during natural disasters like earthquakes, building collapses, or landslides. These robots are built for rugged terrains and play a vital role in reaching areas that are inaccessible or dangerous for human responders.

To ensure accurate movement and positioning, the robot integrates GPS technology for real-time location tracking, which helps in coordinating with rescue teams and mapping disaster zones efficiently. The robot's mobility is enhanced through an inverse kinematics-based control system, allowing precise control of its joints for better stability. The implementation of a trot gait algorithm enables swift, balanced, and agile movement, crucial for navigating complex and unpredictable environments.

The robot is powered by an ESP32 microcontroller, which manages motor control and wireless communication. The ESP32 ensures responsive locomotion and supports real-time operation, while its built-in Wi-Fi and Bluetooth allow remote monitoring and control. This functionality is particularly beneficial in disrupted communication zones, enabling quick adjustments and deployments during rescue missions. Altogether, the combination of adaptive movement, precise positioning, and remote control capabilities makes the quadruped robot a powerful tool for enhancing the safety and efficiency of emergency response operations.

Chapter 2

Literature Survey

- **Yongyong Zhao, Jinghua Wang (Member, IEEE), Guohua Cao, Yi Yuan, Xu Yao, and Luqiang Qi — "Intelligent Control of Multilegged Robot Smooth Motion: A Review":**
  In the journal titled Decision-Making and Locomotion in Quadruped Robots for Search and Rescue, the authors explore advanced locomotion techniques and the integration of real-time sensor fusion to enhance quadruped robot navigation in complex and unpredictable terrains. The study emphasizes the effectiveness of the trot gait algorithm, which enables quadruped robots to move with speed and stability in disaster environments.

- **Hutter et al., IEEE Robotics and Automation Letters (RAL), 2017 — "ANYmal: A Dynamic Quadruped Robot":**
  In the journal titled Agile Locomotion and Terrain Adaptation in Quadrupedal Robots, Hutter et al. present the development of ANYmal, a dynamic quadruped robot equipped with torque-controlled joints and advanced locomotion strategies. The study highlights the importance of real-time feedback and autonomous terrain adaptation for enabling energy-efficient and agile movement in unstructured environments. The integration of torque-controlled actuation enhances the robot's ability to maintain balance and stability on complex terrains.

- **Kim et al., IEEE/RSJ IROS, 2018 — "Cheetah 3: Blind Locomotion":**
  In the study titled Cheetah 3: Blind Locomotion, Kim et al. present a high-performance quadruped robot capable of navigating complex environments using proprioceptive feedback without relying on visual sensors. The research showcases how blind locomotion is achieved through dynamic control strategies, torque-based actuation, and real-time feedback, enabling the robot to maintain stability and agility even without visual input.

- **Kolvenbach et al., Robotics and Automation Letters (RAL), 2019 — "Vision-Assisted Quadruped Control":**

  In the journal titled Vision-Assisted Quadruped Control, Kolvenbach et al. explore the integration of advanced sensing and perception systems to enhance the stability, adaptability, and decision-making capabilities of quadruped robots operating in complex and unpredictable terrains. The study emphasizes how real-time environmental awareness, achieved through onboard sensor data, contributes to dynamic terrain adaptation and precise locomotion control.

- **Raibert et al., IEEE Robotics and Automation, 2008 — "BigDog: The Rough-Terrain Quadruped":**

  In the journal titled BigDog: The Rough-Terrain Quadruped, Raibert et al. present the development of BigDog, a quadruped robot designed for high-performance locomotion in extreme environments. The robot employs hydraulic actuators and dynamic stability control, enabling it to carry heavy loads and traverse rugged, uneven terrain with remarkable robustness. The study showcases how dynamic actuation and terrain-aware control can significantly enhance a robot's adaptability and performance in real-world outdoor conditions.

- **P. M. Wensing, S. Kim, and J. Lee, IEEE/RSJ IROS, 2018 — "Dynamic Locomotion of Quadruped Robots":**

  In the journal titled Dynamic Locomotion of Quadruped Robots, Wensing, Kim, and Lee explore how sensor integration and terrain perception play a crucial role in the planning and execution of gait algorithms for quadruped robots. The study emphasizes the benefits of real-time terrain analysis for selecting optimal gait patterns that ensure stability, agility, and energy efficiency across varying environments.

Chapter 3

Locomotion of A Quadruped Robot

3.1 Project Objective

The objective of this project is to design and develop a quadruped robot capable of achieving stable and efficient locomotion using a predefined trot gait mechanism. The robot is intended to operate effectively on smooth surfaces, serving as a platform to explore legged mobility where traditional wheeled robots may face limitations.

This project specifically focuses on implementing and testing a trot gait, enabling the robot to move in a synchronized and balanced manner using position-controlled servo motors. The gait control has been successfully developed, allowing the robot to perform basic locomotion with consistent step patterns and timing.

An ESP32 microcontroller serves as the core of the control system, executing the gait logic and managing motor commands. The robot uses PCA9685 motor drivers to precisely control high-torque servo motors that actuate each joint of the quadruped's legs. The servo actuation is programmed to follow a coordinated sequence, ensuring smooth and repeatable movement across all four legs. Power is supplied using a 12V 200W SMPS, regulated via a buck converter to match the requirements of the electronic components and actuators.
This quadruped robot offers a modular design that can be enhanced in future iterations with features such as autonomous path planning, terrain adaptation, and AI-based control strategies. By successfully implementing a basic trot gait using low-cost hardware and a lightweight control system, this project contributes to the advancement of quadruped locomotion in robotics and automation

3.2 Problem Statement

Traditional wheeled and tracked robots often struggle to traverse irregular or complex terrains, especially in applications such as search and rescue, surveillance, or exploration, where mobility and adaptability are critical. These limitations restrict their effectiveness in environments with debris, steps, or uneven surfaces.

Quadruped robots offer a promising alternative due to their ability to mimic the legged locomotion of animals, providing enhanced stability and mobility across a wider range of terrains. However, developing a quadruped robot that can execute a stable and efficient gait while maintaining balance and coordination remains a challenging task, particularly when using cost-effective, position-controlled servo motors and resource-constrained microcontrollers.

This project addresses the challenge of implementing a reliable trot gait for a quadruped robot using an ESP32 microcontroller and PCA9685-based servo control, without relying on complex feedback systems or high-end hardware. The aim is to establish a functional and adaptable robotic platform that can perform coordinated leg movements for smooth locomotion on flat surfaces, while laying the groundwork for future upgrades in autonomy and terrain adaptability.

Chapter 4

Methodology

4.1 Methodology

The quadruped robot was developed based on an open-source design, which was analyzed and customized to align with the project's goals. The enhancements were made to optimize the range of motion for each leg and to reduce structural stress during movement. All mechanical parts, including the main chassis, legs, and motor mounts, were 3D-printed using PLA+ material, chosen for its strength-to-weight ratio, ease of printing, and impact resistance—making it suitable for continuous servo actuation and light terrain interaction.

For actuation, high-torque servo motors were selected based on analysis of the torque and speed requirements necessary for lifting and moving the limbs in a coordinated manner. A total of 12 servos were used—three per leg (shoulder, thigh, knee)—to replicate natural quadruped movement. The ESP32 microcontroller served as the central control unit due to its processing capability and built-in wireless communication, which allows remote operation and future scalability. To control multiple servos, the PCA9685 16-channel PWM driver was integrated, ensuring precise and simultaneous control of all servos. Power to the system was provided via a 12V 200W SMPS, with a buck converter used to regulate voltage to the appropriate levels required by the electronics and servos.

The electrical layout included robust wiring, appropriate insulation, and fail-safe power distribution mechanisms to prevent overloads during dynamic movements. The control software was developed using the Arduino framework, programmed onto the ESP32. The software was structured into layers for motor control, gait sequencing, and system state management. A trot gait was implemented as the primary locomotion pattern, chosen for its balance between speed and stability. This gait involves moving diagonal leg pairs in synchrony (front-left with rear-right, and vice versa), enabling efficient traversal of flat or slightly uneven terrain. Servo movements were defined by angle sequences translated into PWM signals, and these sequences were fine-tuned through testing. The code also included basic position transitions to switch between forward and backward motion, along with an idle state for calibration. Though no real-time sensor feedback was implemented in this phase, the system was designed with modularity in mind, allowing for future integration of sensor data for adaptive gait control.

4.2 Design

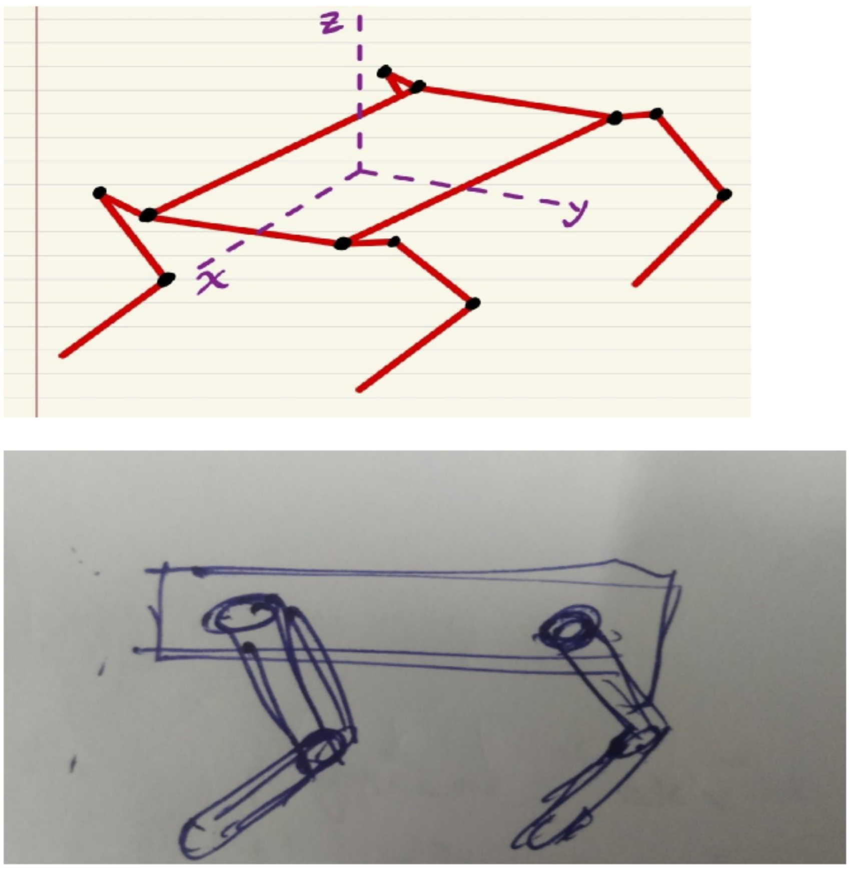

Figure 4.1: Design

4.3 System Architecture

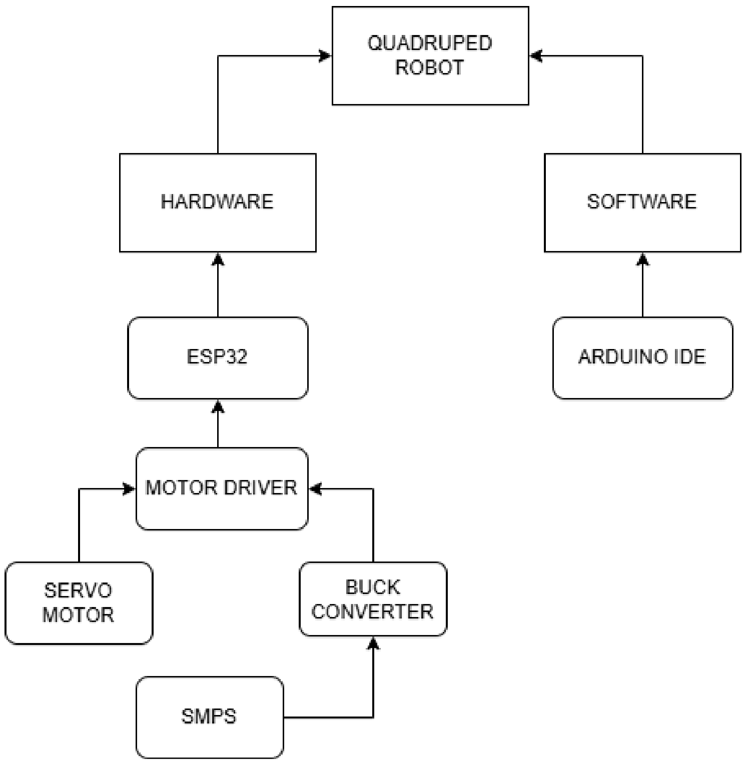

Figure 4.2: Block Diagram

4.3.1 ESP 32

The ESP32 is a powerful microcontroller that features a dual-core processor, integrated Wi-Fi and Bluetooth capabilities, and a wide range of I/O interfaces. It is widely used in robotics and embedded systems due to its high processing power, low energy consumption, and wireless communication features. The ESP32 provides flexibility for real-time control, sensor integration, and remote monitoring, making it an ideal choice for quadruped robot applications.

Architecture of ESP32

- **Microcontroller:**

  The ESP32 is powered by a dual-core Xtensa LX6 processor, operating at up to 240 MHz. It features 4MB of flash memory for program storage and 520KB of SRAM for data handling, making it significantly more powerful than traditional microcontrollers like the ATmega328P.

- **Digital I/O Pins:**

  The ESP32 has 34 programmable GPIO (General Purpose Input/Output) pins, out of which multiple can be used for Pulse Width Modulation (PWM), interrupts, and serial communication. These pins facilitate interfacing with sensors, actuators, LEDs, and displays.

- **Analog Inputs:**
  The ESP32 supports 18 analog input pins (ADC1 & ADC2) with a 12-bit resolution, allowing finer sensor data readings. It also features two Digital-to-Analog Converters (DACs) for analog output generation.

- **Power Supply:**
  The ESP32 operates at 3.3V, unlike the Arduino Uno, which runs on 5V. It can be powered via USB, a 3.3V power source, or a battery, making it suitable for low-power applications.

- **Communication:**
  The ESP32 comes with built-in Wi-Fi (802.11 b/g/n) and Bluetooth (v4.2, BLE), allowing for wireless control and data transmission. It also supports multiple serial communication protocols like UART, SPI, I2C, and I2S for external device integration.

- **Programming:**
  The ESP32 can be programmed using C/C++ via platforms such as the Arduino IDE, Espressif IDF (IoT Development Framework), MicroPython, and PlatformIO. It provides a robust development environment for both beginners and professionals in embedded systems.

- **Expansion:**
  The ESP32 is compatible with a wide range of expansion modules and development boards, which provide additional functionalities such as Wi-Fi connectivity, Bluetooth, LoRa, motor control, and more. It supports direct integration with sensors, actuators, and display modules through its GPIO, I2C, SPI, and UART interfaces, enabling seamless expansion.

- **Open-Source:**
  Being open-source, the ESP32 benefits from a strong developer community and extensive documentation. Numerous libraries and development tools are available, enhancing its usability for rapid prototyping and deployment.

4.3.2 Servo Motor

A servo motor is a type of actuator designed for precise position control using PWM signals. In our project, we use high-torque 35 kg-cm servo motors, which provide accurate angular positioning critical for our quadruped robot.

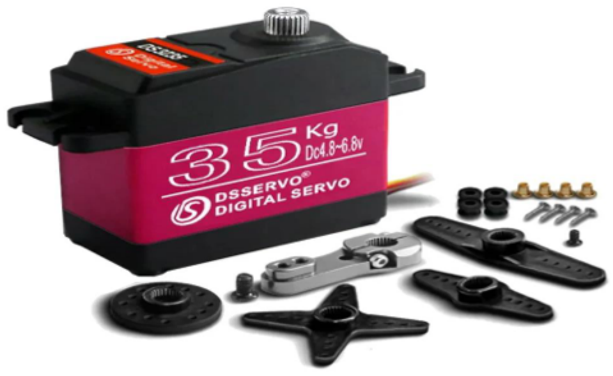

Figure 4.3: Servo Motor

- **Precise Positioning:**
  Servo motors are engineered to move to specific positions based on PWM input signals. This allows for accurate control of the robot's joints, ensuring coordinated and repeatable movements.

- **Integrated Feedback:**
  Many servo motors have built-in feedback mechanisms that help verify that the desired position is reached, enhancing the reliability of the motion control system.

- **Compact and High-Torque Design:**
  The compact design of servo motors enables high torque output without the bulk associated with traditional geared DC motors, making them ideal for applications with limited space.

- **Efficiency in Dynamic Applications:**
  Servos are optimized for dynamic response, providing quick and precise adjustments which are essential for the agile movements of a quadruped robot.

- **Versatility:**
  Servo motors find widespread use in robotics, automation, and various applications that demand fine positional control, making them well-suited for our design requirements.

4.3.3 Motor Diver(PCA9685)

The PCA9685 is a 16-channel, 12-bit PWM driver that controls multiple servo motors simultaneously, communicating with the Raspberry Pi via the I2C interface. This driver is a key component in our project, replacing the need for traditional motor drivers such as the L298N.

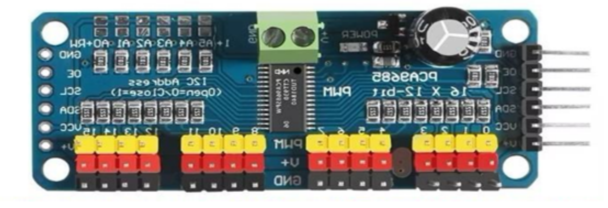

Figure 4.4: Motor Driver

- **Multi-Channel Control:**
  With 16 independent channels, the PCA9685 efficiently manages the 12 servo motors used in our robot, while leaving additional channels available for future expansion.

- **High Resolution PWM Output:**
  The 12-bit resolution provides 4096 discrete steps, ensuring smooth and precise control of each servo's position.

- **Reduced Processor Load:**
  By handling the PWM signal generation externally, the PCA9685 offloads this task from the Raspberry Pi, allowing for more efficient processing and improved overall system performance.

- **Wide Voltage Compatibility:**
  The driver is compatible with both 3.3V and 5V logic levels, simplifying integration with various microcontroller platforms and ensuring versatile application.

- **Built-In Oscillator and Stability:**
  An internal oscillator maintains consistent timing for PWM outputs, which is crucial for the synchronized movements required in our robotic applications.

- **Logic Compatibility:**
  The PCA9685 servo driver is compatible with both TTL (5V) and CMOS (3.3V or 5V) logic levels, making it straightforward to interface with microcontrollers such as the Raspberry Pi, Arduino boards, or other digital control systems.

- **Heat Dissipation:**
  Although the PCA9685 is designed for efficient PWM signal generation, it can generate some heat during extended operation, particularly when driving multiple channels at high loads. It is important to ensure proper PCB layout and ventilation to maintain safe thermal performance.

- **Package:**
  The PCA9685 is typically available in a compact breakout board form factor, which allows for easy integration onto a PCB or breadboard. When implementing the PCA9685 in your project, it is advisable to consult the datasheet and manufacturer guidelines for proper wiring, current management, and overall system integration. This driver is widely used with microcontroller platforms to deliver precise PWM control for multiple servo motors in robotics, CNC machines, motorized vehicles, and other automation projects.

4.4 Software and Hardware Description

4.4.1 Software Requirements

The system requires specific software to achieve optimal results during the project work. Below are the software descriptions:

Arduino IDE and Arduino Programming
The Arduino IDE and programming language provide an accessible platform for beginners to get started with microcontroller programming. Its simplicity, extensive library support, and large community make it a popular choice for hobbyists, students, and professionals alike. The ESP32, used in this project, can be programmed using the Arduino IDE, offering seamless integration for sensor data processing and control.

4.4.2 System Requirement

- **Processor:** Intel Core i5 or above
- **RAM:** 4 GB or more
- **Hard Disk:** 10 GB of available space or more
- **Display:** Dual XGA (1024 × 768) or higher resolution monitors

- **Operating System:** Windows

4.4.3 Hardware Requirements

Base Platform and Frame

The robot consists of a chassis or frame that holds all components together. It is designed to be:

- Sturdy and lightweight
- Provides a stable platform for the quadruped robot

Robot Base

The robot base ensures stability and generally houses the motor or drive system. It is constructed using lightweight and durable materials.

- **Platform Dimensions:** 50cm $\times$ 50cm

Servo Motors (35 kg Torque)

High-torque servo motors are employed for precise leg control, essential for agility and movement across varying terrains.

- **Torque:** 35 kg-cm
- **Specifications:** Suitable for leg actuation and smooth gait transitions

ESP32 Microcontroller

Used instead of the Arduino Uno, the ESP32 offers improved performance and connectivity options.

- **Processor:** Dual-core, 240 MHz
- **Memory:** 520 KB internal SRAM
- **Features:** Built-in Wi-Fi and Bluetooth, ample I/O pins for sensor and actuator integration

Power Supply

The power system consists of a battery and a temporary SMPS for regulation.

- **Battery:** LiPo (Lithium-Polymer)
- **Voltage:** 7.4V (or appropriate for servo motors and ESP32)
- **SMPS:** Switch Mode Power Supply used temporarily until LiPo is fully integrated

4.5 Assembly Process

The assembly process began with 3D-printing all frame and leg components using PLA+ filament. Once printed, each part was thoroughly cleaned, filed, and checked for dimensional accuracy to ensure a proper fit during assembly. The body frame was first assembled using screws and fasteners, followed by the attachment of the upper and lower leg segments for all four legs.

A total of 12 servo motors (3 per leg) were mounted at the joints—hip, thigh, and knee—using appropriate brackets and hardware. Care was taken to align the servos precisely to avoid mechanical binding and to ensure smooth articulation. Servo horns were installed and centered to the neutral position manually before securing them to the limbs.

Each servo motor's signal, VCC, and GND wires were routed cleanly through the frame using spiral wrap and clips to prevent tangling during movement. These wires were connected to a PCA9685 16 channel PWM servo driver module, which handles individual PWM signals for all 12 servos. The PCA9685 was mounted on the robot's frame using standoffs for stability.

A custom power distribution system was installed to regulate and distribute power safely across all components. A 12V, 200W SMPS (Switched-Mode Power Supply) was used as the main power source, connected to the AC mains via a secure three-pin plug. The 12V output from the SMPS was connected to a buck converter, which stepped down the voltage to 6V suitable for the servo motors. A fuse and switch were added for additional safety and power control.

The ESP32 microcontroller was mounted on a perfboard or a custom PCB within the body frame and connected to the PCA9685 via I2C (SCL and SDA lines). The ESP32 was powered separately through a 3.3V/5V regulated line from the buck converter. The wiring ensured isolation between logic-level control and high-current motor power to avoid interference.

Once all electronics were in place, the SMPS was securely mounted on the robot's frame in a position that ensured a low center of gravity and overall balance. Preliminary tests were conducted by powering up the system and checking for correct initialization of the ESP32 and PCA9685. Each servo was then tested for functionality, range, and direction.

The servo motors were calibrated to their neutral (zero) position using a test script to ensure all legs started from a consistent pose. After successful calibration, the final gait control code was uploaded to the ESP32, and functional testing began.Fine-tuning was performed by adjusting servo angles and timing in the code to ensure smooth, synchronized, and stable motion. Final adjustments included tightening all mechanical connections, verifying wire routing, and performing multiple walking tests on flat surfaces

Chapter 5

Challenges Faced

Mechanical Challenges

- **Structural Stability:** Ensuring the robot's frame is both lightweight and strong enough to handle dynamic movement forces.
- **Servo Alignment & Calibration:** Proper positioning and synchronization of the 12 servos is essential for balanced and stable motion.

Motion & Gait Implementation

- **Synchronization of Movements:** Coordinating all four legs to achieve smooth and stable gait patterns.
- **Torque & Power Limitations:** Servos may struggle to lift or move the robot effectively under load, leading to inefficient or failed movements.
- **Slippage & Surface Friction:** Lack of grip on smooth surfaces can cause the robot's feet to slip, reducing walking stability.

Electrical & Hardware Issues

- **Servo Power Management:** High-torque servos draw significant current, which can result in overheating or voltage brownouts.
- **Connections & Wiring Management:** Properly organizing and managing multiple servo wires to avoid tangling or short circuits.

Programming & Control Challenges

- **Gait Algorithm Complexity:** Developing gait algorithms that require precise timing and inter-leg coordination.
- **Servo Control Precision:** Achieving smooth, responsive, and natural-looking motion without jerky or delayed movements.
- **Latency in Commands:** Delays between control commands and actual movements can affect overall robot stability.

Testing & Debugging

- **Unexpected Movements:** Issues such as incorrect joint angles or software bugs may cause erratic behavior.
- **Hardware Failures:** Extended usage can lead to burnt-out or unresponsive servos.
- **Environmental Factors:** Performance may vary across different surfaces such as tile, carpet, or rough terrain.

Chapter 6

Results and Conclusions

The quadruped robot successfully demonstrated basic forward and backward locomotion using a 12-servo gait mechanism. The coordinated movement of its legs validated the effectiveness of its control system, ensuring smooth and stable motion. This achievement marks a significant milestone in its development, as it confirms the robot's ability to execute fundamental movements reliably. The successful implementation of this gait mechanism provides a strong foundation for further enhancements, including optimizing efficiency, improving stability on uneven terrain, and integrating advanced control algorithms for more dynamic and adaptive locomotion.

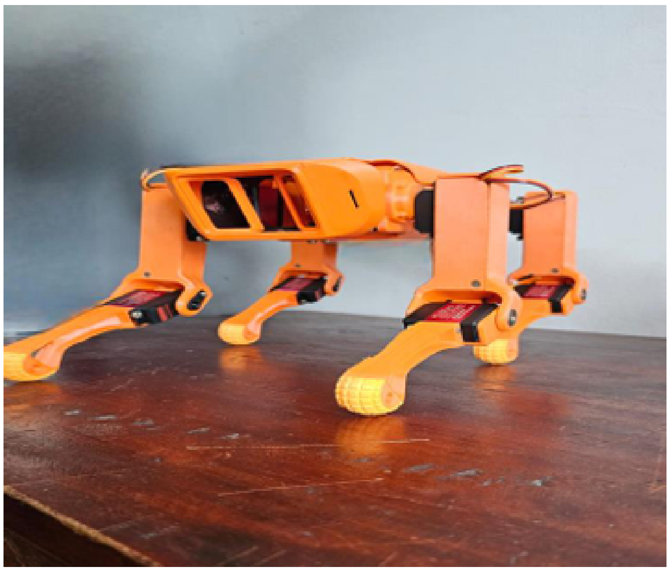

Figure 6.1: Model

Chapter 7

Future Scope

The development of a quadruped robot with basic movement mechanisms has provided valuable insights into robotic locomotion and control. By utilizing 12 servo motors and implementing a fundamental gait mechanism, the project successfully demonstrated basic forward and backward movement. The robot's ability to coordinate leg motions in a structured manner serves as a strong foundation for further advancements in quadruped robotics.

Future improvements can focus on enhancing stability and agility, incorporating real-time feedback sensors for adaptive movement, and implementing more advanced gait algorithms for smoother and more efficient locomotion. Additionally, optimizing power consumption and load distribution will be crucial for improving endurance and practical usability.

This project serves as a steppingstone for further research in robotic mobility, automation, and real world applications such as search and rescue, terrain navigation, and robotic assistance. With continued development, quadruped robots have the potential to contribute significantly to industrial, research, and exploration domains.

Appendix A

Code implementation

Below are the some of the main code snippets used in the project:

A.1 Pseudo Code

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>

// Create PCA9685 driver object
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();

// Servo pulse range
#define SERVO_MIN 100  // Adjust based on your servo’s actual range
#define SERVO_MAX 500

// Define neutral position (zero position) in microseconds
#define ZERO_POS 300  // Adjust this to match your "zero" (usually midpoint)

void setup() {
  Serial.begin(115200);
  Serial.println("Setting up PCA9685...");

  pwm.begin();
  pwm.setPWMFreq(50);  // Standard frequency for servos
  delay(1000);

  // Set all 12 servos to zero position
  for (uint8_t i = 0; i < 12; i++) {
    int pulseLength = ZERO_POS;
    int duty = map(pulseLength, 500, 2500, SERVO_MIN, SERVO_MAX);
    pwm.setPWM(i, 0, duty);

    Serial.print("Servo ");
    Serial.print(i);
    Serial.print(" set to ");
    Serial.println(duty);
  }
}

void loop() {
  // Do nothing
}

/* PSEUDO MOTION SEQUENCE LOGIC (to be implemented with actual function calls)

STEP 1:
MOVE kneeServoLeft TO KNEE_STRAIGHT
WAIT STEP_DELAY

STEP 2:
MOVE hipServoRight TO HIP_BACKWARD
WAIT STEP_DELAY

STEP 3:
MOVE kneeServoRight TO KNEE_BENT
WAIT STEP_DELAY
```

```
MOVE hipServoRight TO HIP_FORWARD
WAIT STEP_DELAY

STEP 4:
MOVE kneeServoRight TO KNEE_STRAIGHT
WAIT STEP_DELAY

MOVE hipServoLeft TO HIP_BACKWARD
WAIT STEP_DELAY

END FUNCTION

// Function: stopMotion()
FUNCTION stopMotion()
  MOVE hipServoLeft TO HIP_NEUTRAL
  MOVE hipServoRight TO HIP_NEUTRAL
  MOVE kneeServoLeft TO KNEE_STRAIGHT
  MOVE kneeServoRight TO KNEE_STRAIGHT
END FUNCTION

*/
```

A.2 Gait Code

```
    #include <Wire.h>
#include <Adafruit_PWMServoDriver.h>

// Initialize the PCA9685 PWM driver
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();

// Define servo min and max pulse lengths
```

```
#define SERVO_MIN 150 // Minimum pulse length
#define SERVO_MAX 600 // Maximum pulse length

// Define leg position names for clarity
#define REAR_LEFT 0 // Pin 0 - Shoulder Rear Left
#define REAR_RIGHT 1 // Pin 1 - Shoulder Rear Right (INVERTED)
#define FRONT_RIGHT 2 // Pin 2 - Shoulder Front Right (INVERTED)
#define FRONT_LEFT 3 // Pin 3 - Shoulder Front Left
#define KNEE_REAR_LEFT 4 // Pin 4 - Knee Rear Left
#define KNEE_FRONT_LEFT 5 // Pin 5 - Knee Front Left
#define KNEE_FRONT_RIGHT 6 // Pin 6 - Knee Front Right (INVERTED)
#define KNEE_REAR_RIGHT 7 // Pin 7 - Knee Rear Right (INVERTED)

// Mark which servos need inverted angle calculations
bool invertedServos[16] = {
  false, true, true, false, false, false, true, true,
  false, false, false, false, false, false, false, false
};

// Standing position angles for all servos
#define STAND_REAR_LEFT_SHOULDER 110
#define STAND_REAR_RIGHT_SHOULDER 110
#define STAND_FRONT_RIGHT_SHOULDER 100
#define STAND_FRONT_LEFT_SHOULDER 100
#define STAND_REAR_LEFT_KNEE 90
#define STAND_FRONT_LEFT_KNEE 90
#define STAND_FRONT_RIGHT_KNEE 80
#define STAND_REAR_RIGHT_KNEE 90

// Rest (low power) position - knees slightly bent
#define REST_KNEE_ANGLE 60

// Define diagonal pairs for optimized movement
```

```
#define DIAGONAL_PAIR_1_SHOULDERS FRONT_LEFT, REAR_RIGHT
#define DIAGONAL_PAIR_1_KNEES KNEE_FRONT_LEFT, KNEE_REAR_RIGHT
#define DIAGONAL_PAIR_2_SHOULDERS FRONT_RIGHT, REAR_LEFT
#define DIAGONAL_PAIR_2_KNEES KNEE_FRONT_RIGHT, KNEE_REAR_LEFT

// Array to track last known positions of servos
int lastKnownPositions[16] = {
  90, 90, 90, 90, 90, 90, 90, 90,
  90, 90, 90, 90, 90, 90, 90, 90
};

// Walking gait parameters - optimized for stability
#define STEP_DELAY 10
#define CYCLE_DELAY 350
#define SHOULDER_SWING 30
#define KNEE_LIFT 40

// Gait pattern parameters
#define DIAGONAL_GAIT 0
#define TRIPOD_GAIT 1
#define RIPPLE_GAIT 2
#define SELECTED_GAIT DIAGONAL_GAIT

// Function prototypes
void setServoPosition(int channel, int angle);
void moveServoSmoothly(int channel, int targetAngle, int delayBetweenSteps);
void centerAllServos();
void moveDiagonalPair(int shoulderA, int shoulderB, int kneeA, int kneeB, int shoulderPo
void optimizedStandUp();
void moveToStandingPosition();
void walkForwardDiagonalOptimized(int steps);
void turnRightOptimized(int steps);
void turnLeftOptimized(int steps);
void restPosition();

void setup() {
  Serial.begin(115200);
  Serial.println("Quadruped robot initialization");

  Wire.begin();
  Wire.setClock(50000);

  if (!pwm.begin()) {
    Serial.println("PCA9685 not detected. Check wiring.");
    while (1);
  }

  pwm.setPWMFreq(50);
  Serial.println("PCA9685 initialized successfully");

  delay(1000);

  Serial.println("Starting optimized stand-up sequence...");
  centerAllServos();
  optimizedStandUp();
  Serial.println("Robot initialized in standing position");
  delay(2000);
}

void centerAllServos() {
  Serial.println("Centering all servos...");
  for (int channel = 0; channel < 8; channel++) {
    setServoPosition(channel, 90);
    delay(100);
  }
  delay(1000);
```

```
}

void setServoPosition(int channel, int angle) {
  angle = constrain(angle, 0, 180);
  int actualAngle = invertedServos[channel] ? 180 - angle : angle;
  lastKnownPositions[channel] = angle;

  int pulseLength;
  if (channel == KNEE_FRONT_RIGHT || channel == KNEE_REAR_RIGHT) {
    pulseLength = map(actualAngle, 0, 180, 125, 625);
  } else {
    pulseLength = map(actualAngle, 0, 180, SERVO_MIN, SERVO_MAX);
  }

  pwm.setPWM(channel, 0, pulseLength);
}

void moveServoSmoothly(int channel, int targetAngle, int delayBetweenSteps) {
  int currentAngle = lastKnownPositions[channel];
  if (currentAngle == targetAngle) return;

  int increment = (currentAngle < targetAngle) ? 1 : -1;

  for (int angle = currentAngle; increment > 0 ? angle <= targetAngle : angle >= targetA
    setServoPosition(channel, angle);
    delay(delayBetweenSteps);
  }

  setServoPosition(channel, targetAngle);
}

void moveDiagonalPair(int shoulderA, int shoulderB, int kneeA, int kneeB, int shoulderPo
  setServoPosition(shoulderA, shoulderPosA);
  setServoPosition(shoulderB, shoulderPosB);
  setServoPosition(kneeA, kneePosA);
  setServoPosition(kneeB, kneePosB);
  delay(delayMs);
}

void optimizedStandUp() {
  Serial.println("Executing optimized diagonal stand-up sequence...");

  moveDiagonalPair(REAR_LEFT, FRONT_RIGHT, KNEE_REAR_LEFT, KNEE_FRONT_RIGHT, 90, 90, RES
  moveDiagonalPair(FRONT_LEFT, REAR_RIGHT, KNEE_FRONT_LEFT, KNEE_REAR_RIGHT, 90, 90, RES

  moveDiagonalPair(FRONT_LEFT, REAR_RIGHT, KNEE_FRONT_LEFT, KNEE_REAR_RIGHT,
                   STAND_FRONT_LEFT_SHOULDER, STAND_REAR_RIGHT_SHOULDER,
                   REST_KNEE_ANGLE, REST_KNEE_ANGLE, 400);

  moveDiagonalPair(FRONT_LEFT, REAR_RIGHT, KNEE_FRONT_LEFT, KNEE_REAR_RIGHT,
                   STAND_FRONT_LEFT_SHOULDER, STAND_REAR_RIGHT_SHOULDER,
                   STAND_FRONT_LEFT_KNEE, STAND_REAR_RIGHT_KNEE, 400);

  moveDiagonalPair(FRONT_RIGHT, REAR_LEFT, KNEE_FRONT_RIGHT, KNEE_REAR_LEFT,
                   STAND_FRONT_RIGHT_SHOULDER, STAND_REAR_LEFT_SHOULDER,
                   REST_KNEE_ANGLE, REST_KNEE_ANGLE, 400);

  moveDiagonalPair(FRONT_RIGHT, REAR_LEFT, KNEE_FRONT_RIGHT, KNEE_REAR_LEFT,
                   STAND_FRONT_RIGHT_SHOULDER, STAND_REAR_LEFT_SHOULDER,
                   STAND_FRONT_RIGHT_KNEE, STAND_REAR_LEFT_KNEE, 400);

  moveServoSmoothly(FRONT_LEFT, STAND_FRONT_LEFT_SHOULDER, STEP_DELAY);
  moveServoSmoothly(FRONT_RIGHT, STAND_FRONT_RIGHT_SHOULDER, STEP_DELAY);
  moveServoSmoothly(REAR_LEFT, STAND_REAR_LEFT_SHOULDER, STEP_DELAY);
  moveServoSmoothly(REAR_RIGHT, STAND_REAR_RIGHT_SHOULDER, STEP_DELAY);
  moveServoSmoothly(KNEE_FRONT_LEFT, STAND_FRONT_LEFT_KNEE, STEP_DELAY);
  moveServoSmoothly(KNEE_FRONT_RIGHT, STAND_FRONT_RIGHT_KNEE, STEP_DELAY);
  moveServoSmoothly(KNEE_REAR_LEFT, STAND_REAR_LEFT_KNEE, STEP_DELAY);
  moveServoSmoothly(KNEE_REAR_RIGHT, STAND_REAR_RIGHT_KNEE, STEP_DELAY);

  Serial.println("Stand-up complete - robot is stable and balanced");
}
```

References

1. Boston Dynamics - Spot Robot.
2. A. Bicchi and L. Marconi, *The Roboticist's Guide to the Control of Quadruped Robots.*
3. L. B. Freeman, J. R. S. Gamboa, and D. H. D. Chien, "Quadruped Robot Design and Control: A Review," *IEEE Transactions on Robotics*, 2019.
4. S.-M. Kim and J. H. Choi, "Quadruped Robot Locomotion: A Survey," *IEEE Access*, 2020.
5. M. J. Schmidt, J. E. DeRuntz, and R. A. Knepper, "The Development and Control of the Quadruped Robot LAURA," *Proc. IEEE IROS*, 2021.
6. K. Y. Lee, J. S. Kim, and H. Y. Lee, *Dynamic Locomotion of Quadruped Robots: Theory and Application.*
7. A. Kalinov and V. Poturaev, "Gait Control for Quadruped Robots Using CPG and Sensors," *Proc. IEEE ICMA*, 2018.
8. J. Zhang and Y. Wang, "Design and Implementation of a Trot-Gait Quadruped Robot," *Proc. IEEE ROBIO*, 2017.
9. B. Siciliano and O. Khatib, *Robotics: Modelling, Planning and Control*, Springer, 2016.
10. M. Hutter, C. Gehring, and M. Bloesch, "StarlETH: A Compliant Quadrupedal Robot," *Proc. IEEE IROS*, 2013.
11. H. Kimura, Y. Fukuoka, and A. H. Cohen, "Adaptive Dynamic Walking of a Quadruped Robot," *International Journal of Robotics Research*, 2007.
12. J. Park and Y. Oh, "Low-Cost Quadruped Robot Design with Trot Gait," *Proc. IEEE ICCAS*, 2015.

13. J. H. Kim and H. S. Lee, "Control Strategies for Quadruped Robots: A Survey," *Proc. IEEE ICAR*, 2002.