

QUADRUPEL LAND MINE DETECTION ASSISTANT AND SURVEILLANCE ROBOT

Submitted in partial fulfillment of the requirements for the
award of

Bachelor of Engineering Degree in
Mechanical Engineering

By

KATNAM YASASWI UTHAM VENKATA ATREYA (38150048)

PULAPA BALA MURALI KRISHNA (38150711)



**DEPARTMENT OF MECHANICAL ENGINEERING
SCHOOL OF MECHANICAL ENGINEERING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MAY - 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF MECHANICAL ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work OF KATNAM YASASWI UTHAM VENKATA ATREYA (38150048) AND PULAPA BALA MURALI KRISHNA (38150711) who carried out the project entitled QUADRUPED LANDMINE DETECTION ASSISTANCE AND SURVEILLANCE ROBOT under our supervision from 13-12-2021 to 08-03-2022.

Dr.R.B.DURAIRAJ

Guide

Dr.G.ARUN KUMAR

Head of the Department

Submitted for Viva voce Examination held on 09-05-2022.

Internal Examiner

External Examiner

DECLARATION

We, KATNAM YASASWI UTHAM VENKATA ATREYA (38150048),and PULAPA BALA MURALI KRISHNA (38150711) hereby declare that the Project entitled QUADRUPED LAND MINE DETECTION ASSISTANT AND SURVEILLANCE ROBOT done by us under the guidance of Dr.R.B.DURAIRAJ is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Mechanical Engineering.

1.

DATE:

2.

PLACE:

SIGNATURE OF THE CANDIDATES

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to the Board of Management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our sincere thanks to **Dr.S.PRAKASH**, M.E., Ph.D., Dean, School of Mechanical Engineering, and **Dr.G.ARUN KUMAR**, M.E., Ph.D., Head of the Department, Department of Mechanical Engineering for providing us with necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude towards our project Guide **Dr.R.B.DURAIRAJ**, M.E., Ph.D., Assistant Professor, Department of Mechanical Engineering for his valuable guidance, suggestions, and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the Department of **Mechanical Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

In these days we need a mobile robot that can be used for surveillance purposes. But Wheeled robots cannot satisfy all the needs as they cannot move well on rough terrains. The Quadruped Surveillance Robot allows us to control it over rough and uneven terrains using its 12 DOF. Each DOF is achieved using a servo motor as an actuator. This robot has a special walking gait which is performed using an algorithm running in a microcontroller unit. It can be controlled using Wi-fi/Bluetooth and can send data in the form of video/images to us live. It uses Inverse kinematics for positional locomotion. It can avoid any obstacle that comes closer to it and changes its motion accordingly with the help of an ultrasonic sensor. It can be used in surveillance of places that are difficult to reach for humans like underground pipelines, mining areas, collapsed buildings using Cam module. It can be used for metal detection. We can use this for various surveillance purposes by making needed modifications. We can add many more features to enhance its abilities. We can include GPS tracking, and send the location of the enemy troops in Warfield.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	i
	LIST OF ABBREVIATIONS	iii
	LIST OF FIGURES	iv
	LIST OF TABLES	v
1	INTRODUCTION	1
	1.1 INVERSE KINEMATICS	3
	1.2 SOFTWARE	4
2	LITERATURE SURVEY	8
3	AIM AND SCOPE OF THE PRESENT INVESTIGATION	13
	3.1 AIM	13
	3.2 SCOPE	13
4	EXPERIMENTAL MATERIALS AND METHODS	14
	4.1 MATERIALS AND METHODS	14
	4.2 COST	18
5	RESULTS AND DISCUSSION	20
	5.1 PROTOTYPE	20
	5.2 TESTING LAYOUT	21
6	SUMMARY AND CONCLUSION	22
7	FURTHER DEVELOPMENTS	23
8	REFERENCES	24
9	ANNEXURE-1	26

LIST OF ABBREVIATIONS

DOF	-	Degree of Freedom
MCU	-	Micro Controller Unit
DC	-	Direct Current
PCB	-	Printed Circuit Board
CAD	-	Computer-Aided Design
IK	-	Inverse Kinematics
PWM	-	Pulse with Modulation
MP	-	Mega Pixel
IO	-	Input-Output
GPS	-	Global Positioning System
CPG	-	Central Pattern Generator

LIST OF FIGURES

Fig No.	Description	Page No.
1.1	Arduino Nano with Nano 328P Expansion Adapter Breakout Board	1
1.2	Schematic Diagram of Arduino Nano connected to Expansion Adapter Breakout board Circuit	2
1.3	Quadruped Battle Assistance Surveillance Robot(CAD Design)	2
1.4	Leg with parameters	3
1.5	Applying Inverse kinematics	3
1.6	Control flow	4
1.7	Numbering of the Legs	4
1.8	Numbering of Servo Motors for each leg	5
1.9	Android Application Main Page	5
1.10	ESP32 CAM Module with OV2640 2MP CAM	6
1.11	ESP32 CAM Module with OV2640 2MP CAM	7
4.1	Connections for MCU operations	14
4.2	Power Connections in the circuit	15
4.3	Arduino Nano	16
4.4	Body Parts of the Robot	17

5.1	Robot Prototype	20
5.2	Robot Prototype	20
5.3	Testing Layout	21

LIST OF TABLES

Table No.	Description	Page No.
1.1	Details of Literature Review	8
2.1	Bill of Materials	18

CHAPTER 1

INTRODUCTION

Nowadays robots are being used extensively for different applications. Our idea was to design a robot that can be used by individuals to reach their desired work and control them in a handy way. The design of this type of robot was inspired by a biological spider, which has the ability to navigate through different terrains and uneven surfaces. Spider robot is ideal for this type of application. We have come up with a capable mechanical design for the robot so that it can freely move around and do the movements needed. The environmental effects were minimized by the reusable electronic parts such as servos and microcontrollers. The software part of the robot is offering logic. We designed the robot in such a way that it has a good speed, smooth and controlled movement. To steer and control the robot we have designed an android application that everyone can use. As it should work for a reliable time period we have installed a battery which lasts longer. The robot can be easily manufactured in a short time for further applications.

We used a Nano 328P Expansion Adapter Breakout Board Shield which allows us to control all the 12 Servos, and power the rest of the components. This will reduce the

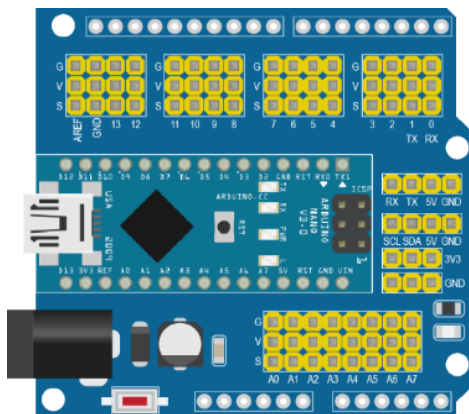


Fig 1.1 : Arduino Nano with Nano 328P Expansion Adapter Breakout Board

complexity of the circuit. Figure 1.1 below shows the Breakout Board, figure 1.2 shows its schematic diagram when connected to Arduino Nano.

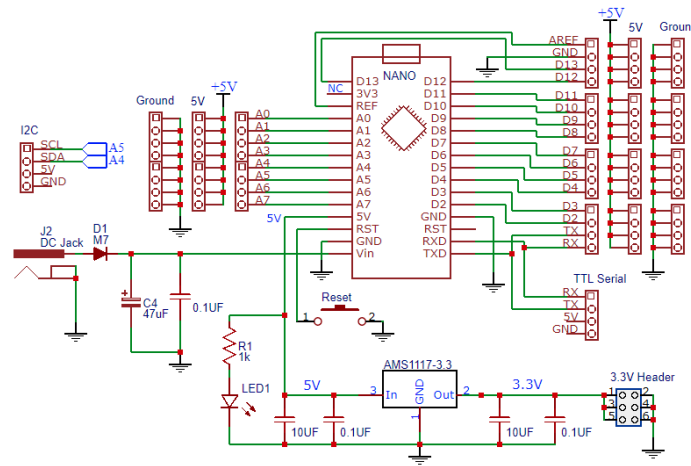


Fig 1.2: Schematic Diagram of Arduino Nano connected to Expansion Adapter Breakout board Circuit

The Robot body was designed in Fusion 360 Software. Each leg of the robot has three DOF (x, y, z) which allows it to move in all directions. Figure 6 shows the parts of the robot (legs, leg connectors, body connectors, and other mounts) which were 3d printed.

The Final CAD design of the Robot is shown in figure 1.3 down below.

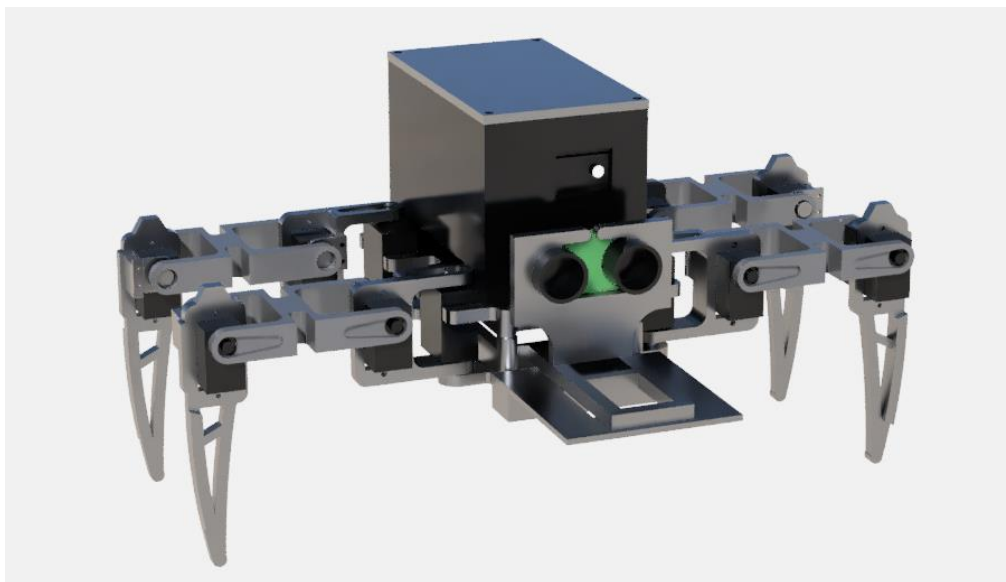


Fig 1.3: Quadruped Battle Assistance Surveillance Robot(CAD Design)

1.1.INVERSE KINEMATICS:

We just consider inverse kinematics for a single leg. The inverse kinematics for other legs can be done by just applying rotation of this leg.



Fig 1.4: Leg with parameters

$$\theta = \tan^{-1}(y/x) , P=X/\cos\theta$$

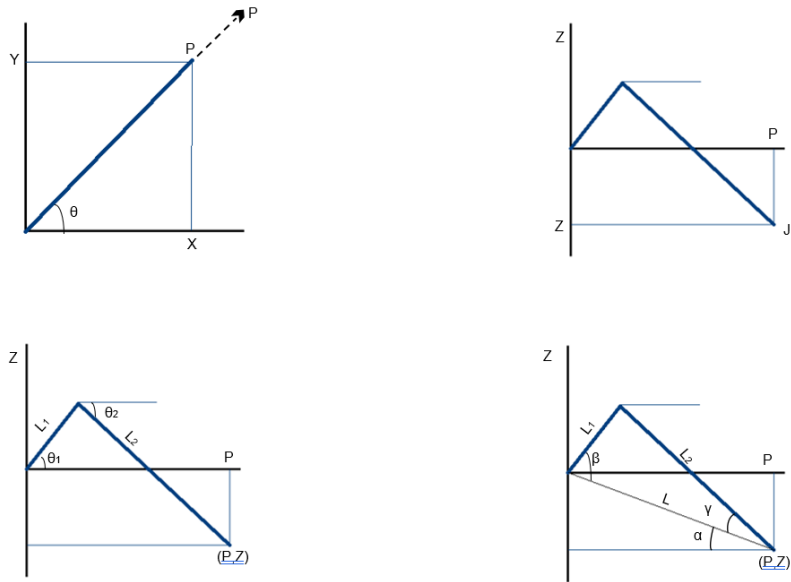


Fig 1.5: Applying Inverse kinematics

$$\theta_1 = \beta - \alpha$$

$$L_2^2 = L_1^2 + L^2 - 2LL_1\cos\beta$$

$$L = \sqrt{P^2 + Z^2}$$

$$\theta_2 = \alpha + \gamma$$

$$\alpha = \sin^{-1}\left(\frac{Z}{L}\right)$$

$$\beta = \cos^{-1}[L_2^2 - L_1^2 - L^2]/[-2L_1L] , \gamma = \cos^{-1}[L_1^2 - L_2^2 - L^2]/[-2L_2L]$$

1.2.SOFTWARE:

The software part of the robot consists of two major sections

- I. Microcontroller logic
- II. Android application to control

In Microcontroller logic, servo motor position and direction are controlled using PWM (Pulse with Modulation) received from the Microcontroller. Firstly, we ran a code to initialize the servos i.e., to identify the 0 position of the servos to clamp the legs as we desired.

Then we upload the code to control the robot using Bluetooth, after that the robot is ready for the commands from the user to get steered and controlled. Figure 1.4 shows the simple schematic of the software process.

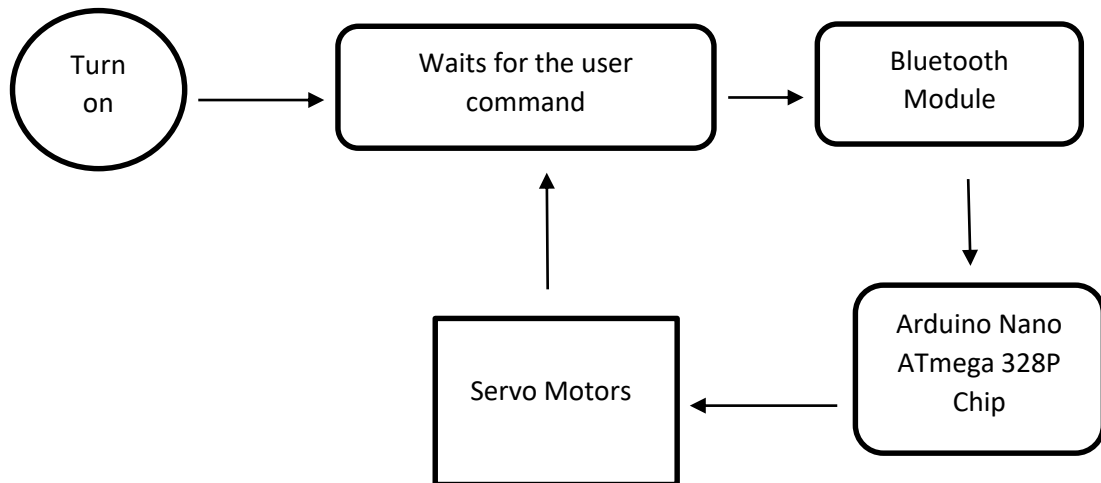


Fig 1.6 : Control flow

When we turn on the switch, the system will start and wait for the command from the user. The Bluetooth module process the command to the microcontroller if any command is received. Then the Arduino Nano (ATmega328P chip) processes and controls the servo motors accordingly. This will run in a loop till we turn off the circuit.

If the robot is in object avoidance mode, it continues to move forward till it reaches an object at 200mm far. Once it encounters an object it will go back 3 steps and turn right/left randomly and continues to go forward in that direction. This loop will continue as long as the robot is in object avoidance mode.

Forward Motion: The robot legs are numbered as shown in figure 1.5 and motors in each leg are numbered as shown in figure 1.6.

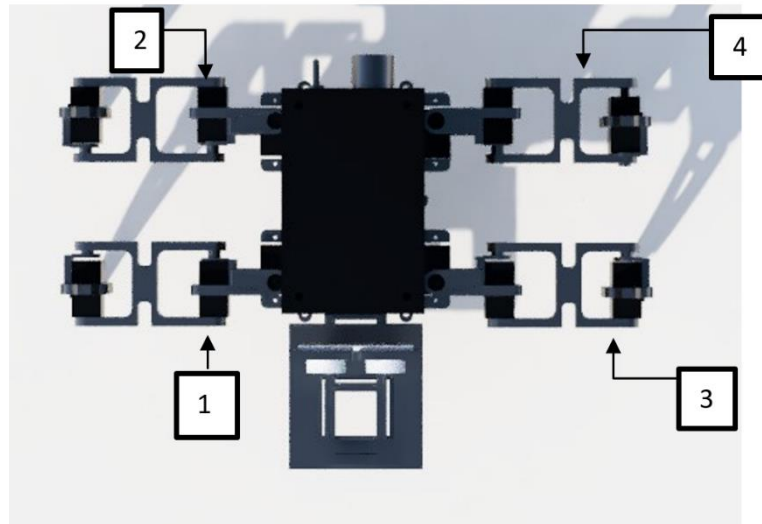


Fig 1.7 : Numbering of the Legs

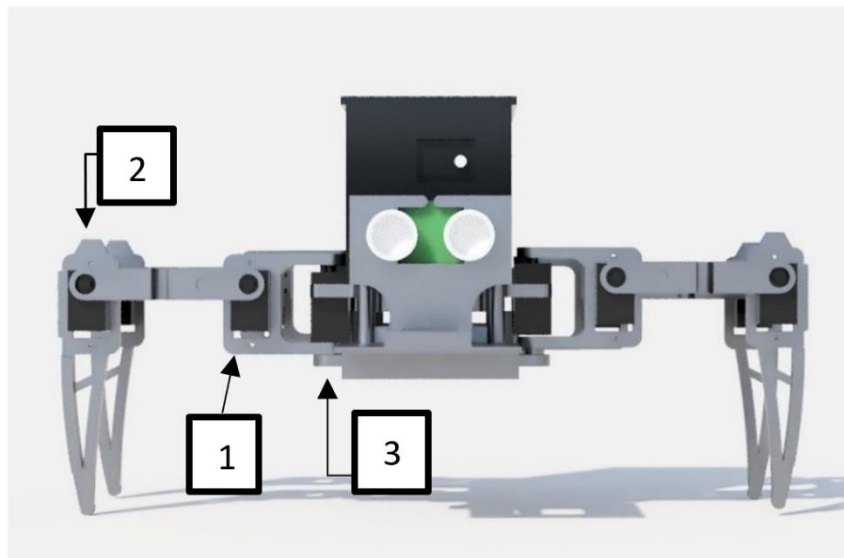


Fig 1.8 : Numbering of Servo Motors for each leg

To understand the gait let's take a look at 6 cycles:

Cycle 0(default starting position): leg 1 will move up, move forward, and move down

Cycle 1: leg 4 will move up and move forward and move down

Cycle 2: leg 3 will move up and move forward and move down

Cycle 3: leg 1 and 4 will move back which makes the robot move forward

Cycle 4: leg 2 will move up, move forward, and move down

Cycle 5: leg 1 will move up and move forward and move down

Cycle 6: leg 2 and 3 will move back which makes the robot move forward

These Cycles repeat in a loop and form a walking gait.

Serial commands from Arduino Nano were sent in specified orders to control servo channels and get the target position of servos. Nano 328P Expansion Adapter Breakout Board is used to extend the Arduino Nano Channels.

smartphone application is used to control and steer the Robot. We used MIT App Inventor to make the professional App. Figure 1.7 below shows the main page of the Application from which the user can control the robot.

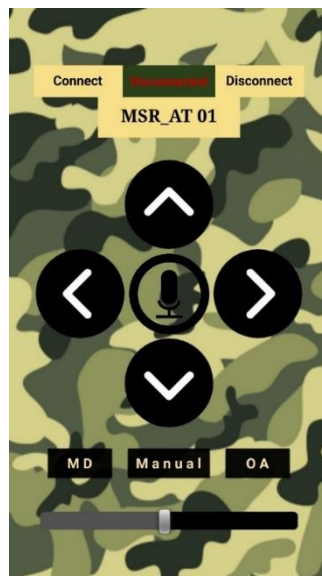


Fig 1.9: Android Application Main Page

There will be four arrow buttons namely, “Forward”, “Backward”, “Left”, “Right” which allow the user to control the robot. The robot moves in the direction of where we want it to move by pressing the arrows. For Communication between the robot and the app, the HC-05 Bluetooth module was used to get values from the app which are sent to the Arduino nano via serial communication. It is shown in figure 1.8 below.

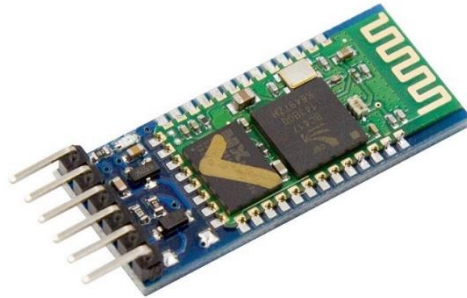


Fig 1.10 : HC-05 Bluetooth Module

We used “FlexiTimer2.h” library to set a timer to manage all servos and “Servo.h” library to define and control servos from Arduino directories. We also used “NewPing.h” library to get the distance readings from the Ultrasonic Distance sensor.

ESP32 Cam Module allows us to get live video/image footage from the robot. We can use ESP32 Boards IP address to access the data shared by the board. It is very crucial for surveillance. You can check the ESP32 Module in Figure 1.9 below.



Fig 1.11 : ESP32 CAM Module with OV2640 2MP CAM

CHAPTER 2

LITERATURE SURVEY

Table 2.1: Details of Literature Review

YEAR	AUTHOR & JOURNAL NAME	INFERENCE
2015	Karakurt, Tolga & Durdu, Akif & Yilmaz, Nihat. (2015). Design of Six Legged Spider Robot and Evolving Walking Algorithms. International Journal of Machine Learning and Computing. 5. 96-100. 10.7763/IJMLC.2015.V5.490.	Basic Idea of the Robot
2019	Cavas, Mehmet & Baballe, Muhammad. (2019). A REVIEW ON SPIDER ROBOTIC SYSTEM. International Journal of New Computer Architectures and their Applications. 9. 19-24. 10.17781/P002604.	Some features like temperature sensing and ultrasonic sensor for proximity.
2016	A. Espinal, H. Rostro-Gonzalez, M. Carpio, E. I. Guerra-Hernandez, M. Ornelas-Rodriguez, H. J. Puga-Soberanes, M. A. Sotelo-Figueroa, P. Melin & <i>Computational Intelligence and Neuroscience</i> , vol. 2016	Body of the Robot is inspired from this paper.
2020	Ragini Sharma, Monish Adhikari, Sanket Deshmuk, Chetan Mhatre & Journal of Advancements in Robotics	Using Bluetooth and wifi communications to control the robot mobility
2011	Zareei Amir & Ghayour Mostafa (2018). Inverse Kinematic Analysis of a Hexapod Spider-like Mobile Robot. Advanced Materials Research	Mobile robot locomotion

(Tolga Karakurt, et al. 2015) has described the creation of a legged robot with features that are required for search and rescue activities. For this objective, a variety of walking algorithms have been developed and tested. The spider-inspired four-legged robot was created with a control mechanism and can perform a variety of walking behaviours. The robot's efficacy is determined by how well it performs on various terrains. A robot is an electromechanical system that can do tasks on its own or according to a set of instructions. Search and rescue robots, space robots, and discovery robots are all possibilities for the four-legged robot. Quadruped robots offer advantages in these domains due to their compact size and practical mobility. One of the fundamental steps to achieving the primary goal is to build walking algorithms for functional and flexible control without being locked in a given limit. Being able to move more readily is one of the basic steps to achieving the main goal. Walking algorithms are used to acquire the required velocity in this scenario.

(Mehmet Cavas, et al. 2019) explained that the National Instrument board assists in gathering data from the sensor installed on the spider robot frame and processing it, allowing him to complete the task. If, for example, this ultrasonic sensor detects an obstruction in the spider robotic movement path, it will automatically transmit the response information to the programmed national instrument board. With the help of this board, it will then process and assess the information in a way that produces an appropriate result, such as enrichment for the spider robot's location in order to avoid the hitch that leads to self-localization. The spider robot will then resume its journey until it reaches its destination or encounters another obstacle along the way.

(Espinal, et al. 2016) Different strategies have been implemented to inherit the locomotion challenges in autonomous robots. Some of them are implemented using wheels, however this method has a number of drawbacks in terms of environmental surfaces and terrains. As a result, several academics have focused on the ease of building a legged robot, which is a form of locomotion system based on legs. This method of creating a legged robot is based on bio-inspired ideas from nature, as this method gives a viable answer for a robot's effective walking.

According to the biologist, vertebrate movement is governed by a central pattern generator (CPG) capable of rhythmic patterns or gaits. Complex control signals generated by CPGs are used to coordinate muscles during rhythmic periods like as walking, running, swimming, and flying. CPGs are commonly utilised to operate a wide range of robots and locomotion modes. Insect locomotion inspired hexapod and octopod CPG model robots. Vertebrates are the inspiration for quadruped robotics. Reptiles have also inspired other types of robots.

(Ragini Sharma, et al. 2020) In this period, robots are becoming more trustworthy and are being downsized, making this technology more ideal for spying and law enforcement purposes. The demand for this Flexi-controlled robot is fast growing in response to the current scenario and concern for human safety. These robots could be employed as spy robots to gather critical information. The goal of this study is to create a spy robot with appropriate sensors and a camera that can do various activities while

being controlled remotely. It can use wireless connectivity to send video and photographs that it has captured.

(Zareei Amir, et al. 2011) The mechanism for a spider-like quadruped mobile robot is described. Geometrical approaches are used to examine the robot's inverse kinematics. The differential kinematics also determines the velocity of the joint variables. The robot's inverse kinematics are then solved for a particular motion. Many scientists have been interested in the research on these walking machines over the last two decades since they have a lot of potential features. These robots can be divided into two groups: biped and multi-legged robots, which have significant mechanical distinctions.

"To display the joint variables of legs by using the position and orientation of the robot platform," meant by inverse kinematic for robots with parallel structure. The limbs of the quadruped spider-like robot are divided into two groups: contact legs that maintain contact with the ground as the robot moves, and noncontact limbs that move across space as the robot walks. The inverse kinematics of position and velocity of each of the two group legs are analysed independently to study the robot's inverse kinematics.

(Arun, et al. 2011) created a new type of rough terrain robots that mimic the mobility, autonomy, and speed of living things. Outdoor terrain that is too steep, rutted, rocky, rainy, muddy, and snowy for traditional vehicles will be traversed by these robots. They will roam around cities and our houses, conducting duties and providing care in places where wheeled vehicles are limited by steps, stairwells, and clutter. Terrain sensors, powerful processing and power systems, advanced actuators, and dynamic controls will all be included in robots that achieve these objectives. Legged robots have the ability to navigate extremely difficult terrain, and great development has been made in this field recently.

(Priyaranjan Biswal, et al. 2021) focuses on several quadrupedal robot design and development approaches, as well as environment sensing techniques. Spot is also one of the most advanced and intelligent quadrupedal robots on the market. The performance of each quadruped robot, as well as the future plan, are detailed.

(Fu Mengyu, et al. 2012) The creation of legged robots that need to go to some unusual walking sites would benefit from bionic research on spiders. As an example, this paper used an eight-legged spider biomimetic robot. Spiders maintain stability with one group of legs while searching for the best supporting place on the ground with another group of legs, according to a simulation. This paper also lists and evaluates simulation findings.

(Yam Geva, et al. 2015) shows how he came up with the idea for a new quadruped robot. The proposed design is characterised by its simplicity, modularity, and ease of interface. The robot is primarily made up of off-the-shelf parts. Four 3-DOF legs, the robot body, and its electronics are all part of the design. The proposed robot can navigate tough terrain while carrying extra payloads. Sensors and computing hardware can both be included in such payloads. The robot's design, control system, and forward and inverse kinematics are all presented.

(Santiago Noriega Álvarez, et al. 2018) Spiders have the ability to enter environments that other creatures and even humans cannot. These characteristics of spiders are used in this project to build and develop a quadruped spider robot that can travel in all directions and perform pre-programmed motions such as ascending, descending, avoiding obstacles, and detecting gases.

(Nikunj S Yagnik, et al. 2017) He talks about the history, early literature, and current developments in the field of quadruped robots. The difficulties of developing and operating quadruped robots are highlighted. These robots' gait mechanisms, path movement methods, static and dynamic stability margins, power supply, posture control, fault tolerant control, and reconfiguration are all covered. Finally, the report concludes with a discussion of quadruped robot development in the future.

(Akash Maity, et al. 2018) talks about how to design a quadruped robot, how to make it with a 3-D printer, and how to analyse it thoroughly. It emphasises the benefits of additive manufacturing over traditional production techniques while also pointing out its drawbacks. This paper examines the pros and downsides of various materials for 3-D printing.

CHAPTER 3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

3.1.AIM:

The main objective of our project is to build a robot that can help as a military spy and a bomb detection agent. After studying and researching different robots, a legged robot is the best suitable one, taking into consideration its natural ability to move around freely.

3.2.SCOPE:

This robot helps to inspect the war zone. It can be used to detect land mines. The robot can inspect collapsed buildings during earthquakes.

CHAPTER 4

EXPERIMENTAL OR MATERIALS AND METHODS USED; ALGORITHMS USED

4.1.MATERIALS AND METHODS:

The hardware of the spider robot consists of two major assemblies, the electrical parts assembly and the Spider body parts assembly.

Below is the list of all electrical parts that have been used.

The electronics part of the Spider robot consisted of these main parts:

- 1) Tower pro MG90s Servo Motors
- 2) Arduino Nano
- 3) ESP32 CAM Module
- 4) 2200mAh, 2S 2C, 7.4V Li-ion Battery
- 5) Arduino Nano 328P Expansion Adapter Breakout Board Shield
- 6) HC-SR04 Ultrasonic Sensor
- 7) A88 Metal Detector Non-Contact Metal Induction Detection Module
- 8) LM2596 DC-DC Buck Converter
- 9) HC-05 Bluetooth Module with TTL Output

Below Diagrams figure 4.1 and figure 4.2 show different connections between the parts.

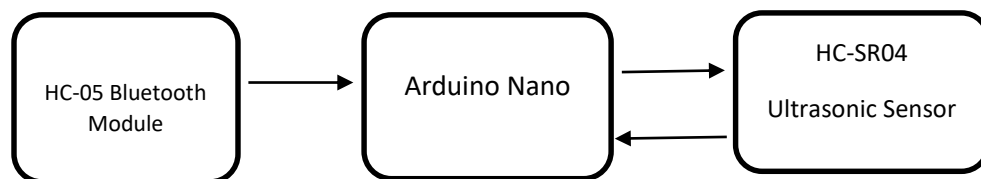


Fig 4.1: Connections for MCU operations

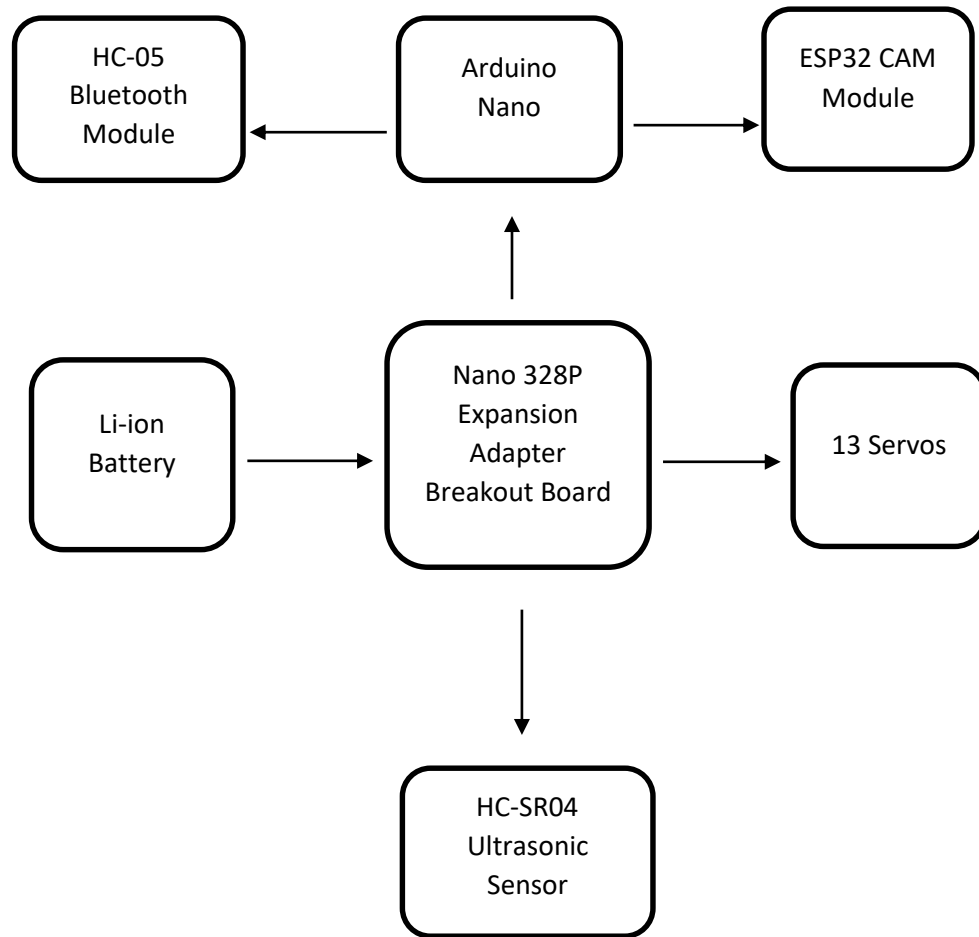


Fig 4.2: Power Connections in the circuit

4.1.1.MG90s Servo Motor:

MG90s are Micro servos that are low in weight that in turn reduce their weight and make them better for controlling. These servos have good torque(2.20Kg-cm), speed(0.11sec/60degree), the dead bandwidth of 1 us, good accuracy.

4.1.2.HC-SR04 Ultrasonic Sensor:

This Ultrasonic sensor is used to detect the distance from the sensor to the object in front of it. This sensor works best with a 2cm-400cm distance. This sensor consists of a sound transmitter (speaker) and a receiver. It sends an ultrasound of 40kHz from a transmitter and receives it back through the

receiver, then calculates the time it takes to come back. Then MCU calculates the distance from the sensor to the object.

4.1.3.Arduino Nano:

Arduino Nano is a compact MCU and is very much used in projects. It operates on 5V and consists of ATmega328P Chip. 32KB Flash memory, 2KB SRAM, 16MHz Clock Speed.



Fig 4.3:Arduino Nano

4.1.4.Nano 328P Expansion Adapter Breakout Board:

It is used as the extension PCB for Arduino Nano and is very much helpful in connecting all the components.

4.1.5.2200 mAh 2S 2C 7.4V Battery:

It is a long-lasting and rechargeable battery that is suitable for this application.

4.1.6.HC-05 Bluetooth Module:

HC-05 Bluetooth Module is used to receive and transmit the data from mobile phone to Arduino Nano using Bluetooth. It uses serial communication to connect with Arduino Nano.

4.1.7.ESP32 CAM Module:

It is used for surveillance. It can connect with wifi/internet and can transmit video/image data to anywhere in the world. It consists of an ESP32 chip and OV2640 2MP Cam. It supports Bluetooth 4.2 standards and Wifi 802.11bgn. It also consists of MCU6050.

4.1.8.LM2596-DC-DC Buck Converter:

It is a DC-DC step-down converter that can drive a 3A load with excellent line and load regulation. The output voltage can be regulated using an in-built potentiometer. For our case, we regulated the output voltage of 5V as Arduino Nano can only work on 5V.

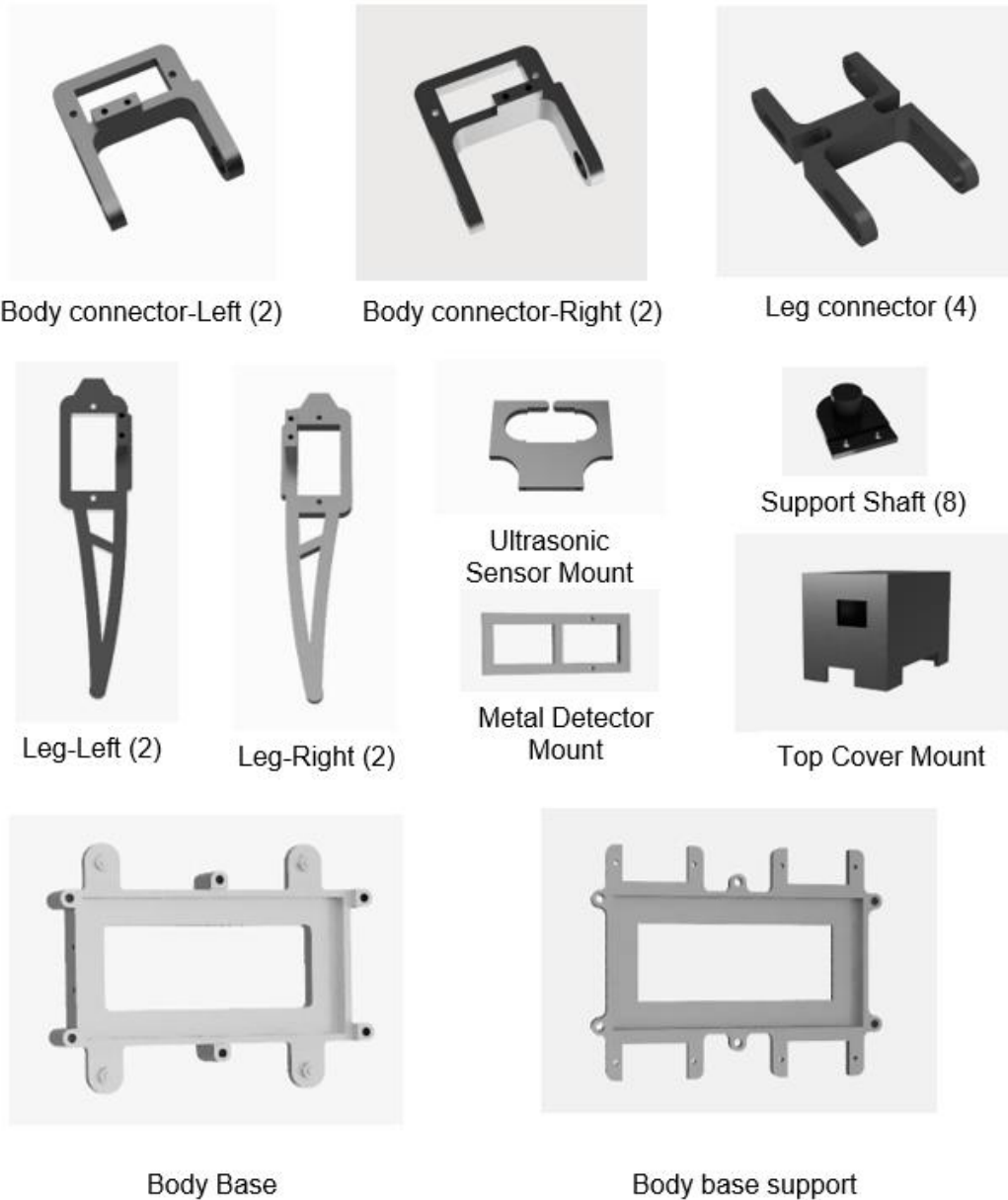


Fig 4.4:Body Parts of the Robot

4.2.COST:

The Project cost will be affordable and can be built with good quality in future industries at nearly the same price. The only thing that costs majorly in the robot is Servos. Although there are much cheaper servo motors on the internet we choose the best of its kind for a good quality product as they should have high torque and good speed. In this robot, 13 servo motors are required which costs Rs.300/each servo. The remaining components and electronics will not cost much.

Table 4.1: Bill of Materials

S.No.	Description	Quantity	Unit Price in Rupees	Price in Rupees
1	Bluetooth Module HC-05	1	650	650
2	LM2596 Buck Converter	1	100	100
3	Ultrasonic Sensor	1	120	120
4	Servos MG90S	15	300	4500
5	Body parts (3d printed)	1	1700	1700
6	ESP32 CAM Module	1	700	700
7	Arduino Nano	1	450	450
8	Nano 328P Expansion Adapter Breakout Board IO Shield: 150	1	150	150
9	Orange 18650 Li-ion 2200mAh 7.4v	1	500	500

	2S Rechargeable Battery	2c			
10	Minute Components: Toggle switch, Jumper wires, Screws, 5mm DC Female jack, Battery Charger, Arduino Cable	-	-	800	
				Total: 9670	

CHAPTER 5

RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

5.1.PROTOTYPE:

We have built a prototype of the robot using 3d printing and standard components as discussed earlier. The final look of the Robot is shown in figure 5.1 below.

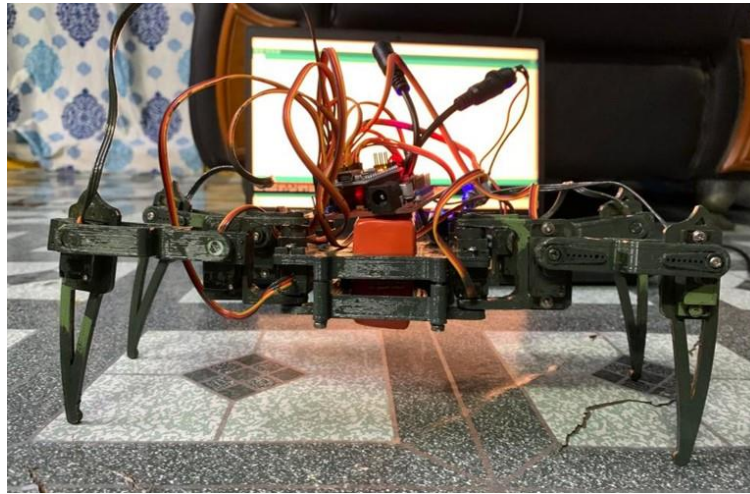


Fig 5.1: Robot Prototype

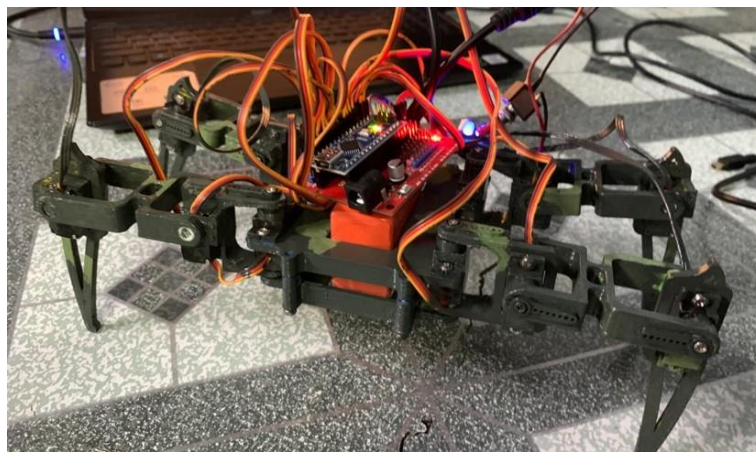


Fig 5.2: Robot Prototype

5.2.TESTING LAYOUT:

We have tested our robot's performance by passing it through various kinds of terrains and scenes. We included some obstacles, spike traps, sand, landmines (metal replicas), slopes, and some camp setups for surveillance. The 2D layout is shown in the figure below.

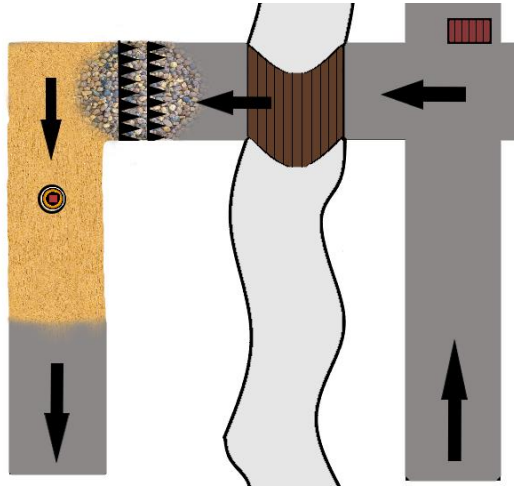


Fig 5.2:Testing Layout

CHAPTER 6

SUMMARY AND CONCLUSIONS

In This Project, we have successfully satisfied the minimum requirements of a military robot enabled with legged locomotion. The movement of the robot is smooth and well controllable. We have made a layout to test the robot under different conditions (rugged terrain, slopes, nail traps, sand, obstacles). The robot successfully passed the test. The surveillance camera is working well with good quality live footage and recording option. The rechargeable and long-lasting battery is an added advantage. It will be ready for manufacturing, after adding some high-end and military mapping-specific features, materials.

CHAPTER 7

FURTHER DEVELOPMENTS

- Lidar mapping can be added
- Autonomous Mobility can be added
- Should be made with much lighter and stronger materials
- GPS live location sending when detecting land mines, person, camps.

CHAPTER 8

REFERENCES

1. A Novel Design of a Quadruped Robot for Research Purposes, Yam Geva, Amir Shapeero, Department of Mechanical Engineering, Ben-Gurion University of the Negev, Israel.
2. Arduino Quadruped Robot Mr.V.Arun¹ , Mr.S.V.S.Prasad² , G.Sridhar Reddy³ , L.Ruthwik Reddy⁴ , M.Venkatesh⁵ , M.Sai Pavan Kumar⁶ ,
1,2Assoc.Prof, MLRIT,Hyderabad, 3,4,5,6Student,MLRIT,Hyderabad
3. Designing, 3D Printing of a Quadruped Robot and Choice of Materials for Fabrication, Akash Maity, Koustav Roy, Dhruvajyoti Gupta.
4. Espinal A., Rostro-Gonzalez H., Carpio M., Guerra-Hernandez E. I., Ornelas-Rodriguez M., H. J. Puga-Soberanes, M. A. Sotelo-Figueroa, P. Melin & *Computational Intelligence and Neuroscience*, vol. 2016
5. Mehmet Cavas, Muhammad Ahmad Baballe & International Journal of New Computer Architectures and their Applications
6. Priyaranjan Biswal, Prases K. Mohanty, Development of quadruped walking robots: A review, Ain Shams Engineering Journal, Volume 12, Issue 2, 2021.
7. Quadruped Robot: Past, Present And Future Nikunj S Yagnik, Dr. Mehul M Gor Automobile Engineering Department, ADIT, Gujarat Technological University.
8. Ragini Sharma, Monish Adhikari, Sanket Deshmuk, Chetan Mhatre & Journal of Advancements in Robotics
9. The kinematics and dynamics simulation of spider robots based on ADAMS, Fu Mengyu, North China Electric Power University.
10. Tolga Karakurt, Akif Durdu, and Nihat Yilmaz & International Journal of Machine Learning and Computing, Vol. 5, No. 2
11. Walking Robot Bio-inspired by Insect's Locomotion for Carbon Dioxide Diagnostic Indexed in the Air Santiago Noriega Álvarez, María Camila

Rojas and Hernando Leon-Rodriguez Electronic Department, Faculty of Engineering, El Bosque University, Bogota, Colombia.

12. Zareei, Amir & Ghayour, Mostafa. (2018). Inverse Kinematic Analysis of a Hexapod Spider-like Mobile Robot. Advanced Materials Research.

ANNEXURE-1

CODE:

```
#include <Servo.h>    //to define and control servos
#include <FlexiTimer2.h> //to set a timer to manage all servos
#include <NewPing.h>   // ultrasonic distance sensor library
/* Servos -----*/
//define 12 servos for 4 legs
char data=0;
Servo servo[4][3];
//define servos' ports
const int servo_pin[4][3] = { {2, 3, 4}, {5, 6, 7}, {8, 9, 10}, {11, 12, 13} };
/* Size of the robot -----*/
const float length_a = 60;
const float length_b = 77.5;
const float length_c = 27.5;
const float length_side = 71;
const float z_absolute = -28;
/* Constants for movement -----*/
const float z_default = -50, z_up = -30, z_boot = z_absolute;
const float x_default = 62, x_offset = 0;
const float y_start = 0, y_step = 40;
/* variables for movement -----*/
volatile float site_now[4][3]; //real-time coordinates of the end of each leg
volatile float site_expect[4][3]; //expected coordinates of the end of each leg
float temp_speed[4][3]; //each axis' speed, needs to be recalculated before
each movement
float move_speed; //movement speed
float speed_multiple = 1.2; //movement speed multiple
const float spot_turn_speed = 4;
const float leg_move_speed = 8;
const float body_move_speed = 3;
```

```

const float stand_seat_speed = 1;
volatile int rest_counter;    //+1/0.02s, for automatic rest
//functions' parameter
const float KEEP = 255;
//define PI for calculation
const float pi = 3.1415926;
/* Constants for turn -----*/
//temp length
const float temp_a = sqrt(pow(2 * x_default + length_side, 2) + pow(y_step,
2));
const float temp_b = 2 * (y_start + y_step) + length_side;
const float temp_c = sqrt(pow(2 * x_default + length_side, 2) + pow(2 * y_start
+ y_step + length_side, 2));
const float temp_alpha = acos((pow(temp_a, 2) + pow(temp_b, 2) -
pow(temp_c, 2)) / 2 / temp_a / temp_b);
//site for turn
const float turn_x1 = (temp_a - length_side) / 2;
const float turn_y1 = y_start + y_step / 2;
const float turn_x0 = turn_x1 - temp_b * cos(temp_alpha);
const float turn_y0 = temp_b * sin(temp_alpha) - turn_y1 - length_side;

//for ultrasonic sensor
#define TRIGGER_PIN  A0
#define ECHO_PIN    A1
#define MAX_DISTANCE 200
int thresh = 20;
long randNumber;

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
int mode=0;
/* -----*/

```

```

/*
  - setup function
  -----*/

void setup()
{
  //start serial for debug
  Serial.begin(9600);
  Serial.println("Robot starts initialization");
  delay(5000);
  //initialize default parameter
  pinMode(14, OUTPUT);
  set_site(0, x_default - x_offset, y_start + y_step, z_boot);
  set_site(1, x_default - x_offset, y_start + y_step, z_boot);
  set_site(2, x_default + x_offset, y_start, z_boot);
  set_site(3, x_default + x_offset, y_start, z_boot);
  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      site_now[i][j] = site_expect[i][j];
    }
  }
  //start servo service
  FlexiTimer2::set(20, servo_service);
  FlexiTimer2::start();
  Serial.println("Servo service started");
  //initialize servos
  servo_attach();
  Serial.println("Servos initialized");
  Serial.println("Robot initialization Complete");
}

```

```

randomSeed(analogRead(0));

    Serial.println("Stand");
    stand();
    delay(500);

    hand_wave(3);
    delay(1000);
}

void servo_attach(void)
{
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            delay(50);
            servo[i][j].attach(servo_pin[i][j]);
            delay(50);
        }
    }
}

void servo_detach(void)
{
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 3; j++)
        {

```

```

        servo[i][j].detach();
        delay(100);
    }
}
}
/*
- loop function
-----*/

void loop()
{
    if(Serial.available() > 0)
    {
        data = Serial.read();
        Serial.print(data);
        Serial.print("\n");
    }

    if(data == 'M'){mode=0;} //Manual Android Application and IR Remote
Control Command
    else if(data == 'A'){mode=1;} //Auto Obstacle Avoiding Command

    if (mode==0){
        if(data == 'F'){step_forward(1); } // if the bt_data is '1' the DC motor will go
forward
        else if(data == 'B'){step_back(1);} // if the bt_data is '2' the motor will Reverse
        else if(data == 'L'){turn_left(3);} // if the bt_data is '3' the motor will turn left
        else if(data == 'R'){turn_right(3);} // if the bt_data is '4' the motor will turn right
//else if(data == 'S'){Stop(); } // if the bt_data '5' the motor will Stop
        else if(data == 'U'){Serial.println("Stand");stand();}
        else if(data == 'D'){Serial.println("Sit");sit;}
        else if(data == 'S' ){ }
    }
}

```

```

else if(data == 'W'){ digitalWrite(14, HIGH);}
else if(data == 'w'){ digitalWrite(14, LOW);}
else if(data == 'H'){ Serial.println("Hand wave");hand_shake(3);}
else if(data == 'h'){ Serial.println("Hand wave");hand_shake(3);}
//else if(data == 'A'){ Serial.println("Body dance");body_dance(10);}
//else if(data == 'a'){ Serial.println("Body dance");body_dance(10);}
while(Serial.available()) {Serial.read();}
}

if(mode==1){

    int uS = sonar.ping_cm();
    // Serial.print("Ping: ");
    // Serial.print(uS);
    // Serial.println("cm");
    if (uS < thresh && uS > 1){
        step_back(3);
        delay(500);

    randomNumber = random(1,3);
    Serial.println(randomNumber);

    if(randomNumber == 1){
        turn_right(5);
        delay(500);
    }
    else if(randomNumber == 2){
        turn_left(5);
        delay(500);
    }
}
}

```

```

        else {
            step_forward(1);
        }
    }

    delay(10);
}

/*
- sit
- blocking function
-----*/
void sit(void)
{
    move_speed = stand_seat_speed;
    for (int leg = 0; leg < 4; leg++)
    {
        set_site(leg, KEEP, KEEP, z_boot);
    }
    wait_all_reach();
}

/*
- stand
- blocking function
-----*/
void stand(void)
{
    move_speed = stand_seat_speed;

```

```

for (int leg = 0; leg < 4; leg++)
{
    set_site(leg, KEEP, KEEP, z_default);
}
wait_all_reach();
}

/*
- spot turn to left
- blocking function
- parameter step steps wanted to turn
-----*/

#
void turn_left(unsigned int step)
{
    move_speed = spot_turn_speed;
    while (step-- > 0)
    {
        if (site_now[3][1] == y_start)
        {
            //leg 3&1 move
            set_site(3, x_default + x_offset, y_start, z_up);
            wait_all_reach();

            set_site(0, turn_x1 - x_offset, turn_y1, z_default);
            set_site(1, turn_x0 - x_offset, turn_y0, z_default);
            set_site(2, turn_x1 + x_offset, turn_y1, z_default);
            set_site(3, turn_x0 + x_offset, turn_y0, z_up);
            wait_all_reach();

```

```
set_site(3, turn_x0 + x_offset, turn_y0, z_default);  
wait_all_reach();
```

```
set_site(0, turn_x1 + x_offset, turn_y1, z_default);  
set_site(1, turn_x0 + x_offset, turn_y0, z_default);  
set_site(2, turn_x1 - x_offset, turn_y1, z_default);  
set_site(3, turn_x0 - x_offset, turn_y0, z_default);  
wait_all_reach();
```

```
set_site(1, turn_x0 + x_offset, turn_y0, z_up);  
wait_all_reach();
```

```
set_site(0, x_default + x_offset, y_start, z_default);  
set_site(1, x_default + x_offset, y_start, z_up);  
set_site(2, x_default - x_offset, y_start + y_step, z_default);  
set_site(3, x_default - x_offset, y_start + y_step, z_default);  
wait_all_reach();
```

```
set_site(1, x_default + x_offset, y_start, z_default);  
wait_all_reach();
```

```
}
```

```
else
```

```
{
```

```
//leg 0&2 move
```

```
set_site(0, x_default + x_offset, y_start, z_up);  
wait_all_reach();
```

```
set_site(0, turn_x0 + x_offset, turn_y0, z_up);  
set_site(1, turn_x1 + x_offset, turn_y1, z_default);  
set_site(2, turn_x0 - x_offset, turn_y0, z_default);  
set_site(3, turn_x1 - x_offset, turn_y1, z_default);
```

```

wait_all_reach();

set_site(0, turn_x0 + x_offset, turn_y0, z_default);
wait_all_reach();

set_site(0, turn_x0 - x_offset, turn_y0, z_default);
set_site(1, turn_x1 - x_offset, turn_y1, z_default);
set_site(2, turn_x0 + x_offset, turn_y0, z_default);
set_site(3, turn_x1 + x_offset, turn_y1, z_default);
wait_all_reach();

set_site(2, turn_x0 + x_offset, turn_y0, z_up);
wait_all_reach();

set_site(0, x_default - x_offset, y_start + y_step, z_default);
set_site(1, x_default - x_offset, y_start + y_step, z_default);
set_site(2, x_default + x_offset, y_start, z_up);
set_site(3, x_default + x_offset, y_start, z_default);
wait_all_reach();

set_site(2, x_default + x_offset, y_start, z_default);
wait_all_reach();
}
}
}

/*
- spot turn to right
- blocking function
- parameter step steps wanted to turn
-----*/

```

```

void turn_right(unsigned int step)
{
    move_speed = spot_turn_speed;
    while (step-- > 0)
    {
        if (site_now[2][1] == y_start)
        {
            //leg 2&0 move
            set_site(2, x_default + x_offset, y_start, z_up);
            wait_all_reach();

            set_site(0, turn_x0 - x_offset, turn_y0, z_default);
            set_site(1, turn_x1 - x_offset, turn_y1, z_default);
            set_site(2, turn_x0 + x_offset, turn_y0, z_up);
            set_site(3, turn_x1 + x_offset, turn_y1, z_default);
            wait_all_reach();

            set_site(2, turn_x0 + x_offset, turn_y0, z_default);
            wait_all_reach();

            set_site(0, turn_x0 + x_offset, turn_y0, z_default);
            set_site(1, turn_x1 + x_offset, turn_y1, z_default);
            set_site(2, turn_x0 - x_offset, turn_y0, z_default);
            set_site(3, turn_x1 - x_offset, turn_y1, z_default);
            wait_all_reach();

            set_site(0, turn_x0 + x_offset, turn_y0, z_up);
            wait_all_reach();

            set_site(0, x_default + x_offset, y_start, z_up);
            set_site(1, x_default + x_offset, y_start, z_default);

```

```

set_site(2, x_default - x_offset, y_start + y_step, z_default);
set_site(3, x_default - x_offset, y_start + y_step, z_default);
wait_all_reach();

set_site(0, x_default + x_offset, y_start, z_default);
wait_all_reach();
}
else
{
    //leg 1&3 move
    set_site(1, x_default + x_offset, y_start, z_up);
    wait_all_reach();

    set_site(0, turn_x1 + x_offset, turn_y1, z_default);
    set_site(1, turn_x0 + x_offset, turn_y0, z_up);
    set_site(2, turn_x1 - x_offset, turn_y1, z_default);
    set_site(3, turn_x0 - x_offset, turn_y0, z_default);
    wait_all_reach();

    set_site(1, turn_x0 + x_offset, turn_y0, z_default);
    wait_all_reach();

    set_site(0, turn_x1 - x_offset, turn_y1, z_default);
    set_site(1, turn_x0 - x_offset, turn_y0, z_default);
    set_site(2, turn_x1 + x_offset, turn_y1, z_default);
    set_site(3, turn_x0 + x_offset, turn_y0, z_default);
    wait_all_reach();

    set_site(3, turn_x0 + x_offset, turn_y0, z_up);
    wait_all_reach();

```

```

    set_site(0, x_default - x_offset, y_start + y_step, z_default);
    set_site(1, x_default - x_offset, y_start + y_step, z_default);
    set_site(2, x_default + x_offset, y_start, z_default);
    set_site(3, x_default + x_offset, y_start, z_up);
    wait_all_reach();

    set_site(3, x_default + x_offset, y_start, z_default);
    wait_all_reach();
}
}
}

/*
- go forward
- blocking function
- parameter step steps wanted to go
-----*/
void step_forward(unsigned int step)
{
    move_speed = leg_move_speed;
    while (step-- > 0)
    {
        if (site_now[2][1] == y_start)
        {
            //leg 2&1 move
            set_site(2, x_default + x_offset, y_start, z_up);
            wait_all_reach();
            set_site(2, x_default + x_offset, y_start + 2 * y_step, z_up);
            wait_all_reach();
            set_site(2, x_default + x_offset, y_start + 2 * y_step, z_default);
            wait_all_reach();

```

```
move_speed = body_move_speed;
```

```
set_site(0, x_default + x_offset, y_start, z_default);  
set_site(1, x_default + x_offset, y_start + 2 * y_step, z_default);  
set_site(2, x_default - x_offset, y_start + y_step, z_default);  
set_site(3, x_default - x_offset, y_start + y_step, z_default);  
wait_all_reach();
```

```
move_speed = leg_move_speed;
```

```
set_site(1, x_default + x_offset, y_start + 2 * y_step, z_up);  
wait_all_reach();  
set_site(1, x_default + x_offset, y_start, z_up);  
wait_all_reach();  
set_site(1, x_default + x_offset, y_start, z_default);  
wait_all_reach();
```

```
}
```

```
else
```

```
{
```

```
//leg 0&3 move
```

```
set_site(0, x_default + x_offset, y_start, z_up);  
wait_all_reach();  
set_site(0, x_default + x_offset, y_start + 2 * y_step, z_up);  
wait_all_reach();  
set_site(0, x_default + x_offset, y_start + 2 * y_step, z_default);  
wait_all_reach();
```

```
move_speed = body_move_speed;
```

```
set_site(0, x_default - x_offset, y_start + y_step, z_default);
```

```

    set_site(1, x_default - x_offset, y_start + y_step, z_default);
    set_site(2, x_default + x_offset, y_start, z_default);
    set_site(3, x_default + x_offset, y_start + 2 * y_step, z_default);
    wait_all_reach();

    move_speed = leg_move_speed;

    set_site(3, x_default + x_offset, y_start + 2 * y_step, z_up);
    wait_all_reach();
    set_site(3, x_default + x_offset, y_start, z_up);
    wait_all_reach();
    set_site(3, x_default + x_offset, y_start, z_default);
    wait_all_reach();
}
}
}

/*
- go back
- blocking function
- parameter step steps wanted to go
-----*/
void step_back(unsigned int step)
{
    move_speed = leg_move_speed;
    while (step-- > 0)
    {
        if (site_now[3][1] == y_start)
        {
            //leg 3&0 move
            set_site(3, x_default + x_offset, y_start, z_up);

```

```

wait_all_reach();
set_site(3, x_default + x_offset, y_start + 2 * y_step, z_up);
wait_all_reach();
set_site(3, x_default + x_offset, y_start + 2 * y_step, z_default);
wait_all_reach();

```

```

move_speed = body_move_speed;

```

```

set_site(0, x_default + x_offset, y_start + 2 * y_step, z_default);
set_site(1, x_default + x_offset, y_start, z_default);
set_site(2, x_default - x_offset, y_start + y_step, z_default);
set_site(3, x_default - x_offset, y_start + y_step, z_default);
wait_all_reach();

```

```

move_speed = leg_move_speed;

```

```

set_site(0, x_default + x_offset, y_start + 2 * y_step, z_up);
wait_all_reach();
set_site(0, x_default + x_offset, y_start, z_up);
wait_all_reach();
set_site(0, x_default + x_offset, y_start, z_default);
wait_all_reach();

```

```

}

```

```

else

```

```

{

```

```

//leg 1&2 move

```

```

set_site(1, x_default + x_offset, y_start, z_up);
wait_all_reach();
set_site(1, x_default + x_offset, y_start + 2 * y_step, z_up);
wait_all_reach();
set_site(1, x_default + x_offset, y_start + 2 * y_step, z_default);

```

```

wait_all_reach();

move_speed = body_move_speed;

set_site(0, x_default - x_offset, y_start + y_step, z_default);
set_site(1, x_default - x_offset, y_start + y_step, z_default);
set_site(2, x_default + x_offset, y_start + 2 * y_step, z_default);
set_site(3, x_default + x_offset, y_start, z_default);
wait_all_reach();

move_speed = leg_move_speed;

set_site(2, x_default + x_offset, y_start + 2 * y_step, z_up);
wait_all_reach();
set_site(2, x_default + x_offset, y_start, z_up);
wait_all_reach();
set_site(2, x_default + x_offset, y_start, z_default);
wait_all_reach();
}
}
}

// add by RegisHsu

void body_left(int i)
{
    set_site(0, site_now[0][0] + i, KEEP, KEEP);
    set_site(1, site_now[1][0] + i, KEEP, KEEP);
    set_site(2, site_now[2][0] - i, KEEP, KEEP);
    set_site(3, site_now[3][0] - i, KEEP, KEEP);
    wait_all_reach();
}

```

```
}
```

```
void body_right(int i)
```

```
{  
    set_site(0, site_now[0][0] - i, KEEP, KEEP);  
    set_site(1, site_now[1][0] - i, KEEP, KEEP);  
    set_site(2, site_now[2][0] + i, KEEP, KEEP);  
    set_site(3, site_now[3][0] + i, KEEP, KEEP);  
    wait_all_reach();  
}
```

```
void hand_wave(int i)
```

```
{  
    float x_tmp;  
    float y_tmp;  
    float z_tmp;  
    move_speed = 1;  
    if (site_now[3][1] == y_start)  
    {  
        body_right(15);  
        x_tmp = site_now[2][0];  
        y_tmp = site_now[2][1];  
        z_tmp = site_now[2][2];  
        move_speed = body_move_speed;  
        for (int j = 0; j < i; j++)  
        {  
            set_site(2, turn_x1, turn_y1, 50);  
            wait_all_reach();  
            set_site(2, turn_x0, turn_y0, 50);  
            wait_all_reach();  
        }  
    }
```

```

    set_site(2, x_tmp, y_tmp, z_tmp);
    wait_all_reach();
    move_speed = 1;
    body_left(15);
}
else
{
    body_left(15);
    x_tmp = site_now[0][0];
    y_tmp = site_now[0][1];
    z_tmp = site_now[0][2];
    move_speed = body_move_speed;
    for (int j = 0; j < i; j++)
    {
        set_site(0, turn_x1, turn_y1, 50);
        wait_all_reach();
        set_site(0, turn_x0, turn_y0, 50);
        wait_all_reach();
    }
    set_site(0, x_tmp, y_tmp, z_tmp);
    wait_all_reach();
    move_speed = 1;
    body_right(15);
}
}

```

```

void hand_shake(int i)
{
    float x_tmp;
    float y_tmp;
    float z_tmp;

```

```

move_speed = 1;
if (site_now[3][1] == y_start)
{
    body_right(15);
    x_tmp = site_now[2][0];
    y_tmp = site_now[2][1];
    z_tmp = site_now[2][2];
    move_speed = body_move_speed;
    for (int j = 0; j < i; j++)
    {
        set_site(2, x_default - 30, y_start + 2 * y_step, 55);
        wait_all_reach();
        set_site(2, x_default - 30, y_start + 2 * y_step, 10);
        wait_all_reach();
    }
    set_site(2, x_tmp, y_tmp, z_tmp);
    wait_all_reach();
    move_speed = 1;
    body_left(15);
}
else
{
    body_left(15);
    x_tmp = site_now[0][0];
    y_tmp = site_now[0][1];
    z_tmp = site_now[0][2];
    move_speed = body_move_speed;
    for (int j = 0; j < i; j++)
    {
        set_site(0, x_default - 30, y_start + 2 * y_step, 55);
        wait_all_reach();
    }
}

```

```

    set_site(0, x_default - 30, y_start + 2 * y_step, 10);
    wait_all_reach();
}
set_site(0, x_tmp, y_tmp, z_tmp);
wait_all_reach();
move_speed = 1;
body_right(15);
}
}

```

/*

- micros servos service /timer interrupt function/50Hz
- when set site expected, this function move the end point to it in a straight line
- temp_speed[4][3] should be set before set expect site, it make sure the end point

move in a straight line, and decide move speed.

-----*/

```

void servo_service(void)
{
    sei();
    static float alpha, beta, gamma;

    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            if (abs(site_now[i][j] - site_expect[i][j]) >= abs(temp_speed[i][j]))
                site_now[i][j] += temp_speed[i][j];
            else

```

```

        site_now[i][j] = site_expect[i][j];
    }

    cartesian_to_polar(alpha, beta, gamma, site_now[i][0], site_now[i][1],
site_now[i][2]);
    polar_to_servo(i, alpha, beta, gamma);
}

rest_counter++;
}

/*
- set one of end points' expect site
- this founction will set temp_speed[4][3] at same time
- non - blocking function
-----*/
void set_site(int leg, float x, float y, float z)
{
    float length_x = 0, length_y = 0, length_z = 0;

    if (x != KEEP)
        length_x = x - site_now[leg][0];
    if (y != KEEP)
        length_y = y - site_now[leg][1];
    if (z != KEEP)
        length_z = z - site_now[leg][2];

    float length = sqrt(pow(length_x, 2) + pow(length_y, 2) + pow(length_z, 2));

    temp_speed[leg][0] = length_x / length * move_speed * speed_multiple;
    temp_speed[leg][1] = length_y / length * move_speed * speed_multiple;

```

```

temp_speed[leg][2] = length_z / length * move_speed * speed_multiple;

if (x != KEEP)
    site_expect[leg][0] = x;
if (y != KEEP)
    site_expect[leg][1] = y;
if (z != KEEP)
    site_expect[leg][2] = z;
}

/*
- wait one of end points move to expect site
- blocking function
-----*/

void wait_reach(int leg)
{
    while (1)
        if (site_now[leg][0] == site_expect[leg][0])
            if (site_now[leg][1] == site_expect[leg][1])
                if (site_now[leg][2] == site_expect[leg][2])
                    break;
}

/*
- wait all of end points move to expect site
- blocking function
-----*/

void wait_all_reach(void)
{
    for (int i = 0; i < 4; i++)
        wait_reach(i);
}

```

```

}

/*
- trans site from cartesian to polar
- mathematical model 2/2
-----*/

void cartesian_to_polar(volatile float &alpha, volatile float &beta, volatile float
&gamma, volatile float x, volatile float y, volatile float z)
{
//calculate w-z degree
float v, w;
w = (x >= 0 ? 1 : -1) * (sqrt(pow(x, 2) + pow(y, 2)));
v = w - length_c;
alpha = atan2(z, v) + acos((pow(length_a, 2) - pow(length_b, 2) + pow(v, 2) +
pow(z, 2)) / 2 / length_a / sqrt(pow(v, 2) + pow(z, 2)));
beta = acos((pow(length_a, 2) + pow(length_b, 2) - pow(v, 2) - pow(z, 2)) / 2 /
length_a / length_b);
//calculate x-y-z degree
gamma = (w >= 0) ? atan2(y, x) : atan2(-y, -x);
//trans degree pi->180
alpha = alpha / pi * 180;
beta = beta / pi * 180;
gamma = gamma / pi * 180;
}

/*
- trans site from polar to microservalos
- mathematical model map to fact
- the errors saved in eeprom will be add
-----*/

void polar_to_servo(int leg, float alpha, float beta, float gamma)

```

```

{
  if (leg == 0)
  {
    alpha = 90 - alpha;
    beta = beta;
    gamma += 90;
  }
  else if (leg == 1)
  {
    alpha += 90;
    beta = 180 - beta;
    gamma = 90 - gamma;
  }
  else if (leg == 2)
  {
    alpha += 90;
    beta = 180 - beta;
    gamma = 90 - gamma;
  }
  else if (leg == 3){
    alpha = 90 - alpha;
    beta = beta;
    gamma += 90;
  }
  servo[leg][0].write(alpha);
  servo[leg][1].write(beta);
  servo[leg][2].write(gamma);
}

```