QUADRUPED ROBOT WITH A ROBOTIC ARM

By: Natnael Asmare

date: Dec 11, 2020

Abstract

In recent decades, robots have become one of the fastest growing industries, we have wheeled robots, tracked robots, legged robots for traversing different trains. Comparing this different type of robots legged robots have the upper hand when it comes to unstructured terrains. The main focus of this thesis is to propose a relatively cheap design for a general-purpose quadruped robot that could be used as a teaching aid and also for research purposes.

To develop the robot a 3D model was developed using solid works. Then the kinematic model for the robot was devised. The forward and inverse kinematic calculations for the kinematic model was shown. The torque requirements are figured out by devising a torque calculation for the worst loading scenario. A stable gait pattern for walking and turning the robot has been proposed and the proper algorithms to implement the gait patterns has been devised. A simplistic android application for control was developed. The experimental results demonstrate the effectiveness of the proposed algorithms.

Table of content

List of Figures

List of Tables

CHAPTER ONE

Introduction and Background

Since the beginning of the development of bionic robots in the 1960, the delicate structure, movement mechanism, and behavior of organisms have become the objective of robotic imitation. The introduction of bionics robots has improved the adaptability of robots in unstructured and unknown environments. Legged robots are a product of imitation of the structure and movement of the legs and feet of mammals, insects, and amphibians. Due to its motion trajectory, it is more maneuverable and adaptable in a nonstructural environment and has better flexibility, stability, and fault tolerance than a wheeled and crawler robot. In recent years the development and construction of legged robots has become a hot topic in the field of robotics. From the different kind of legged robots, the quadruped robot has better load and high stability than bipedal robot and less complexity, less redundancy than the multi-legged robots. As the famous Japanese foot robot research scholar Shigeo Hirose believes that the quadruped robot is the best form of legged robots form the aspects of stability and control difficulty and manufacturing cost. Compared to wheeled or tracked mobile robots multi-legged robot possesses a greater potential.

Recently, several quadruped robots are being developed which are equipped with hydraulic actuators or electric actuators. One of the leading companies in robotic research Boston Dynamics developed a series of robots equipped with hydraulic actuators with high energy density. Hydraulic drive quadruped robots are limited in their outdoor use due to their large noise and size limits. Hydraulic actuation in robotics gives them better payload carrying capacity but adds to their complexity due to having oil circuits, which often makes the structure more complex. Quadruped robots equipped with electric actuators can even be used in indoor environments, such as SpotMini, ANYmal, and MIT cheetah. Now a days there is a large number of active researches programs in the field of legged locomotion. Apart from designing and building the robot itself, the main challenges are the planning and implementation of the legs' motions in generating a walking sequence, namely how to plan the steps of the robot so that it moves in a desirable way while maintaining equilibrium constraints (quasi-static walking) or else maintaining the stability of the robot (dynamic walking).

Figure 1: Spot mini

Figure 2: ANYmal

Figure 3: MIT mini cheetah

The majority of the research that is being done on quadruped robots in this big research institutions is on a scale comparable to a mature dog. Working on research on this scale will require a considerable amount of budget. Our goal is to present a much smaller and much simpler legged robot platform with most of the functionality still intact. The size of our robot will be comparable to a mature cat. The major reason for choosing this scale is to reduce the cost of building the robot. In addition to their scale these robots which are built in the large institution use custom made actuators and different electronic parts which makes it hard to replicate. In these papers the design presented can be built from off -the shelf products. Therefore, it can be used immediately in a simple and cost-effective manner.

Many researchers develop their own robot algorithms and theories but in order to test their finding they will need an actual robot. It may be possible to test some aspects in a simulation software but not all. So, this cheap small-scale robot will be a great help for researchers whose main focused is in theory and algorithm development. The main advantages of the presented robotic platform are design simplicity, structure modularity, low cost and a simple interface.

The paper is structured as follows:

- First, we will start by making a 3D model
- Then we will make a kinematic model to represent the robot
- Forward and Inverse kinematic calculations
- Torque requirement
- Designing Gait pattern for walking and turning
- Construction of robot
- Android application for control

Motivation

Most of the developed countries are already working on robotic system and have managed to utilize this machine in their industries and homes to some extent. And our country is trying to become more industrialized but no visible efforts have been made to utilize robots in industries which could increase a company profit significantly. One of the reasons for not being part of such a technology is the cost to design and develop robotic system. If we could come up with chipper ways to design and develop robots more and more people will be involved with the technology which later translates to advances in the field. The more research and developments are done in this field more robust robotic systems will be developed which can be commercialized. Having a commercial product will again reinforce the advance of the technology.

When one thinks of the future, he thinks about general purpose robots. Robots helping around in the day to day activities of a conventional house. Mass use of robots in the industries. What is the end goal of the human race? From our point of view if the development of general-purpose robots is sufficiently advanced then robots can do what people do in the industry or the office. People may not need to go to work. A sufficiently advanced general-purpose robotic system could do most if not all of the things that a human can do. Some of this task could be a robot farmer, fully automated industries or a robot assistance in our homes. One who controls this technology will control the world. Whether we become part of it or don't either ways the world is going to embrace the essence of robots in our daily lives and the industry. It is just a matter of time.

Problem Statement

Researchers who are interested in theory and algorithm development need a cheaper and modular platform to do research.

Student in higher institution learn robotics and electromechanically systems with little exposure to actual robotic systems with may be attributed to the larger costs of robots.

Designing specific mechanism for individual problems will unnecessary and waste of money if it is possible to develop modular robots with multiple applications with little hardware and software modification.

Objective

General objective

Our main goal for this project is that to prove a general-purpose robot could be built with relatively cheap components which can be found easily in the market. With most of the functionality of a general-purpose robot intact. This robot will be of service for students studying robotic systems to have a practical experience for their study and also for researchers who want to focus on the theory and algorithm development to test their finding on an actual hardware system.

Specific objective

Although our plan in this project is to build a platform for other developers to participate in the solution development, we will be programming the robot to demonstrate our claims.

Some of the application areas for such a robot are:

- Planting trees (for ready-made holes)
- Surveillance (using inbuilt camera)
- Defuse a bomb (cut a wire with remote control)

CHAPTER TWO

Components used

Electronics

- 16 micro servo motor
- Arduino Mega
- Ultrasonic sensor
- Accelerometer
- Dc to Dc buck converter
- Bluetooth module

Equipment

- Multimeter
- Soldering iron
- Hot glue gun
- Lead
- Computer

Software

- Proteus
- SolidWorks
- Vs code

Cost break down

Electronic device	***cost***
16xMircro servo motors	$7x16 = $112
Arduino Mega	$30
Ultrasonic sensor	$2
Accelerometer	$2
Dc to Dc buck converter	$3
Bluetooth module	$4
	Total = $153

Table 1: Cost break down

The majority of the cost comes from the servo motors which account more than 70% of the cost. The total cost for building the robot is $153 in total not including the manpower.

CHAPTER THREE

Methodology

A problem in robotics is concerned in multiple disciplines. To develop a successful robotic system, one has to work on the mechanical system design since there is a lot of joints and moving part this has to be designed with care. And then comes the electrical system design how the actuators and sensors should be connected to the central controlling micro controller, the power supply design since robots use a large number of actuators the power consumption will be usually high. And finally, the software system has to be designed how the different actuator should be controlled with different algorithms and preparing the API for further developments.

- The first thing to do for the project is design the mechanical system on a 3D modeling software. How the actuator should be arranged in the body of the robotic system. Also, the aesthetic of the robot should be in consideration.

- Then comes design of the electrical system. First the power supply for the whole robot will be made and then how the different actuator and sensors are connected to the micro-controller and the power supply unit. This will be done on a simulation software like proteus.

- Then the software program design will be made both for the robot system and the mobile application control. The software program will include a walking algorithm for the robot movement.

- After doing the 3D model and software simulation we will start on developing the actual hardware.

- Although we may have a working simulation in reality there will be a lot of errors, we have to do a lot of testing and calibrating the robotic system and improving the design.

- After the hardware implementation passes the testes then it will be ready to be programed with functionality. The test functionality that we will be programming are surveillance system, bomb defusing system, tree planting system.

CHAPTER FOUR

Implementation

Design of the quadruped locomotive

Requirements:

- The body should be able to contain all the electronics
- A single step of at least 5cm
- Hight adjustable
- Mimic a quadrupedal animal's gait and form
- Self-Balancing

Leg design

The main design problem in designing a quadruped robot is in the design of the legs. As of the requirements the leg design should be able to support at least a single step of 5 cm, 3DOF, must be height adjustable and the form should resemble of a quadrupedal animal.

The below sketch can be taken as the general default form for the quadruped locomotive viewed from the side. The robot leg will have two joints in each leg as it is shown in the sketch with red circles. For the height to be adjustable the robot's joints default position will be 45° in the shoulders measured from the normal of robot body to ground and 90° in the elbow measured form the upper leg section. This angle values are chosen so that the robot is at its mid height by default.

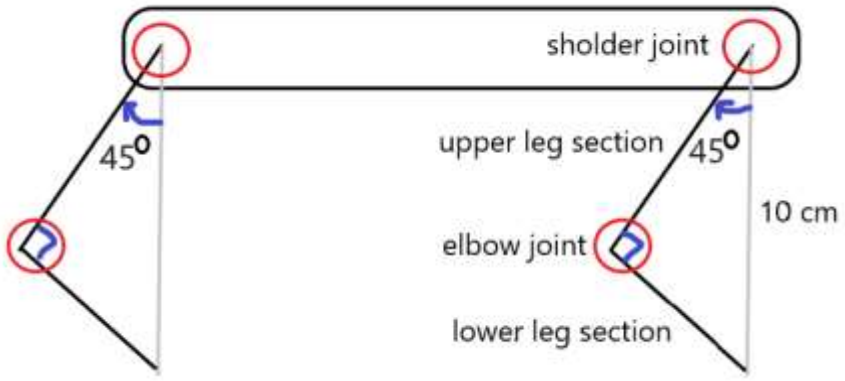


Figure 4: General default form for the quadruped locomotive

There will be two extreme cases during height adjusting. One is when the shoulder joint is 90^o from the normal of robot body to ground and 0^o in the elbow measured from the upper leg section. This is the minimum height achieved by the robot (body flat to ground). And the other is when the shoulder joint is 0^o from robot body to ground and 180^o in the elbow measured from the upper leg section. This is when the robot will have the maximum height. Both of these cases are show in the below sketches.

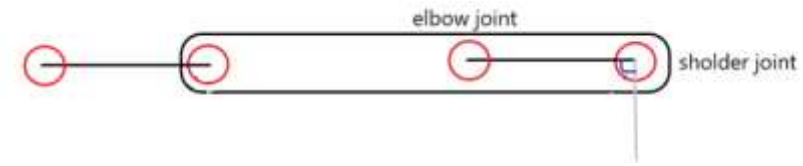


Figure 5: Minimum height (body flat to ground)

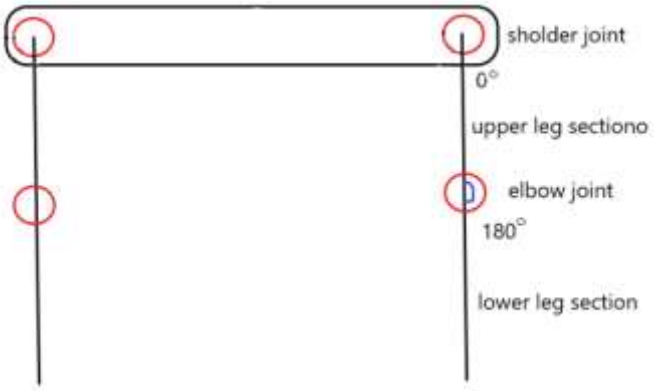


Figure 6: Maximum height

The other requirement in the design of the quadruped locomotive is that the legs should support at least a single step distance of 5cm. This distance is measured from the normal of the robot body to ground. This design constraint and the robot body to ground distance will give as the minimum length of the robot legs. Quadruped robots are superior than wheeled robot in their ability to maneuver in rough train. One example for this is making the robot able to maneuver in decently trimmed grass ground. And for this we need to have enough ground clearance. Taking in consideration that we are modeling our robot after a cat and to make sure that it will be able to maneuver in grass ground, we have chosen to make the ground clearance to be 10cm. This is the distance from the shoulder joint to the ground when the robot is in its default height as it is shown on fig 1. Using these two constraints we can determine the minimum length of the robot legs.

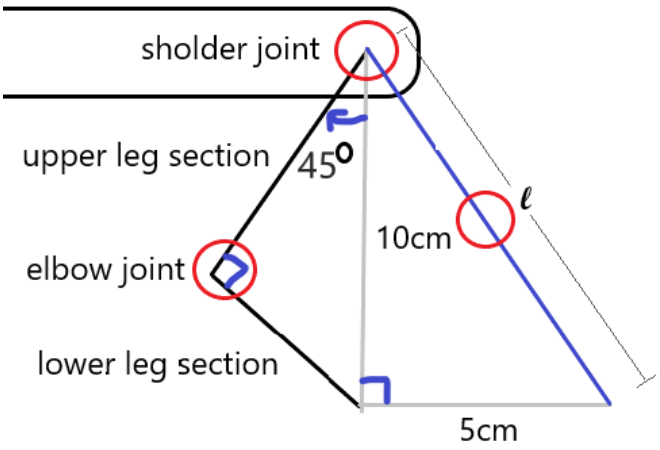


Figure 7: Minimum leg of the robot leg

As it can be seen from Figure 7 the minimum leg length of the robot leg can be calculated from the two constraints (minimum single step distance and robot default height or ground clearance) using Pythagorean theorem.

$$10^2 + 5 = l^2$$

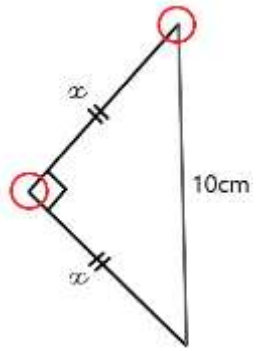
$$l^2 = 125$$

$$l = \sqrt{125} \approx 11 \cdot 18\ cm$$

From the above calculation we found that the minimum leg length should be around 11.2 cm. The maximum leg length should be determined by how much torque the shoulder joint servo motor can generate, that is the torque generated should be able to lift the hole leg as well as support the weight of the body.

Any leg length between the minimum and the maximum leg length will be applicable but it is best not to be too close to the two extremes in order to compensate for errors.

Another requirement in our design is to mimic the form of a quadruped animal and good aesthetics. For this will make the upper leg section and lower leg section equal. And also, the angle between the upper and lower leg section 90° in default stance. From this it is possible to calculate the length of the upper and lower leg sections. Again, we will be using the Pythagorean theorem to determine the length.



$$x^2 + x^2 = 100$$

$$2x^2 = 100$$
$$x^2 = 50$$
$$x \approx 7.07cm$$

Figure 8: Leg section calculation

Form the above calculation we can see that the length for each leg section should be around 7cm. And the total leg length will be 14cm. Earlier we have found that the minimum leg length for the robot should be greater than 11.2cm to satisfy the minimum of 5cm single step constraint and since 14cm is indeed greater than 11.2cm it is possible to use it as the robot leg length.

We have to note that this number values are dimension measured between the rotation axis's and ground the actual leg part will be larger for the benefit of mechanical design cohesion. But still the length between the joints and ground will still be the same as the value determined in the above calculations.

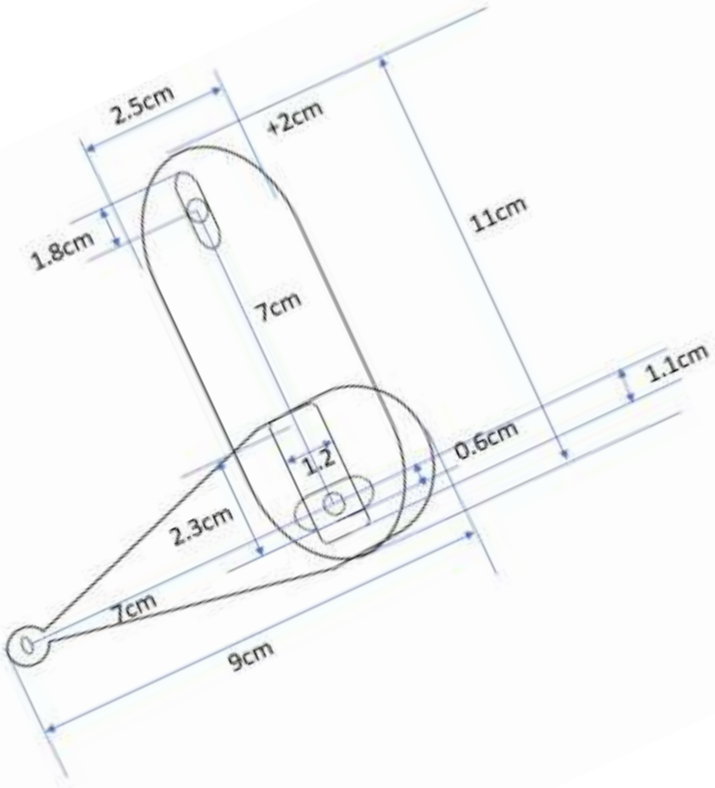


Figure 9: Leg dimensions for actual development

Now that we have determined all the necessary dimensions, we can start designing the leg parts. Before developing the physical parts, we will use a 3D simulation software for the purpose of reducing error and rapid prototyping. The software of choice for this purpose is Solidworks.

Upper leg section

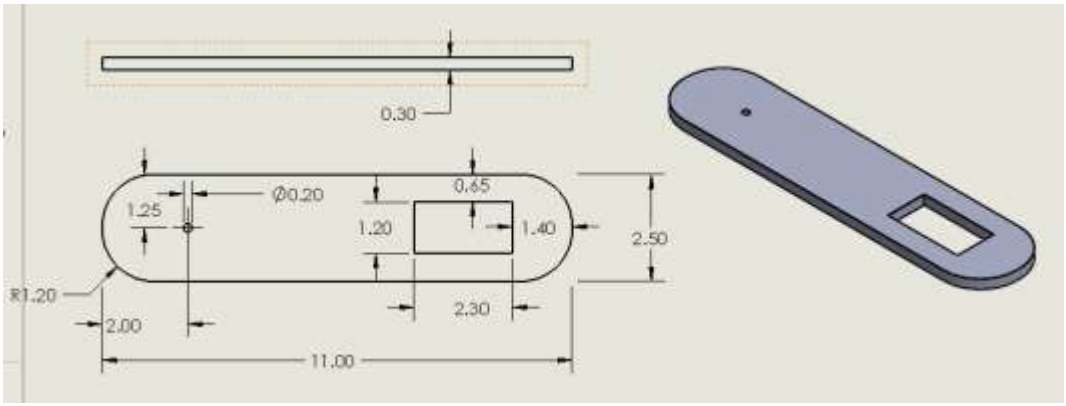


Figure 10: Upper leg section (all measurements are in centimeters)

As it can be seen form figure 10 the total length of the upper leg section is 11 cm but the distance between the rotation axis's will be 7 cm as determined in our calculations.

Lower leg section

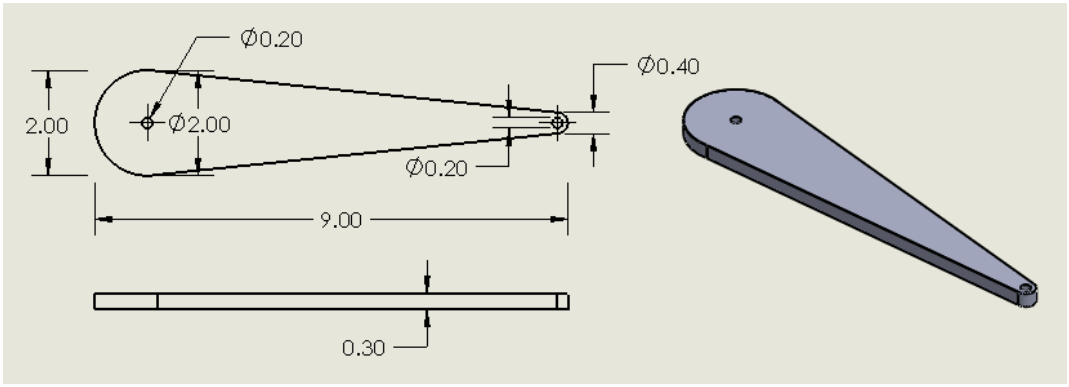


Figure 11: Lower leg section (all measurements are in centimeters)

As it can be seen form figure 11 the total length for the lower leg section is 9 cm but the distance between the rotation axis of the joint and ground will be 7 cm as determined in our calculations.

Body design

The robot body should be able to contain all the important components and house them. The body must be strong enough to protect the internal components from physical damages. It is also an added bonus if design aesthetics is pleasing. The different components that body must enclose are listed below.

- Arduino Mega (make sure the design also supports Raspberry pie)
- 9 V battery (to power the micro controller)
- Bower Bank (to power the servos)
- Bluetooth module
- Accelerometer
- Ultrasonic sensor
- Servo motor

In order to design the robot body, the dimensions for the individual components that are to be enclosed in must be determined. The dimensions for each individual component are listed below.

Arduino dimensions

- Height 12.45 mm ~ 1.3 cm
- Length 107.95 mm ~ 10.8 cm
- Width 53.42 mm ~ 5.4 cm

Raspberry pie dimensions

- Height 16.72 mm ~ 2 cm
- Length 87.37 mm ~ 8.8 cm
- Width 57.3 mm ~ 6 cm

Arduino with prototyping shield dimensions

- Height 12.45 mm ~ 2.7 cm
- Length 107.95 mm ~ 12 cm
- Width 53.42 mm ~6 cm

Ultrasonic sensor dimensions

- Length = 4.6 cm
- Width = 2 cm
- Height = 1.8 cm

To support both Arduino and Raspberry pie the dimension dedicated to microcontroller in the body design should be

- Height 2.7 cm
- Length 12 cm
- Width 6 cm

In designing the robot body, we must make sure it will contain all the internal components. So, we must model all the internal elements that will be in the body to find the total volume or dimensions for the robot body. The major elements in the robot body to take space are the Arduino with shield, 9 v battery, power bank, ultrasonic sensor and servo motors.

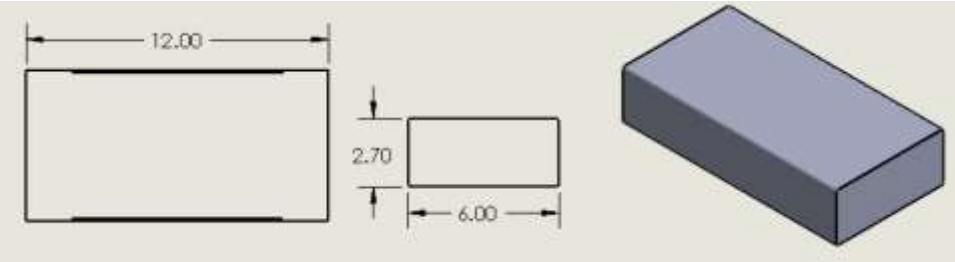


Figure 12: Arduino with the shield model (all measurements are in centimeters)

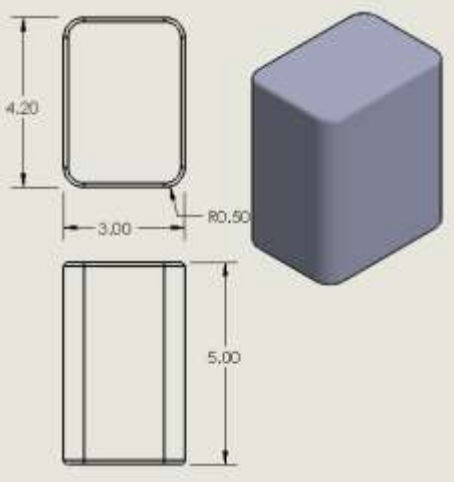


Figure 13: 9v battery model (all measurements are in centimeters)

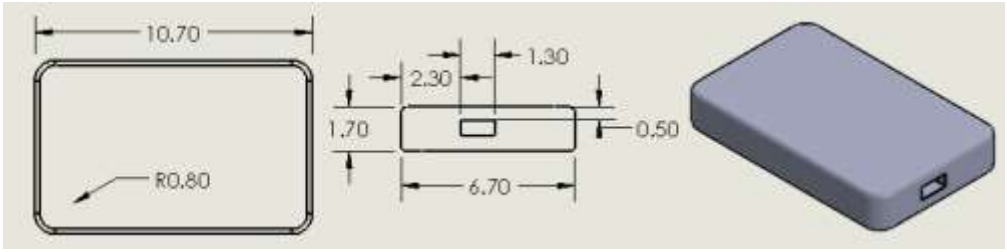


Figure 14: Power bank model (all measurements are in centimeters)

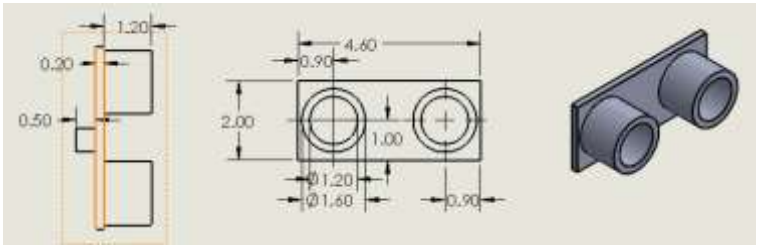


Figure 15: Ultrasonic sensor model (all measurements are in centimeters)

Body internals arrangement v1

All stacked on top on of each other

Figure 16: Body internal arrangement v1

The Height becomes 7.4 cm

- Good center of mass
- Bad ascetics (height is too long)

Body internals arrangement v2

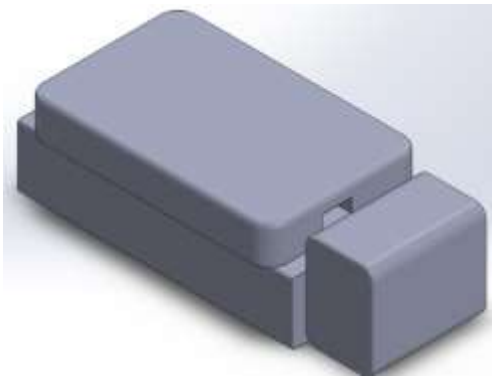


Figure 17: Body internal arrangement v1

- The Height becomes 4.4 cm
- Good ascetics
- Length = 15 cm
- Width = 6.7 cm (determined by the power bank width)

Body enclosure design v1 by using Body internals arrangement v2

- Servo in body 23mm x 2 = 46 mm = 4.6 cm
- Design v2 length = 15cm + 4.6 cm = **19.6 cm**
- Width = **7.3 cm**
- Height = **4.4 cm**

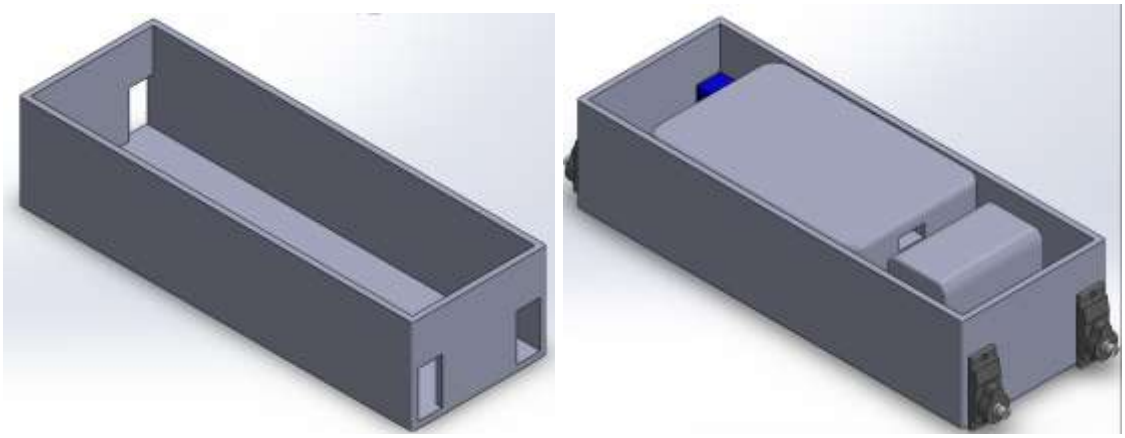


Figure 18: Body enclosure design v1 by using Body internals arrangement v2

Body enclosure design v2 by using Body internals arrangement v2 with different servo configuration

- Width = Arduino with shield + 2X servo width + encloser thickness (for both sides)
  - 6cm + (1.2cm * 2) + 0.6cm = 9cm
- Length = Body internals arrangement v2 + encloser thickness (for both sides)
  - 15cm + 0.6cm = 15.6cm
- Height = 4.4 cm

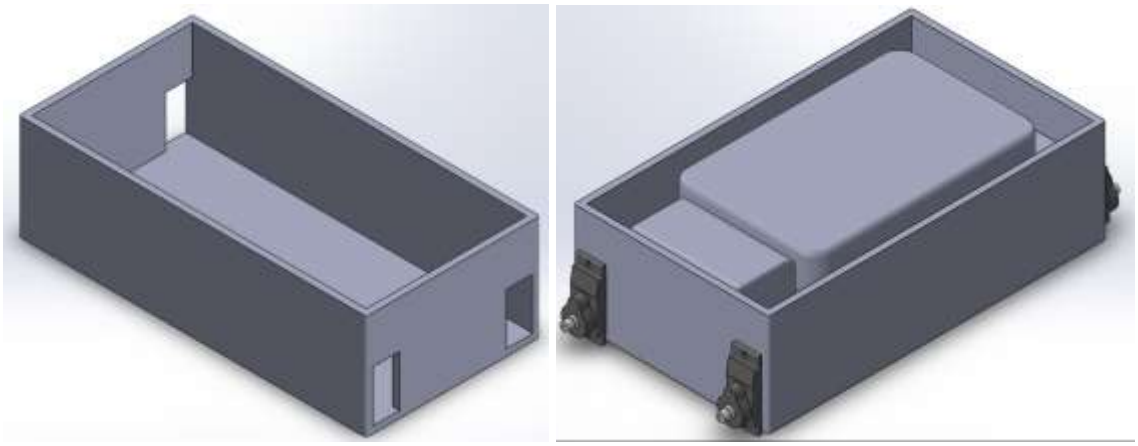


Figure 19: Body enclosure design v2 by using Body internals arrangement v2 with different servo configuration

Comparing the two body designs we have chosen the second design (body enclosure design v2). We have chosen this configuration due to its reduced length. Now that we have the designs for the legs and the robot body, we can combine them to form the final design for the quadruped locomotive. We can see the final 3D renderings in the below figures.

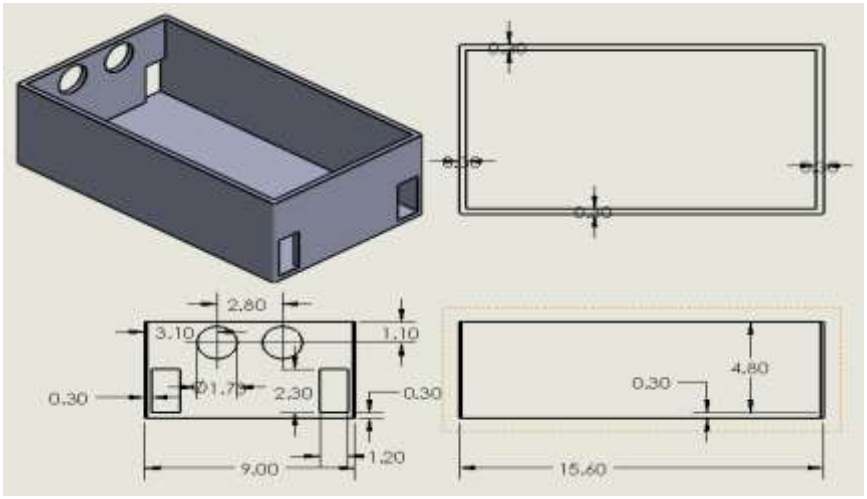


Figure 20: Enclosure dimensions

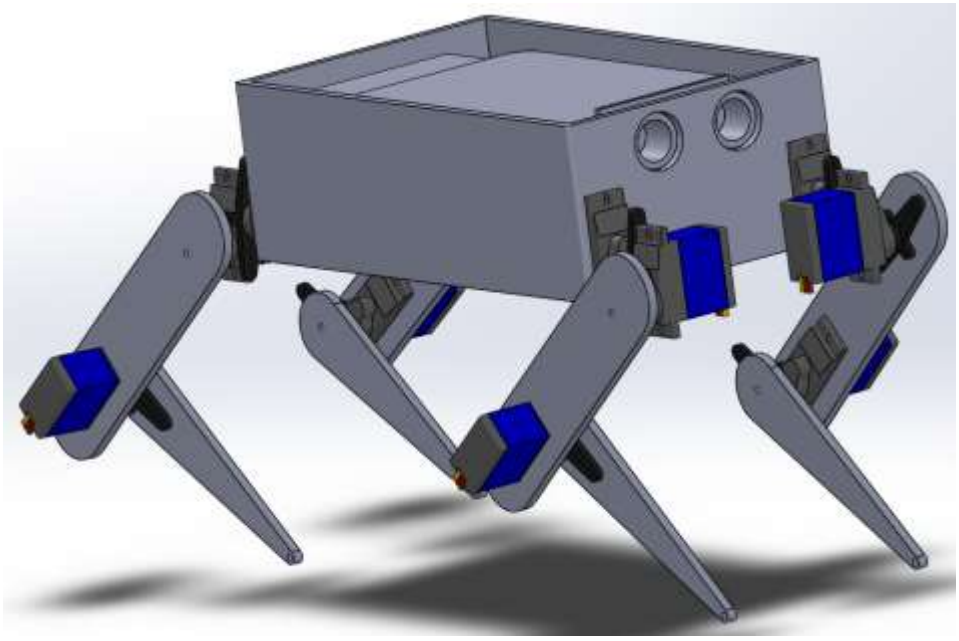


Figure 21: Final 3D rendering for the quadruple locomotive

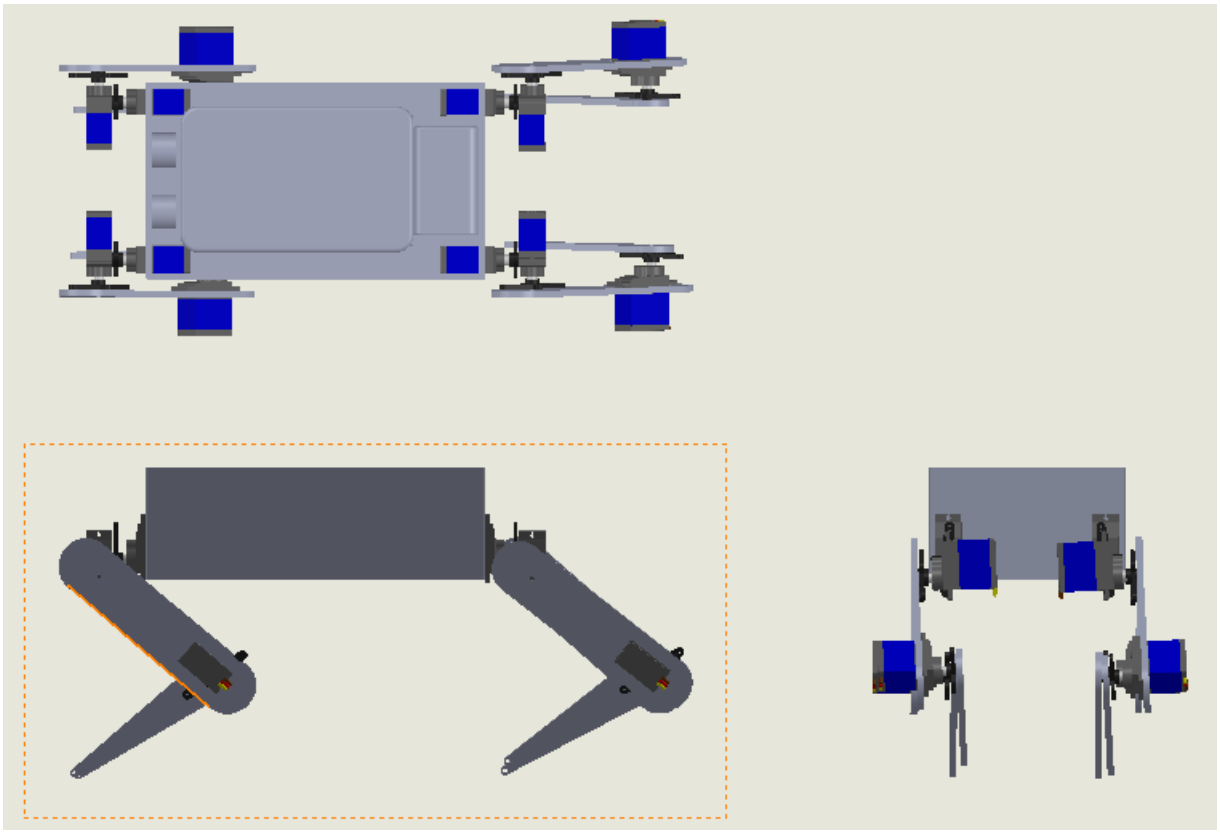


Figure 22: Full lower body build

Design of the robotic arm

We have designed our robot to be mobile thus the quadruped locomotive. Using the quadruped locomotive our robot can navigate through rough train and go to different places. In order for the robot to be of any use it must be able to manipulate objects and move them in space. A robotic arm will be mounted on top of the quadruped locomotive to make the robot able to grab and move objects in space. Due to its mobility it can go to different places and due to the robotic arm, it will be able to manipulate objects in different locations.

In designing the robotic arm, we need to consider different requirements to come up with the optimal dimensions of each joint.

Requirements:

- 3 DOF?
- Able to reach to objects on ground level
- Interchangeable end effector design

The robotic arm kinematic arrangement of choice is the three revolute joint Articulated configurations (RRR). We are choosing this configuration due to its relatively large freedom of movement in a compact space and also easier implantation using servo motor which are widely available in the market. The joint configuration and workspace are shown on the images below.

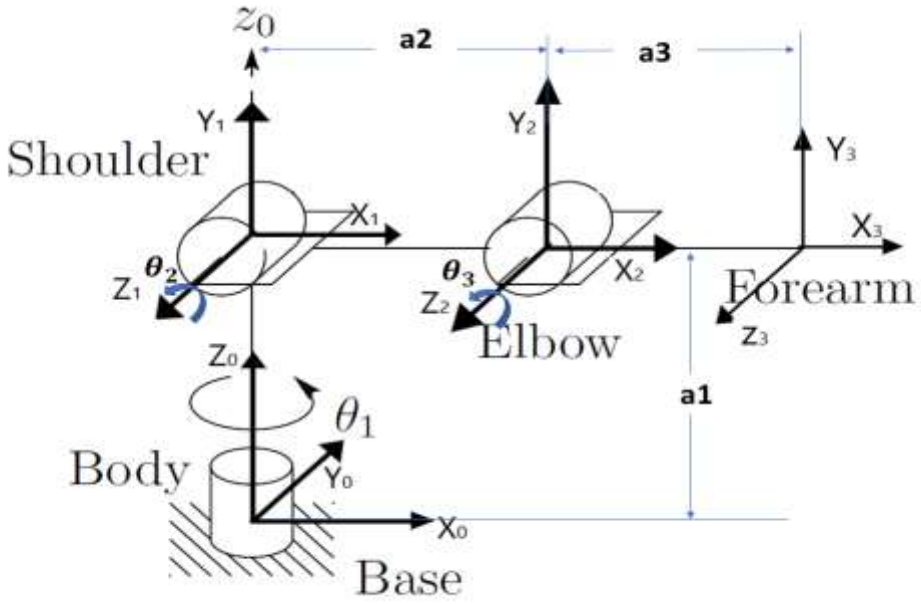


Figure 23: Articulated arm model

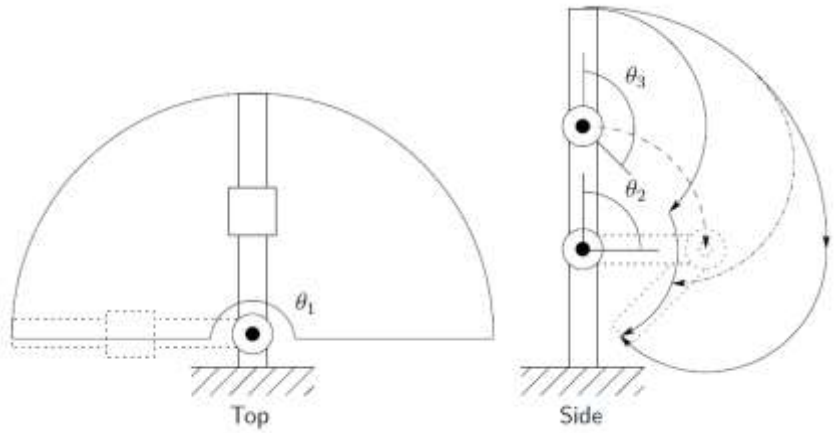


Figure 24: Working area for an articulated arm

The robotic arm will be mounted on top of the quadruped locomotive and to satisfy our requirement of the robotic arm being able to reach to objects on the ground we have to determine the height of the locomotive. Even though the quadruped locomotive is designed so that it is height adjustable the robotic arm must be able to rich the ground level when the locomotive is at its default height which is 10 cm from the axis of rotation at the shoulder joint in leg to the ground. The distance from the axis of rotation on the shoulder joint to the top of the body is 5 cm as it can be seen on the image below. The total height of the quadruped locomotive will be 15 cm from ground to the top of the body at default stance.

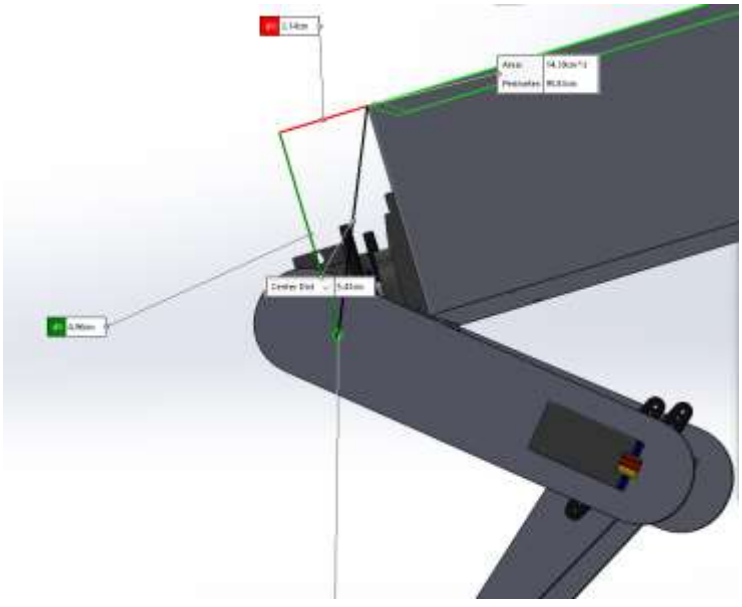


Figure 25: Shoulder joint to robot top distance

The robot dimensions can be divided in to three parts as it can be seen form figure y. The first being the dimension from the body to shoulder. In our case this dimension value is pre-defined by the type of servo motors we are using, that is 5cm from the base to the rotation axis of the shoulder joints. The rest of the robot length is to be determined by the requirement that the robot must be able to reach the ground level while in default stance. As it can be seen from the below figure the combined length for the rest of the robot has to be at list 20.5cm to suffice the requirement. We can round this value to 20cm considering there will be an end effector attached at the end point. Taking this value, we can proportionally divide this between the last two section to get a good working area and reachability. This value will be the distance between the rotation axis and the physical implementation will be larger for mechanical cohesion. The detailed used dimensions are shown in the images below even though the dimensions seem to be larger than the actual calculated values the distance between the rotation arises is still the same.

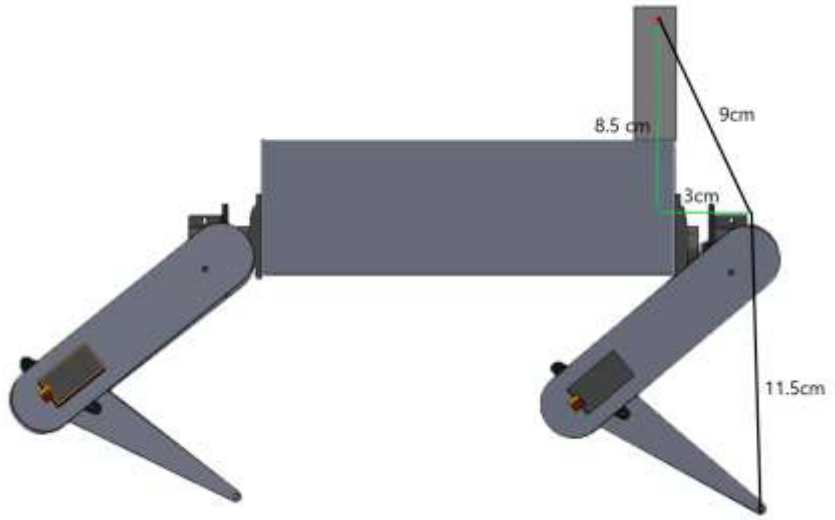


Figure 26: Arm length requirement

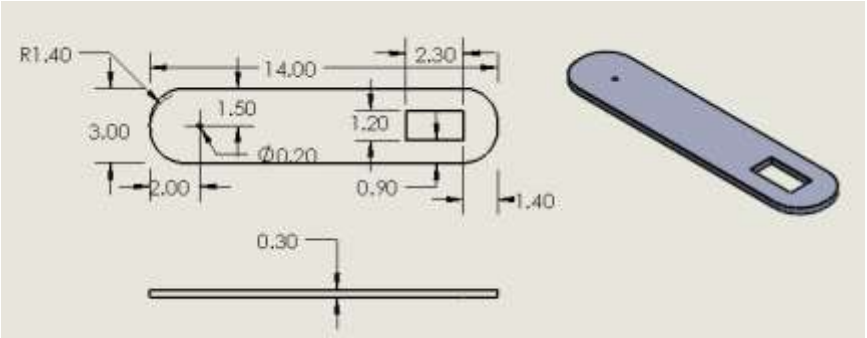


Figure 27: Lower arm section (all measurements are in centimeters)

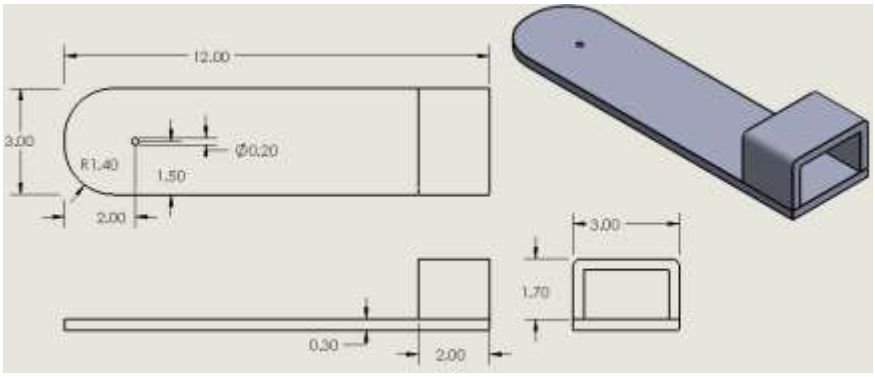


Figure 28: Upper arm section (all measurements are in centimeters)

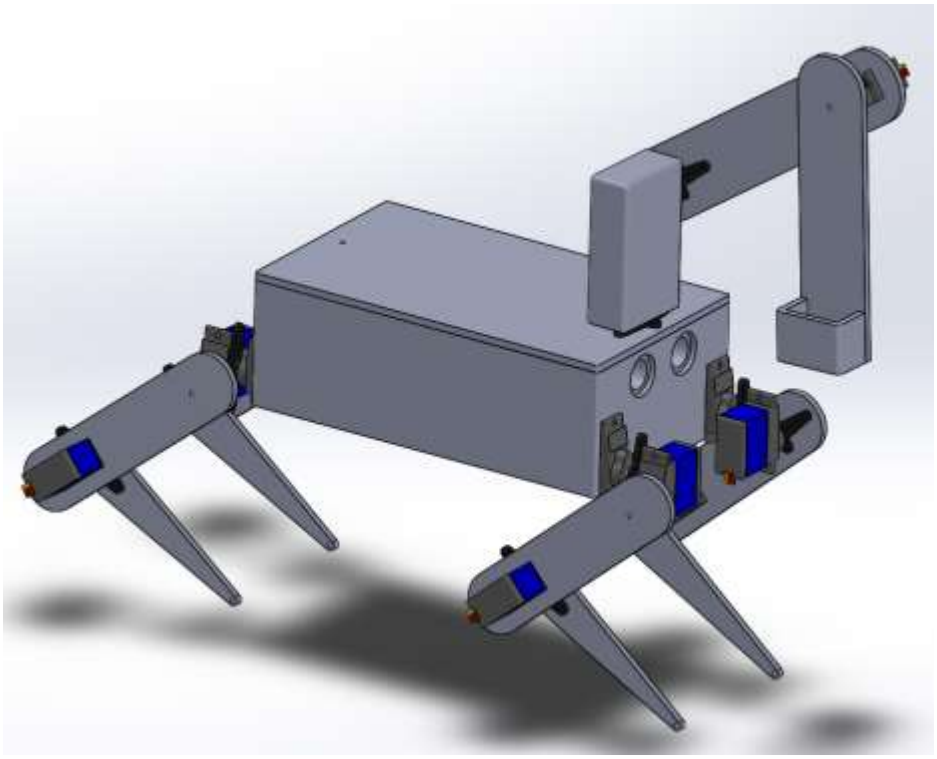


Figure 29: Full body build

Robot Kinematics

Forward Kinematics Actuate

Forward kinematics calculates the position and orientation of the end-effector with respect to the given reference frame, given the set of joint-link parameters. There are two different methods used, the first one is the Denavit-Hartenberg (D-H) parameter and the second one is successive screw displacements. D-H method and geometric methods are used in this paper for implementing forward kinematics.

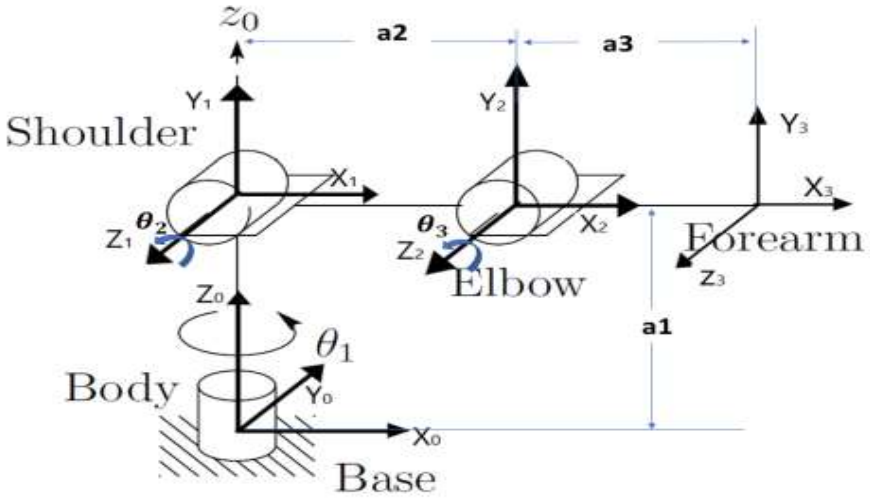


Figure 30: Articulated arm model

	θ	α	r	d
1	θ_1	90^0	0	a_1
2	θ_2	0^0	a_2	0
3	θ_3	0^0	a_3	0

Table 2: D-H parameter

$$H_1^0 = \begin{bmatrix} c\theta_1 & 0 & s\theta_1 & 0 \\ s\theta_1 & 0 & -c\theta_1 & 0 \\ 0 & 1 & 0 & a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_2^1 = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & a_2c\theta_2 \\ s\theta_2 & c\theta_2 & 0 & a_2s\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_3^2 = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & a_3c\theta_3 \\ s\theta_3 & c\theta_3 & 0 & a_3s\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_3^0 = H_1^0 H_2^1 H_3^2$$

$$H_3^0 = \begin{bmatrix} c\theta_1 c\theta_2 c\theta_3 - c\theta_1 s\theta_2 s\theta_3 & -c\theta_1 c\theta_2 s\theta_3 - c\theta_1 s\theta_2 c\theta_3 & s\theta_1 & a_3 c\theta_1 c\theta_2 c\theta_3 - a_3 c\theta_1 s\theta_2 s\theta_3 + a_2 c\theta_1 c\theta_2 \\ s\theta_1 c\theta_2 c\theta_3 - s\theta_1 s\theta_2 s\theta_3 & -s\theta_1 c\theta_2 s\theta_3 - s\theta_1 s\theta_2 c\theta_3 & c\theta_1 & a_3 s\theta_1 c\theta_2 c\theta_3 - a_3 s\theta_1 s\theta_2 s\theta_3 + a_2 s\theta_1 c\theta_2 \\ s\theta_2 c\theta_3 + c\theta_2 s\theta_3 & -s\theta_2 s\theta_3 + c\theta_2 c\theta_3 & 0 & a_3 s\theta_2 c\theta_3 + a_3 c\theta_2 s\theta_3 + a_2 s\theta_2 + a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Leg kinematics

Each of the four legs have 3 DOF using the three revolute joints. As it can be seen form the 3D models all the legs can be modeled as an articulated arm. The kinematic diagram that shows how the links and joints are connected together when all of the joint variables have an angle value of 0 is shown below.

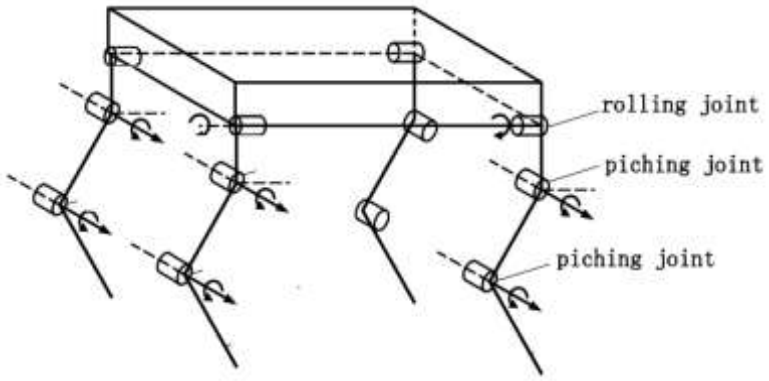


Figure 31: Kinematic modeling of the quadruped locomotive

As it can be seen from the above image the modeling for the four legs will be the same. Even though we can model all the legs as an articulated arm the Inverse kinematics for the front legs will be in the elbow down configuration and the back legs will be in the elbow up configuration. This is because the bending of the knee joint is in the positive direction of z axis in the back legs and negative direction of z axis in the front legs.

Inverse Kinematics is the procedure in which the joints are controlled in order to achieve the end position, given the position and orientation of the robotic arm. Solving Inverse kinematics is important compared to forward kinematics, as it can move the leg to the target position, which would be helpful in placing the leg at the target location. There are two different methods in solving inverse kinematics: geometrical and algebraic methods. As algebraic methods result in complicated solution as the Degree of Freedom (DOF) increases, we have used geometrical method in this paper.

The kinematic diagram of an articulated arm in elbow down configuration is shown below. The inverse kinematics is done using the graphical method and is as shown below.

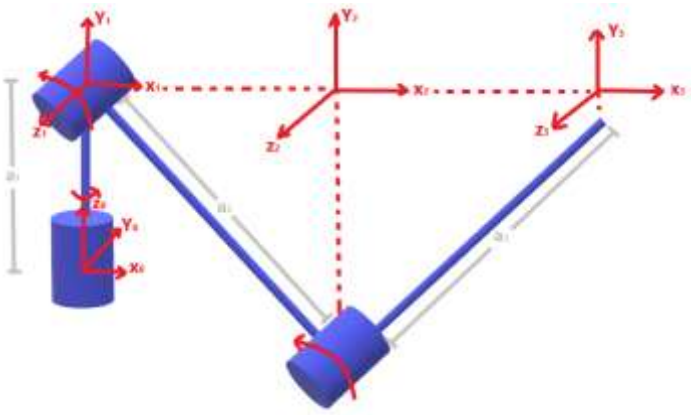


Figure 32: Kinematic model of the robot legs (for the front legs)

Elbow down

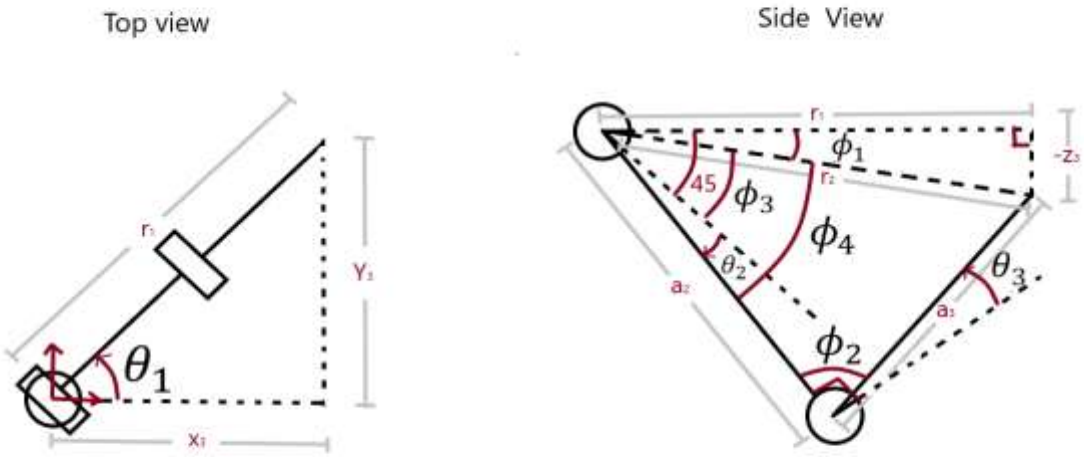


Figure 33: Top and side view for elbow down configuration

$$\theta_1 = \tan^{-1}\left(\frac{Y_3}{X_3}\right)$$

$$r_1 = \sqrt{x_3^2 + y_3^2}$$

$$\phi_1 = \tan^{-1}\left(\frac{-z_3}{r_1}\right)$$

$$r_2 = \sqrt{r_1^2 + z_3^2}$$

$$r_2^2 = a_2^2 + a_3^2 - 2a_2a_3\cos\phi_2$$

$$\phi_2 = \cos^{-1}\left(\frac{a_2^2 + a_3^2 - r_2^2}{2a_2a_3}\right)$$

$$\phi_2 + \theta_3 = 90^0$$

$$\theta_3 = 90^0 - \phi_2$$

$$a_3^2 = r_2^2 + a_2^2 - 2r_2a_2\cos\phi_4$$

$$\phi_4 = \cos^{-1}\left(\frac{r_2^2 + a_2^2 - a_3^2}{2r_2a_2}\right)$$

$$\phi_3 + \phi_1 = 45^0$$

$$\phi_3 = 45^0 - \phi_1$$

$$\phi_4 = \phi_3 + \theta_2$$

$$\theta_2 = \phi_4 - \phi_3$$

Elbow Up

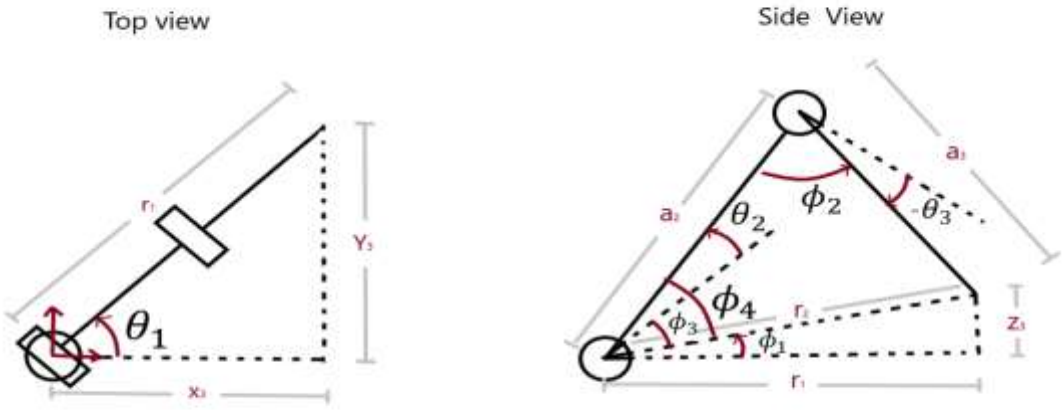


Figure 34: Top and side view for elbow up configuration

$$\theta_1 = \tan^{-1}\left(\frac{Y_3}{X_3}\right)$$

$$r_1 = \sqrt{x_3^2 + y_3^2}$$

$$\phi_1 = \tan^{-1}\left(\frac{z_3}{r_1}\right)$$

$$r_2 = \sqrt{r_1^2 + z_3^2}$$

$$r_2^2 = a_2^2 + a_3^2 - 2a_2a_3\cos\phi_2$$

$$\phi_2 = \cos^{-1}\left(\frac{a_2^2 + a_3^2 - r_2^2}{2a_2a_3}\right)$$

$$\phi_2 - \theta_3 = 90^0$$

$$\theta_3 = \phi_2 - 90^0$$

$$a_3^2 = r_2^2 + a_2^2 - 2r_2a_2\cos\phi_4$$

$$\phi_4 = \cos^{-1}\left(\frac{r_2^2 + a_2^2 - a_3^2}{2r_2a_2}\right)$$

$$\phi_3 + \phi_1 = 45^0$$

$$\phi_3 = 45^0 - \phi_1$$

$$\phi_4 = \phi_3 + \theta_2$$

$$\theta_2 = \phi_4 - \phi_3$$

Joint Driving torque

Torque is defined as a turning or twisting "force" and is calculated using the following relation $T = F * L$ The force (F) acts at a length (L) form a pivot point. In a vertical plane, the force acting on an object (causing it to fall) is the acceleration due to gravity (g = 9.81m/s^2) multipled by its mass: $F = m * g$ The force above is also considered the object's weight (w). $W = m * g$ The torque required to hold a mass at a given distance form a pivot is there fore: $T = (m * g) * L$ This can be found similarly by doing a torque balance about a point. Note that the length L is the perpendicular length form the pivot to the force[3].

Quadruped locomotive

In the early development stage of a quadruped robot, the joint driving forces of all joints are significant references to design the mechanism and actuators of the robot. They need not have high precision, so they can be computed from simplified but not complicated models.

We will be using the crawling gait. But for joint torque forces the trotting gait will be considered. This is because in trotting gait only 2 legs are in stance phase and the torque calculated will be sufficient for crawling gait which has at any time at least 3 legs in contact with ground.

While a quadruped robot runs with trotting gait, one pair of the diagonal feet are in stance phase and other pair of diagonal feet are in swing phase and other pair of diagonal feet are in swing phase simultaneously. Under the assumption that there is no slippage between the supporting feet and ground, it can be considered that the supporting feet and ground are connected with hinge joints [1]. So, the motions in sagittal plane of the pair of diagonal legs in stance phase and the torso are equivalent to a five-bar mechanism as shown in the figure below.

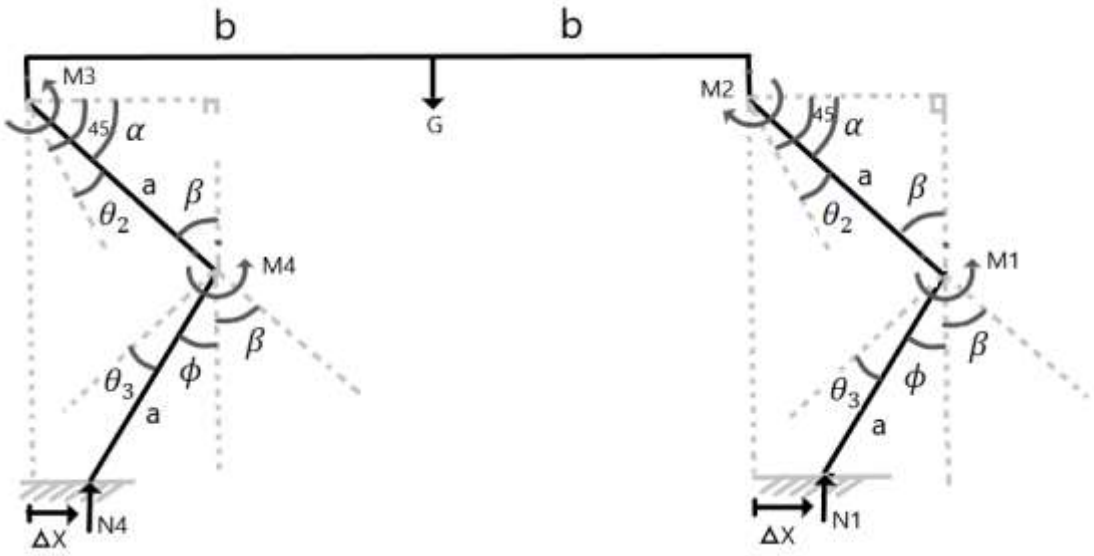


Figure 35: Torque distribution for quadruped locomotive

Four assumptions listed as below are defined to simplify the computing process of joint driving forces. There is no slippage between the supporting feet and ground. The quadruped robot is running with trotting gait. The quadruped robot is running straight at a constant velocity, so it can be considered that the reaction forces from ground to the supporting feet of the robot only have exponents along vertical direction. The weight of all components of the quadruped robot is concentrated to the torso. All links of legs are mass less [1].

For the entire quadruped robot simplified to a five-bar mechanism, we can get a force and torque equilibrium equations as below

$$N_1 + N_4 = G$$

$$(b + \Delta x)N_1 - (b - \Delta x)N_4 = 0$$

where, *G* presents the gravity of the entire robot concentrated to torso, *b* presents half of the torso length, a represents length of the leg links, N1 and N4 present the reaction forces from ground to the fore and rear supporting feet respectively.

From equations above we can get the reaction forces N1 and N4 as shown below

$$N_1 = \{(b - \Delta x)G\}/(2b)$$

$$N_4 = \{(b + \Delta x)G\}/(2b)$$

The reaction forces N1 and N4 can be migrated to the torso as two new forces N2 and N3.

$$N_2 = N_1$$

$$N_3 = N_4$$

$$M_2 = N_1\Delta x = \{(b - \Delta x)\Delta xG\}/(2b)$$

$$M_3 = N_4\Delta x = \{(b + \Delta x)\Delta xG\}/(2b)$$

$$\alpha = 45^0 - \theta_2$$

$$\beta = 90^0 - \alpha$$

$$\phi = 90^0 - \beta - \theta_3$$

$$M_1 = N_1 a \sin(\phi)$$

$$M_4 = N_4 a \sin(\phi)$$

Thus, the needed driving torques for four rotary joints of two supporting legs is obtained. The torque requirement of motors for other legs will remain the same.

Joint torque calculation

In the previous subsection we derived the equations for calculating the torque required at each joint. These torque equations are same for all the legs. Using these equations and adding the mass of individual components we can calculate the torque at each joint. Once the torque is known
we would be able to select from the range of servo motors available. Let's start from the total weight of the robot G. The total mass of the robot is around 750 grams so the weight will be at 7.35 N. Our goal when designing the robot was to achieve a stepping distance of 5cm so will have $\Delta x = 5\text{cm}$. To find the values of $\theta_1 \; and \; \theta_2$ we have to substitute the cartesian coordinates of the legs end position in the equation that we found for the inverse kinematics. Our default height of robot is at 10cm as determined in the design section then the cartesian coordinates for the angle's calculation will be $(10,0,5)$.

$$G = 7.35N$$

$$b = 10cm$$

$$a = 7cm$$

$$N_1 = 1.84N$$

$$N_4 = 5.52N$$

$$M_1 = 0.023Nm$$

$$M_2 = 0.092Nm$$

$$M_3 = 0.276Nm$$

$$M_4 = 0.07Nm$$

Maximum torque required = $0.276Nm$ = $2.8kg/cm$ Now that we have the maximum torque requirement of joints, we can select the servo motor.

Different types of RC servo motors are available for working on robots. These servo motors are classified according to their construction and working. Based on working these are classified into analog and digital servos. The digital servos are precise than analog. Based on the construction these are classified into servo with plastic gear and servo with metal gear. From above calculations, the maximum torque requirement is $2.8kg/cm$. With this torque figure we have selected the Tower pro MG92b High-Torque Servo [4]. This motor has metal gears for increased torque handling.

The key specifications are Mentioned in the table below.

Servo motor specification	
Size	Length: 22.8mm, Width: 12.0mm, Height: 31.0mm
Weight	13.8g
Type	Digital
Max Voltage	6.6V
Stall torque at 6V	5.0v: 3.10kg-cm, 6.0v: 3.5010kg-cm
Speed	5.0v: 0.13sec/60, 6.0v: 0.08 sec/60

Table 3: Servo motor specification

Robotic Arm

The robotic arm that is mounted on the robot body is to be used for manipulating objects and lifting objects of light weight. Taking the size and weight of our robot in to consideration the robot is design to lift at least 500 grams (half a kilogram) of weight that is 5 cm apart from the robot arm base, this distance is in the horizontal direction perpendicular to the weight of the object. Since the robot is mobile it can approach any object close enough to suffice this criterion. This will reduce the cost of the servo motor to be used at the joints relative to a fixed robot arm with no mobility where it needs to extend the arm to reach objects.

As we did for the quadruped locomotive calculation, we have made the assumption that the joint and links of the robot arm are mass less. This is done because the weight of object to be lifted is significantly larger than the weights of the links and joint (servo motors). This assumption will simplify the calculation and reduce unnecessary complexity. Due to the distance from the object to be lifted the largest torque requirement is at the shoulder pitching joint as it can be seen from figure x. Using this assumption, we can calculate the amount of torque that is required at the shoulder joint as follows.

$$W = m * g$$

$$W = 4.9N$$

$$T = F * d$$

$$T = 0.24525Nm$$

Maximum torque required is $0.245Nm = 2.5kg/cm$ meaning that the same servo motor that is used for the quadruped locomotive is also applicable in the robotic arm.

Planning the gait

We have modeled and derived the kinematic equations. Using the kinematic equation each leg can be controlled to a desired location relative to the robot body. And by using the inverse kinematic equation we can position each leg by giving the cartesian coordinates, which the Inverse kinematic equation turns it to angle values for the servo motors. At each point of the robot body is kept in balance by three legs on ground, making it very stable when moving robot. Robot motion with gait is two-phase discontinuous [2]. The body is propelled forward/backward with three of the feet securely placed on the ground and a leg is transferred with all other three legs and body halted. For every phase of robot body will propel forward once.

Walking straight gait

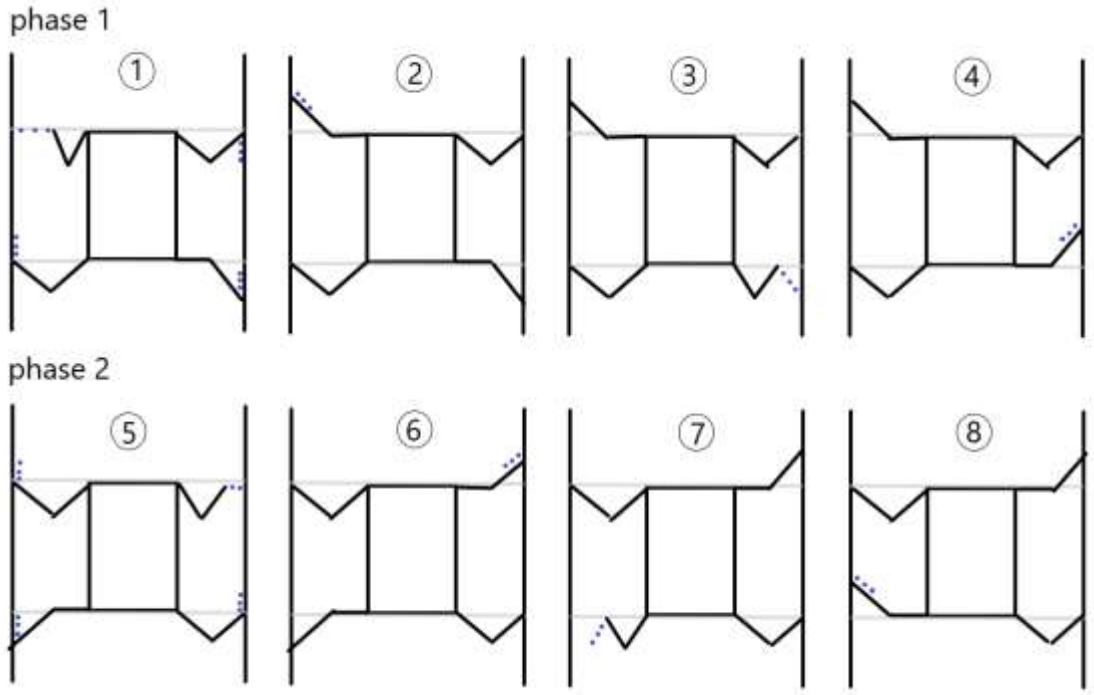


Figure 36: Walking straight gait pattern

Implementing the above gait pattern on an algorithm the robot can be directed to move forward or backward if both the left side legs and right-side legs stepping distance is the same. When developing an algorithm for the above gait pattern it is possible to control the stepping distances for the left side leg and right-side legs independently. That is using different stepping distance values on the right and left side legs we can use this gait pattern not only for directing the robot in the forward and backward direction but also to direct it to turn to the right direction or the left direction.

By making the left side leg stepping distance value smaller or even negative relative to the right-side leg stepping distance it is possible to make the robot turn to the left direction. And by making the right-side leg stepping distance value smaller or even negative relative to the left side leg stepping distance it is possible to make the robot turn to the right direction. So, it is possible to use this gait pattern for turning the robot or we can develop independent one with more control for the turning angle. In the next section we will develop an independent gait pattern for turning the robot with more control to turning angle.

The below figure shows how the independent turning gait works. Using this gait pattern, we can control the amount of angle the robot is to turn. The red dot shows the initial position of the leg foot print and the light blue dot shows the final position of the leg foot print after a turn is made. The value of θ is the amount of angle the the robot is to turn. The amount of distance between the front legs is represented by "l" and the distance between the front legs and back legs is represented by "w".

Using basic geomery and trigonometry we can calculate the position of the light blue dot (leg foot print after a turn is made) from the given θ value and the dimentions of robot relative to red dot (leg foot print before a turn is made). As it is shown in the diagram below the sequence of the legs turning is first from the fornt-left leg then to the back-right leg then the front-right leg and finally the back-left leg.

This sequencing is choosen to keep the robot balanced while turning although this is not the only sequencing possible. The equations for calculating the position of the light blue dots relative to the red dots is shown below.

Turning gait

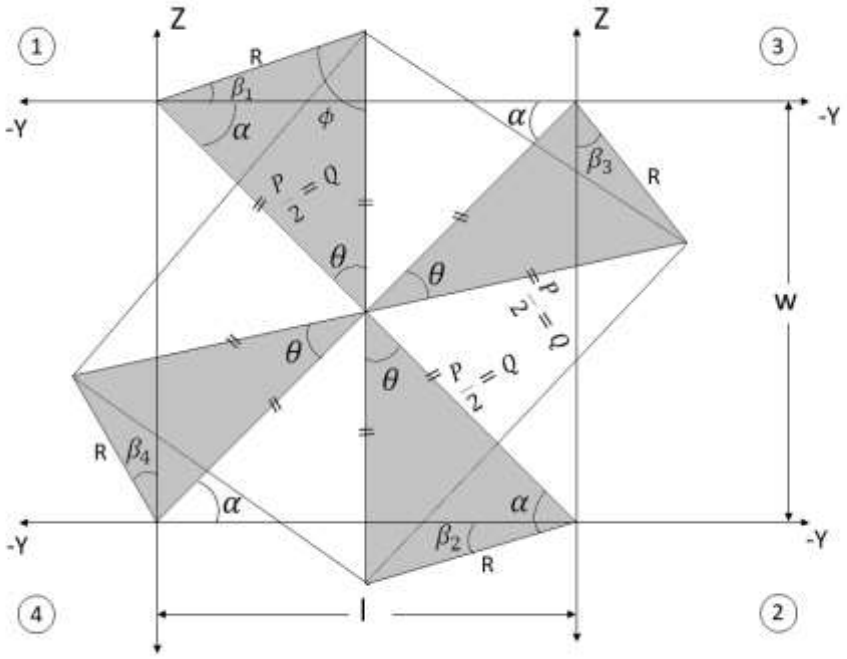


Figure 37: Rotation on a fixed axis gait pattern

$$Q = \frac{P}{2}$$

Where: P - is the diagonal length of the robot

Q – length from the center of the robot body to the shoulder joints of each leg

$$P = \sqrt{l^2 + \omega^2}$$

$$\beta_1 = \phi - \alpha$$

$$\beta_3 = \phi - (90 - \alpha)$$

$$\beta_2 = \phi - \alpha = \beta_1$$

$$\beta_4 = \phi - (90 - \alpha)$$

$$Z_1 = +R\sin(\beta_1)$$

$$Y_1 = +R\cos(\beta_1)$$

$$Z_3 = -R\cos(\beta_3)$$

$$Y_3 = -R\sin(\beta_3)$$

$$Z_2 = +R\sin(\beta_2)$$
$$Y_2 = +R\cos(\beta_2)$$
$$Z_4 = -R\cos(\beta_4)$$
$$Y_4 = -R\sin(\beta_4)$$

Controller

In order to control the robot, we have chosen to develop an android application that will run on a smart phone. We have chosen to use a smart phone application because of their touch controls. A touch screen can have many interfaces depending on our needs and this makes it easy to change the robot settings. We have developed the application using an app development website called mitappinventor. The method of communication is through Bluetooth.

The controller is designed so that it would be simple and easy to use. Many of us are familiar with game controller (joy sticks) layout so we have based our designs from them. Anyone who have previously had some experience with game controllers will easily learn to control the robot.

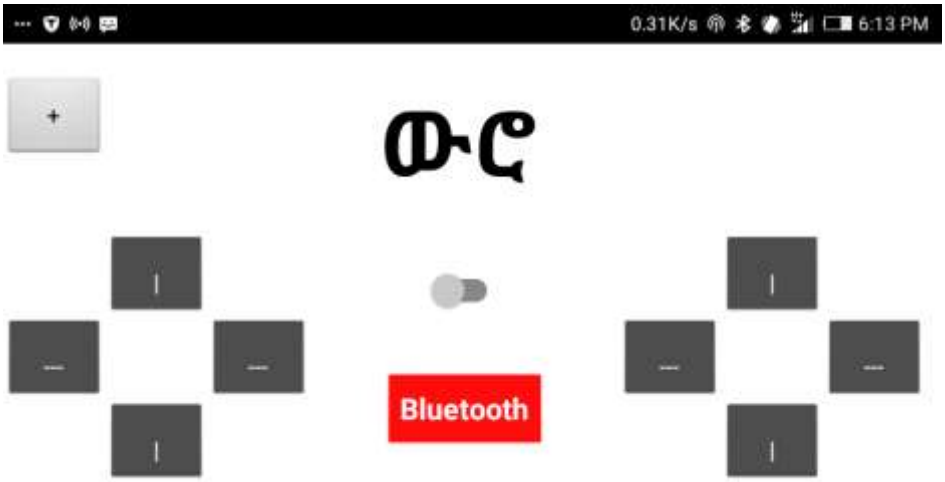


Figure 38: Android application for controlling robot (home screen)

From the above image we can see that we have two button configurations one is on the right and one is on the left, formatted on a cross pattern, just as it is found on game controllers. To make the controlling of the robot easier we have made two settings in controlling the robot. The first setting, which is activated

by default it to control the quadruped locomotive that is moving the robot body and the second setting, which is activated by toggling the switch in the middle to the right is to control the robotic arm.

The first setting can be activated by toggling the switch on the middle to the left. This mode is also on by default. In this mode the robot body can be directed to move forward, backward, rotate right, rotate left. Although this are the function that we have programmed, the robot can be programed to move in different ways, like sliding to the right and left. The specific buttons to control the robot are very intuitive. For the forward motion the top button on the left button configuration is used. For the backward motion the button on the bottom of the left button configuration is used. For left rotation the button on the left side of the left button configuration is used. For the right rotation the button on the right side of the left button configuration is used. The button configuration on the right are reserved for future robot motion like sliding to the right and left.

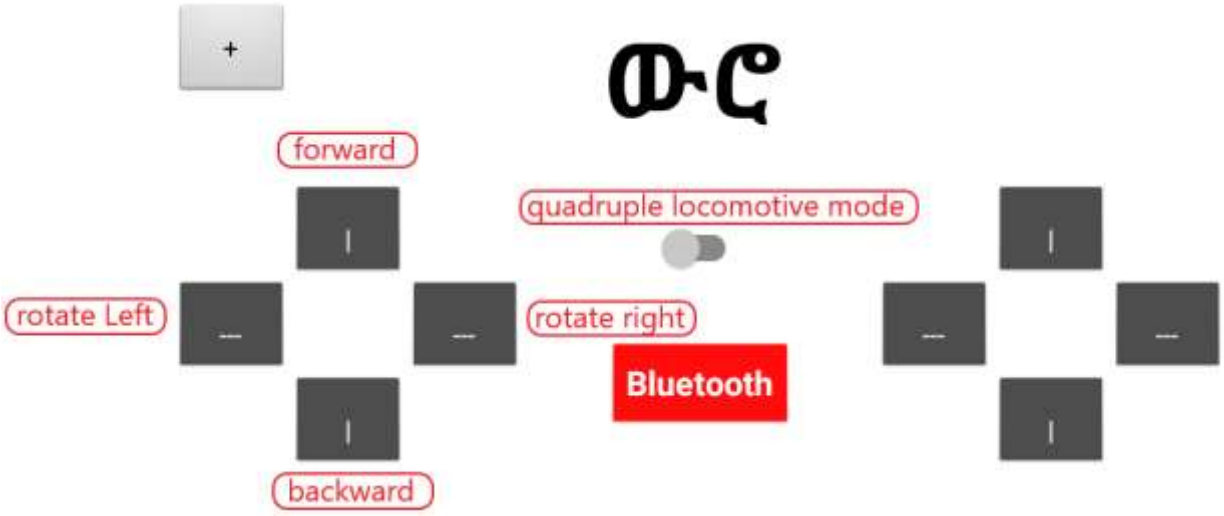


Figure 39: Android application quadruple locomotive controls

The second setting which is activated by switching the middle switch to the right is used to control the robot arm. Again, controlling the robotic arm is very intuitive. The robotic arm is controlled by increasing and decreasing the vales in each axis. The button configuration on the left are used to change the values in the x-y plane.

The top button on the left configuration is used increase the x value and the bottom button to decrease the x value. The left button on the left configuration is used to increase the y value and the right button on the left configuration is used to decrease the y value. The top button on the right configuration is used

to increase the z value and the bottom button is used to decrease the z value. The left button on the right configuration is used to open the end effector and the right button to close it.

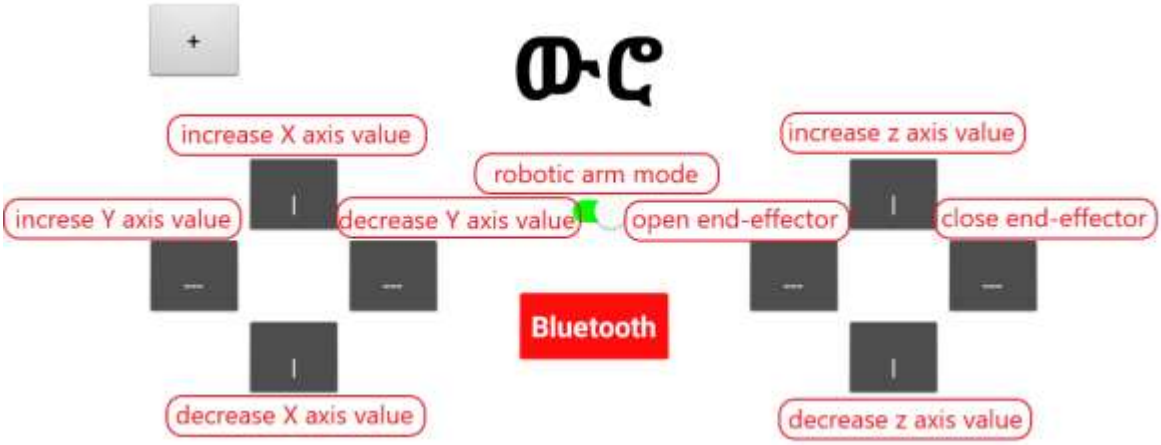


Figure 40: Android application robotic arm controls

For the basic functioning of the robot the above-mentioned controls are enough but for advanced users we have made it so that the different parameters in the walking algorithm can be changed at real time form the controller. This advanced controls can be accessed by pressing on the plus button on the top left corner of the home screen. The algorithms running on the robot are very generic, meaning we have built them so that the different parameters of walking can be changed. The parameters that could be changed in the advanced setting are Delay, resolution, stepping distance for left legs, stepping distance for right legs, height, leg lift, Angle of rotation. See Appendix for details on each parameter. The parameters can be set for the individual motion by selecting the motion desired.

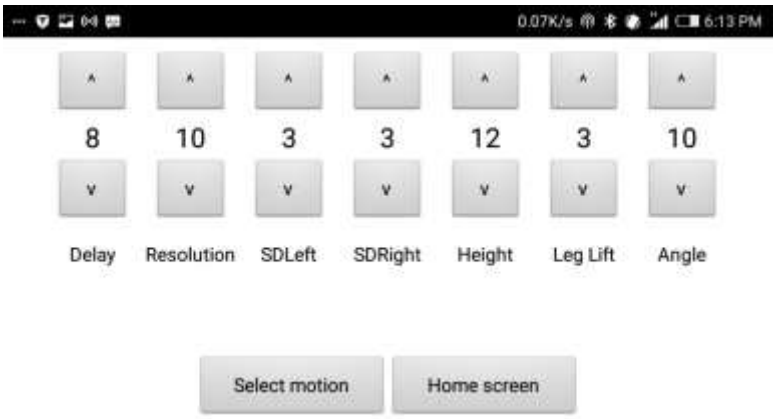


Figure 41: Advanced Setting interface

Figure 42: Select motion interface

Construction

On the construction of the robot body we have recycled a broken plastic table. Recycling the plastic table will be good for the environment and also eliminate the cost for 3D printing. We have molded the plastic using a soldering iron. Using low grade sanding the contact points are made smooth and clean. Most of the servo motors are bolted to the body and some are connected using a hot glue. To increase friction on the contact points we have used old sandals material on the legs and the end effector gripper.

Figure 43: Construction pieces

Figure 44: Recycled Plastic for construction

Figure 45: Final construction of Robot (Wero)

Power consumption

The power consumption of the robot depends on many factors.

- The efficiency of the running algorithms
- The difficulty of the train its walking
- Loading
- Speed and also different factors

The robot runs on two voltage levels, 5V for the microcontroller, sensors (Ultrasonic, IMU, Bluetooth) and 6V for the servo motors. The current draw can run up to 3.4A on maximum loading.

CHAPTER FIVE

Result and Conclusion

The main aim for this project was to build a cheap and affordable general-purpose robot for research and development. We have started the project by designing the robot body and different configuration on a simulation software. The kinematic model for the robot was developed. The forward and inverse kinematics for the robotic arm and the quadruped locomotive has been derived and implemented in code. The torque calculation has been done in order to choose appropriate servo motors. A walking gait for the robot has been derived using static stability and the appropriate generic algorithm for moving forward and backward has been implemented. A turning gait has been derived and the proper calculations has been done, and the corresponding turning algorithm for both the right and left direction has been implemented in code. An android application for controlling the robot has been developed. In developing the robot, we have made it to be as generic as possible. This is done to encourage further development and improvements for the robot. All the algorithms are developed so that every single parameter for the algorithm can be changed. The end effector is designed to be modular to encourage new end effector designs.

The robot body was made using reusable old plastic table. We have tried to build all the parts as precisely as we possibly can but building the body manually has it side effects. All the dimensions were not exactly the same as the 3D model on solid works. This was shown when we executed the walking algorithms, the robot does not walk straight and has some deviation to the side. This is attributed to the imprecision build of material and also the suboptimal choice of materials like the friction reduction material on the legs. We have tried to mitigate these errors in code. We have also seen the imprecision build errors express them self's in the rotation algorithms, even though it was hard to quantify the error values, it was fairly obvious to see.

This thesis has proved that it is indeed possible to develop and do research on quadruped robots very cheaply. All the materials used in this project are easily accessible in the market and can be constructed following our designs.

Using this project, it is possible to teach students the basics of robotics in practice. Student can develop their own algorithms and applications for the robot. We believe this will encourage student to do more robotics research.

CHAPTER SIX

Future Works

The results of this work open variety of interesting areas and the quadruped robot can be further developed in the following aspects.

- Improvement of the body design

The existing body design have some limitations due to the lack of an access to a 3D printer. Using a 3D printer will give more design freedom and a better precision can be achieved.

- Improvement of the robot leg structure

The servo motors on the elbow joints can be designed to be closer to the shoulder joints. This will reduce the amount of torque necessary to drive the legs which translates to a faster actuation.

- Improvement of the robotic arm

The existing robotic arm has only 3 Degree of freedom, which only gives as control to the position of the end effector. The robotic arm could be designed to have 6 degrees of freedom which give as control to the position and also the orientation of the end effector.

- Machine learning

Artificial intelligence is one of the cutting-edge technologies used in robotics. Using this technology in the robot will reduce the amount of control needed from a human operator.

- Selection of better actuator and sensor

A more expensive actuator and sensor will increase the robot payload capacity and also help in increasing the size of the robot. Adding force sensors will help in devising dynamic balancing of the robot and better walking algorithms.

- Control over the internet and camera on the robot

Applying a control over the internet will give as the ability to control the robot from anywhere using an internet connection. A camera system will help in using the robot as a surveillance system and also vision processing can be done from the images.

References

[1] Ganesh Kale, Sanjay Gandhe, Pravin Dhulekar, Sushant Pawar, “Designing a Quadruped with Stair Parameter Analyzing and Climbing Capability”, 2015 International Conference on Computing Communication Control and Automation Nasik, India

[2] Than Trong Khanh Dat and 1Tran Thien Phuc,” A Study on Locomotions of Quadruped Robot”, Faculty of Mechanical Engineering, Ho Chi Minh City University of Technology, Vietnam

[3] Robot arm torque tutorial, “https://www.robotshop.com/community/tutorials/show/robot-arm-torque-tutorial”, (accessed on august 20 2020)

[4] Tower pro MG92b, “https://www.towerpro.com.tw/product/mg92b/”, (accessed on august 20 2020)

[5] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar, “Robot Dynamics and Control”, Second Edition January 28, 2004

APPENDICES

Appendix A ON THE CODE LIBRARY METHODS

In this section we will explain the important methods that are implemented in code. The explanation will be at a conceptual and design level in order for users to use this method and experiment with them.

writer():

This method is called to apply the changes made to angle values. This will write the angles calculated to the actual servo motors.

arm(x,y,z):

This method is used to calculate the inverse kinematics for the robotic arm. The method accepts cartesian coordinates as an argument. Make sure to call the writer(); method after using this method.

frontRight(x,y,z):

This method is used to control the front right leg independently. The method accepts cartesian coordinate as an argument. Make sure to call the writer(); method after using this method.

frontLeft(x,y,z):

This method is used to control the front left leg independently. The method accepts cartesian coordinate as an argument. Make sure to call the writer(); method after using this method.

backLeft (x,y,z):

This method is used to control the back-left leg independently. The method accepts cartesian coordinate as an argument. Make sure to call the writer(); method after using this method.

backRight (x,y,z):

This method is used to control the back-left leg independently. The method accepts cartesian coordinate as an argument. Make sure to call the writer(); method after using this method.

elbowDownInverseKinematics(x,y,z):

This method is the code implementation of the inverse kinematics calculations done for the elbow down configuration on the kinematic section. The method accepts cartesian coordinates as an argument.

elbowUpInverseKinematics(x,y,z):

This method is the code implementation of the inverse kinematics calculations done for the elbow up configuration on the kinematic section. The method accepts cartesian coordinates as an argument.

setTurn(angle):

This method is used to set the angle of rotation for the rotate right and rotate left functions. The method accepts angle value in degrees as an argument.

turnRight(delayer,resolution,height,legLift):

This method is the code implementation for calculations made on the rotation gait on the above section. The method takes multiple arguments.

The robot motion is composed of small discreate motions, which we called resolutions. The more resolution the motion has the smother the locomotion but at the same time the motion becomes slow. If we set the resolution to 10 it means to perform a single leg motion it will make 10 smaller steps. After every single resolution step, we have a small delay. This delay value is the waiting time after a single resolution step which is measured in milliseconds. The height is the distance for the leg shoulder joint to ground measured in centimeters. Leg lift is the distance the leg will raise above the ground when in motion measured in centimeters. All these parameters can be set for the turning algorithm as arguments.

turnLeft(delayer,resolution,height,legLift):

This method is the same as the trunRight method only the rotation is in the right direction.

walkForward(delayer, resolution, stepDistanceLeft, stepDistanceRight, height, legLift):

This method is the code implementation of the walking gait derived in the previous sections for walking forward. As the turning method the walking methods are developed to be as generic as possible. The method will take multiple arguments to set the different parameters of the walking algorithm. The arguments delayer, resolution, height, legLift will have the same function as described for the turning algorithms.

StepDistanceLeft is the amount of length a single step of the left legs will move. StepDistanceRight is the amount of length a single step of the right legs will move. The right and left legs stepping distances are differentiated this way for two reasons. The first one is to correct some errors that occur at the physical construction of the robot parts, since we did not use 3D printers there will be some discrepancies in the measurements of parts.

The second reason is we can use different stepping distance on the right and left legs to make the robot turn. Notice that the rotation function will turn the robot maintain its current location.

walkBackward(delayer, resolution, stepDistanceLeft, stepDistanceRight, height, legLift);

This method is the same as the walkForward method but in the backward direction.

User guide Common problems

There may be some vibrations on robot due to some ac voltage leaks, due to this the screws may get lose so please try to tighten the screws on a regular basis.

If you are having problems with the Bluetooth communication

- Try resetting the Arduino bord using onboard button
- Change the 9v battery on the robot used to power the Arduino