Kynex
(More Than just a Robotic Dog)

Group Members:

Zain Ali (231859)

Jawad Ahmad (231802)

Abdul Wasay (231775)

BE MECHATRONICS (Session 2023 - 2027)

Project Supervisor :

Mr. Irfan Minhas

Supervisor Name :

Dr. Neelum Yousuf

DEPARTMENT OF MECHATRONICS ENGINEERING

FACULTY OF ENGINEERING

AIR UNIVERSITY, ISLAMABAD

Kynex
(More Than just a Robotic Dog)

Group Members:

Zain Ali (231859)

Jawad Ahmad (231802)

Abdul Wasay (231775)

BE MECHATRONICS (Session 2023 - 2027)

Project Supervisor :

Mr. Irfan Minhas

Supervisor Name :

Dr. Neelum Yousuf

DEPARTMENT OF MECHATRONICS ENGINEERING

FACULTY OF ENGINEERING

AIR UNIVERSITY, ISLAMABAD

Acknowledgements

First and foremost, we express our sincere gratitude to Allah Almighty, the ultimate source of wisdom and guidance. His blessings have illuminated our path throughout the development of the **Kynex** project, enabling us to face challenges with perseverance and determination.

We would like to express our sincere gratitude to **Dr. Neelum Yousuf**, our Faculty Supervisor, for her continuous guidance, encouragement, and valuable insights throughout the duration of this project. Her expertise and constructive feedback played a vital role in shaping the direction and successful completion of our work.

We are also deeply thankful to **Mr. Irfan Minhas**, our Project Supervisor, for his technical support, practical guidance, and constant motivation during the development of this project. His hands-on approach and willingness to share knowledge greatly enhanced our learning experience.

Furthermore, we would like to acknowledge the support provided by our institution and all those who directly or indirectly contributed to the completion of this project. Their cooperation and encouragement were instrumental in making this project a success.

Abstract

This project presents the design and development of **Kynex**, a robotic dog designed to demonstrate basic principles of robotics, motion control, and human–robot interaction. The robot is capable of performing fundamental actions such as sitting, standing, walking, and barking, which are achieved through coordinated control of actuators and embedded systems.

The primary objective of this project is to create an interactive and educational robotic platform that mimics simple animal behaviors
.
The system integrates mechanical design, electronic hardware, and software control to ensure smooth and stable movement. User interaction is enabled through predefined motion sequences, allowing Kynex to respond in a realistic and engaging manner. This project highlights the practical implementation of robotics concepts, including actuator control, synchronization, and behavioral programming.

The successful development of Kynex demonstrates the potential of low-cost robotic companions for learning, entertainment, and future research in legged robotics. The project also lays a foundation for further enhancements such as sensor-based interaction, autonomous behavior, and advanced motion planning.

Section: BEMTS – 4A – 23

Department: MTS

Team Member	Roll No.	Role	Signature
Zain Ali	231859	• Team Lead • Coding • Simulation	
Jawad Ahmad	231802	• CAD Design • Components Outsourcing • Troubleshooting	
Abdul Wasay	231775	• Circuit Designing • PCB Designing • Project Management	

Table of Content

Page

Chapter 8- Project management

8.1.1 Everyone mus write about how he executed his role in project

8.1.2 Comment individually about success /failure of your project

8.1.3 A paragraph about other team member positive and negative aspects

8.1.4 Final Bill of material list and paragraph about project budget allocation

8.1.5 Risk management that you learned

Chapter 09- Feedback for project and course

Chapter 10- Detailed YouTube video

References

Kynex
(More Than just a Robotic Dog)

Chapter 1 - Preliminaries

1.1.1 Proposal

Robotics has become an essential field in education, research, and entertainment. This project focuses on developing **Kynex**, a robotic dog that can perform basic actions such as sitting, standing, walking, barking, and half-sitting. The aim is to combine mechanical design, electronics, and programming to create an interactive robotic companion that demonstrates fundamental robotics concepts.

Following are the objectives of our project:

- Design and build a legged robotic platform capable of performing predefined motions.
- Implement actuator and motor control to achieve smooth and stable movements.
- Program interactive behaviors like barking and responding to simple commands.
- Provide an educational tool for learning robotics principles and motion control.

Our Methodology for the project will be:

- **Mechanical Design:** Construct a stable, quadruped frame using lightweight materials for durability and mobility.
- **Electronics:** Integrate microcontrollers, motors, sensors, and actuators to enable movement and interaction.
- **Software:** Develop motion sequences, behavioral programming, and control algorithms for actions such as sit, stand, walk, and bark.

- **Testing:** Perform iterative testing to ensure stability, accuracy of movement, and responsiveness.

Following are the outcomes, we are expecting from this project:

- A functional robotic dog capable of basic actions.
- Demonstration of robotics concepts including motion control, actuator coordination, and behavioral programming.
- A platform for future research and enhancements, such as sensor-based interaction and autonomous movement.

This project bridges theoretical knowledge and practical application in robotics. Kynex can be used for educational purposes, entertainment, and as a foundation for future intelligent robotic companions

1.1.2 Initial feasibility

The initial feasibility analysis of the **Kynex Robotic Dog** project indicates that the proposed system is technically, economically, and operationally viable within the given constraints. The project utilizes commonly available electronic components, lightweight mechanical materials, and standard microcontroller platforms, making development practical within the available time and budget.

From a **technical perspective**, the required movements such as sitting, standing, walking, and barking can be achieved using servo motors and basic motion control algorithms. The design does not require highly complex sensors or advanced artificial intelligence, which ensures reliable implementation using existing knowledge and tools.

From an **economic standpoint**, the project is cost-effective as it relies on low-cost hardware and open-source software tools. The overall cost remains manageable, making it suitable for academic and prototype-level development.

From an **operational perspective**, the robotic dog is simple to assemble, operate, and maintain. The system is safe for indoor use and can be easily tested and demonstrated. Therefore, the initial feasibility study confirms that the Kynex project is achievable and suitable for successful completion within the proposed scope.

1.1.3 Technical Standards

The design and development of the **Kynex Robotic Dog** adhere to recognized engineering and technical standards to ensure safety, reliability, and interoperability. Relevant standards from **IEEE**, **ISO**, and general embedded system practices are considered during the project lifecycle.

- **IEEE Standards:**
  The system design follows principles outlined in **IEEE 830 / IEEE 29148** for software requirements specification, ensuring clear definition of functional behaviors and system constraints. Software development practices align with **IEEE 1016** for software design documentation to maintain modularity and readability.

- **Embedded Systems and Communication:**
  Microcontroller programming and digital signal handling conform to standard embedded system design practices as recommended by **IEEE embedded system guidelines**. Signal integrity and timing constraints are maintained to ensure reliable actuator control.

- **Electrical and Power Safety:**
  The robotic dog operates on **low-voltage DC power**, complying with general electrical safety practices similar to **IEC 60950 / IEC 62368** standards for electronic equipment. Proper insulation, grounding, and current protection are implemented to minimize electrical risks.

- **Mechanical and Robotics Standards:**
  Mechanical design considerations align with **ISO 8373** (Robots and robotic devices – Vocabulary) to ensure consistency in robotic terminology and design approach. Structural stability and load limits are maintained to prevent mechanical failure.

- **Software Quality and Testing:**
  Software testing and validation follow structured verification practices inspired by **IEEE 829 / IEEE 29119**, ensuring correct

execution of motion sequences such as sitting, standing, walking, and barking.

By considering these standards, the Kynex project ensures a structured, safe, and academically compliant approach to robotic system development.

1.1.4 Team Roles & Details

The project was developed by a team of three dedicated students from the Department of Mechatronics Engineering, Air University Islamabad. Each member contributed to distinct yet collaborative roles to ensure the successful completion of the project.

- **Zain Ali – Project Lead**
  Zain Ali served as the Project Lead, overseeing overall project execution and coordination. He was primarily responsible for coding and system integration, assembling the robotic dog, and performing troubleshooting. Zain also contributed to component sourcing and financial planning, ensuring smooth progress of the project.

- **Jawad Ahmed – Design and Troubleshooting**
  Jawad Ahmed designed the CAD model of the robotic dog and played a key role in troubleshooting during the development phase. He also contributed to component sourcing and financial management, sharing equal responsibility in these areas.

- **Abdul Wasay – CAD Design Project Management, Documentation**
  Abdul Wasay assisted in the CAD design and was mainly responsible for report writing, project management, and design marketing-related tasks. His contributions ensured proper documentation, structured planning, and effective presentation of the project.

- **Shared Responsibilities**
  Component sourcing and financial management were equally divided among **Team Members**, ensuring cost efficiency and timely availability of required materials.

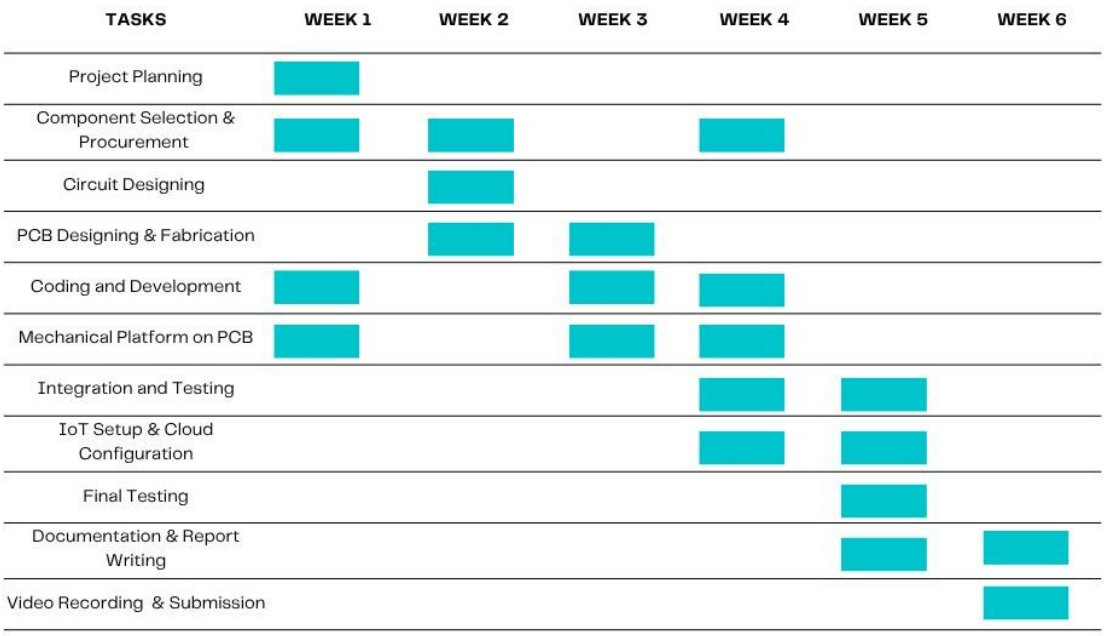
1.1.5 Gantt Chart

The Gantt chart provides a visual timeline of the entire project lifecycle, divided by weeks and tasks. It outlines when each team member should complete specific tasks and how different project phases overlap.

Tasks such as hardware design, component integration, and cloud setup are scheduled in parallel to optimize time. Firmware programming and testing are also staggered to allow feedback incorporation.

Critical paths are highlighted, including cloud integration and sensor interfacing, which must be completed before final testing. Buffer time is allocated toward the end to accommodate unexpected delays.

This project scheduling tool helps maintain team accountability, streamline communication, and ensure timely project delivery.



1.1.6 Estimated Budget

The estimated budget for the project has been carefully calculated to ensure efficient resource allocation while maintaining cost-effectiveness. The major components, along with their quantities and approximate prices in Pakistani Rupees (PKR), are listed below:

No.	Item Name	No. of Items	Price Per Unit (Rs)	Total Cost (Rs)
1	MG90	11	450	4950
2	PCA 986	1	950	950
3	ESP 32	1	1200	1200
4	Ultrasonic Sensor	1	250	250
5	ISD 1820	1	500	500
6	3D Printing	-	-	6000
7	Battery 1800mah	1	600	600
8	3.7 V– 5V Boost Converter	1	100	100
9	5v Charging Module	1	250	250
10	Misc.	-	-	2500
Total Cost			Rs 17300	

Chapter 2 - Project Conception

2.1.1 Introduction

Robotics is a rapidly advancing field that integrates mechanical design, electronics, and software to create intelligent systems capable of performing tasks autonomously or semi-autonomously. One of the emerging areas in robotics is the development of interactive and bio-inspired robots that mimic the behavior and movement of living beings. Such systems play an important role in education, research, and entertainment by demonstrating real-world applications of engineering principles.

This project focuses on the design and development of **Kynex**, a robotic dog capable of performing basic actions such as sitting, standing, walking, barking, and half-sitting. The robot is designed to simulate simple animal-like behavior through coordinated motion control and embedded programming. The primary goal of this project is to provide a functional and interactive robotic platform that helps in understanding fundamental concepts of robotics, including actuator control, motion sequencing, and system integration.

The development of Kynex combines mechanical modeling, electronic circuit design, and software implementation. Emphasis is placed on creating a stable, safe, and cost-effective robotic system using readily available components. The project not only demonstrates practical robotics implementation but also lays the foundation for future enhancements such as sensor-based interaction, autonomous decision-making, and advanced motion control techniques.

2.1.2 Literature Review

Legged robotics has been widely studied due to its ability to navigate complex environments where wheeled robots face limitations. Quadruped robots, in particular, offer improved stability and balance, making them suitable for bio-inspired motion research and educational applications. Early research in legged robotics established the fundamental principles of balance, coordination, and locomotion control that are still applied today [1].

Raibert's pioneering work on dynamically balanced legged robots introduced key concepts of gait planning and stability control, which remain central to quadruped robot design [1]. These principles have been extended in modern robotic systems to achieve smoother and more efficient motion using simplified mechanical structures and control strategies [2].

Recent research has focused on low-cost and educational quadruped robots that emphasize predefined motion sequences rather than complex artificial intelligence. Such designs allow students and researchers to understand actuator coordination, embedded system control, and behavior-based programming without the need for expensive hardware or advanced computation [3]. These systems are particularly suitable for academic projects and prototype development.

Commercial robotic pets such as Sony's AIBO demonstrate the potential of robotic companions in entertainment and interaction. However, their complex architecture and high cost limit their accessibility for educational purposes [5]. In contrast, academic and prototype-level projects aim to

develop affordable robotic platforms that replicate essential behaviors while remaining easy to design, assemble, and program.

Based on the reviewed literature, it is evident that a quadruped robotic system with predefined behaviors is a practical and effective approach for learning robotics fundamentals. The **Kynex Robotic Dog** builds upon these studies by implementing a cost-effective, modular, and educational robotic platform capable of performing basic movements such as sitting, standing, walking, and barking.

2.1.3 List of Features and Operational Specification of Your Project

A. Key Features

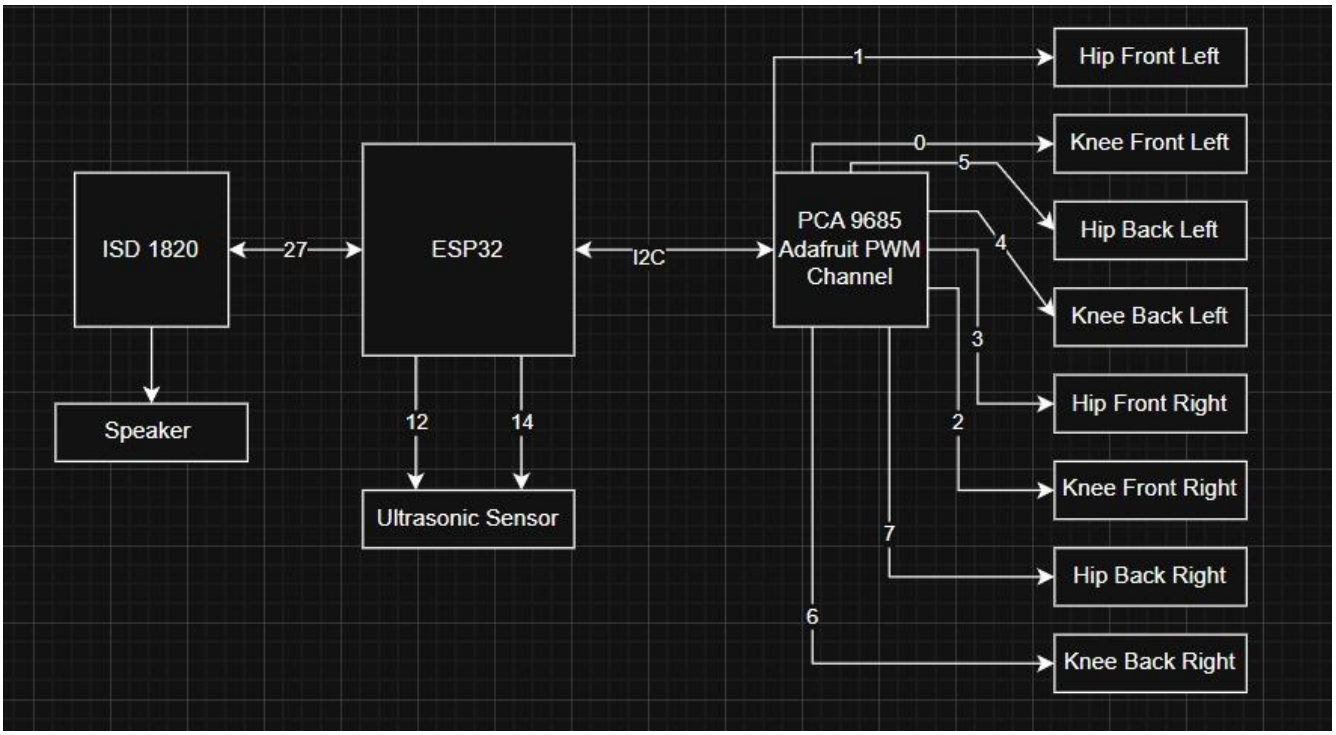
- Quadruped robotic dog design inspired by animal locomotion
- Ability to perform basic actions such as **sit, stand, walk, half-sit, and bark**
- Predefined motion sequences for stable and repeatable movements
- Embedded system–based control using a microcontroller
- Smooth actuator coordination for realistic motion
- Interactive behavior suitable for educational and demonstration purposes
- Compact, lightweight, and safe design for indoor operation
- Cost-effective construction using readily available components

B. Operational Specifications

- **Control System:** Microcontroller-based embedded system
- **Actuation:** Servo motors for leg and body movement
- **Locomotion Type:** Quadruped legged motion
- **Power Supply:** Low-voltage DC power source
- **Operating Environment:** Indoor, flat surfaces
- **Modes of Operation:** Manual / Pre-programmed motion control
- **User Interaction:** Button-based or programmed command execution
- **Safety Features:** Low-voltage operation and controlled movement limits

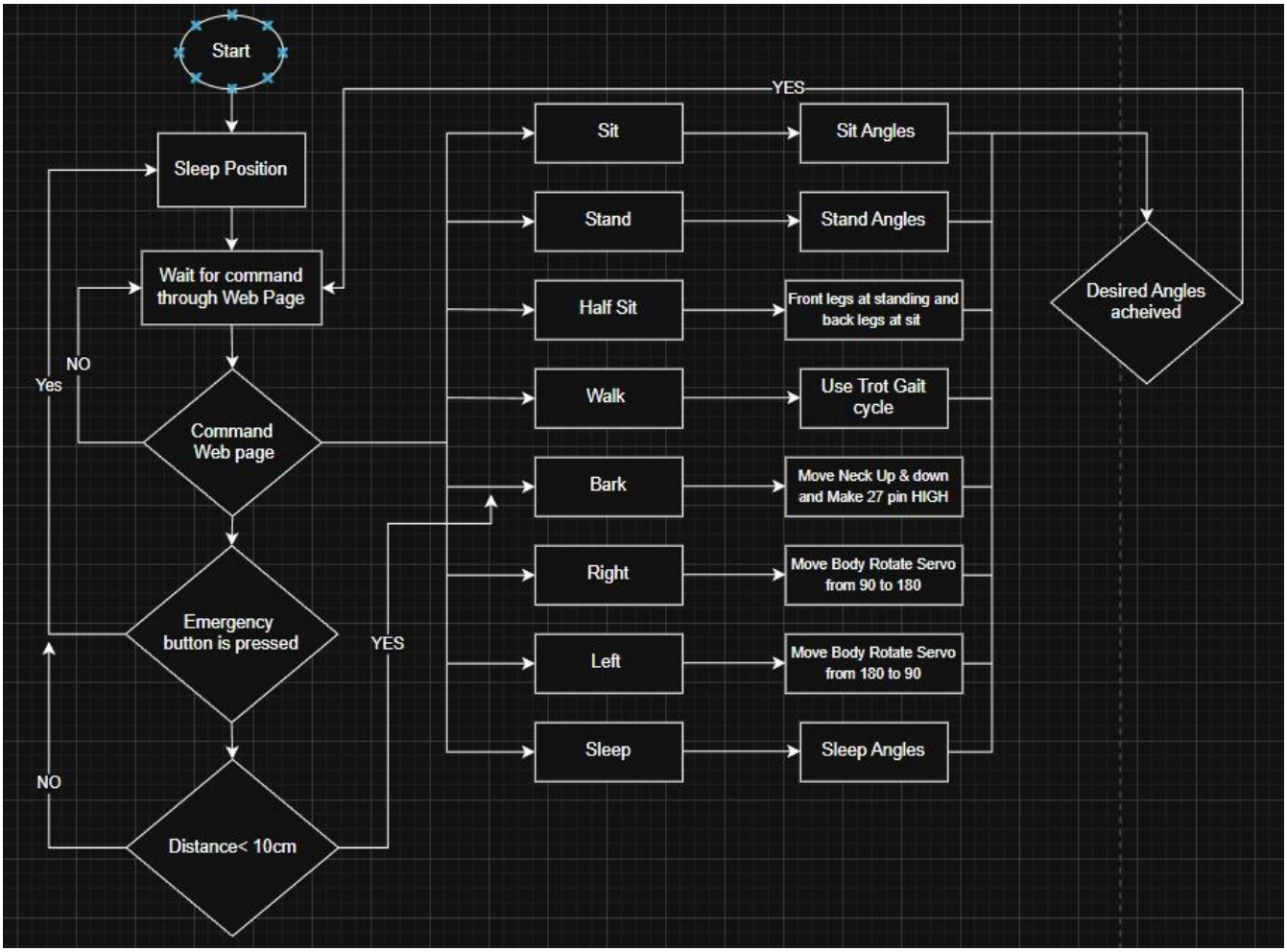
2.1.4 Block Diagram

The block diagram of the Kynex Robotic Dog illustrates the overall system architecture and interaction between its major components. The microcontroller serves as the central control unit, processing predefined motion commands and generating control signals for the servo motors. These motors actuate the legs and body of the robot to perform actions such as sitting, standing, walking, and barking. A power supply unit provides regulated low-voltage DC power to all electronic components, ensuring safe and stable operation. User input or programmed instructions initiate specific behaviors, while all subsystems work together in a coordinated manner to achieve smooth and reliable robotic motion.



2.1.5 Flow Chart

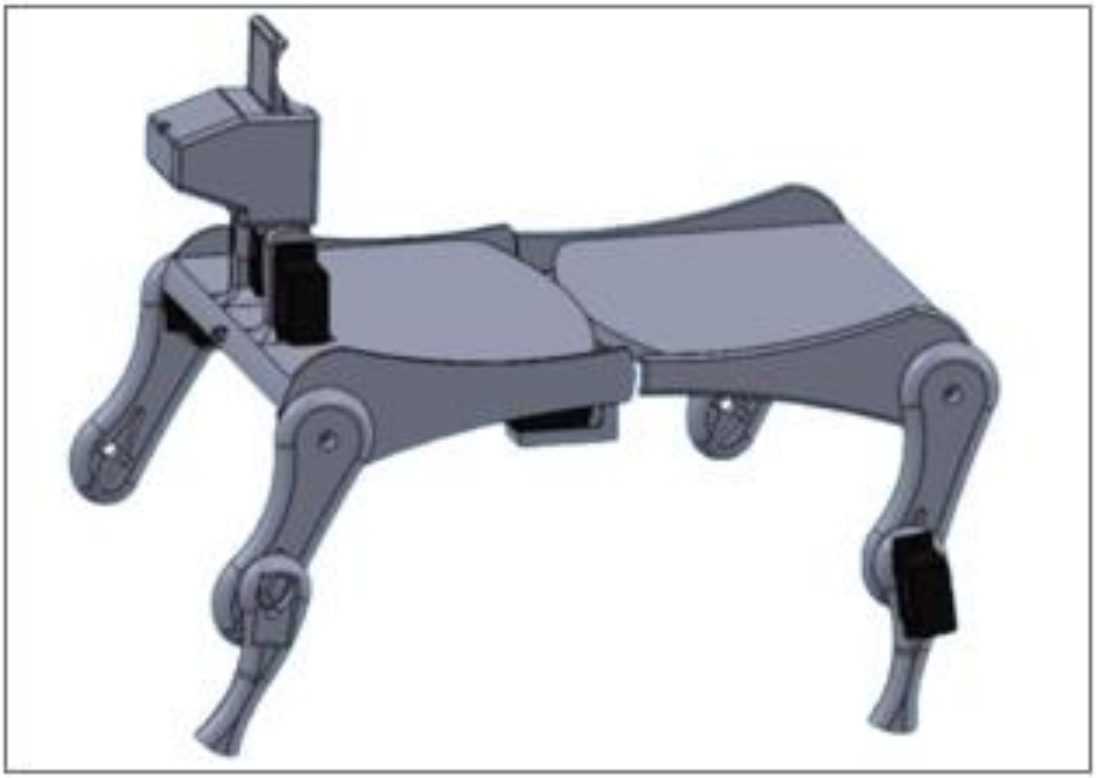
The flowchart of the **Kynex Robotic Dog** represents the logical sequence of operations from start to finish. It begins with system initialization, where the microcontroller sets up all necessary components and checks for proper functionality. Once initialized, the system waits for user input or executes predefined motion sequences. Based on the selected command, the microcontroller processes the instructions and sends signals to the servo motors, controlling the robot's actions such as sitting, standing, walking, or barking. After completing each action, the system either returns to the idle state awaiting the next command or continues with the next programmed sequence, ensuring smooth, coordinated, and reliable operation



Chapter 3- Mechanical Design

3.1.1 Platform Design

The platform design of the **Kynex Robotic Dog** focuses on creating a stable, lightweight, and durable structure that supports smooth motion and reliable operation. The robot features a quadruped frame, with each leg connected to a servo motor to enable precise articulation for actions like walking, sitting, and standing. Materials for the platform are chosen to balance strength and weight, ensuring that the actuators can efficiently move the robot without overloading. The design also considers proper weight distribution, center of gravity, and joint placement to maintain balance during dynamic movements. Additionally, the platform provides sufficient space and mounting points for electronic components, including the microcontroller, sensors, and power supply, facilitating easy assembly, maintenance, and future upgrades.



3.1.2 Material Selection and Choices

The structural components of the **Kynex Robotic Dog**, including the frame and legs, are primarily made from **PLA (Polylactic Acid) plastic**. PLA is chosen for its **lightweight, durable, and easy-to-fabricate properties**, making it ideal for 3D printing the robot's parts. Its rigidity provides sufficient support for smooth motion while minimizing the load on servo motors, which is critical for efficient actuation of walking, sitting, and standing actions.

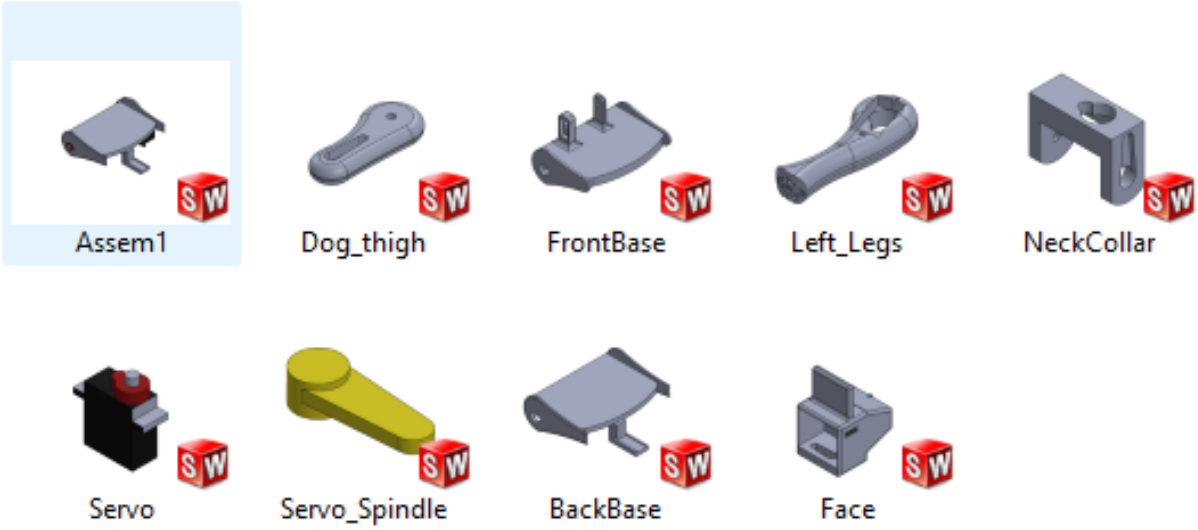
The joints and brackets are also fabricated using PLA, ensuring precise alignment and repeatable motion sequences. Fasteners and connectors are made from standard metal components to maintain secure assembly. PLA's ease of customization allows modifications during the design iteration, making it cost-effective for prototyping. Overall, the choice of PLA ensures that Kynex is robust, lightweight, safe for indoor use, and suitable for educational and prototype applications.

3.1.3 3D CAD Design

Although the project did not include 3D printing due to cost and practicality concerns, a complete 3D CAD model of project was developed using SolidWorks. This digital design helped visualize component placement, motor orientation, wheel alignment, and wire routing before physical implementation.

The CAD model served as a planning and verification tool, allowing the team to finalize dimensions and placements before PCB fabrication. It also supported better collaboration by providing a clear reference for hardware arrangement and mechanical feasibility.

By creating a 3D model despite not producing a printed chassis, the team demonstrated engineering foresight and ensured that all spatial and mechanical considerations were validated during the design phase.



Chapter 4- Electronics Design and Sensor Selections

4.1.1 Component Selection

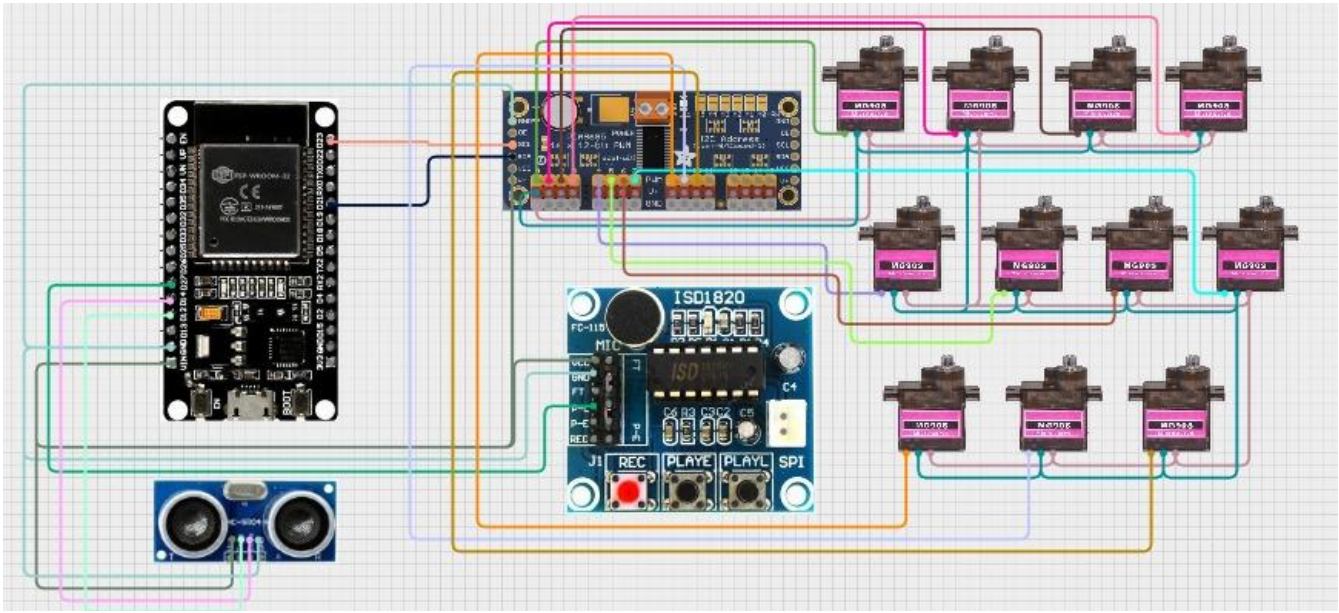
The selection of components for the **Kynex Robotic Dog** was made to ensure efficient performance, cost-effectiveness, and compatibility with the overall system design. Each component was chosen based on its function, reliability, and suitability for a quadruped robotic platform.

1. **MG90 Servo Motors (11 units)** – These servos were selected for their compact size, sufficient torque, and precise control, enabling smooth leg and body movements necessary for walking, sitting, and standing actions.
2. **PCA9685 Servo Driver (1 unit)** – This module allows the microcontroller to control multiple servos simultaneously with accurate PWM signals, simplifying the coordination of all leg joints.
3. **ESP32 Microcontroller (1 unit)** – Chosen for its processing power, multiple GPIO pins, and wireless communication capabilities, the ESP32 acts as the central control unit for executing motion sequences and handling inputs.
4. **Ultrasonic Sensor (1 unit)** – Included for basic obstacle detection and environment sensing, enhancing the robot's interactivity and safety.
5. **ISD1820 Voice Module (1 unit)** – Enables the robot to play pre-recorded sounds, such as barking, to make Kynex more interactive and lifelike.
6. **3D Printed Components** – The robot's structural parts, legs, and brackets are 3D printed using **PLA plastic**, providing lightweight, durable, and customizable mechanical support.
7. **Battery (1800 mAh, 1 unit)** – A rechargeable Li-ion battery powers the robot, providing sufficient energy for servo motors and electronics during operation.
8. **3.7V–5V Boost Converter (1 unit)** – Ensures the regulated voltage required by the servos and microcontroller for stable operation.
9. **5V Charging Module (1 unit)** – Facilitates safe recharging of the battery, ensuring convenient and reliable power management.

10. **Miscellaneous Components** – Additional items such as wires, screws, connectors, and fasteners were included to complete assembly and integration.

4.1.2 Schematic View

The schematic view illustrates the complete electrical connectivity of the system, showing how the ESP32, sensors, stepper motor drivers, and power modules are interconnected. It provides a logical representation that guides circuit understanding and PCB development.



Chapter 5 – Electronics Design and Sensor Selections

5.1.1 Controller Selections with Features

For the **Kynex Robotic Dog**, the **ESP32 microcontroller** was selected as the central control unit due to its robust capabilities, versatility, and suitability for real-time robotic applications. The selection was based on the requirements of handling multiple servo motors, managing sensors, and executing predefined motion sequences efficiently.

Controller Selected: ESP32 Microcontroller

Key Features:

1. **High Processing Power:** Dual-core processor capable of executing real-time control algorithms and motion sequences for multiple actuators simultaneously.
2. **Multiple GPIO Pins:** Sufficient digital and analog I/O pins to interface with 11 servo motors, ultrasonic sensor, voice module, and other peripherals.
3. **PWM Support:** Capable of generating precise Pulse Width Modulation signals necessary for accurate servo control.
4. **Wireless Communication:** Integrated Wi-Fi and Bluetooth modules enable future expansions such as remote control or interaction with mobile devices.
5. **Low Power Consumption:** Optimized energy usage ensures longer operational time when powered by the onboard battery.

6. **Ease of Programming:** Supports Arduino IDE and MicroPython, allowing flexible and rapid development of motion control programs.

7. **Integration Capability:** Easily interfaces with external modules like the PCA9685 servo driver for controlling multiple servos with minimal wiring complexity.

The **ESP32** was chosen over other microcontrollers because it provides a balance between computational capability, connectivity, ease of use, and cost-effectiveness. Its ability to handle multiple tasks simultaneously makes it ideal for controlling a quadruped robotic dog with multiple degrees of freedom and interactive features.

5.1.2 Code

```
#include <Wire.h>
#include <WiFi.h>
#include <WebServer.h>
#include <SPIFFS.h>
#include <Adafruit_PWMServoDriver.h>

Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver(0x40);

// ---------------- CONFIG ----------------
#define SERVO_MIN  50
#define SERVO_MAX  3200
#define SERVO_FREQ 50

int angleToPulse(int angle)
{
  return map(angle, 0, 180, SERVO_MIN, SERVO_MAX);
}


// ================== WIFI ==================
const char* ssid = "QuadrupedDog";
const char* password = "12345678";

WebServer server(80);


// ---------------- CHANNEL MAP ----------------
#define KNEE_FL 0
#define HIP_FL  1
#define KNEE_FR 2
#define HIP_FR  3
```

```
#define KNEE_BL 4
#define HIP_BL  5
#define KNEE_BR 6
#define HIP_BR  7



#define BODY_ROTATE 8
#define NECK_UD     9

#define TRIG_PIN 12
#define ECHO_PIN 14
#define BARK_PIN 27
#define Bark_Full 26

// ---------------- KNEE DIRECTION ----------------
#define KNEE_FL_DIR -1
#define KNEE_FR_DIR +1
#define KNEE_BL_DIR -1
#define KNEE_BR_DIR +1

// ---------------- SITTING ----------------
int sitAngles[8] =
{
  0,   180,
  180, 0,
  180, 0,
  0,   180
};

// ---------------- STANDING ----------------
int standAngles[8] =
{
  98,  148,
```

```
  90,  50,
  110, 60,
  100, 150
};

// ---------------- HALFSIT ----------------
int Halfsit[8] =
{
  150, 95,
  35,  90,
  180, 0,
  0,  180
};

// ---------------- GAIT CONTROL ----------------
bool trotEnabled = false;
char turnMode = 'S';   // S=straight, L=left, R=right


int trotStepDelay = 80;   // SPEED CONTROL
int kneeLift      = 12;
int hipRelief     = 3;

// ---------- WEB SPEED CONTROL ----------
int webSpeed = 80;              // default trot speed (ms)

// ================================================
//                LOW LEVEL SERVO
// ================================================
void setServo(int ch, int angle)
{
  pwm.writeMicroseconds(ch, angleToPulse(angle));
  delay(5);
```

```
}

// ================================================
//                  SIT
// ================================================
void goSit(int stepDelay = 60)
{
 int cur[8];
 for(int i=0;i<8;i++) cur[i]=standAngles[i];
 bool moving=true;

 while(moving)
 {
  moving=false;
  for(int ch : {HIP_FL, HIP_FR, HIP_BL, HIP_BR})
  {
   if(cur[ch]<sitAngles[ch]) cur[ch]++;
   else if(cur[ch]>sitAngles[ch]) cur[ch]--;
   if(cur[ch]!=sitAngles[ch]) moving=true;
   setServo(ch,cur[ch]);
  }
  delay(stepDelay);
 }

 delay(120);

 moving=true;
 while(moving)
 {
  moving=false;
  for(int ch : {KNEE_FL, KNEE_FR, KNEE_BL, KNEE_BR})
  {
   if(cur[ch]<sitAngles[ch]) cur[ch]++;
```

```
      else if(cur[ch]>sitAngles[ch]) cur[ch]--;
      if(cur[ch]!=sitAngles[ch]) moving=true;
      setServo(ch,cur[ch]);
    }
    delay(stepDelay);
  }
}

// ================================================
//                STAND
// ================================================
void goStandFrontBack(int stepDelay = 60)
{
  Serial.println("STAND");

  int cur[8];
  for(int i=0;i<8;i++) cur[i]=sitAngles[i];
  bool moving;

  // FRONT HIPS
  moving=true;
  while(moving)
  {
    moving=false;
    for(int ch : {HIP_FL, HIP_FR})
    {
      if(cur[ch]<standAngles[ch]) cur[ch]++;
      else if(cur[ch]>standAngles[ch]) cur[ch]--;
      if(cur[ch]!=standAngles[ch]) moving=true;
      setServo(ch,cur[ch]);
    }
    delay(stepDelay);
  }
```

```
// BACK HIPS
moving=true;
while(moving)
{
  moving=false;
  for(int ch : {HIP_BL, HIP_BR})
  {
    if(cur[ch]<standAngles[ch]) cur[ch]++;
    else if(cur[ch]>standAngles[ch]) cur[ch]--;
    if(cur[ch]!=standAngles[ch]) moving=true;
    setServo(ch,cur[ch]);
  }
  delay(stepDelay);
}

// FRONT KNEES
moving=true;
while(moving)
{
  moving=false;
  for(int ch : {KNEE_FL, KNEE_FR})
  {
    if(cur[ch]<standAngles[ch]) cur[ch]++;
    else if(cur[ch]>standAngles[ch]) cur[ch]--;
    if(cur[ch]!=standAngles[ch]) moving=true;
    setServo(ch,cur[ch]);
  }
  delay(stepDelay);
}
```

```
  // BACK KNEES
  moving=true;
  while(moving)
  {
    moving=false;
    for(int ch : {KNEE_BL, KNEE_BR})
    {
      if(cur[ch]<standAngles[ch]) cur[ch]++;
      else if(cur[ch]>standAngles[ch]) cur[ch]--;
      if(cur[ch]!=standAngles[ch]) moving=true;
      setServo(ch,cur[ch]);
    }
    delay(stepDelay);
  }
}

// ================================================
//              HALF SIT
// ================================================
void goHalfSit(int stepDelay = 60)
{
  Serial.println("HALF SIT");

  int cur[8];
  for(int i=0;i<8;i++) cur[i]=standAngles[i];
  bool moving=true;

  // ---- ALL HIPS ----
  while(moving)
  {
    moving=false;
    for(int ch : {HIP_FL, HIP_FR, HIP_BL, HIP_BR})
    {
```

```
    if(cur[ch]<Halfsit[ch]) cur[ch]++;
    else if(cur[ch]>Halfsit[ch]) cur[ch]--;
    if(cur[ch]!=Halfsit[ch]) moving=true;
    setServo(ch,cur[ch]);
   }
   delay(stepDelay);
  }

  delay(120);

  // ---- ALL KNEES ----
  moving=true;
  while(moving)
  {
   moving=false;
   for(int ch : {KNEE_FL, KNEE_FR, KNEE_BL, KNEE_BR})
   {
    if(cur[ch]<Halfsit[ch]) cur[ch]++;
    else if(cur[ch]>Halfsit[ch]) cur[ch]--;
    if(cur[ch]!=Halfsit[ch]) moving=true;
    setServo(ch,cur[ch]);
   }
   delay(stepDelay);
  }
}

// ================================================
//            TROT STEP (WITH HIP RELIEF)
// ================================================
void trotDiagonal(int kneeA, int kneeB, int hipA, int hipB)
{
  // HIP RELIEF
  setServo(hipA, standAngles[hipA] + hipRelief);
```

```
  setServo(hipB, standAngles[hipB] + hipRelief);

  for(int i=0;i<=kneeLift;i++)
  {
   int liftA = (turnMode=='L' && kneeA==KNEE_FR) ? i/2 : i;
   int liftB = (turnMode=='R' && kneeB==KNEE_BL) ? i/2 : i;

   setServo(kneeA, standAngles[kneeA] + liftA);
   setServo(kneeB, standAngles[kneeB] + liftB);
   delay(trotStepDelay);
  }

  // REMOVE HIP RELIEF
  setServo(hipA, standAngles[hipA]);
  setServo(hipB, standAngles[hipB]);

  for(int i=kneeLift;i>=0;i--)
  {
   setServo(kneeA, standAngles[kneeA] + i);
   setServo(kneeB, standAngles[kneeB] + i);
   delay(trotStepDelay);
  }
}

// ================================================
//              TROT LOOP
// ================================================
void trotLoop()
{
  if(!trotEnabled) return;

  trotDiagonal(KNEE_FL, KNEE_BR, HIP_FR, HIP_BL);
  trotDiagonal(KNEE_FR, KNEE_BL, HIP_FL, HIP_BR);
```

```
}


// ================================================
//              DANCE
// ================================================

void dance(int stepDelay = 60, int loops = 4)
{
   int cur[10];  // 8 legs + body + neck
   for(int i=0;i<8;i++) cur[i] = standAngles[i];
   cur[BODY_ROTATE] = 90;
   cur[NECK_UD] = 90;

   for(int l=0;l<loops;l++)
   {
      // Forward swing hips + body/neck
      for(int step=0; step<=5; step++)
      {
         for(int i : {HIP_FL, HIP_FR, HIP_BL, HIP_BR})
            setServo(i, standAngles[i] + step);

         // Knees stay at standing angles
         for(int i : {KNEE_FL, KNEE_FR, KNEE_BL, KNEE_BR})
            setServo(i, standAngles[i]);

         setServo(BODY_ROTATE, 90 + step*9); // 45->135 approx
         setServo(NECK_UD, 90 + step*9);    // 45->135 approx

         delay(stepDelay);
      }

      // Backward swing hips + body/neck
```

```
        for(int step=5; step>=-5; step--)
        {
            for(int i : {HIP_FL, HIP_FR, HIP_BL, HIP_BR})
                setServo(i, standAngles[i] + step);

            for(int i : {KNEE_FL, KNEE_FR, KNEE_BL, KNEE_BR})
                setServo(i, standAngles[i]);

            setServo(BODY_ROTATE, 90 + step*9);
            setServo(NECK_UD, 90 + step*9);

            delay(stepDelay);
        }

        // Return hips to standing
        for(int step=-5; step<=0; step++)
        {
            for(int i : {HIP_FL, HIP_FR, HIP_BL, HIP_BR})
                setServo(i, standAngles[i] + step);

            for(int i : {KNEE_FL, KNEE_FR, KNEE_BL, KNEE_BR})
                setServo(i, standAngles[i]);

            setServo(BODY_ROTATE, 90 + step*9);
            setServo(NECK_UD, 90 + step*9);

            delay(stepDelay);
        }
    }
}

// ================================================
//        ULTRASONIC SENSOR AND DISTANCE
```

```
// ============================================================

void setupUltrasonic()
{
   pinMode(TRIG_PIN, OUTPUT);
   pinMode(ECHO_PIN, INPUT);

   digitalWrite(TRIG_PIN, LOW);

   pinMode(BARK_PIN, OUTPUT);
   digitalWrite(BARK_PIN, LOW);

   pinMode(Bark_Full, OUTPUT);
   digitalWrite(Bark_Full, LOW);
}

long readDistanceCM()
{
   digitalWrite(TRIG_PIN, LOW);
   delayMicroseconds(2);
   digitalWrite(TRIG_PIN, HIGH);
   delayMicroseconds(10);
   digitalWrite(TRIG_PIN, LOW);

   long duration = pulseIn(ECHO_PIN, HIGH, 30000); // 30ms timeout
   long distance = duration * 0.034 / 2; // cm
   return distance;
}

//Call this in loop() periodically
void checkObstacle()
{
   long dist = readDistanceCM();
```

```
    Serial.print("Distance: ");
    Serial.print(dist);
    Serial.println(" cm");

    if(dist > 0 && dist < 10) // valid measurement <10cm
    {
       Bark(HIGH);
    }
    else
    {
       Bark(LOW);
    }
}

// ========================================================
//              BARK
// ========================================================

void Bark(bool x)
{
  if(x==HIGH)
  {
    digitalWrite(BARK_PIN, HIGH);
    Serial.print("Barking...");
  }



  else
  digitalWrite(BARK_PIN, LOW);

}
void Barking()
{
```

```
  digitalWrite(Bark_Full, HIGH);
}

// ================================================================
//                Move LEFT / RIGHT
// ================================================================
void MoveLeft(int stepDelay = 30)
{
   int cur = 90;
   while(cur < 110)
   {
      cur++;
      setServo(BODY_ROTATE, cur);
      delay(stepDelay);
   }
}

void MoveRight(int stepDelay = 30)
{
   int cur = 90;
   while(cur > 70)
   {
      cur--;
      setServo(BODY_ROTATE, cur);
      delay(stepDelay);
   }
}

void MoveCenter(int stepDelay = 30)
{
   int cur = 0;
   cur = 90;
```

```
    // smooth transition to 90
    int pos = cur;
    if(pos < 90)
    {
       while(pos < 90)
       {
          pos++;
          setServo(BODY_ROTATE, pos);
          delay(stepDelay);
       }
    }
    else if(pos > 90)
    {
       while(pos > 90)
       {
          pos--;
          setServo(BODY_ROTATE, pos);
          delay(stepDelay);
       }
    }
    else
    {
       setServo(BODY_ROTATE, 90); // already centered
    }
}


// ================================================================
//                  WEBPAGE
// ================================================================
void handleCommand(char cmd)
{
  switch(cmd)
```

```
  {
    case '1': goSit(); break;
    case '2': goStandFrontBack(); break;
    case '3': trotEnabled = true; break;
    case '4': trotEnabled = false; break;
    case '5': goHalfSit(); break;

    case 'L': turnMode = 'L'; break;
    case 'R': turnMode = 'R'; break;

    case 'Q': MoveLeft(); break;
    case 'W': MoveRight(); break;
    case 'C': MoveCenter(); break;

    case 'D': dance(); break;
    case 'B': Barking(); break;
  }
}

void setSpeedFromWeb(int value)
{
  webSpeed = constrain(value, 30, 200);   // safe range
  trotStepDelay = webSpeed;
}

// ================================================
//                HTML & JAVA
// ================================================

void handleRoot()
{
  String html =
    "<!DOCTYPE html>\n"
```

```
  "<html>\n"
  "<head>\n"
 "  <title>Quadruped Control</title>\n"
  "  <meta name=\"viewport\" content=\"width=device-width, initial-
scale=1\">\n"
 "  <style>\n"
 "    body { font-family: Arial; text-align: center; background:#111;
color:#fff; }\n"
 "    button {\n"
 "      width: 120px; height: 45px;\n"
 "      margin: 6px; font-size: 16px;\n"
 "      border-radius: 8px; border: none;\n"
 "    }\n"
  "  </style>\n"
 "</head>\n"
 "<body>\n"
 "<h2> Quadruped Robot Control</h2>\n"
 "<h3>Posture</h3>\n"
 "<button onclick=\"send('1')\">Sit</button>\n"
 "<button onclick=\"send('2')\">Stand</button>\n"
 "<button onclick=\"send('5')\">Half Sit</button>\n"
 "<h3>Gait</h3>\n"
 "<button onclick=\"send('3')\">Start Trot</button>\n"
 "<button onclick=\"send('4')\">Stop Trot</button>\n"
 "<h3>Turning</h3>\n"
 "<button onclick=\"send('L')\">Left</button>\n"
 "<button onclick=\"send('R')\">Right</button>\n"
 "<h3>Speed Control</h3>\n"
 "<input type=\"range\" min=\"30\" max=\"200\" value=\"80\"\n"
 "       oninput=\"setSpeed(this.value)\">\n"
 "<p>Speed: <span id=\"sp\">80</span> ms</p>\n"
 "<h3>Body</h3>\n"
 "<button onclick=\"send('Q')\">Move Left</button>\n"
```

```
 "<button onclick=\"send('W')\">Move Right</button>\n"
 "<button onclick=\"send('C')\">Move Center</button>\n"
 "<h3>Actions</h3>\n"
 "<button onclick=\"send('D')\">Dance</button>\n"
 "<button onclick=\"send('B')\">Bark</button>\n"
 "<script>\n"
 "function send(cmd)\n"
 "{\n"
 "  fetch(\"/cmd?c=\" + cmd);\n"
 "}\n"
 "function setSpeed(v)\n"
 "{\n"
 "  document.getElementById(\"sp\").innerHTML = v;\n"
 "  fetch(\"/speed?v=\" + v);\n"
 "}\n"
 "</script>\n"
  "</body>\n"
  "</html>\n";

  server.send(200, "text/html", html);
}


// ================================================================
//                    SETUP
// ================================================================
void setup()
{
  Serial.begin(115200);
  Wire.begin();
  Wire.setClock(400000);

  pwm.begin();
```

```
pwm.setPWMFreq(SERVO_FREQ);
delay(300);

for(int ch=0; ch<8; ch++)
  setServo(ch, sitAngles[ch]);

// -------- SPIFFS --------
if(!SPIFFS.begin(true))
{
  Serial.println("SPIFFS Mount Failed");
  return;
}

// -------- WIFI AP --------
WiFi.softAP(ssid, password);
Serial.print("AP IP: ");
Serial.println(WiFi.softAPIP());

// -------- ROOT WEBPAGE --------
server.on("/", handleRoot);

// -------- COMMAND API --------
server.on("/cmd", HTTP_GET, []()
{
  if(server.hasArg("c"))
    handleCommand(server.arg("c")[0]);

  server.send(200, "text/plain", "OK");
});

// -------- SPEED API --------
server.on("/speed", HTTP_GET, []()
{
```

```
    if(server.hasArg("v"))
      setSpeedFromWeb(server.arg("v").toInt());

    server.send(200, "text/plain", "OK");
  });

  server.begin();

  setupUltrasonic();
  Serial.println("READY");
}


// ============================================================
//                    LOOP
// ============================================================
void loop()
{
  checkObstacle();
  server.handleClient();

if(Serial.available())
  handleCommand(Serial.read());

  trotLoop();
}
```

5.1.3 Forward Kinematics

```
clc;
clear;
close all;

%
==========================================================
=
% Single Leg Forward Kinematics using Peter Corke Toolbox
%
==========================================================
=

%% Leg Parameters (mm)
thigh_length = 71.83;
shank_length = 64.96;

% Joint angles (radians)
theta1 = deg2rad(325);  % Hip pitch angle
theta2 = deg2rad(-55);  % Knee pitch angle

%% Create 2-DOF Leg Robot using DH parameters
% DH Convention: [theta, d, a, alpha]
% Link 1: Thigh (revolute joint at hip)
% Link 2: Shank (revolute joint at knee)

L(1) = Revolute('d', 0, 'a', thigh_length, 'alpha', 0, 'offset', 0);
L(2) = Revolute('d', 0, 'a', shank_length, 'alpha', 0, 'offset', 0);

% Create serial link robot
robot = SerialLink(L, 'name', 'Dog Leg');
```

```
%% Joint Configuration
q = [theta1, theta2];  % [hip_angle, knee_angle]

%% Forward Kinematics
T = robot.fkine(q);

% Extract end-effector position
try
    foot_pos = T.t';  % RTB 10.x
catch
    foot_pos = transl(T)';  % RTB 9.x
end

%% Display Results
fprintf('=== DH Parameter Table ===\n');
fprintf('Link |  θ (deg)  |  d (mm)  |  a (mm)  |  α (deg)\n');
fprintf('-----|-----------|----------|----------|----------\n');
fprintf('  1  |  %.2f   |   %.2f   |  %.2f  |   %.2f\n', rad2deg(q(1)), L(1).d,
L(1).a, rad2deg(L(1).alpha));
fprintf('  2  |  %.2f   |   %.2f   |  %.2f  |   %.2f\n', rad2deg(q(2)), L(2).d,
L(2).a, rad2deg(L(2).alpha));
fprintf('\nNote: θ values shown are the current joint angles\n');
fprintf('      In DH convention, θ is the variable for revolute joints\n\n');

fprintf('=== Forward Kinematics Results ===\n');
fprintf('Joint Angles: [%.2f°, %.2f°]\n', rad2deg(q(1)), rad2deg(q(2)));
fprintf('Thigh Length: %.2f mm\n', thigh_length);
fprintf('Shank Length: %.2f mm\n', shank_length);
fprintf('\nFoot Position (mm):\n');
fprintf('  X: %.2f\n', foot_pos(1));
fprintf('  Y: %.2f\n', foot_pos(2));
fprintf('  Z: %.2f\n', foot_pos(3));
```

```
fprintf('\n=== Homogeneous Transformation Matrix (T₀²) ===\n');
fprintf('End-effector transform from base to foot:\n\n');
if isobject(T)
    % RTB 10.x - SE3 object
    try
        T_matrix = T.T;
    catch
        T_matrix = double(T);
    end
else
    % RTB 9.x - already a matrix
    T_matrix = T;
end
fprintf('T = \n');
fprintf('  [%8.4f  %8.4f  %8.4f  %8.4f]\n', T_matrix(1,:));
fprintf('  [%8.4f  %8.4f  %8.4f  %8.4f]\n', T_matrix(2,:));
fprintf('  [%8.4f  %8.4f  %8.4f  %8.4f]\n', T_matrix(3,:));
fprintf('  [%8.4f  %8.4f  %8.4f  %8.4f]\n', T_matrix(4,:));

fprintf('\nRotation Matrix (R):\n');
fprintf('  [%8.4f  %8.4f  %8.4f]\n', T_matrix(1:3,1)');
fprintf('  [%8.4f  %8.4f  %8.4f]\n', T_matrix(1:3,2)');
fprintf('  [%8.4f  %8.4f  %8.4f]\n', T_matrix(1:3,3)');

fprintf('\nPosition Vector (P):\n');
fprintf('  [%8.4f  %8.4f  %8.4f]ᵀ mm\n', T_matrix(1:3,4)');

fprintf('\n=== Individual Link Transformations ===\n');
% Calculate individual transformations
T0_1 = robot.links(1).A(q(1));
T1_2 = robot.links(2).A(q(2));

fprintf('\nT₀¹ (Base to Knee):\n');
```

```
if isobject(T0_1)
    try
        T0_1_matrix = T0_1.T;
    catch
        T0_1_matrix = double(T0_1);
    end
else
    T0_1_matrix = T0_1;
end
fprintf('  [%8.4f  %8.4f  %8.4f  %8.4f]\n', T0_1_matrix(1,:));
fprintf('  [%8.4f  %8.4f  %8.4f  %8.4f]\n', T0_1_matrix(2,:));
fprintf('  [%8.4f  %8.4f  %8.4f  %8.4f]\n', T0_1_matrix(3,:));
fprintf('  [%8.4f  %8.4f  %8.4f  %8.4f]\n', T0_1_matrix(4,:));

fprintf('\nT₁² (Knee to Foot):\n');
if isobject(T1_2)
    try
        T1_2_matrix = T1_2.T;
    catch
        T1_2_matrix = double(T1_2);
    end
else
    T1_2_matrix = T1_2;
end
fprintf('  [%8.4f  %8.4f  %8.4f  %8.4f]\n', T1_2_matrix(1,:));
fprintf('  [%8.4f  %8.4f  %8.4f  %8.4f]\n', T1_2_matrix(2,:));
fprintf('  [%8.4f  %8.4f  %8.4f  %8.4f]\n', T1_2_matrix(3,:));
fprintf('  [%8.4f  %8.4f  %8.4f  %8.4f]\n', T1_2_matrix(4,:));

fprintf('\nVerification: T₀² = T₀¹ × T₁²\n');

%% Visualization
figure('Name', 'Leg Forward Kinematics', 'Position', [100 100 1000 800]);
```

```
hold on; grid on; axis equal;
xlabel('X [mm]'); ylabel('Y [mm]'); zlabel('Z [mm]');
title('2-DOF Leg: Forward Kinematics');

% Hip position (origin)
hip_pos = [0, 0, 0];

% Calculate knee position using first link
knee_x = thigh_length * cos(theta1);
knee_y = thigh_length * sin(theta1);
knee_pos = [knee_x, knee_y, 0];

% Calculate foot position using both links
foot_x = knee_x + shank_length * cos(theta1 + theta2);
foot_y = knee_y + shank_length * sin(theta1 + theta2);
foot_pos_manual = [foot_x, foot_y, 0];

% Draw hip joint
plot3(hip_pos(1), hip_pos(2), hip_pos(3), 'ro', ...
    'MarkerSize', 15, 'MarkerFaceColor', 'r', 'LineWidth', 2);
text(hip_pos(1), hip_pos(2), hip_pos(3)+5, 'Hip', 'FontSize', 12,
'FontWeight', 'bold');

% Draw thigh
plot3([hip_pos(1), knee_pos(1)], [hip_pos(2), knee_pos(2)], [hip_pos(3),
knee_pos(3)], ...
    'b-', 'LineWidth', 6);
text(knee_x/2, knee_y/2, 5, sprintf('Thigh (%.1fmm)', thigh_length),
'FontSize', 10);

% Draw knee joint
plot3(knee_pos(1), knee_pos(2), knee_pos(3), 'ko', ...
    'MarkerSize', 12, 'MarkerFaceColor', 'k', 'LineWidth', 2);
```

```
text(knee_pos(1), knee_pos(2), knee_pos(3)+5, 'Knee', 'FontSize', 12, 'FontWeight', 'bold');

% Draw shank
plot3([knee_pos(1), foot_pos_manual(1)], [knee_pos(2), foot_pos_manual(2)], ...
    [knee_pos(3), foot_pos_manual(3)], 'g-', 'LineWidth', 6);
text((knee_x+foot_x)/2, (knee_y+foot_y)/2, 5, sprintf('Shank (%.1fmm)', shank_length), 'FontSize', 10);

% Draw foot
plot3(foot_pos_manual(1), foot_pos_manual(2), foot_pos_manual(3), 'mo', ...
    'MarkerSize', 12, 'MarkerFaceColor', 'm', 'LineWidth', 2);
text(foot_pos_manual(1), foot_pos_manual(2), foot_pos_manual(3)+5, 'Foot', 'FontSize', 12, 'FontWeight', 'bold');

% Draw ground reference
plot3([-20, foot_x+20], [0, 0], [0, 0], 'k--', 'LineWidth', 1);

% Add angle annotations
% Hip angle arc
theta_arc = linspace(0, theta1, 20);
arc_radius = 15;
arc_x = arc_radius * cos(theta_arc);
arc_y = arc_radius * sin(theta_arc);
plot3(arc_x, arc_y, zeros(size(arc_x)), 'r-', 'LineWidth', 2);
text(arc_radius*1.5*cos(theta1/2), arc_radius*1.5*sin(theta1/2), 0, ...
    sprintf('θ₁=%.0f°', rad2deg(theta1)), 'FontSize', 10, 'Color', 'r');

% Knee angle arc
theta_arc2 = linspace(theta1, theta1+theta2, 20);
arc_x2 = knee_x + arc_radius * cos(theta_arc2);
```

```
arc_y2 = knee_y + arc_radius * sin(theta_arc2);
plot3(arc_x2, arc_y2, zeros(size(arc_x2)), 'g-', 'LineWidth', 2);
text(knee_x + arc_radius*1.5*cos(theta1+theta2/2), ...
    knee_y + arc_radius*1.5*sin(theta1+theta2/2), 0, ...
    sprintf('θ₂=%.0f°', rad2deg(theta2)), 'FontSize', 10, 'Color', 'g');

view(2);  % Side view (2D)
xlim([-50 200]);
ylim([-200 100]);

%% Show robot using built-in plot function
figure('Name', 'Robot Toolbox Visualization');
robot.plot(q, 'workspace', [-100 200 -200 100 -50 50]);
title('Peter Corke Toolbox Visualization');

fprintf('\n=== Forward Kinematics Complete ===\n');
```

5.1.4 Matlab Code

```
  clc;
clear;
close all;

% ---------------------------
% Base dimensions (mm)
base_length = 85;
base_width  = 110;
base_height = 6; % thickness

% ---------------------------
% Updated Thigh + Shank Parameters
thigh_length = 71.83;        % mm
thigh_angle_deg = 30;        % NEW THIGH ANGLE
thigh_angle = deg2rad(thigh_angle_deg);

shank_length = 64.96;        % mm
shank_angle_deg = 63;        % NEW SHANK ANGLE
shank_angle = deg2rad(shank_angle_deg);

% ---------------------------
figure;
hold on;
grid on;
axis equal;
xlabel('X [mm]');
ylabel('Y [mm]');
zlabel('Z [mm]');
```

```
title('Robotic Dog Model');

% Sphere for joints
[sx, sy, sz] = sphere(20);
r = 5; % joint sphere radius



%
==========================================================
==================
% BASE 1
%
==========================================================
==================
[X1, Y1] = meshgrid([0 base_length], [0 base_width]);
Z1 = zeros(2);

surf(X1, Y1, Z1, 'FaceColor', [0.8 0.8 0.8]);
surf(X1, Y1, Z1+base_height, 'FaceColor', [0.9 0.9 0.9]);

plot3([0 base_length base_length 0 0], [0 0 base_width
base_width 0], ...
      [0 0 0 0 0], 'k', 'LineWidth', 2);

% Base 1 joints (FRONT)
joint1_base1 = [0, 0, base_height];
joint2_base1 = [85, 0, base_height];
```

```
%
==========================================================
==================
% BASE 2
%
==========================================================
==================
base2_offset = [0, base_width + 20, 0];

[X2, Y2] = meshgrid([0 base_length], [0 base_width]);
Z2 = zeros(2);

surf(X2 + base2_offset(1), Y2 + base2_offset(2), Z2, 'FaceColor',
[0.7 0.7 0.9]);
surf(X2 + base2_offset(1), Y2 + base2_offset(2),
Z2+base_height, 'FaceColor', [0.6 0.6 0.8]);

plot3([0 base_length base_length 0 0] + base2_offset(1), ...
      [0 0 base_width base_width 0] + base2_offset(2), ...
      [0 0 0 0 0], 'k', 'LineWidth', 2);

% Base 2 joints (BACK)
joint1_base2 = [0, base_width, base_height] + base2_offset;
joint2_base2 = [85, base_width, base_height] + base2_offset;



%
==========================================================
==================
```

```
% Middle Revolute Joint (between Base1 & Base2)
%
================================================
==================
joint_between = [base_length/2, base_width + 10, base_height];

all_base_joints = [joint1_base1; joint2_base1; ...
                   joint1_base2; joint2_base2; ...
                   joint_between];

for i = 1:size(all_base_joints,1)
    surf(sx*r + all_base_joints(i,1), ...
         sy*r + all_base_joints(i,2), ...
         sz*r + all_base_joints(i,3), 'FaceColor', 'r');
end



%
================================================
==================
% THIGHS (updated angles)
%
================================================
==================
thigh_vec_front = [0,  thigh_length*cos(thigh_angle), -
thigh_length*sin(thigh_angle)];
thigh_vec_back  = [0, -thigh_length*cos(thigh_angle), -
thigh_length*sin(thigh_angle)];
```

```
knee1_base1 = joint1_base1 + thigh_vec_front;
knee2_base1 = joint2_base1 + thigh_vec_front;
knee1_base2 = joint1_base2 + thigh_vec_back;
knee2_base2 = joint2_base2 + thigh_vec_back;

knee_joints = [knee1_base1; knee2_base1; knee1_base2;
knee2_base2];

% Draw thighs
plot3([joint1_base1(1) knee1_base1(1)], [joint1_base1(2)
knee1_base1(2)], [joint1_base1(3) knee1_base1(3)], 'b',
'LineWidth', 4);
plot3([joint2_base1(1) knee2_base1(1)], [joint2_base1(2)
knee2_base1(2)], [joint2_base1(3) knee2_base1(3)], 'b',
'LineWidth', 4);
plot3([joint1_base2(1) knee1_base2(1)], [joint1_base2(2)
knee1_base2(2)], [joint1_base2(3) knee1_base2(3)], 'b',
'LineWidth', 4);
plot3([joint2_base2(1) knee2_base2(1)], [joint2_base2(2)
knee2_base2(2)], [joint2_base2(3) knee2_base2(3)], 'b',
'LineWidth', 4);

% Draw knee joints
for i = 1:size(knee_joints,1)
    surf(sx*r + knee_joints(i,1), ...
        sy*r + knee_joints(i,2), ...
        sz*r + knee_joints(i,3), 'FaceColor', 'r');
end
```

```
%
==========================================================
==================
% SHANKS (updated 63°)
%
==========================================================
==================
shank_vec_front = [0,  shank_length*cos(thigh_angle +
shank_angle), ...
                   -shank_length*sin(thigh_angle + shank_angle)];

shank_vec_back = [0, -shank_length*cos(thigh_angle +
shank_angle), ...
                  -shank_length*sin(thigh_angle + shank_angle)];

% Draw shanks (green)
plot3([knee1_base1(1) knee1_base1(1)+shank_vec_front(1)], ...
     [knee1_base1(2) knee1_base1(2)+shank_vec_front(2)], ...
     [knee1_base1(3) knee1_base1(3)+shank_vec_front(3)], 'g',
'LineWidth', 4);

plot3([knee2_base1(1) knee2_base1(1)+shank_vec_front(1)], ...
     [knee2_base1(2) knee2_base1(2)+shank_vec_front(2)], ...
     [knee2_base1(3) knee2_base1(3)+shank_vec_front(3)], 'g',
'LineWidth', 4);

plot3([knee1_base2(1) knee1_base2(1)+shank_vec_back(1)], ...
     [knee1_base2(2) knee1_base2(2)+shank_vec_back(2)], ...
```

```
    [knee1_base2(3) knee1_base2(3)+shank_vec_back(3)], 'g',
'LineWidth', 4);

plot3([knee2_base2(1) knee2_base2(1)+shank_vec_back(1)], ...
    [knee2_base2(2) knee2_base2(2)+shank_vec_back(2)], ...
    [knee2_base2(3) knee2_base2(3)+shank_vec_back(3)], 'g',
'LineWidth', 4);



%
==========================================================
==================
% UPDATED NECK REVOLUTE JOINT (absolute position)
%
==========================================================
==================
neck_joint = [29.3, 227.5, 32];  % NEW EXACT LOCATION

surf(sx*r + neck_joint(1), ...
    sy*r + neck_joint(2), ...
    sz*r + neck_joint(3), ...
    'FaceColor', 'r');

plot3([neck_joint(1) neck_joint(1)], ...
    [neck_joint(2) neck_joint(2)], ...
    [0 neck_joint(3)], 'k--', 'LineWidth', 2);
```

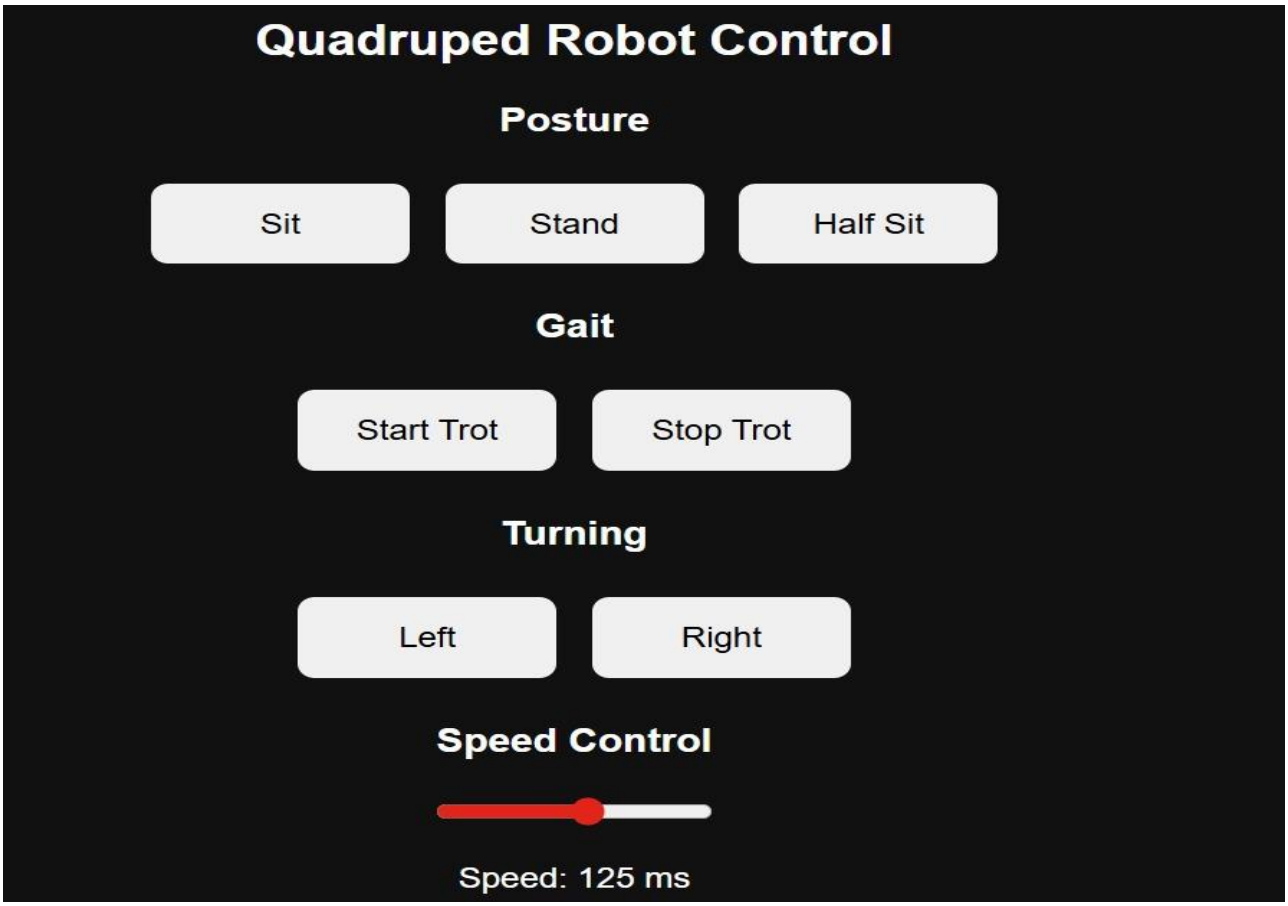
```
%
==========================================================
==================
% ADD 38 mm LINK ALONG +X AXIS FROM NECK JOINT
%
==========================================================
==================

link_length = 38;         % length of neck link
link_vec = [link_length, 0, 0];   % ALONG +X AXIS

link_end = neck_joint + link_vec; % end-point of link

% Draw link
plot3([neck_joint(1) link_end(1)], ...
      [neck_joint(2) link_end(2)], ...
      [neck_joint(3) link_end(3)], ...
      'm', 'LineWidth', 5);      % magenta link

%
==========================================================
==================
% NEW REVOLUTE JOINT AT SPECIFIED POSITION
%
==========================================================
==================
new_joint = [54.5, neck_joint(2), neck_joint(3)+19]; % [x, y, z]

% Draw the new revolute joint
```

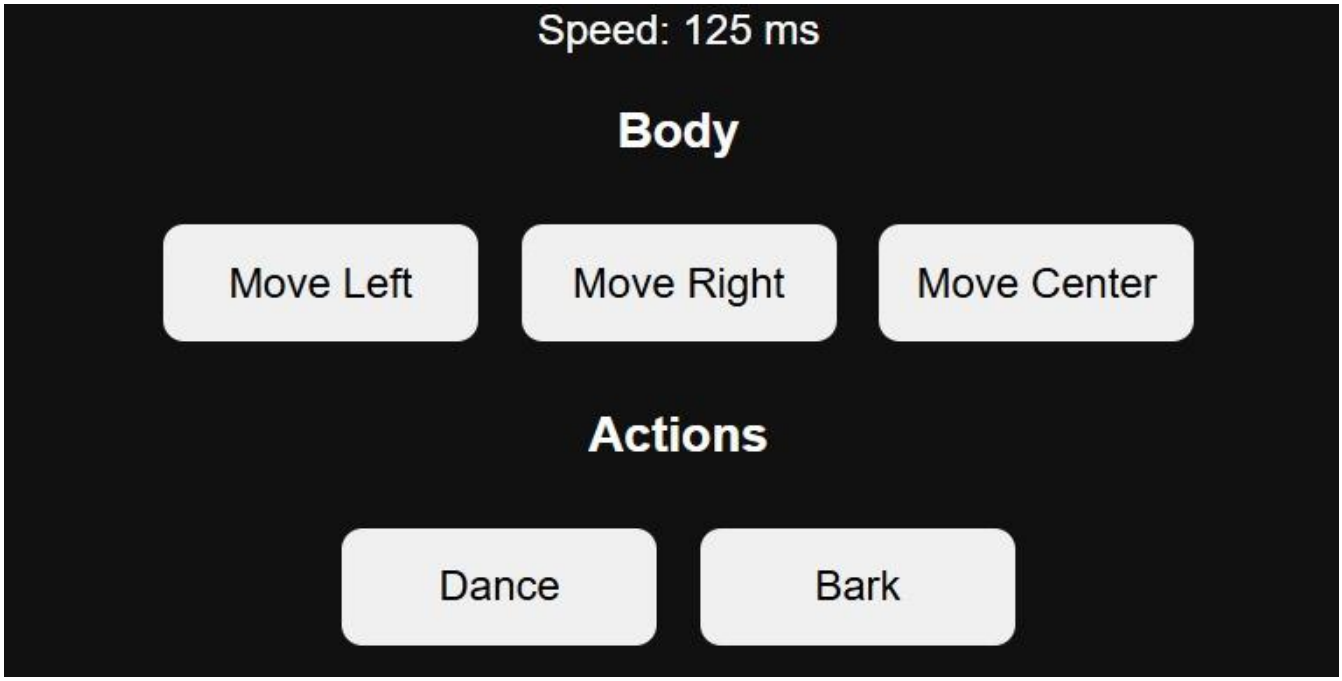
```
surf(sx*r + new_joint(1), ...
    sy*r + new_joint(2), ...
    sz*r + new_joint(3), 'FaceColor', 'r');

% Draw the link directly connecting neck joint to new joint
plot3([neck_joint(1) new_joint(1)], ...
     [neck_joint(2) new_joint(2)], ...
     [neck_joint(3) new_joint(3)], '', 'LineWidth', 5); % magenta
link



view(3);
```

Chapter 6- Simulations and final Integrations

6.1.1 Web View

The **Kynex Robotic Dog** incorporates a **web-based control interface** to enable easy and intuitive user interaction. The ESP32 microcontroller hosts a lightweight web server that allows users to connect to the robot through a web browser using a smartphone, tablet, or computer. Once connected, the web interface displays control buttons for executing predefined actions such as **sit, stand, walk, half-sit, and bark**. When a command is selected, the corresponding instruction is sent wirelessly to the ESP32, which processes the request and activates the appropriate servo motor sequences. This web-based control eliminates the need for additional hardware controllers or applications, providing a simple, platform-independent, and user-friendly method for operating the robotic dog.




Speed: 125 ms
Body
Move Left
Move Right
Move Center
Actions
Dance
Bark

6.1.2 Trot Gait Cycle Explanation

The trot gait cycle implemented in this quadruped robotic dog is a coordinated walking pattern in which diagonally opposite legs move together. This gait is commonly used in four-legged animals because it provides a good balance between stability, speed, and smooth motion, making it suitable for robotic locomotion.

Diagonal Leg Pairing

In the implemented system, the legs are divided into two diagonal pairs:

- **Pair 1: Front Left (FL) knee and Back Right (BR) knee**
- **Pair 2: Front Right (FR) knee and Back Left (BL) knee**

  Only one diagonal pair is lifted at a time, while the other pair remains on the ground to support the robot's body weight. This ensures continuous stability during movement.

Motion Sequence

Each trot cycle consists of the following steps:

1. **Hip Relief Phase**

   Before lifting the knees, a small offset (hip relief) is applied to the opposite hips. This slightly shifts the robot's center of mass and reduces load on the lifting legs, improving balance and reducing servo stress.

2. **Knee Lift Phase**

   The selected diagonal knee pair is lifted gradually by increasing the knee angles in small increments. This creates a smooth lifting motion instead of a sudden step.

3. **Knee Lowering Phase**

   After reaching the maximum lift height, the knees are gradually returned to their original standing angles, completing one step of the gait.

4. **Diagonal Switch**

   The system then switches to the opposite diagonal pair and repeats the same sequence, resulting in continuous forward motion.

Turning During Trot

Directional turning is achieved by asymmetrically reducing the knee lift on one side:

- For a left turn, the front right leg lift is reduced.
- For a right turn, the back left leg lift is reduced.

This causes uneven stride lengths, naturally steering the robot without disrupting gait stability.

Speed Control

The walking speed of the trot gait is controlled by adjusting the step delay between servo movements. A lower delay results in faster motion, while a higher delay produces slower and more controlled movement. This speed parameter can be modified in real time via the web interface.

Advantages of the Trot Gait

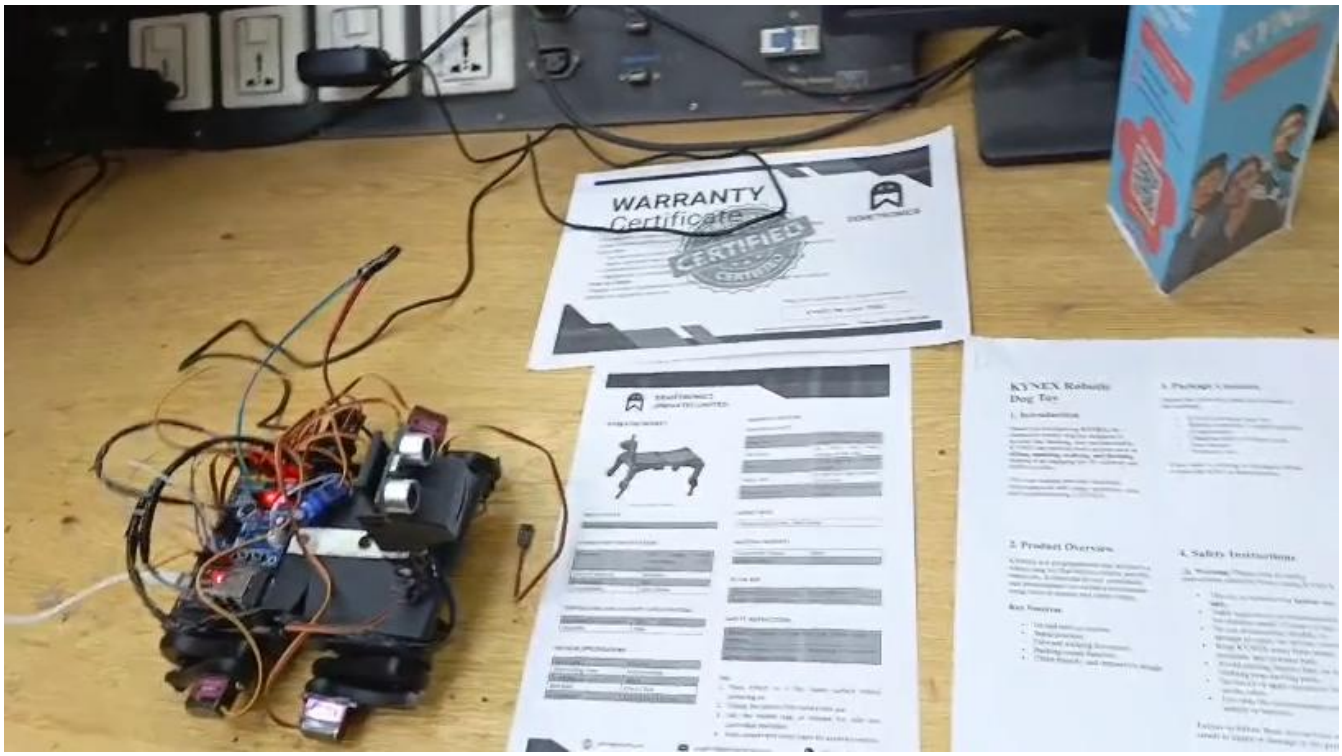
- Maintains continuous stability with at least two legs on the ground
- Produces smooth and natural motion
- Reduces mechanical stress on servos
- Allows easy implementation of turning and speed control

Chapter 7- System Test phase

7.1.1 Final testing

The final testing phase of the Kynex Robotic Dog was carried out to ensure that all system components operated correctly and met the project objectives. Individual testing of hardware modules, including servo motors, sensors, power supply, and the ESP32 controller, was performed before full system integration. After integration, all predefined action sit, stand, walk, half-sit, and barkwere tested repeatedly to verify consistency and reliability.

The robot was tested on flat indoor surfaces to evaluate balance, gait stability, and smoothness of motion. The web-based control interface was also tested to confirm accurate and responsive command execution. Power stability and component temperature were monitored throughout testing to ensure safe operation. The results of final testing confirmed that Kynex performed all intended functions successfully and was ready for demonstration and evaluation.



7.1.2 Things that are questionable and get burned again and again

During the development of the **Kynex Robotic Dog**, several components and design aspects were identified as questionable due to repeated failures and overheating issues. One of the major concerns was **servo motor overloading**, which occurred when excessive torque was required during improper gait execution or incorrect joint alignment. This led to overheating and, in some cases, permanent damage to the servo motors.

Another recurring issue was related to **power management**, where unstable voltage levels caused components such as the boost converter and control modules to heat up or malfunction. Improper current distribution and simultaneous actuation of multiple servos placed high demand on the power supply, increasing the risk of component burnout. Additionally, **wiring and connection faults**, including loose connections and inadequate insulation, contributed to short circuits and repeated failures during testing.

These issues highlighted the importance of proper load analysis, current rating verification, thermal management, and systematic testing. Through repeated troubleshooting, the team learned to implement better power regulation, safer wiring practices, and gradual testing of components to reduce the risk of damage and improve system reliability.

Chapter 8- Project management

8.1.1 Individual Execution of Roles

The successful completion of the **Kynex Robotic Dog project** was achieved through the dedicated execution of roles by each team member. Responsibilities were clearly defined at the start, allowing efficient workflow and accountability throughout the project.

- **Zain Ali – Project Lead**
  Zain led the project by coordinating all team activities, managing timelines, and supervising the integration of hardware and software components. He executed the coding, assembly, and system testing, ensuring the robot's motion sequences were accurate and reliable.

- **Jawad Ahmed – Design and Troubleshooting**
  Jawad focused on designing the CAD model of Kynex and assembling mechanical components. He actively performed troubleshooting to resolve servo synchronization issues and hardware alignment problems. His contributions ensured the structural stability and functionality of the robotic dog.

- **Abdul Wasay – Project Management, Documentation, and Marketing**
  Abdul executed project management tasks by maintaining schedules, organizing team meetings, and coordinating component procurement. He also handled report writing and documentation, as well as preparing design marketing materials to showcase the project effectively.

Overall Execution

The clearly defined roles enabled the team to work efficiently with minimal overlap and conflicts. Collaborative efforts in troubleshooting, testing, and component sourcing strengthened team performance, ensuring that the

project milestones were achieved successfully. Each member’s execution of their respective responsibilities contributed significantly to the overall success of the Kynex Robotic Dog project.

8.1.2 Project Success and Failure – Individual Reflections

The **Kynex Robotic Dog project** provided valuable insights into both the successes and challenges encountered during the development process. Each team member reflected on their personal learning and contributions, identifying areas of achievement and aspects that could be improved in future projects.

- **Zain Ali – Project Lead**
  Zain reflects that the project's success was largely due to effective coordination and timely execution of coding and assembly tasks. Challenges included managing the integration of multiple servos and debugging complex motion sequences. Through this, Zain learned the importance of systematic testing, patience, and iterative problem-solving.

- **Jawad Ahmed – Design and Troubleshooting**
  Jawad considers the CAD modeling and structural design of Kynex a major success, as it provided a stable platform for motion. However, initial troubleshooting of servo synchronization revealed the need for better pre-planning of actuator sequencing. This experience emphasized to Jawad the value of thorough simulation before hardware implementation.

- **Abdul Wasay – Project Management, Documentation, and Marketing**
  Abdul views the project's documentation and planning as highly successful, ensuring clarity in workflow and report preparation. Challenges included aligning technical progress with planned timelines and coordinating component sourcing. This experience highlighted to Abdul the importance of time management, clear communication, and flexibility in handling unexpected delays.

8.1.3 Team Member Evaluations

The team functioned cohesively, with responsibilities clearly divided and collaboration effectively managed. All members demonstrated technical competence, problem-solving skills, and proactive communication, which contributed to the timely and successful completion of the Kynex project. The project reflects a balanced combination of leadership, technical skill, and teamwork.

8.1.4 Final Bill of Materials and Budget Allocation

- **Mechanical & Structural Components:** 6,000 Rs (3D printing & PLA)
- **Actuation & Control Electronics:** 8,900 Rs (MG90 servos, PCA9685, ESP32, sensors, voice module)
- **Power System & Accessories:** 950 Rs (battery, boost converter, charging module)
- **Miscellaneous & Assembly:** 2,500 Rs (fasteners, wires, connectors, and other small items)

The careful allocation of the budget ensured that **Kynex** was developed in a **cost-effective and efficient manner**, while meeting the technical requirements for motion, control, and interactivity.

8.1.5 Risk Management Lessons Learned

During the development of the **Kynex Robotic Dog**, we gained valuable experience in **risk management**, which proved essential for the successful completion of the project. We identified potential risks such as hardware failure, incorrect wiring, software bugs, servo overload, battery issues, and delays in component sourcing. By analyzing these risks early, we were able to implement preventive measures and contingency plans to minimize their impact.

We learned the importance of **systematic testing** at each stage, including hardware assembly, software integration, and motion sequence verification, to detect and resolve issues before they escalated. Proper documentation of tasks, responsibilities, and timelines helped mitigate delays and resource mismanagement. Additionally, the division of responsibilities among team members ensured that risks were monitored and addressed collaboratively.

Overall, this project reinforced the value of **anticipating problems, planning solutions, and maintaining flexibility** in engineering design. The skills gained in risk assessment and mitigation are directly applicable to future projects and professional work, fostering a proactive and responsible approach to technical challenges.

Chapter 9 - Feedback for project and course

The Kynex Robotic Dog project provided an excellent opportunity to apply theoretical concepts of robotics, electronics, and embedded systems into a practical, hands-on experience. Through this project, we gained valuable skills in mechanical design, circuit integration, programming, and project management, which strengthened our understanding of how multidisciplinary components work together to create a functional robotic system.

Working on this project also enhanced our problem-solving abilities and teamwork skills, as each team member contributed to design, coding, assembly, and troubleshooting tasks. The project highlighted the importance of careful planning, component selection, and iterative testing to achieve desired outcomes.

The course itself offered a solid foundation in robotics principles, control systems, and programming, which directly supported the successful completion of this project. The structured curriculum, combined with practical lab sessions, enabled us to connect theory with real-world application, fostering both technical proficiency and innovation.

Overall, the project and course together provided a comprehensive learning experience, encouraging creativity, technical exploration, and professional growth. They have prepared us to approach future engineering challenges with confidence and a systematic methodology.

Chapter 10 - Detailed YouTube video

A comprehensive YouTube video has been created to showcase the project. The video covers:

- Project overview and objectives
- Detailed explanation of hardware components and sensors
- Step-by-step demonstration of the assembly process
- Software walkthrough highlighting code functionality for line following, obstacle avoidance, and cloud data uploading
- Real-time demonstration of the robot in action, showcasing its performance on different tracks and obstacle scenarios
- Discussion of challenges encountered and solutions implemented
- Final remarks and future improvement ideas

The video serves as an engaging supplement to the written report, allowing viewers to visually grasp the project workflow and technical depth.

Here are the links:

Zain Ali

https://youtube.com/@zainali1835?si=igBnx9XNib0nLXWY

Jawad Ahmed

https://youtube.com/playlist?list=PL2lLntThJ06b6nZY2AastYcY4EtK5Xqka&feature=shared

Abdul Wasay

https://youtube.com/playlist?list=PLPqOMWdbuucVcuF1ox1ofC2cUcYNqt7I1&si=xyXncvapeZ7eEF9A

References

[1] M. H. Raibert, *Legged Robots That Balance*, MIT Press, 1986.

[2] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, Springer, 2016.

[3] C. Semini, N. G. Tsagarakis, M. Focchi, and D. G. Caldwell, "Design of HyQ – A Hydraulically and Electrically Actuated Quadruped Robot," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 225, no. 1, pp. 10–25, 2011.

[4] J. Kimura, Y. Fukuoka, and H. Cohen, "Adaptive Dynamic Walking of a Quadruped Robot," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 1205–1210.

[5] Sony Corporation, "AIBO: The World's First Entertainment Robot," Technical White Paper, 1999.