A Project Stage-II Report on

Asteroid – The Humanoid Robot

By

Mr. Suhrud Parag Joglekar

Mr. Varad Vishnu Gole

Mr. Atharva Prasanna Sambhus

Guide

Prof. Dr. Manmohan M. Bhoomkar

Department of Mechanical Engineering

PVG's college of engineering & technology

[2021-22]

PVG's COLLEGE OF ENGINEERING & TECHNOLOGY

C E R T I F I C A T E

This is to certify that **Mr. Joglekar Suhrud Parag, Mr. Gole Varad Vishnu, Mr. Sambhus Atharva Prasanna,** has successfully completed the Project Stage – II entitled **"Asteroid – The Humanoid Robot"** under my supervision, in the partial fulfillment of Bachelor of Engineering - MechanicalEngineering of University of Pune.

Date: 19/05/2022

Place: Pune

Prof. Dr. Manmohan M. Bhoomkar

Guide **Internal Examiner**

Prof. Dr. Manmohan M. Bhoomkar

Dr. M.R. Tarambale

Head of Department **I/C Principal**

External Examiner

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got all of this while working on the project. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

We owe our deep gratitude to our project guide **Prof. Dr. Manmohan M. Bhoomkar** who took keen interest on our project work and guided by providing all the necessary information for developing a good mechanical design.

We would further like to extend our gratitude towards **PI Lab** who provided us with all the resources for executing the steps needed for the project. We would also like to thank our **Prof. Dr. Manmohan M. Bhoomkar**, Head of the Department for giving us the opportunity to work on this project in PI Lab.

We are also thankful to all teaching and non-teaching staff members of Mechanical Engineering of PVG's College of Engineering and Technology, Pune for their valuable time, support, comments, suggestions and persuasions.

LIST OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Humanoid robot has wide range of applications. It can be used for medical research purpose for building various types of prosthesis and also studying various human joints and motions. It also has applications in fields like disasters management, mine working, human assistant in various professions and also as emotional support.

In this project, we plan to design and build a low-cost humanoid robot which will be able to perform basic human movements like walking, self-balancing, pick and dropping. We are designing parts using SolidWorks platform. After designing joints and various parts, we are going to carry out simulations and analysis using software like MATLAB. Manufacturing of the robot will be done using 3D printer available in our lab. Assembling the parts along with all electronic components will be done. Which will help to detect and pick objects, avoid obstacles and communicate with people.

At the end of the project we will have at least 19DOF's humanoid robot designed with smooth and dexterous motions. This humanoid will have an ability to sense its environment and interact with it. It will be useful in various applications.

1. INTRODUCTION

Robotics is an emerging field of science in which robots are fabricated and programmed to do varied tasks. In those robots, Humanoid robots are the well sophisticated robots that perform tasks similar to humans and interact with them.

Humanoid robot is a robot with its body shape built to resemble that of the human body. In general, humanoid robots have a torso, a head, two arms, and two legs. Some humanoid robots may also have heads designed to replicate human facial features such as eyes and mouth.

A humanoid design might be for functional purposes, such as interacting with human tools and environments, for experimental purposes, such as the study of bipedal locomotion, or for other purposes. The attempt to the simulation of the human body leads to a better understanding of it to develop computational models of humanbehavior we learn Human cognition, a field of study which is focused on how humanslearn from sensory information in order to acquire perceptual and motor skills. The critical phase in the development of Humanoid robots is to design it effectively and anthropomorphic.

1.1 Problem Statement

The main motive is to design and build a low-cost humanoid robot and to achieve basic movements like walking, balancing and pick and dropping which are similar to humans along with making it completely mobile and wireless.

1.2 Objective

The main objective of the project is to build humanoid robot and perform various motions like walking, balancing and pick and dropping. This task can be divided into following sub objectives.

1. Study of human gait, motion and current humanoids.
2. Fixing Degrees of Freedoms and their locations.
3. CAD modelling of various parts and joints.
4. Kinematic and Structural analysis.
5. Motion Generation
6. Electronic component selection and Circuit designing
6. Interfacing and coding framework
7. Manufacturing by FDM 3D Printer
8. To enable Asteroid to perform various human activities like:
    i. Walking
    ii. Balancing
    iii. Pick and dropping

1.3 Scope

1. Build Humanoid robot to fulfil mentioned objectives and its testing and calibration of the scope of the project.

2. Design and generation of effective Walking Gait

3. Path generation for various other movements

4. Designing and building reliable servo controller frame work.

1.4 Methodology

Phase 1: Study and Research

We studied following things:

1. Human joints and Gait.
2. Various existing humanoid robots.
3. Kinematics and Dynamics
4. Various Actuators and sensors.
5. Interfacing and Coding Frameworks

Phase 2: CAD modelling and Designing

Then we decided necessary degrees of freedom required for our objective. We also decided dimensions of various parts such that it would look like human. After deciding all dimensions and joint we modelled all parts in CAD software.

Phase 3: Simulation

We performed Kinematic analysis (Forward Kinematic analysis and Inverse Kinematics) on end effectors to validate functionality of design. We also performed structural simulation to confirm validity of design.

Phase 4: control scheme and Manufacturing

Then we built interfacing framework for actuator control and use C++ for coding. At last we used FDM 3D Printing for manufacturing various parts and Joints.

2. LITERATURE REVIEW

2.1 Study of Human and Human Gait

Human Body

We studied Human Body as It is what we are expecting to mimic. The human body is the structure of a human being. The human body has four limbs (two arms and two legs), a head and a neck which connect to the torso. The body's shape is determined by a strong skeleton made of bone and cartilage, surrounded by fat, muscle, connective tissue, organs, and other structures [1]. Which inspired our 2-stage structure of humanoid i.e. Stronger Skeleton and casing which will consist of our organs (electronic Power and controlling units).

Human body has 244 degrees of freedom. Basically, free motion is work of joints and ligaments present in the body. The number of joints is 230. Many of them having single DOF and some have more than one [2]. Studding this We decided necessary DOFs of human which are necessary to perform desired task and included them in our Humanoid.

Human Gait

Human gait refers to locomotion achieved through the movement of human limbs. Human gait is defined as bipedal, biphasic forward propulsion of center of gravity of the human body. Different gait patterns are characterized by differences in limb-movement patterns, overall velocity, forces and changes in the contact with the surface (ground, floor, etc.) [3][4].

Types of Human Gait

Human gaits are the various ways in which a human can move, either naturally or as a result of specialized training. The so-called natural gaits, in increasing order of speed, are the walk, jog, skip, run, and sprint. All-natural gaits are designed to propel a person

forward, but can also be adapted for lateral movement. Walk is most important gait in our project point of view.

Walk

Walking involves having at least one foot in contact with the ground at all times. There is also a period of time within the gait cycle where both feet are simultaneously in contact with the ground. When a foot is lifted off the ground, that limb is in the swing phase of gait. When a foot is in contact with the ground, that limb is in the stance phase of gait. A mature walking pattern is characterized by approximately 60% of the gait cycle being the stance phase of gait while 40% in swing phase.

double limb stance

when both feet are in contact with the ground and the center of mass is within a person's base of support polygon i.e. region formed by enclosing all the contact points between the feet and the ground by using a convex shape.

Single Limb Stance

only one foot is in contact with the ground and the center of mass is in front of that foot and moving towards the leg that is in the swing phase.

As single Limb Stance end Center of mass travels from one limb to another limb in Double Limb Stance. Then Single Limb Stance of other limb Starts and this cycle continues.

2.2 Study of Humanoid Robots

This chapter is overview of the various humanoid robot we studied, Humanoid robot still a new technology with many challenges. Only few humanoid robots are commercially available, often at high cost. This is a very important aspect; researches had been done in the field of Robotics and AI for humanoid until recent years. We also study the human structure and behaviour to build humanoid robot. The realistic looks of humanoid robot are all inspired from the human body structure and having it's of functional purpose. The study of humanoid robot includes the many of humanoid robot which helps us in this project [5].

Leonardo da Vinci designs a humanoid automaton that looks like an armoured knight knows as Leonardo's robot (1495) ,there are many such human like robots design automation mentioned in the history but world's first full-scale humanoid robot with artificial mouth was initiated by Waseda University(Japan) in 1967 project name-WABOT-1(1972).In addition this project WABOT-2 humanoid was devolved as the advanced version of aWABOT-1 in addition to basic human gait motions it has the artistic activity of in playing keyboard instruments [6][7].

Honda developed many humanoid robots the seven series of robots named as Experimental model E0 to E6 which all has the basic android design structure and performed the basic human gait motion this series devolved 1986-1993. The Prototype Model series P1, P2 and P3 an evolution from E series, with upper limbs (1997)[8].

The 11th bipedal humanoid robot 'ASIMO (Advanced Step in Innovative mobility)'in 2000, able to run, 130cm tall and weighs 54kg with operating time one-hour. It has ability to recognize moving object and face recognition ASIMO respond to sound and it is named in the world's most advanced humanoid robots. The name ASIMO was chosen in honour of Isaac Asimov [9].

NAO (2008) is one of best commercial humanoids in the research, education and entertainment sector. NAO is an open source humanoid robot which has very attractive mechanical design frame with 25 degree of freedom [10].

Poppy humanoid robot (2013) first open source 3D-printed humanoid robot in the world It is a 25-degrees of freedom humanoid robot with a fully actuated vertebral column with 83cm tall and weighs 5.3kg. It is used for education, research (walk, human- robot interaction) or art (dance, performances). From a single arm to the complete humanoid, this platform is actively used in labs, engineering schools, and artistic projects. ABS material is used in 3d printed manufacturing for its mechanical frame. Dynamixel motors are used as actuators. As Poppy humanoid project is available in our college's PI lab, we study many aspects of humanoid robot with basic functioning and the complex mechanical structural design and gain hands of experience of the working of the Poppy humanoid robot. After studying POPPY, we observed some of the drawbacks such as-the hand(palm) of the poppy is having zero degree of freedom as its do not have gripping function, also due high DOF it is hard to balance its C.G. and control its bipedal motion [14].

There are also some advanced robots we studied –ATLAS (Boston dynamics2013) it is use to do many difficult activities like climb ladder, a double flipped jump, drive utility vehicle at site.

SOPHIA developed by Hanson Robotics; it is most advanced AI implemented humanoid robot which has ability to display 60 facial expressions. Sophia is conceptually similar to the computer program ELIZA which was one of the first attempt at simulating human conversations. Sophia is able to have conversation using natural language subsystem. In 2018 Sophia was upgraded with functional leg [15].

We also studied two made in India humanoid robots MANAV (2014) a 2 ft tall and weight 2kg is India's first humanoid manufactured by 3d printed technology made up of Acrylonitrile butadiene styrene plastic (ABS) with total 21 DOF. The robot comes with lithium polymer battery also equipped with Wi-Fi and Bluetooth connectivity [19].

RASHMI one of the most advanced AI implemented humanoid robot with 83 facial expressions with its unique 6 axis neck movement. Facial recognition ,3d mapping and OCR. Rashmi used to speak 4 different language English, Hindi, Marathi, Bhojpuri, etc. Rashmi has also a functional hand with finger movement, she doesn't have legs motion at present. Rashmi will be used in ISROs Gaganyaan2022 space mission.

2.3 Kinematic Study

2.3.1 Kinematics

Kinematics is field of study that describes the motion of points, bodies (objects), and systems of bodies (groups of objects) without considering the forces that cause them to move. In this report we have performed Kinematic study on single foot and arm for motion programming and analysis [21].

Coordinate Systems

first task is to clearly define the position and orientation of each part of the robot. To do this we define a special point on every link of robot, considering this point as origin we define a coordinate system for each link.

Global Coordinate System

It is fixed coordinate system located on base link of robot and it is used for determining location of any point in space with reference to fixed base. Positions defined using World Coordinates are called Absolute Positions.

Local Coordinate System

Every link of robot is assigned with its own local coordinate system so we can relate any link to base link via transformation matrix. Any position of point on link defined using local coordinates is called relative position.

Homogeneous Transformation Matrix

We can represent link local coordinate system (B) with respect to fixed global coordinate system (S) by specifying position of B in S coordinates i.e. P vector and also orientation of B in S coordinate i.e. R matrix. We gather this together in 4x4 matrix, called homogeneous transformation matrix or transformation matrix [22].

$$T = \begin{bmatrix} R & P \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & P_x \\ R_{21} & R_{22} & R_{23} & P_y \\ R_{31} & R_{32} & R_{33} & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Chain Rule

What we described above in Homogeneous transformation matrix can be generalised. Let us assume we have a mechanism which connects the local coordinates S_1 through S_n. If we have a Homogeneous Transformation which describes next local coordinates S_i and S_{i+1} as,

$$_{i+1}^{i}T$$

by reiterating through the above process, we get the following equation,

$$_{n}^{1}T = {}_{2}^{1}T * {}_{3}^{2}T \ldots * {}_{n}^{n-1}T$$

Here $_{n}^{1}T$ is a homogeneous transformation that describes the position and attitude of the Nth joint. To add another link to the end of this joint weneed to multiply it with the homogeneous transformation matrix from the right [22].

This method of multiplying the homogeneous transformation in order to calculate the coordinate transform matrix is called the chain rule. The chain rule enables us to calculate the kinematics of an arm with multiplejoints without too much complication.

For example,

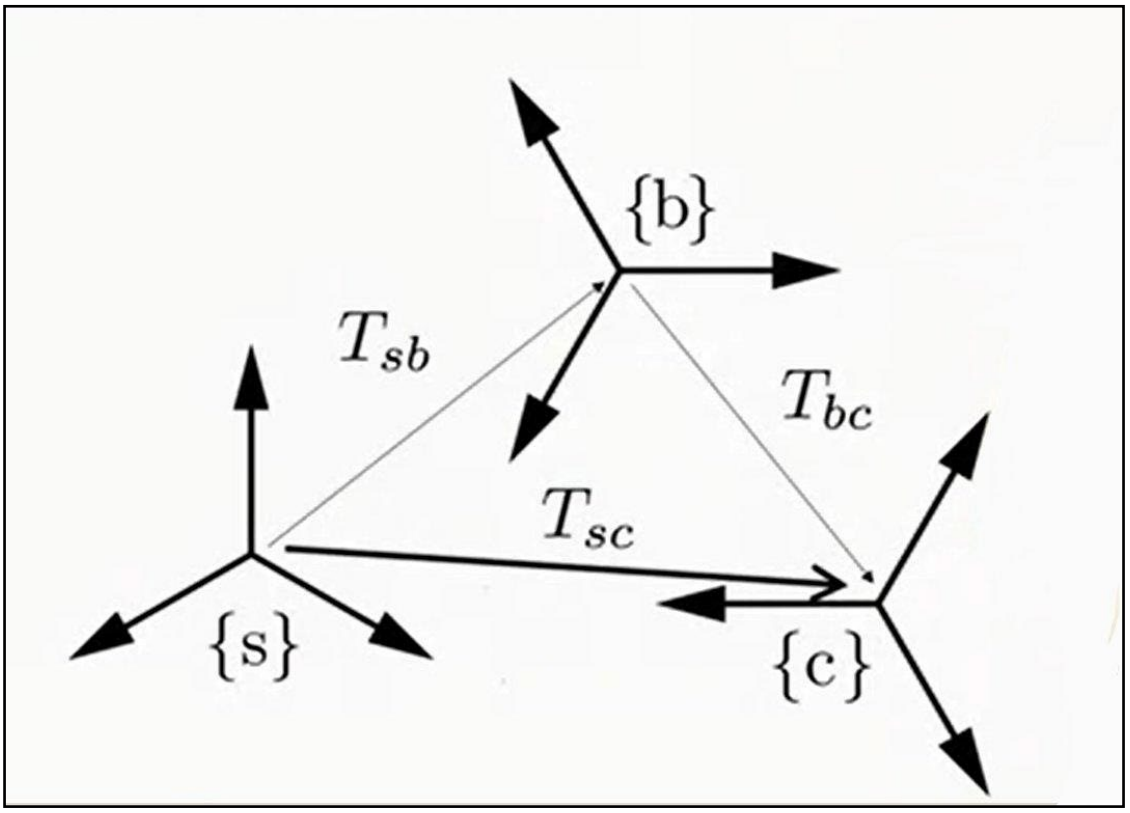


Figure 2.1: Transformation Matrices

For above system, Final homogeneous transformation matrix is,

$$T_{sc} = T_{sb} * T_{bc}$$

Like this we can assign coordinate systems on every link of robot and define every link with respect to other mathematically.

2.3.2 Forward Kinematics

Forward kinematics refers to the use of the kinematic equations of a robot to compute the position of the end-effector from specified values for the joint parameters (Joint angle). For forward kinematics we generally begin with fixed base link and find transformation matrix for each link and combine them to find final transformation matrix. If we put our joint parameters (i.e. joint angles) in transformation matrix we can find position and orientation of end effector for those particular parameters.

Workspace of a Robot (Manipulator)

workspace analysis of serial manipulators is of great interest since the workspace geometry can be considered not only a fundamental issue for manipulator design but for robot placement in a working environment and trajectory planning.

It is defined as the set of points that can be reached by its end-effector considering type of joints and physical constraint on them.

We can find out workspace using manual graphical analysis or by iteration method in MATLAB.

2.3.3 Inverse Kinematics

Inverse kinematics makes use of the kinematics equations to determine the joint parameters that provide a desired position for each of the robot's end effectors. When we compare final transformation matrix (4x4 matrix) with position and orientation (4x4 matrix) we need, we can solve those simultaneous equations to find out joint parameters.

Denavit Hartenberg Representation

While it is possible to carry out all of the forward kinematics using an arbitrary frame (Coordinate systems) attached to each link of the robot, it is helpful to be systematic in the choice of these frames. A commonly used convention for selecting

frames of reference in robotic applications is the Denavit-Hartenberg, or D-H convention.

In this convention, coordinate systems are assigned using certain rules such that any link can be described using only four parameters. Whereas normal 3-Dimensional system every link may have 6 degrees of freedom and might require 6 parameters to describe link/transformation of link.

Rules for assigning coordinate system to $link_i$ in D-H notations are as follows:

i) Z_i is an axis about which the rotation is considered or along which the translation takes place.

ii) If Z_{i-1} and Z_i axes are parallel to each other, X axis will be directed from Z_{i-1} to Z_i along their common normal.

iii) If Z_{i-1} and Z_i axes intersect each other, X axis can be selected along either of two remaining directions.

iv) If Z_{i-1} and Z_i axes act along a straight line, X axis can be selected anywhere in a plane perpendicular to them.

v) Y axis is decided as Y = ZxX [23].

After assigning coordinate system we can find out DH parameters for each which are as follows:

Link Parameters

Length of $link_i$ (a_i): It is the mutual perpendicular distance between $Axis_{i-1}$ and $Axis_i$

Angle of twist of $link_i$ (α_i): It is defined as the angle between $Axis_{i-1}$ and $Axis_i$.

Joint Parameters

Offset of $link_i$ (d_i): It is the distance measured from a point where a_{i-1} intersects the $Axis_{i-1}$ to the point where a_i intersects the $Axis_{i-1}$ measured along the said axis.

Joint Angle (θ_i): It is defined as the angle between the extension of a_{i-1} and a_i measured about the $Axis_{i-1}$ [23].

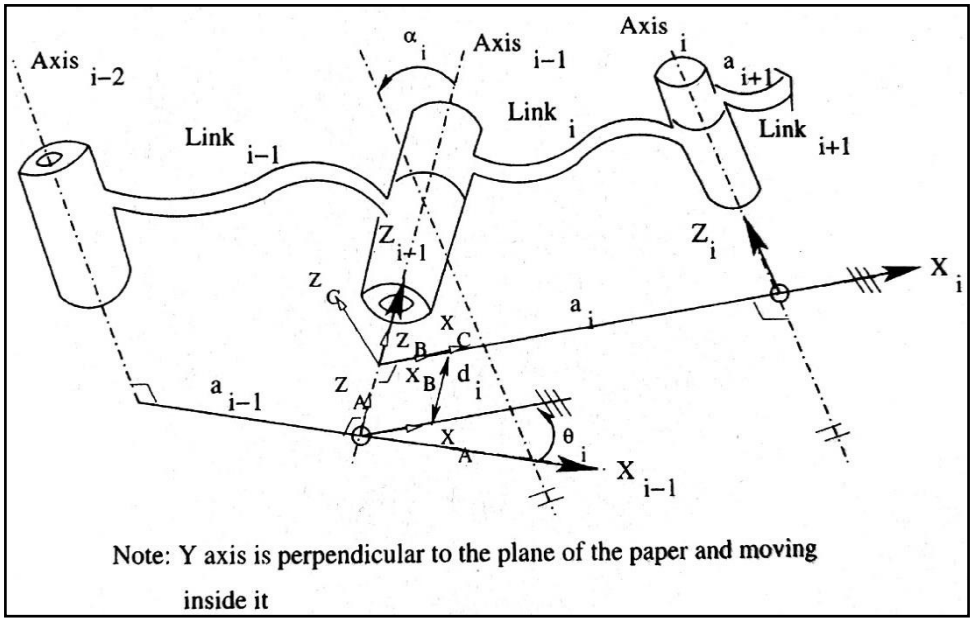


Figure 2.2: Denavit-Hartenberg Parameters

Now for given DH parameters transformation matrix can be found out as follows:

$$ {}^{i-1}_{i}T = {}^{i-1}_{A}T\,{}^{A}_{B}T\,{}^{B}_{C}T\,{}^{C}_{i}T $$

$$ = ROT(Z, \theta_i)TRANS(Z, d_i)ROT(X, \alpha_i)TRANS(X, a_i) $$

$$ = Screw_Z Screw_X $$

This transformation matrices makes forward kinematic calculations much easier than normal.

2.4 Study of Humanoid Walk

Humanoids walking is moving along at a straight line with moderate pace by lifting up and putting down each foot in turn, so that one foot is on the ground while the other is being lifted and center of mass is always in equilibrium [4].

There exist two kind of walking, namely, static walking and dynamic walking. In “static walking”, the projection of the center of mass never leaves the support polygon during the walking. In “dynamic walking”, there exist periods when the projection of the center of mass leaves the support polygon.

Static walking is comparatively stable waling pattern which can be used without any uses of stabilizers. But also, it results in Slower walking speeds. Whereas Dynamic walking is relatively unstable walking pattern and may cause Humanoid to fall while walking. To prevent this, it might require special system called as stabilizers.

Stabilizers is type of program, which modifies the walking pattern by using gyros, accelerometers, force sensors or other devices to tolerate for any imbalance. Dynamic Walking with combination of stabilizer may result in faster walking pattern [22].

2.4.1 Static Walking

The center of gravity of the robot is always within the area bounded by the feet that are touching the ground. There exist 2 stages of walking, namely, Single Support Phase (SSP) and Dual Support Phase (DSP). In SSP only one leg is touching the ground and CG is balanced above that leg. In DSP both feet are touching the ground and CG can move within support polygon i.e. from one leg to another. Walking Pattern is generated such that CG is always above supporting Phase.

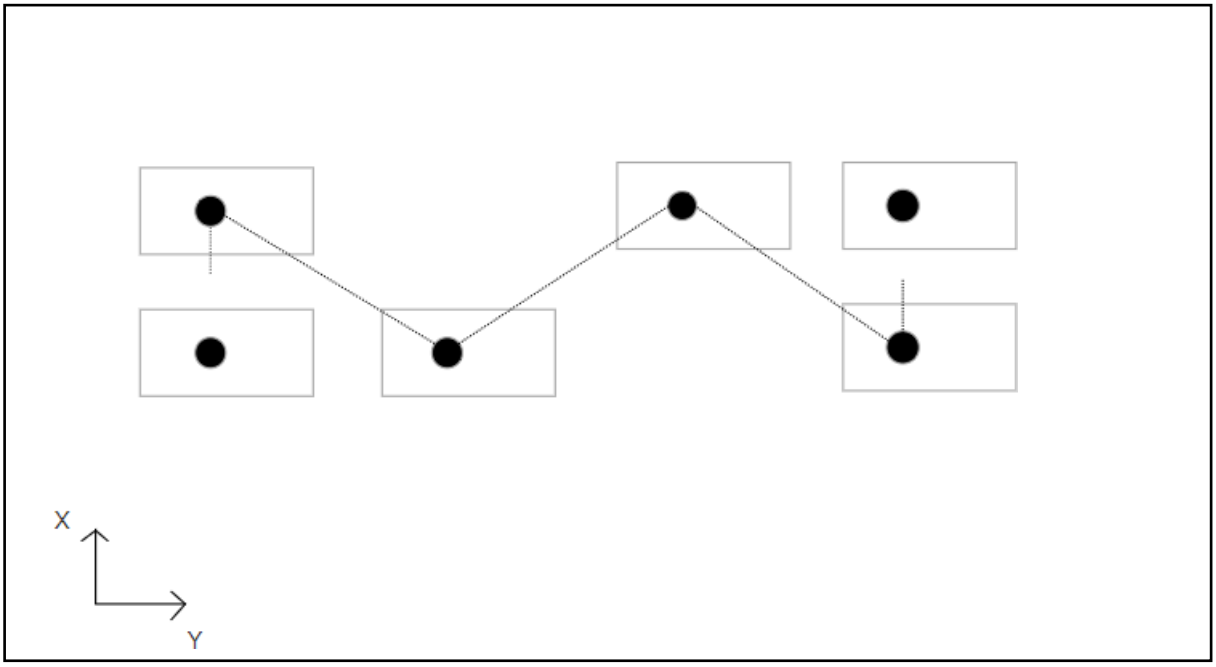


Figure 2.3:Walking Pattern for Static Walking

2.4.2 Dynamic Walking

There are few Dynamic walking pattern generation methods. The most known of which is Inverse pendulum method for Dynamic Walking.

Inverse Pendulum method

It is very complex to Dynamic study humanoid considering every joint and mass. So inverted pendulum method makes 3 assumptions which simplifies dynamic calculations of Humanoid. These assumptions are as follows:

i) We assume that all the mass of the robot is concentrated at its center ofmass (CoM).

ii) We assume that the robot has massless legs, whose tips contact the groundat single rotating joints.

iii) We assume the robot motion is constrained to the sagittal plane *and* defined by the axis of walking direction and vertical axis.

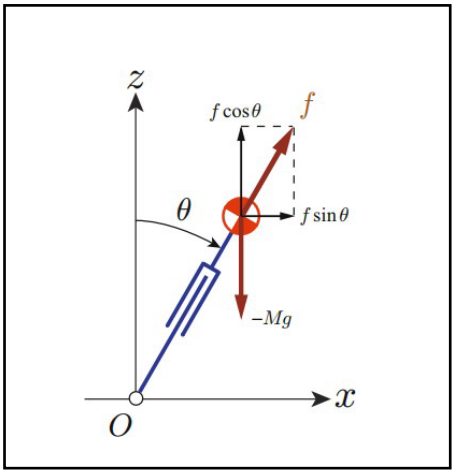


Figure 2.4: Inverse pendulum model

The inputs of the pendulum are the torque τ at the pivot and the kick force f at the prismatic joint along the leg. One of the important limitations is we cannot use big torque τ since the feet of biped robot is very small. If a walking robot has a point contact like a stilt, we have τ = 0.

In this case, the pendulum will almost always fall down, unless the CoM is located precisely above the pivot. Even with such a simple pendulum, we have a variety of falling patterns corresponding to different kick forces, f.

For Kick Force, f = (mg/cosθ), we get gravitational force balanced by kick force and CoM travels horizontally. Intuitively, we can say the pendulum is keeping the CoM height by extending its leg as fast as it is falling. We call this the Linear Inverted Pendulum [22].

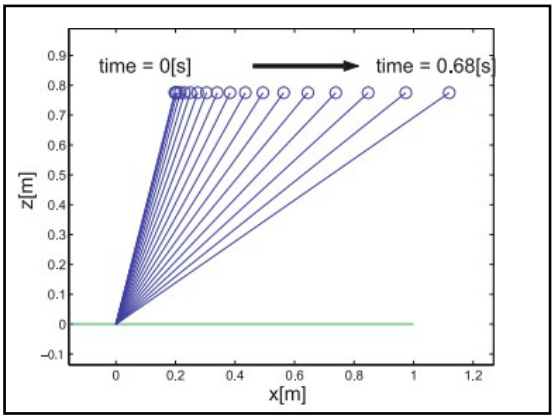


Figure 2.5: Horizontal Motion in LIP

2.5 Electronic Circuit and Components

2.5.1 MX28AT

INTRODUCTION

The MX-28T Dynamixel Robot Servo Actuator is the newest generation of Robotis Dynamixel actuator; equipped with an onboard 32bit 72mhz Cortex M3, a contact-less magnetic encoder with 4x the resolution over the AX/RX series, and up to 3mpbs using the new TTL 2.0 bus. Each servo has the ability to track its speed, temperature, shaft position, voltage, and load. The newly implemented PID control algorithm is used to maintain shaft position can be adjusted individually for each servo, allowing controlling the speed and strength of the motor's response [24].

Why Mx28T?

Mx28AT is a medium sized motor having dimensions (35.6x50.6x35.5mm) and gives high torque about to 2.3 to 3.1 Nm. Considering our requirement Mx28AT is best suitable for hand and leg joints where torque required is high, the weight of motor must be low and it also satisfy our design parameters. Hence 6 MX28AT are used for hand joints and 8 MX64AT are used for leg joint.

Figure 2.6: Servo MX28AT

SPECIFICATIONS (Mx28AT)

SR.NO	ITEM	SPECIFICATION
1.	MCU	ARM CORTEX-M3 (72 [MHz], 32Bit)
2.	Motor	Coreless
3.	Baud Rate	8,000 [bps] ~ 4.5 [Mbps]
4.	Control Algorithm	PID Control
5.	Resolution	4096 [pulse/rev]
6.	Weight	77g
7.	Dimensions (W x H x D)	35.6x50.6x35.5mm
8.	Gear ratio	193:1
9.	Stall Torque	2.3[Nm] (at 11.1V, 1.3A) 2.5[Nm] (at 12.0V, 1.4A) 3.1[Nm] (at 14.8V, 1.7 A)
10.	No Load Speed	50[rev/min] (at 11.1V) 55[rev/min] (at 12.0V) 67[rev/min] (at 14.8 V)
11.	Operating Temperature	-5 ~ +80 [°C]
12.	Input Voltage	12V
13.	Feedback	Position, Temperature, Load, Input Voltage, etc.
14.	Material	Full Metal Gear Engineering Plastic (Front, Middle, Back)
15	Standby Current	100 [mA]

Table 2.1: motor (Mx2AT) specifications

PERFORMANCE GRAPH

SR.NO.	ITEM	DATA	UNIT
1.	Weight	165	g
2.	Dimension	40.2x61.6x41	mm
3.	Gear ratio	200:1(spur/material)	Type/material
4.	Network	TTL/RS-485	-
5.	Position sensor	Contactless absolute encoder (360º / 4096)	-
6.	Operating voltage	10~14.8	V
7.	Stall torque	6.0 at 12V	Nm
8.	Stall current	4.1 at 12V	Nm
9.	No load speed	63 at 12V	RMP

Table 2.2: Performance Chart MX28AT

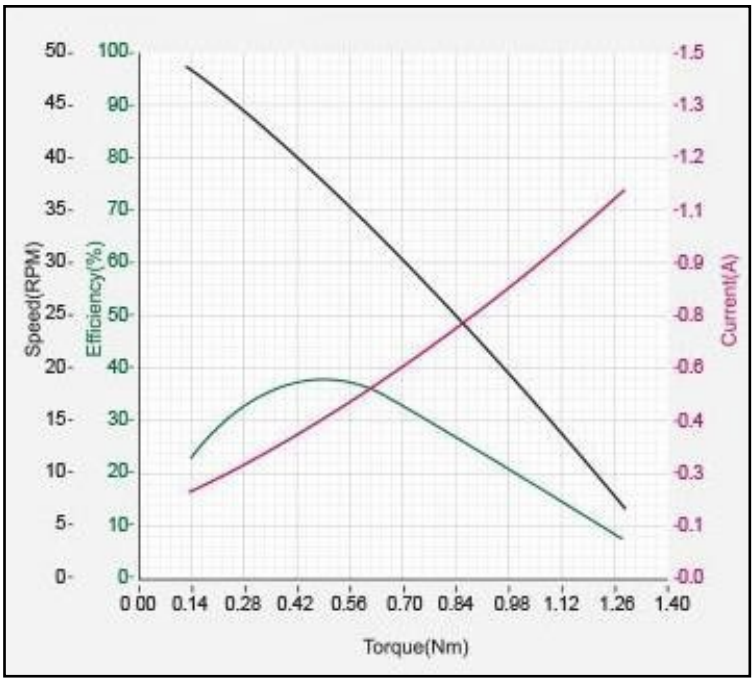


Figure 2.7:Performance graph (MX28AT)

2.5.2 MX64AT

INTRODUCTION

The MX-64T Dynamixel Robot Servo Actuator is the newest generation of Robotis Dynamixel actuator. Each servo has the ability to track its speed, temperature, shaft position, voltage, and load. The newly implemented PID control algorithm used to maintain shaft position can be adjusted individually for each servo, allowing controlling the speed and strength of the motor's response. All MX Series servos use 12v nominal voltage, so MX Dynamixels can be mixed without having to worry about separate power supplies. All of the sensor management and position control is handled by the servo's built-in microcontroller. This distributed approach leaves your main controller free to perform other functions [25].

Why MX64AT?

MX64AT is a large sized motor having dimensions (40.2x61.1x41mm) and gives very high torque about 5.5 to 7.3 Nm. Considering the requirement of high torque between hip and torso joint that is spline to balance cg during walk MX64AT is used.

Figure 2.8:Servo MX64AT

SPECIFICATIONS

SR.NO	ITEM	SPECIFICATION
1.	MCU	ARM CORTEX-M3 (72 [MHz], 32Bit)
2.	Motor	Coreless
3.	Baud Rate	8,000 [bps] ~ 4.5 [Mbps]
4.	Control Algorithm	PID Control
5.	Resolution	4096 [pulse/rev]
6.	Weight	165g
7.	Dimensions (W x H x D)	40.2x61.1x41mm
8.	Gear ratio	200:1
9.	Stall Torque	5.5[Nm] (at 11.1 [V], 3.9 [A]) 6.0[Nm] (at 12.0 [V], 4.1 [A]) 7.3[Nm] (at 14.8 [V], 5.2 [A])
10.	No Load Speed	58[rev/min] (at 11.1[V]) 63[rev/min] (at 12.0[V]) 78[rev/min] (at 14.8 [V])
11.	Operating Temperature	-5 ~ +80 [°C]
12.	Input Voltage	10.0~14.8V
13.	Feedback	Position, Temperature, Load, Input Voltage, etc.
14.	Material	Full Metal Gear Engineering Plastic (Front, Middle, Back)
15	Standby Current	100 [mA]

Table 2.3: motor (Mx64AT) specifications

PERFORMANCE GRAPH

SR.NO.	ITEM	DATA	UNIT
1.	Weight	165	g
2.	dimension	40.2x61.6x41	mm
3.	Gear ratio	200:1(spur/material)	Type/material
4.	Network	TTL/RS-485	-
5.	Position sensor	Contactless absolute encoder (360° / 4096)	-
6,	Motor	Maxon motor	-
7.	Operating voltage	10~14.8	V
8.	Stall torque	6.0 at 12V	Nm
9.	Stall current	4.1 at 12V	Nm
10.	No load speed	63 at 12V	RMP

Table 2.4: Data (MX64AT)

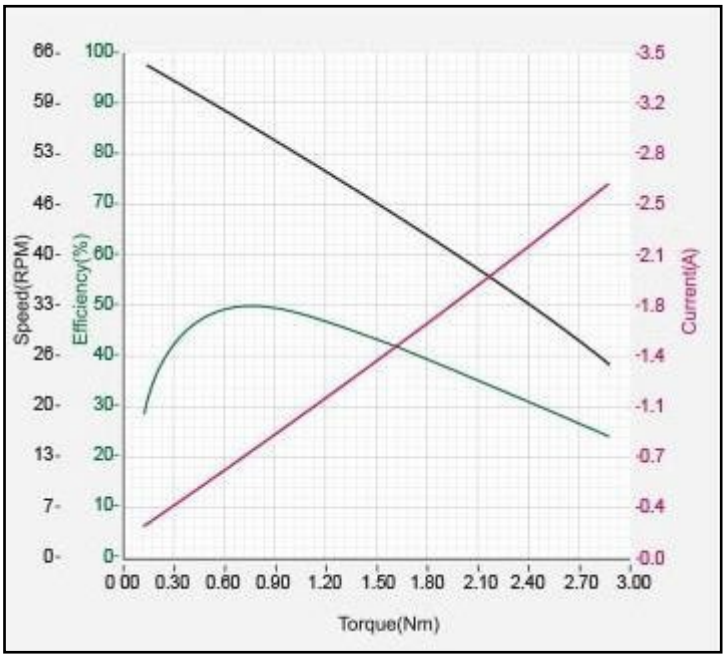


Figure 2.9: Performance graph (MX64AT)

2.5.3 AX12

INTRODUCTION

The AX-12A servo actuator from Robotis is the most advanced actuator on the market in this price range. The AX-12A robot servo has the ability to track its speed, temperature, shaft position, voltage, and load. As if this weren't enough, the control algorithm used to maintain shaft position on the ax-12 actuator can be adjusted individually for each servo, allowing you to control the speed and strength of the motor's response. All of the sensor management and position control is handled by the servo's built-in microcontroller. This distributed approach leaves your main controller free to perform other functions [26].

Why AX12?

AX12 is a light weight motor weighing up to 53.5g, having dimensions (32x50x40mm) and having low torque. The torque provided by the motor is up to 1.5 Nm. Two ax12 motor are used one inside head for up down movement and another as a head neck joint giving side wise motion. As the Torque requirement for above motion is low AX12 are suitable for the purpose.

Figure 2.10: Servo AX12

SPECIFICATIONS (Ax12)

SR.NO	ITEM	SPECIFICATION
1.	Resolution	0.29º
2.	Motor	Coreless
3.	Baud Rate	7843 [bps] ~ 1 [Mbps]
4.	ID	0~253
5.	Running degree	0º ~ 300º endless turn
6.	Weight	53.5g
7.	Dimensions (W x H x D)	32x50x40mm
8.	Gear ratio	254:1
9.	Stall Torque	1.5[Nm] (at 12 [V], 1.5 [A])
10.	No Load Speed	59[rev/min] (at 12[V])
11.	Operating Temperature	-5 ~ +70 [°C]
12.	Input Voltage	9~12V
13.	Feedback	Position, Temperature, Load, Input Voltage, etc.
14.	Material	Engineering Plastic
15	Protocol type	Half Duplex Asynchronous Serial Communication (8bit, 1stop, No Parity)

Table 2.5: motor (Ax12) specifications

2.5.4 SG 90 Servo

INTRODUCTION

A servo motor is a rotary actuator or a motor that allows for a precise control in terms of the angular position, acceleration, and velocity. Basically, it has certain capabilities that a regular motor does not have. Consequently, it makes use of a regular motor and pairs it with a sensor for position feedback. It is a self-contained electrical device that rotates parts of machine with high efficiency and great precision. The Tower Pro SG90 9g Mini Servo is 180° rotation servo. It equips sophisticated internal circuitrythat provides good torque, holding power, and faster updates in response to external forces.

Advantages: Sg90 servo motor has good speed control characteristics, smooth control in the entire speed range, almost no oscillation, high efficiency, low heat generation, high speed control, and high precision position control [27].

WHY SG 90?

Considering above advantages SG90 is used in wrist to control finger movements where the space available to mount the motor is less and torque required is sufficient. Sg 90 is used to pull string mechanism of palm.

Figure 2.11: Servo SG 90

SPECIFICATIONS (SG 90)

SR.NO	ITEM	SPECIFICATION
1.	A	32mm
2.	B	23mm
3.	C	28.5mm
4.	D	12mm
5.	E	32mm
6.	F	19.5mm
7.	Speed	0.1 sec
8.	Torque	2.5 kg.cm
9.	Weight	14.7
10.	Voltage	4.8-6

Table 2.6: specification (SG 90)

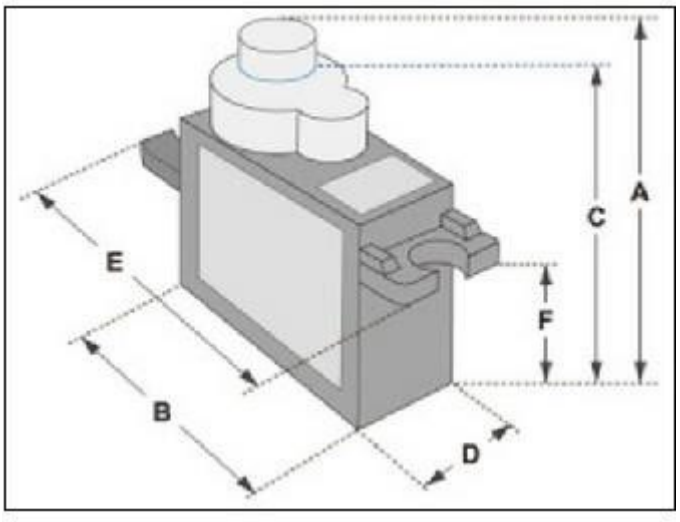


Figure 2.12: Specification SG 90

WIRE DESCRIPTION

Wire Number	Wire Colour	Description
1	Brown	Ground wire connected to the ground of system
2	Red	Powers the motor typically +5V is used
3	Orange	PWM signal is given in through this wire to drive the motor

Table 2.7: wire description

MOTOR ROTATION

PWM signal produced should have a frequency of 50Hz that is the PWM period should be 20ms. Out of which the On-Time can vary from 1ms to 2ms. So, when the on-time is 1ms the motor will be in 0° and when 1.5ms the motor will be 90°, similarly when itis 2ms it will be 180°. So, by varying the on-time from 1ms to 2ms the motor can be controlled from 0° to 180°

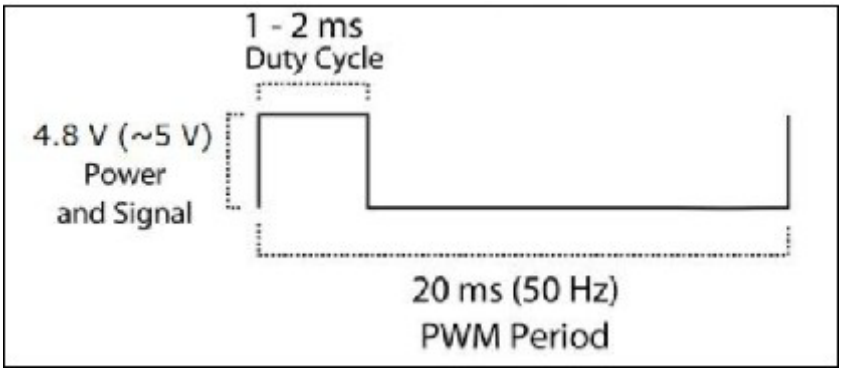


Figure 2.13: PWM period

2.5.5 U2D2

Introduction

U2D2 is a small size USB communication converter that enables to control and operate DYNAMIXEL with PC.

U2D2 can be connected to the USB port of the PC with the enclosed USB cable. It supports both 3Pin TTL connector and 4Pin RS-485 connector to link up with various Dynamixel. U2D2 does not supply power to Dynamixel; therefore, an external power supply should provide power to Dynamixel [28].

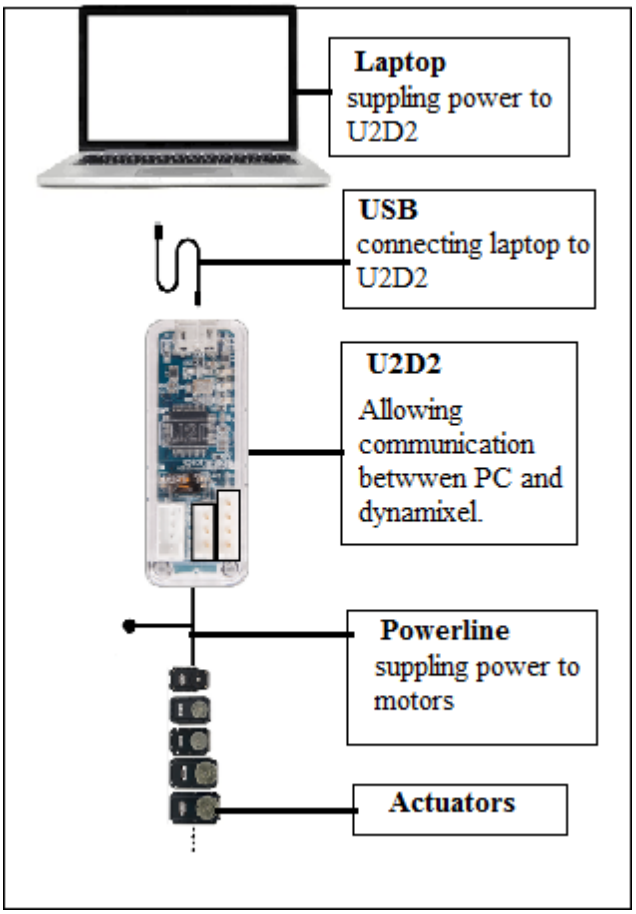


Figure 2.14: Connection for U2D2

LAYOUT

Following is the table showing description of various ports that are available on U2D2.

SR.NO	PORTS	DESCRIPTION
1.	4Pin UART	Convert USB and UART
2.	3Pin TTL Level	Connect to DYNAMIXEL with 3Pin TTL Level Communication
3.	4Pin RS-485	Connect to DYNAMIXEL with 4Pin RS-485 Communication
4.	Status LED	Display status of Power supply, TxD (Data write) and RxD (Data Read)
5.	Micro-B USB	Connect to the PC with USB cable

Table 2.8: Wire Description

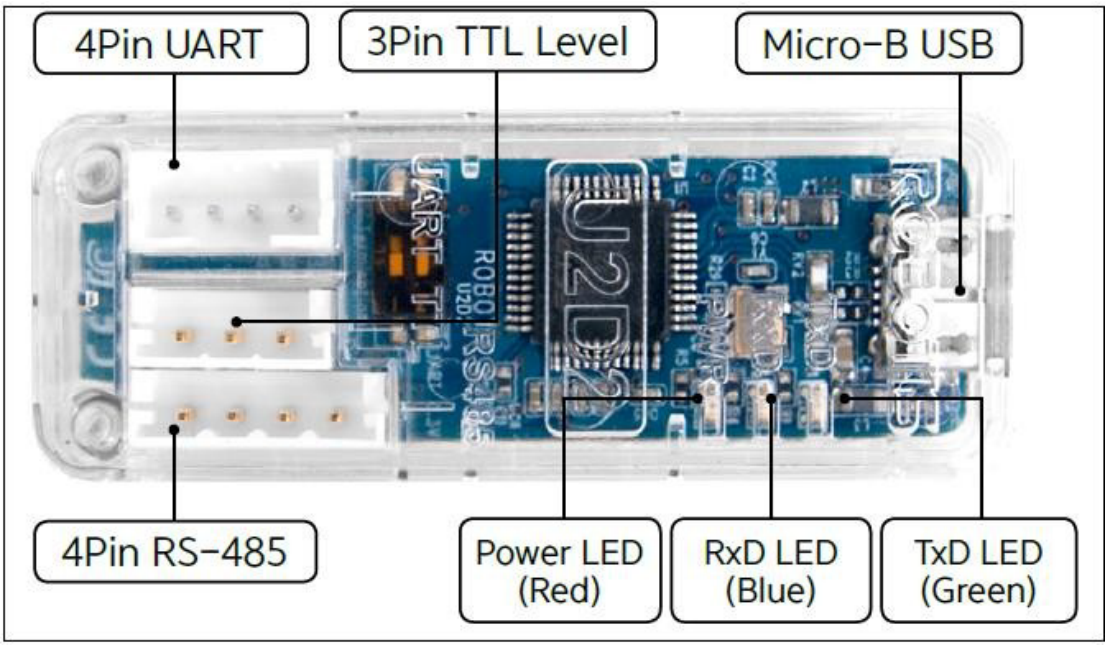


Figure 2.15: U2D2 layout

2.5.6 SMPS2Dynamixel

Following are some of the important features of SMPS2Dynamixel.

- This device provides power from the SMPS to the DYNAMIXEL.
- Connect the SMPS to the DC terminal and connect the DYNAMIXEL using the cable.
- 3P connector for AX series and 4P connector for DX/RX/EX series are mounted on the SMPS2Dynamixel. The power and communication line are all connected making it possible to play the role of the DYNAMIXEL expansion bus.
- When controlling the DYNAMIXEL with the USB2Dynamixel, power can be easily supplied to the DYNAMIXEL.
- Can be used by connecting to a proper DC terminal. Can be used according to the proper voltage of the DYNAMIXEL.
- A shrink tube is used to cover the circuit part to protect it from short circuit from metal substances [29].

Figure 2.16: SMPS2Dynamixel

2.5.7 Battery

We have decided to use Lithium Polymer LiPo Battery of capacity 5200mAh to power our entire circuit. A lithium polymer battery (lithium-ion polymer battery) isa rechargeable battery of lithium-ion technology using a polymer electrolyte instead ofa liquid electrolyte. High conductivity semisolid (gel) polymers form this electrolyte. These batteries provide higher specific energy than other lithium battery types and are used in applications where weight is a critical feature. It Consists of number of cells where each cell has nominal Voltage of 3.7 Volts. Fully Charged battery cell has voltage of 4.2 and Fully depleted cell has voltage around 3.0 volts. There no specific instrument to measure charge of batteries. Voltage can be used to get measure about charge in battery.

Figure 2.17: LiPo Battery

Specification

1) Capacity – 5200 mAh
2) Number of Cells – 3
3) Voltage – 12.6V (Max)/ 9V (Min)
4) Dimension- 28*44*137mm
5) Weight-360gm

2.6 3D PRINTING

2.6.1 Introduction

The 3D printing process builds a three-dimensional object from a computer-aided design (CAD) model, usually by successively adding material layer by layer, which is why it is also called additive manufacturing.

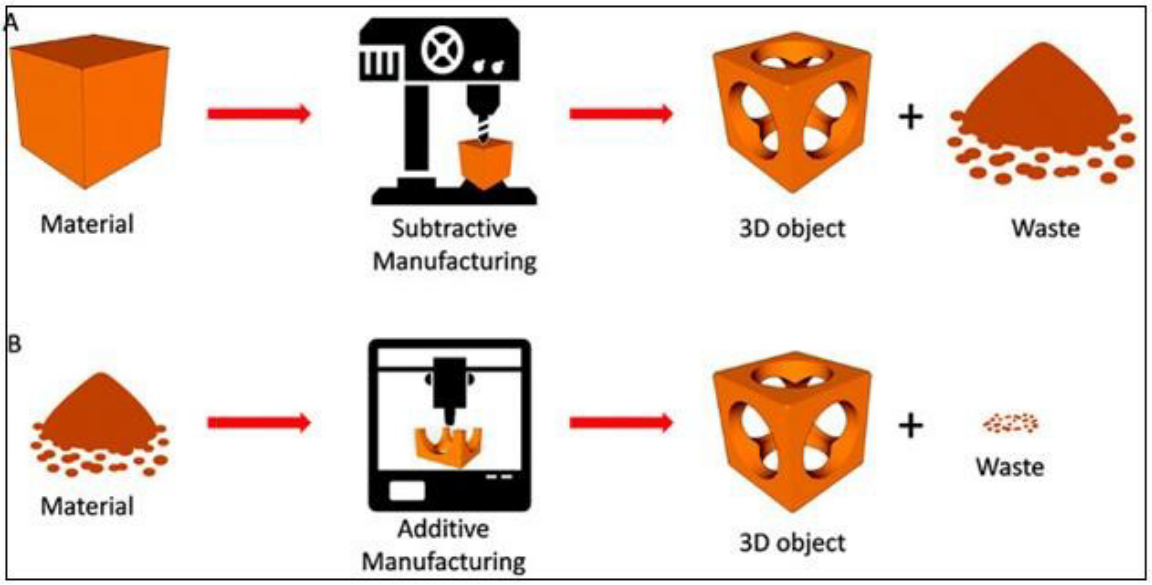


Figure 2.21: Showing difference between subtractive and additive manufacturing

Above figure clearly shows that additive manufacturing is more economical as it utilizes raw material to the fullest and produces less waste. 3D printing is one of the types of additive manufacturing. One of the key advantages of 3D printing is the ability to produce very complex shapes or geometries; including hollow parts or parts with internal truss structures to reduce weight, and a prerequisite for producing any 3D printed part is a digital 3D model or a CAD file. 3D printing is suitable and economical for rapid prototyping and hence we have used FDM (Fused Deposition Modelling) method a type of 3D printing to manufacture our humanoid.

2.6.2 FDM

Fused deposition modelling (FDM), as one of the most commonly used low-cost three-dimensional (3D) printing technologies, has been a significant method to realize the transformation from conceptualization to the products. The parts can be manufactured rapidly and directly from computer aided design (CAD) model without geometry limitation and specific tooling and with high material utilization.

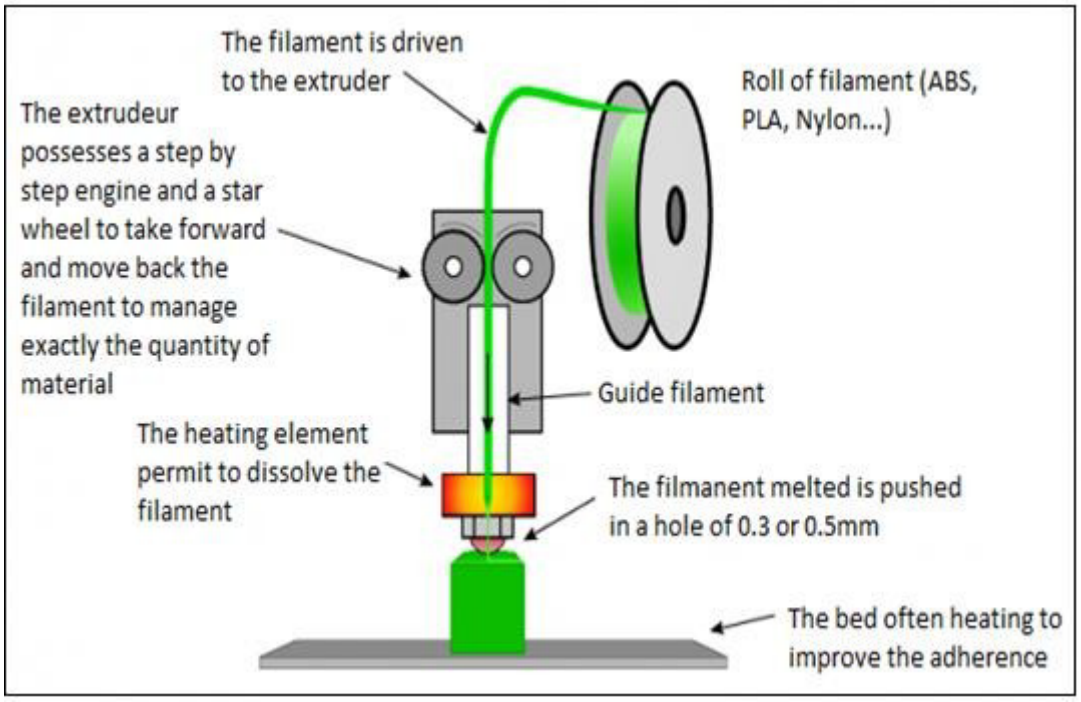


Figure 2.22: working of FDM (Fused deposition modelling) process

The above figure shows working of FDM, In the FDM process, the polymeric filament is continuously fed into the nozzle and heated to a semi liquid state, and then the thermoplastic material is extruded onto the previous layer along the cross-section contour and the filling trajectory. At the same time, the extruded material rapidly solidifies and adheres with the surrounding material to accumulate the required complex plastic parts. Hence, in comparison with the conventional fabrication process of composites characterized by impregnation and solidification of the matrix, FDM provides a possibility for manufacturing complex functional and structural parts with CFRTPCs (Continuous fibre reinforced thermoplastic composites).

2.6.3 3D printer and material used

The LulzBot TAZ 6 is the most reliable, easiest-to-use desktop 3D printer ever, featuring innovative self-levelling and self-cleaning, and a modular tool head design forflexible and multi-material upgrades. LulzBot TAZ 6 has been used for entire manufacturing process of humanoid. Material used for 3D printing is PLA (Polylactic Acid).

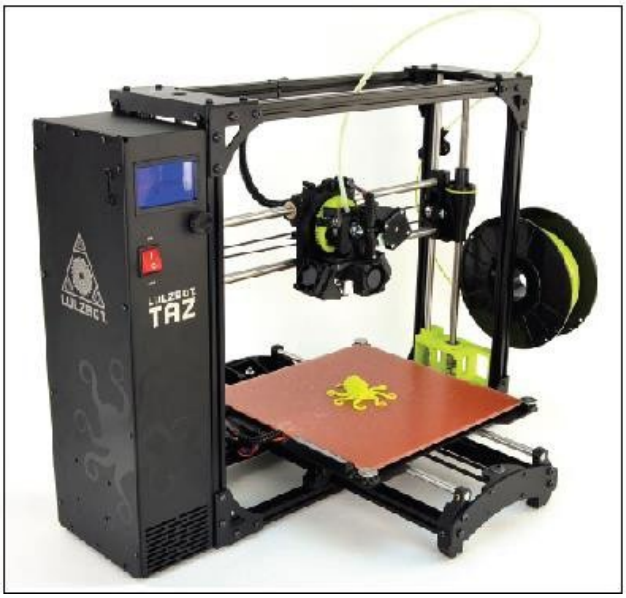


Figure 2.23: LulzBot TAZ 6 3D printer

Specifications of printer

Sr. No.	Type	Specification
1.	Print volume dimensions	280 mm x 280 mm x 250 mm
2.	Maximum Traveling speed	200mm/sec
3.	Connectivity	USB Serial and Included 8gb SD Card
4.	Nozzle Diameter	0.5mm
5.	Nozzle Temperature	Up to 290^0
6.	Maximum Print Bed Temperature	Up to 120°C

Table2.10: LulzBot TAZ 6 3D Printer specifications

2.6.4 Software used for printing

Cura is designed for Fused Filament Fabrication (FFF) 3D printers. Fused Filament Fabrication is the term for the process of laying down successive layers of extruded filament to create a 3-dimensional object. As each layer of molten plastic is extruded into place, it fuses with the previous layer.

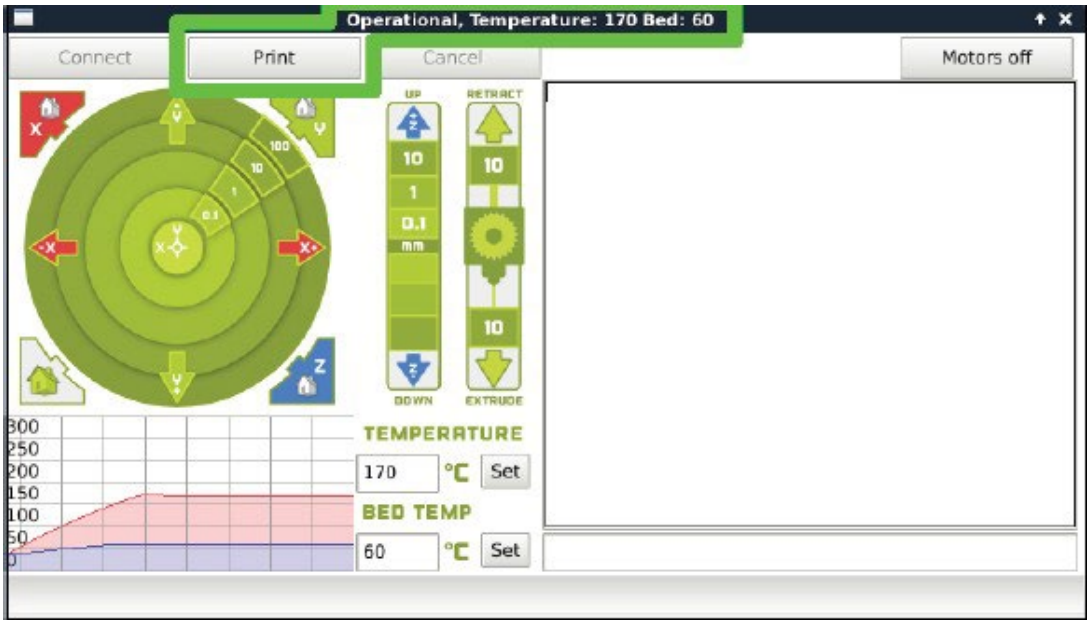


Figure 2.24: control screen of Cura software

2.6.5 Printing Parameters

1) Layer Height

The thickness of each printed layer is known as the "Layer Height". The smaller the layer height, the smoother curves will appear. Larger layer heights are better for bridging and

Overhang.

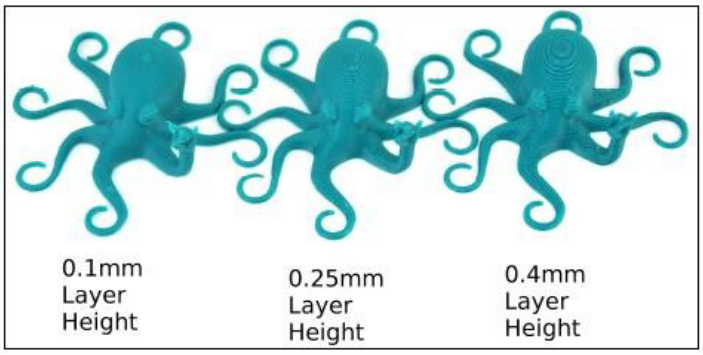


Figure 2.25: Different Layer height

2) Shell Thickness

Shells are the number of layers on the outside of a print. For FDM shells are always the first areas to be printed per layer. Strength can be added by increasing shells thickness. Most slicer programs allow shell thickness to be adjusted even allowing areas of high stress to be customized with a high shell density offering localized areas of high strength.

Any increase in the number of shells also increase the amount of time and material required to print the model increasing overall part cost. Shells typically consist of a specified number of nozzle diameters. It is always good to design shells to be a multiple of nozzle diameter to prevent voids from being formed.

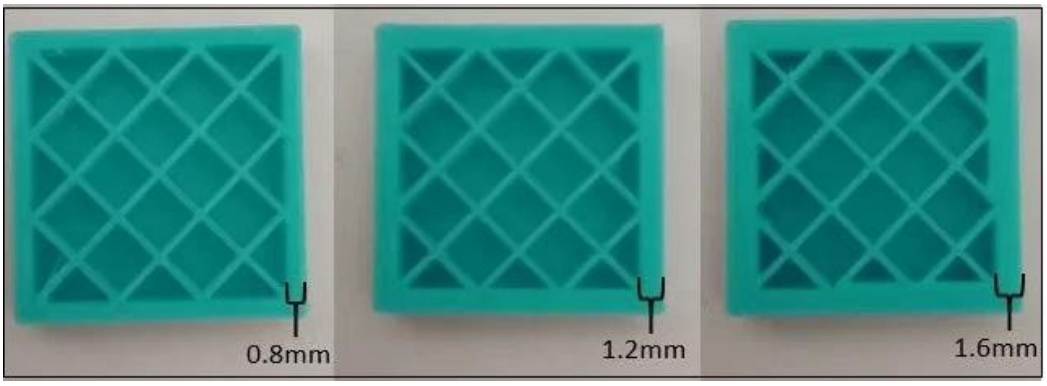


Figure 2.26: Different Shell thickness

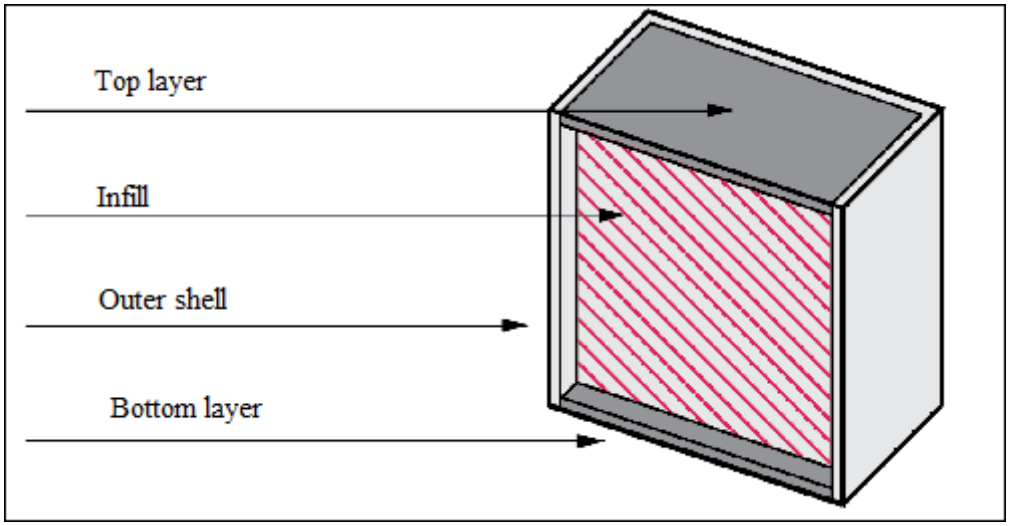


Figure 2.27: figure showing different layers of 3D printed part

3) Fill Density

The strength of a design is directly related to infill percentage. A part with 50% infill compared to 25% is typically 25% stronger while a shift from 50% to 75% increases part strength by around 10%. A prototype where form is important can be printed with very low infill saving significantly on cost and time whereas a bracket that will

experience loading will need a higher infill percentage. As mentioned above, the standard 20% that most printers use as a default should be acceptable for most application [32].

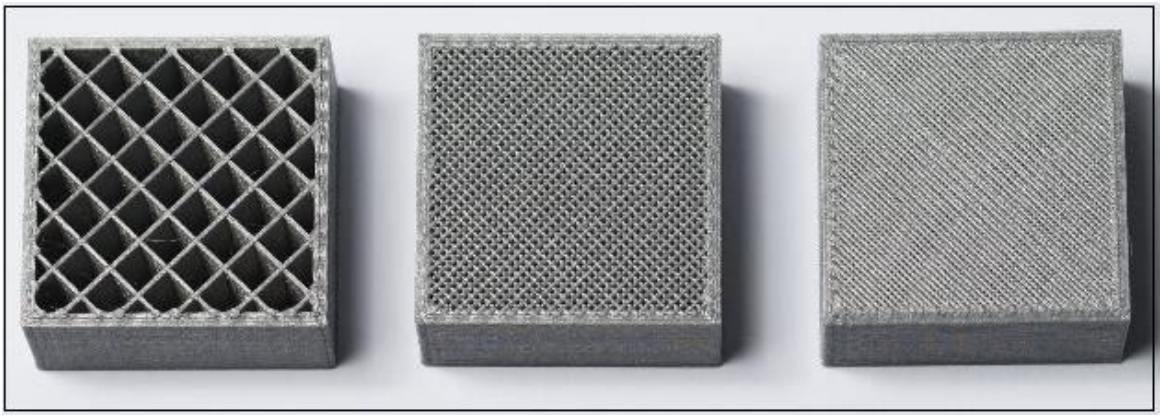


Figure 2.28: 3D parts showing different fill density

3. DESIGN OF HUMANOID

Mechanical structure of humanoid robot is an important task in the development of humanoid robot. The main objective is to design humanoid structure that can easily manipulate and capable imitate equivalent human motion. Stiffness and compliance consist with humanoid decide the flexibility of structure. Human dimension is taken into account as reference because their proportion allow for stable walking and optimal distribution of forces actuating while a human is walking. Biomechanics gives us the relation between human height and length of each link and in the same way for the mass. Design of humanoid robot should be Compact in size and light weight. The actuators of the Humanoid Robot used are Dynamixel Robot Servo Actuator. The design parameters include dimension aspect ratio, design of link, types of joint, forces on each joint, degree of freedom of each link, mass and height of body and mimic shape of human.

DESIGN OF HUMANOID			
ASPECT RATIO	DOF (DEGREE OF FREEDOM)	CAD MODELING	STRUCTURAL ANALYSIS

Table 3.1: Building blocks (Design of humanoid)

3.1 Aspect Ratio

We study the human body and its motion to help in the development of humanoid robot Human body has 244 degrees of freedom (acronym is: DOF) Basically, free motion is work of joints and ligaments present in the body the number of joints are 230 many of them having single DOF and some have more than one. Consider weight and balancing of robot we decided to restrict height of robot up to 70cm [33].

Accordingly, ratios are taken.

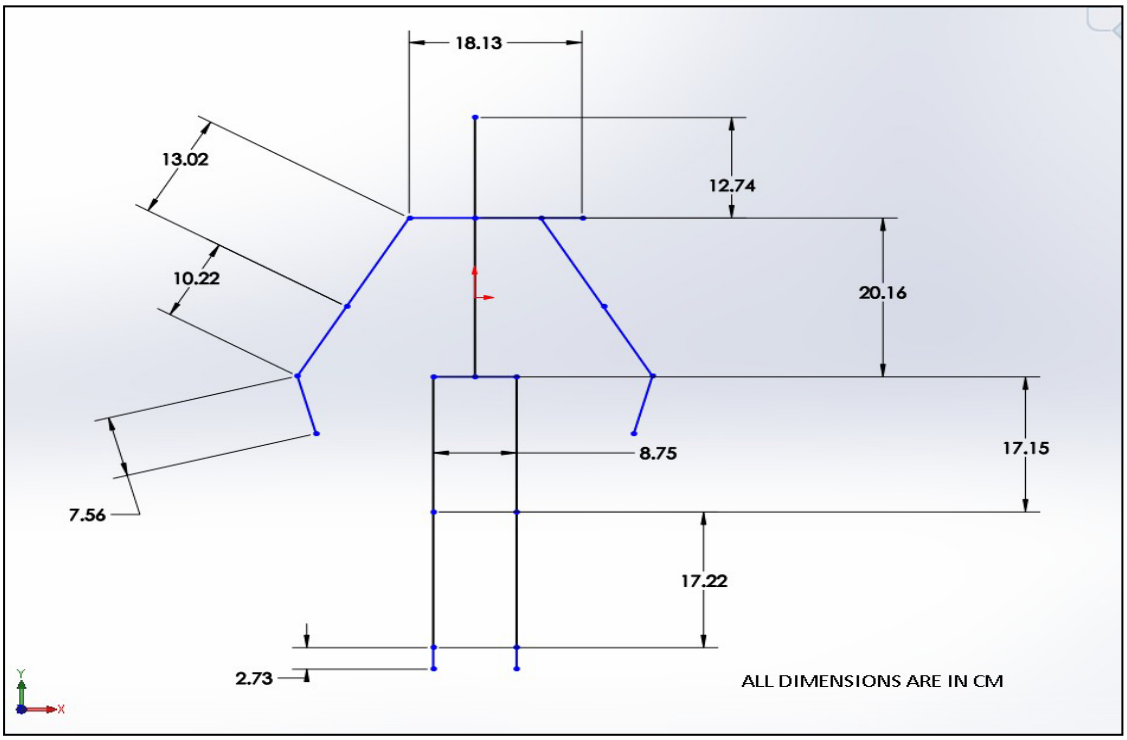


Figure 3.1: Aspect Ratio

3.2 Degrees of Freedom

The mechanical structure plays an important role in the humanoid's performance. DOF of mechanical system is the number of independent parameters that defines its configuration or state. Besides providing the general support to the main passive inactive subsystems of the humanoid, imitating the role of the human skeleton, it comprises a set of simple kinematics joints mimicking the most important degrees of freedom (DOF) of the human body.

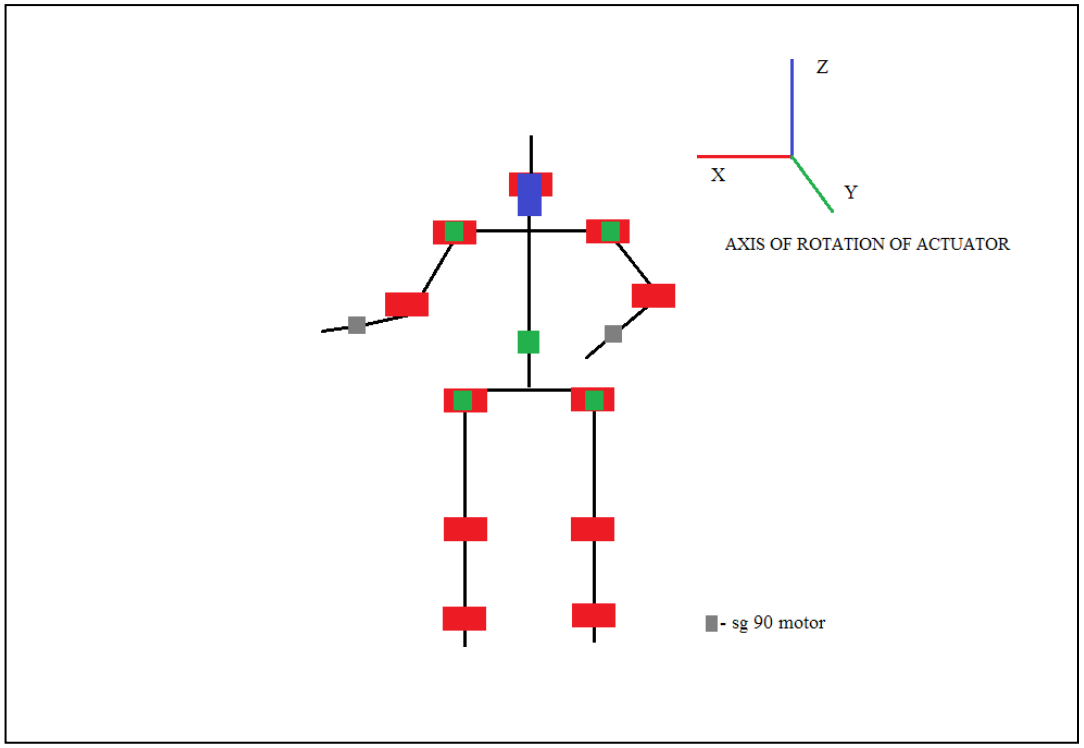


Figure 212: Motor location in Mechanical Design

HEAD	NECK (PAN & TILT)	2DOF×1=2DOF
ARM	SHOULDER(PITCH/YAW)	2DOF×2=4DOF
	ELBOW(PITCH)	1DOF×2=2DOF
HAND	WRIST(GRASP)	1DOF×2=2DOF
TORSO	WAIST(YAW)	1DOF
LEG	HIP(PITCH/YAW)	2DOF×2=4DOF
	KNEE(PITCH)	1DOF×2=2DOF
	ANKEL(PITCH)	1DOF×2=2DOF
TOTAL		=19DOF

Table 3.2: Degree of freedom of humanoid robot

The mass of our humanoid is considered as 3kg and height is 70 cm. It has total 19 degree of freedom Head has 2 (movement in x and y axis) degree of freedom. Waist has 1 degree of freedom. Each leg has 4 degree of freedom and each arm has 3 degree of freedom and 1DOF of gripper for grasp or hold the objects. First degree of freedom of leg is inside the waist which contributes in balancing the robot while walking. The total height of our humanoid is 70 cm so according to biomechanics the length of each part is calculated.

3.3 CAD (Computer-Aided Design) Modelling

CAD modelling is use to create conceptual design, product layout, strength and kinematic analysis of assembly and the manufacturing processes themselves, produce detailed engineering designs through 3D and 2D drawings of the physical components of manufactured products. In order to design the humanoid robot, Solidworks software is used to draw the 3D model. The location and orientation of the motor is defined and embedded into the design. The design process has been repeatedly improved to obtain the best output. We use SOLIDWORKS 2017 edition software for CAD modelling.

Cad modelling is done by considering all above-mentioned parameter and following:

1. Aspect ratio and Degree of freedom of humanoid Robot.
2. Manufacturing process –Additive manufacturing by 3D printing technology (FDM- Fluid Deposition Modelling).

Considering above parameters, we decided that modelling should be done in 2 stages

Stage 1: Skeleton

Skeleton is the core of our humanoid which consists of motor holders, clamps and rigid links. The main power house of humanoid robot 'Battery' also at the section of skeleton along with wires and connectors. And main purpose of this Skeleton stage is to try to maintain Center of Mass (COM) of humanoid at the middle of body which is base of stable bipedal walking. The links will bear most of the forces and stresses which leads to high resistance and more stiffness. It also modelled such that it will be easier to 3D print these Parts. These are main load bearing parts of Humanoid.

Stage 2: Casing

Casing will consist of loosely attached parts that will mostly cover all joints and will act as protection and shock absorber for Skeleton. It will give aesthetic look to our humanoid. To make humanoid design more likely to human shape and for safety of the internal core skeleton structure this stage of aesthetic look is attached over a skeleton structure which includes the head also.

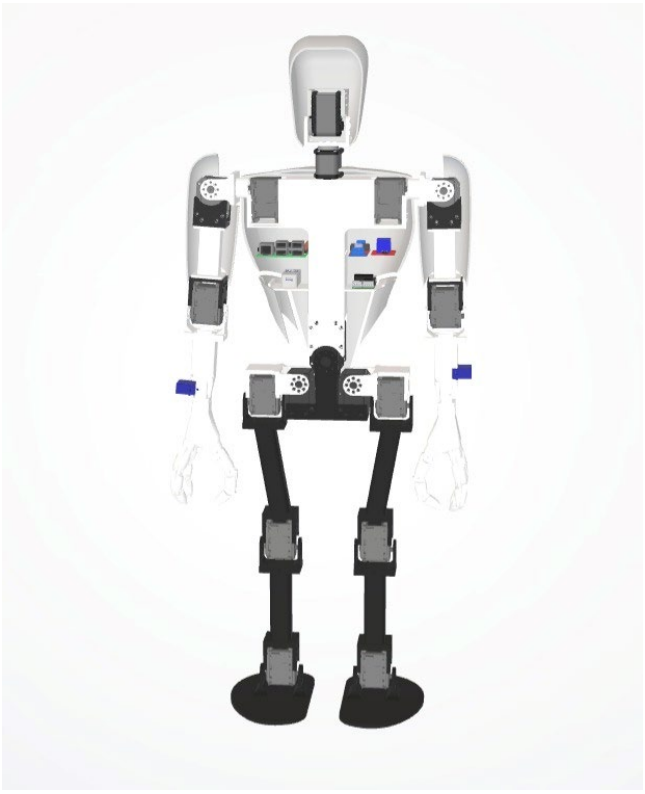


Figure 3.3: 3D CAD of Full Skeleton structure of Humanoid Robot

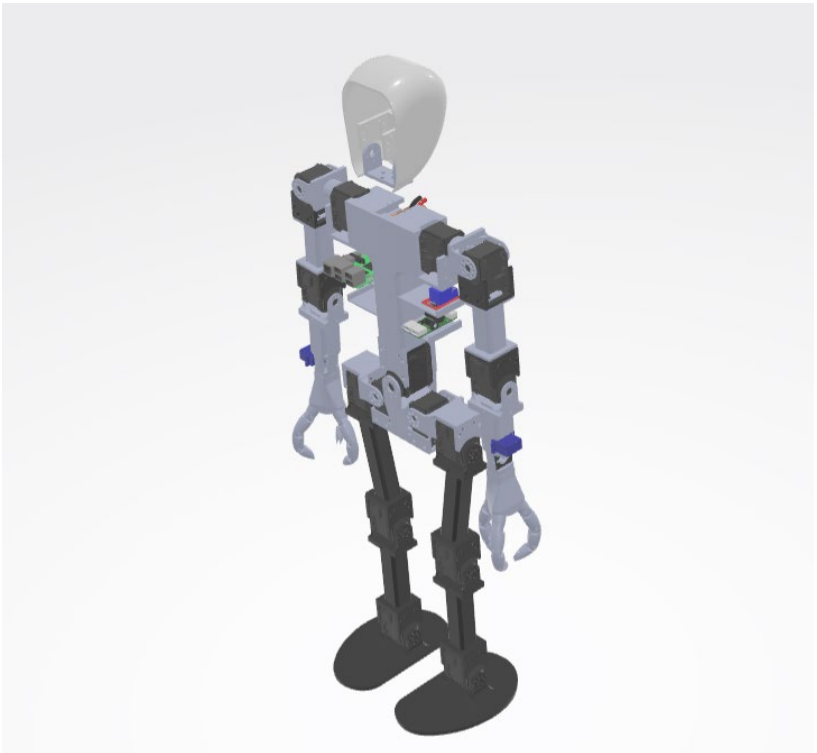


Figure 3.4: 3D CAD isometric view of Full Skeleton structure of Humanoid Robot

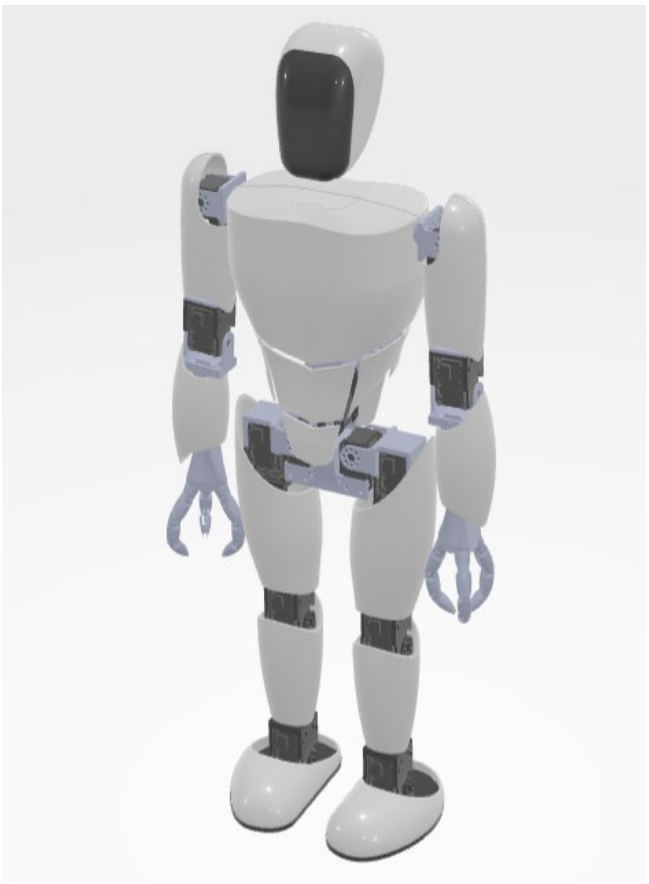


Figure 3.5: Casing of Humanoid

3.4 Clamp and Holder

For joining one component of Skelton structure with another and to hold the Actuators of the humanoid robot Clam and Holder system used. This allows relative movement with two of the joining components. This system allows the joints to move in a rotary motion along the axis of the dc motor actuator Used. It is the most important part of the skeleton structure.

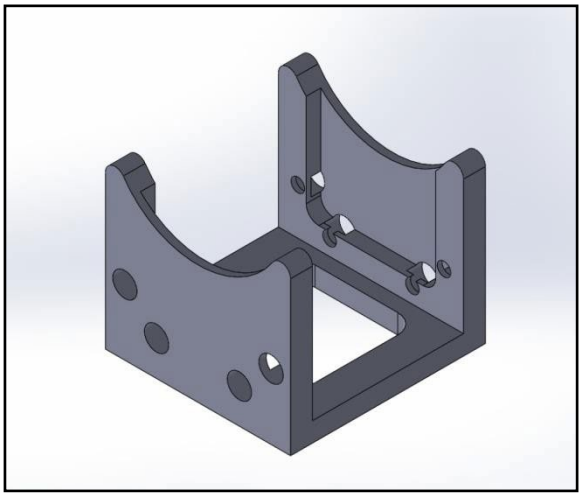


Figure 3.6: Holder

There actuators used are- MX-28AT Dynamixel Robot Servo Actuator and MX-64AT Dynamixel Robot Servo Actuator, Tower pro sg90. The actuator sg90 is assembled in the cavity section of hand. But both the MX-28AT and MX-64AT Dynamixel Robot Servo Actuator requires clamp holder system. Nut (M 2.5) and Bolt (wrench bolt M2.5*6) are used to assemble actuator and clamp-holder.

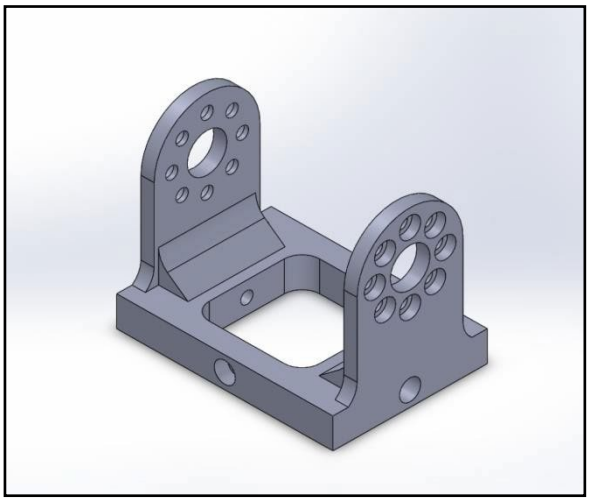


Figure 3.7: Clamp

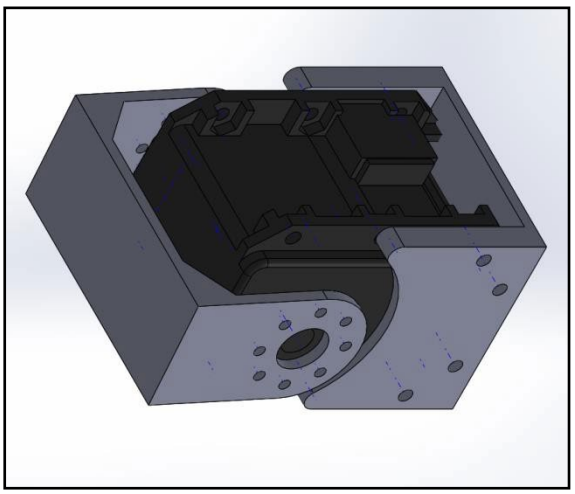
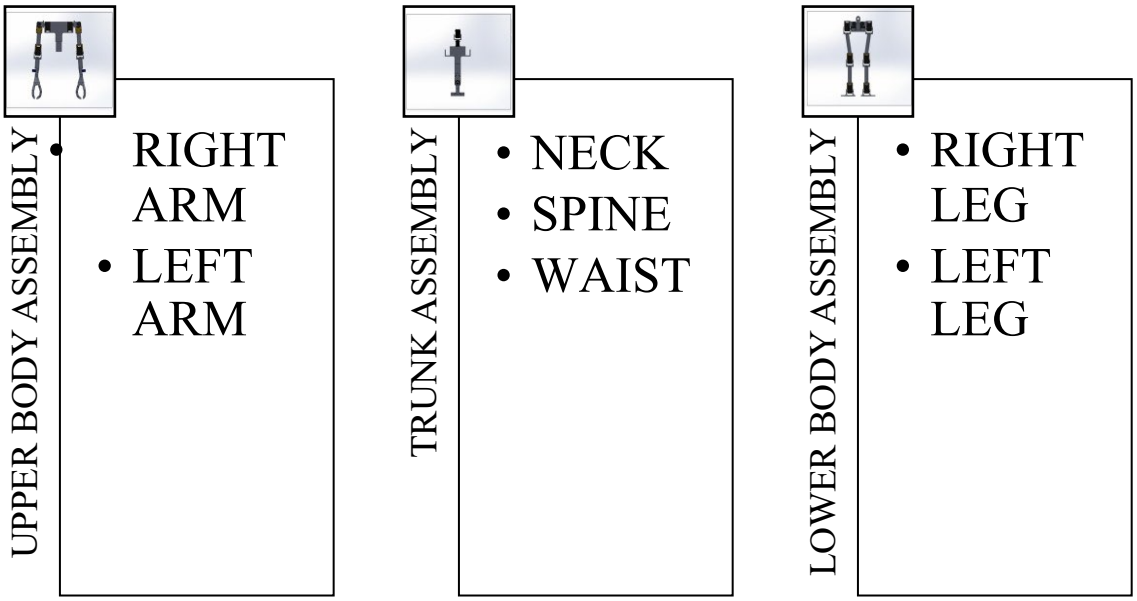


Figure 3.8: Clamp and Holder Assembly

3.5 Complete CAD Assembly of humanoid Robot

Parts of skeleton structure like Thigh, Calf, Forearm, etc are having shape of Rounded square to give more strength to structure makes humanoid deign safe and stable to and helps to manage bear a consider stress COM at centre. The CAD model ismade by considering the manufacturing process used -3d printing technology and properties of the Polylactic Acid Polymer plastic used.

The Assembly are divided into main parts as given below:



3.5.1 Upper Body Assembly:

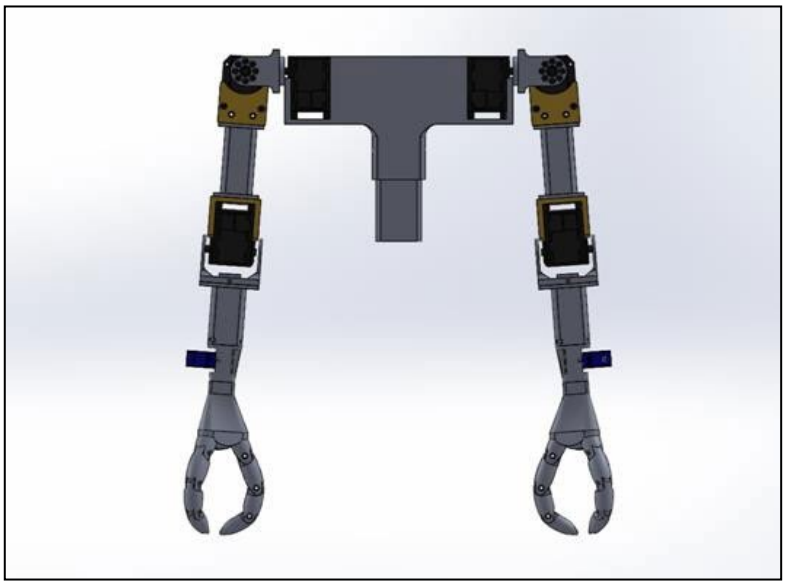


Figure 3.9: Upper Body Assembly (Front View)

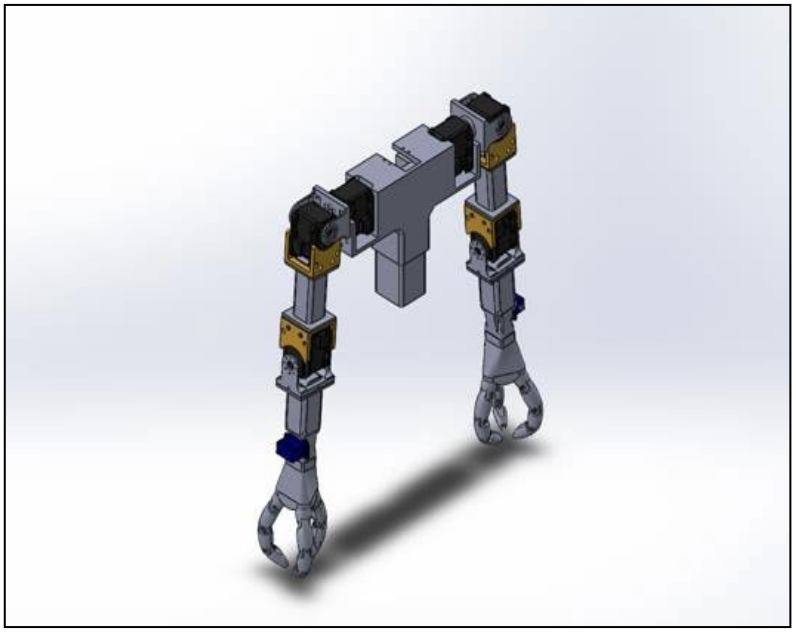


Figure 3.10: Upper Body Assembly (Isometric View)

- **Right Arm:**

Sub-assembly name	Motor name	Motor Type	ID
Right upper arm/ shoulder	r_shoulder_X	MX-28AT	9
	r_shoulder_Y	MX-28AT	10
Right upper arm/elbow	r_elbow_X	MX-28AT	11
Right upper arm/wrist	r_wrist_Y	Sg90	-

Table 3.3: Right Arm assembly

- **Left Arm:**

Sub-assembly name	Motor name	Motor Type	ID
Left upper arm/ shoulder	l_shoulder_x	MX-28AT	12
	l_shoulder_Y	MX-28AT	13
Left upper arm/elbow	l_elbow_X	MX-28AT	14
Left upper arm/wrist	l_wrist_Y	Sg-90	-

Table 3.4: Left Arm assembly

3.5.2 Trunk Assembly:

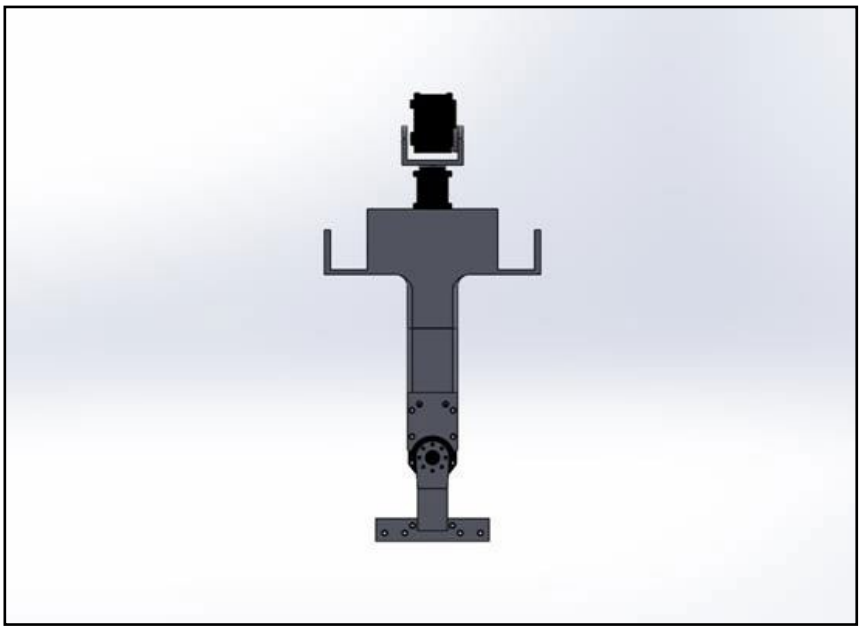


Figure 3.11: Trunk Assembly (Front View)

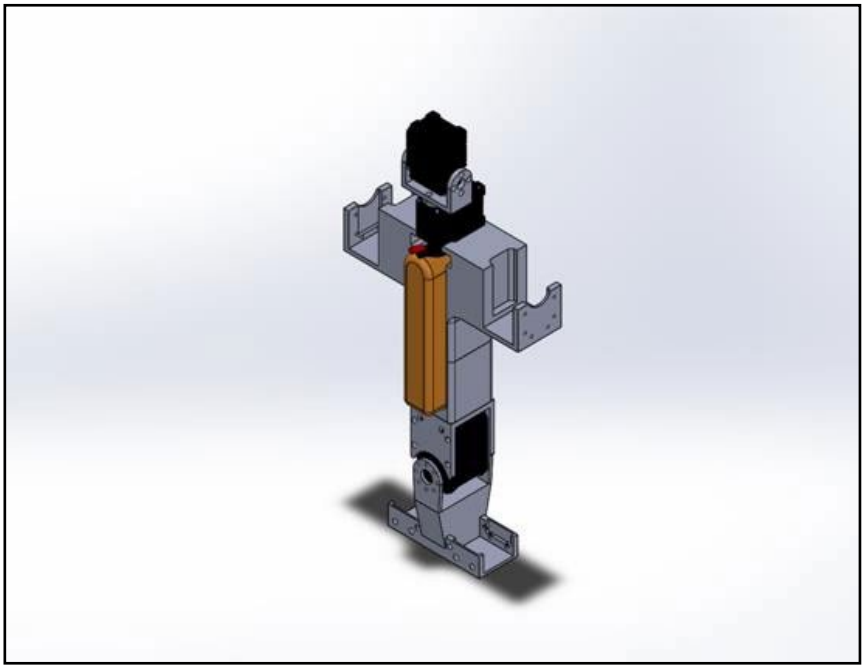


Figure 3.12: Trunk Assembly (Isometric View)

Sub-assembly name	Motor name	Motor Type	ID
Trunk/neck	n_trunk_X	MX-28AT	51
	n_trunk_Z	MX-28AT	52
Trunk/waist	w_trunk_Y	MX-64AT	31

Table 3.5: Trunk Assembly

Trunk assembly includes the neck-spin-waist Assembly here the neck and waist assembly are explained as above and the spine is connected to the main trunk by screw and with the help of Cyanoacrylate adhesive (superglue). Two pelvis Actuators are also hold by the trunk. The battery is attached to trunk at the back.

3.5.3 Lower Body Assembly:

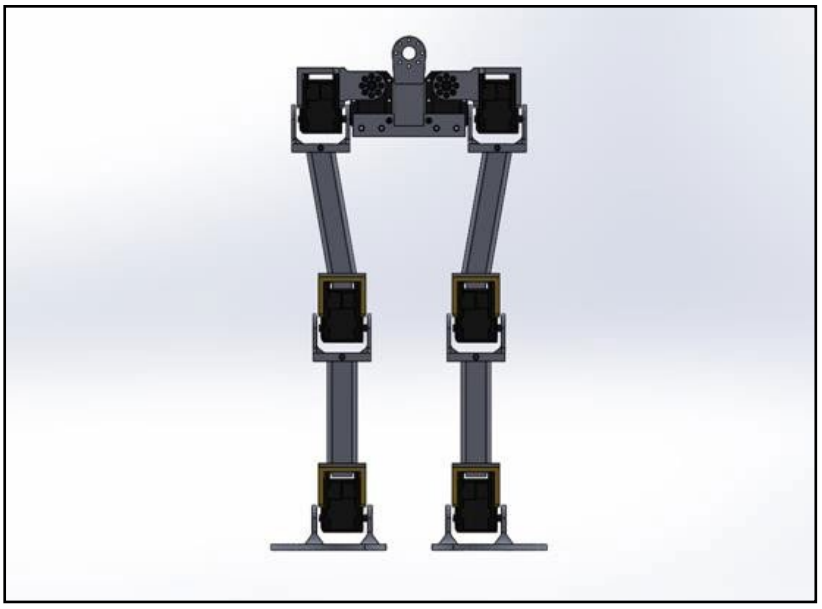


Figure 3.13: Leg Assembly (Front View)

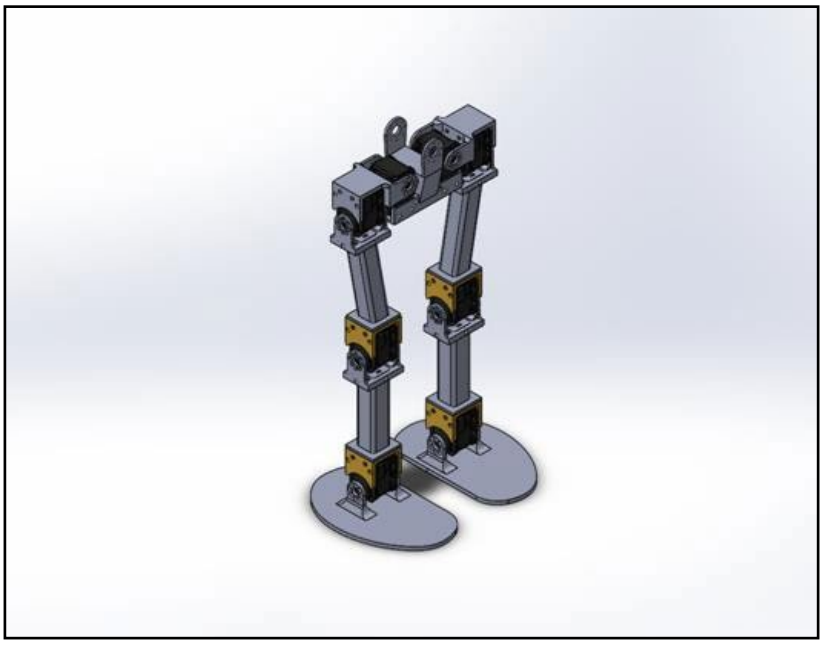


Figure 3.14: Leg Assembly (Isometric View)

- **Pelvis:**

Sub-assembly name	Motor name	Type	ID
Pelvis	r_hip_Y	MX-28AT	8
Pelvis	l_hip_Y	MX-28AT	4

Table 3.6: Pelvis Assembly

- **Right Leg:**

Sub-assembly name	Motor name	Type	ID
Right hip	r_hip_X	MX-28AT	7
Right knee	r_knee_X	MX-28AT	6
Right ankel	r_ankle_X	MX-28AT	5

Table 3.7: Right Leg Assembly

- **Left Leg:**

Sub-assembly name	Motor name	Type	ID
Left hip	l_hip_X	MX-28AT	1
Left knee	l_knee_X	MX-28AT	3
Left ankle	l_ankle_X	MX-28AT	2

Table 3.8: Left Leg Assembly

3.6 Design of Gripper

One of the important parts of humanoid design is design of its gripper which will help it to grab various objects. We decided to design completely symmetrical 3 finger hand so that it can grasp any type of objects without worrying about its orientation. We decide to use string mechanism so that many joints can be controlled passively without consuming extra space and actuators.

Another aspect of gripper design was to control it. Due to space consideration we decided to use only single active controlling actuation which is a servo motor to control actuation of three fingers simultaneously.

All three fingers are string controlled and combine to form single string which is pulled or released by servo motor to move fingers. Due to this hand has Grasp and Release as 2 positions.

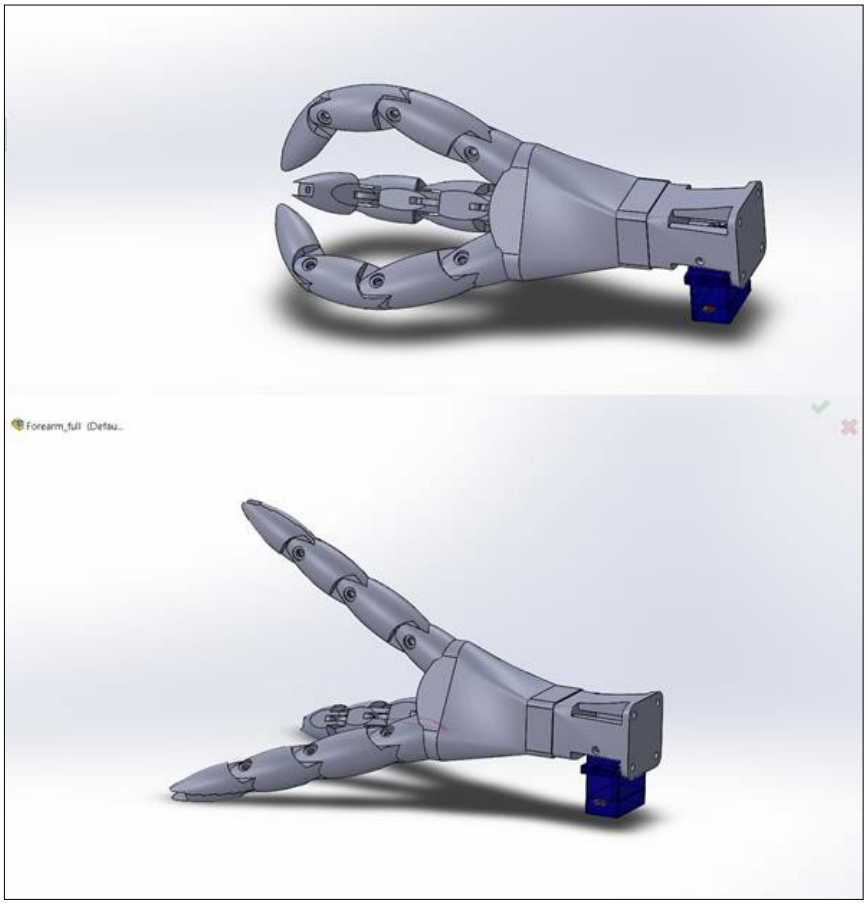


Figure 3.15: Flexion and Extension of Arm

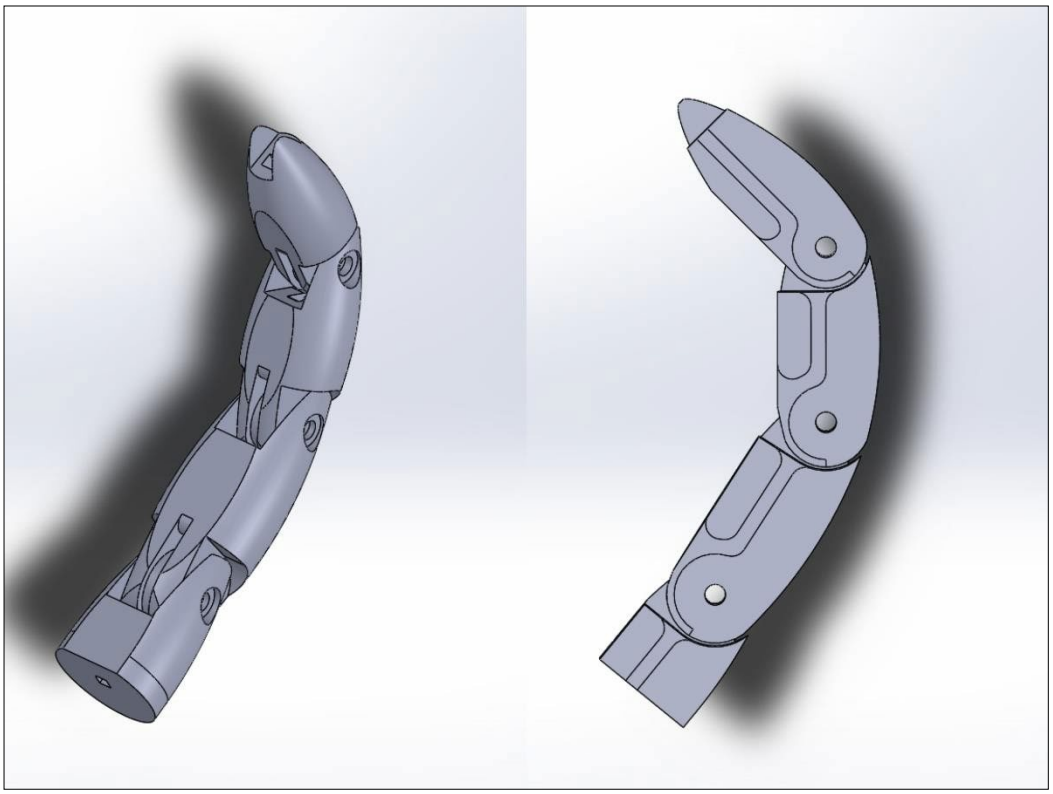


Figure 3.16: Design of 3 DOF Finger with Cross-Section View

There are total 9 degrees of freedom passively controlled by actuator due to this ability of string mechanism hand can easily grasp any shape object. According to shape of objects 9 joints will have different rotations and object will be gripped.

For extension or release purpose spring will be used to extend fingers when servo pull is no longer in effect. When servo will return to its extension position spring force will force finger to extend and achieve extension position.

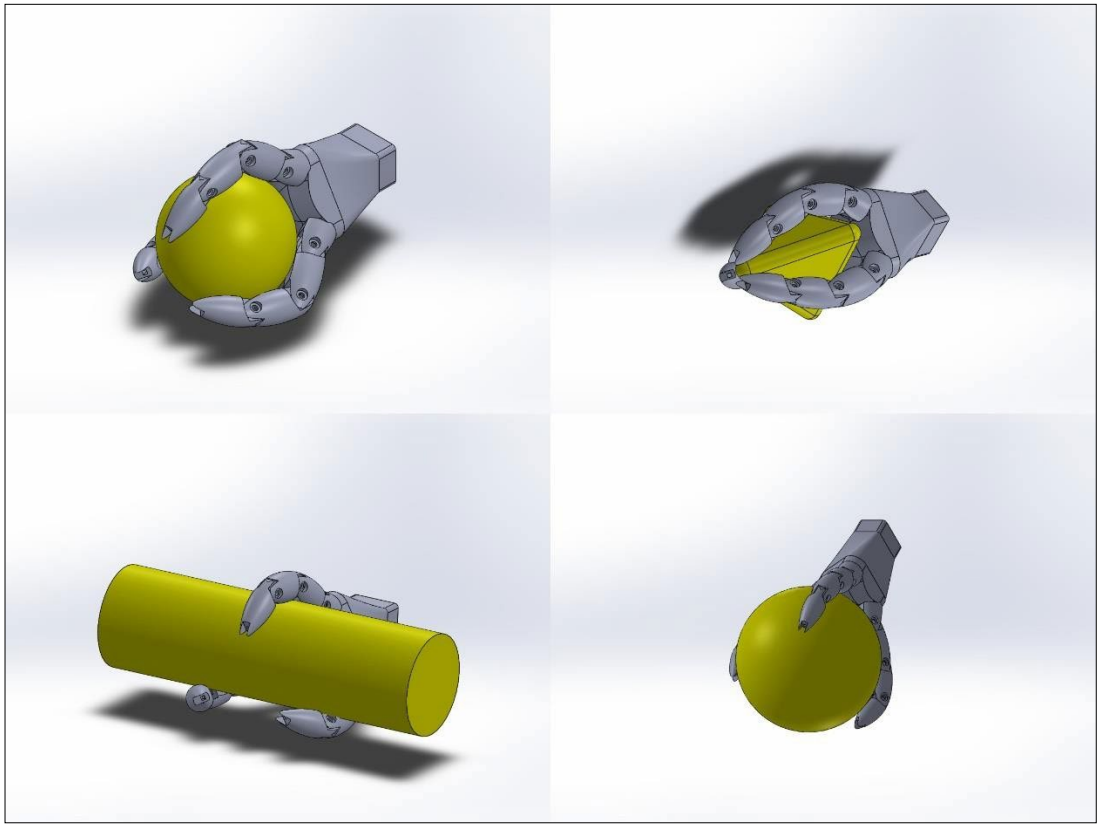


Figure 3.17: Variety of Objects Gripped by Arm

4. SIMULATIONS

4.1 Kinematic Study on Right Arm

As we are going to use our hands for pick, drop and various other application it is necessary to perform kinematic analysis on arm so we can use that framework in programming to automatically move hand to required end positions of hand.

4.1.1 Modelling of Arm

First, we modelled our arm in CAD software and gave required amount of DOFs at required positions so that it will be dextrous and can perform all necessary actions which human arm does.

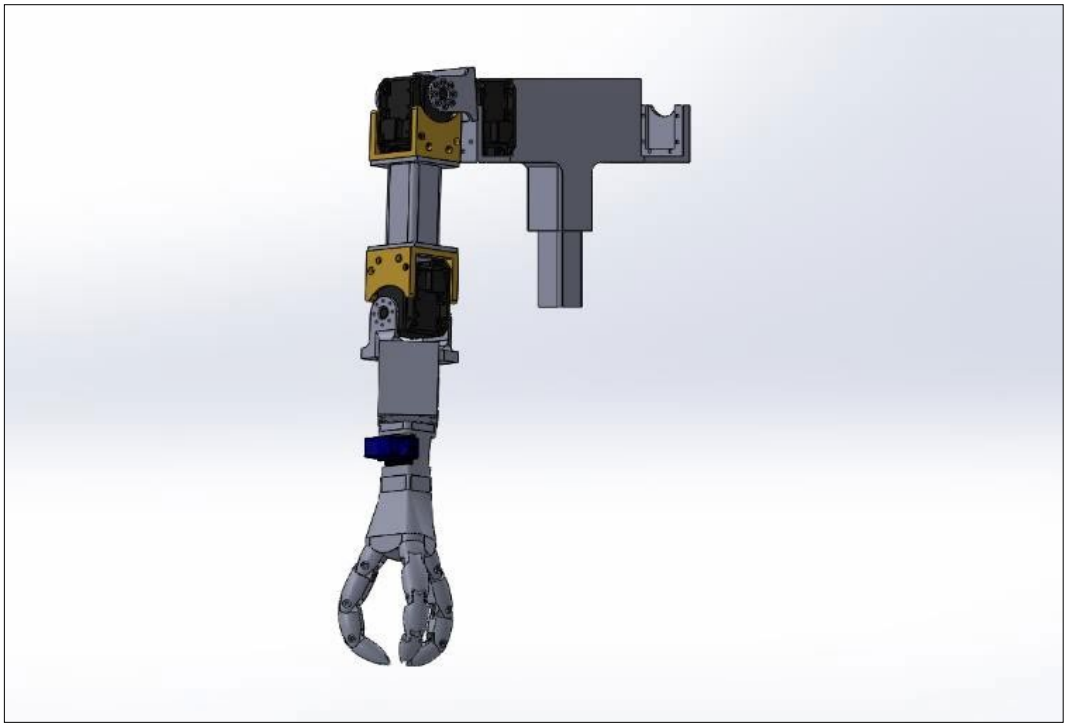


Figure 4.1: CAD Modelling of Right Arm

4.1.2 Kinematic Diagram

Then we converted that model into simpler kinematic diagram to perform kinematic analysis.

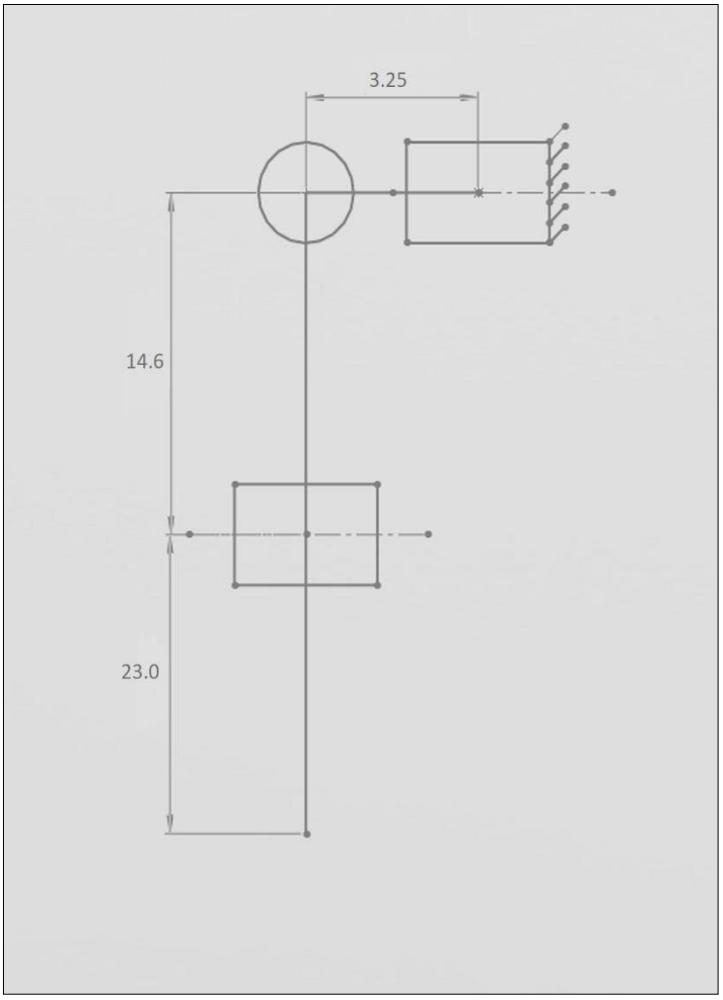


Figure 4.2: Kinematic Diagram of Right Arm

Our arm is 3 revolute joint system (RRR). It has three revolute joints, 2at shoulder and 1at elbow joint. Due to revolute joint every joint angle (θ_i) is variable and taken as θ_1, θ_2, θ_3 respectively.

Further Coordinate systems where assigned according to Denavit-Hartenberg notations.

After assigning coordinate system to all links we can find all Denavit-Hartenberg parameters i.e. link and joint parameters [34].

Now DH parameter Table For given assembly are:

Link	θ_i	d_i	α_i	a_i
1	θ_1	3.125	90	0
2	θ_2	0	90	14.6
3	θ_3	0	90	23

Table 4.1: Denavit-Hartenberg Parameters for Right Arm

4.1.3 Forward Kinematics

Now for Forward Kinematics, we know

$$^{Base}_{3}T = {}^{Base}_{1}T * {}^{1}_{2}T * {}^{2}_{3}T$$

Now,

$$^{Base}_{1}T = (\bar{Z}, \theta_1)Trans(\bar{Z}, 3.125)Rot(\bar{X}90)$$

$$= \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & 25/8 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Also,

$$^{1}_{2}T = (\bar{Z}, \theta_2)Trans(\bar{X}14.6)Rot(\bar{X}90)$$

$$= \begin{bmatrix} C_2 & 0 & S_2 & (73 * C_2)/5 \\ S_2 & 0 & -C_2 & (73 * S_2)/5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Also,

$$^{2}_{3}T = (\bar{Z}, \theta_3)Trans(\bar{X}23)Rot(\bar{X}90)$$

$$= \begin{bmatrix} C_3 & 0 & S_3 & 23 * C_3 \\ S_3 & 0 & -C_3 & 23 * S_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now,

$$^{Base}_{3}T = {}^{Base}_{1}T * {}^{1}_{2}T * {}^{2}_{3}T$$

$$= \begin{bmatrix} FS\,S + C\,C\,C & C\,S & C\,C\,S - C\,S & \frac{73 * C_1C_2}{5} + 23 * S\,S + 23 * C\,C\,C\,1 \\ I^{1\ 3} \quad {}^{1\ 2\ 3} & {}^{1\ 2} & {}^{1\ 2\ 3} \quad {}^{3\ 1} & {}^{1\ 3} \quad {}^{1\ 2\ 3}\ I \\ I C\,C\,S - C\,S & S\,S & C\,C + C\,S\,S & \frac{73 * C_2S_1}{5} - 23 * C\,S + 23 * C\,C\,S\ I \\ I^{1\ 3\ 1} \quad {}^{1\ 3} & {}^{1\ 2} & {}^{1\ 3} \quad {}^{2\ 1\ 3} & {}^{1\ 3} \quad {}^{2\ 3\ 1}\ I \\ I & & & \frac{73 * S_2}{5} \\ I \quad C_3S_2 & -C_2 & S_2S_3 & + 23 * C_3S_2 + 25/8 \quad I \\ [\quad 0 & 0 & 0 & 1 \quad] \end{bmatrix}$$

This is final equation for forward Kinematics of position and orientation of hand corresponding to the Shoulder joint.

4.1.4 Inverse Kinematics

Now for Inverse Kinematics,

$$ {}^{Base}_{3}T = \begin{bmatrix} R_{11} & R_{12} & R_{13} & q_x \\ R_{21} & R_{22} & R_{23} & q_y \\ R_{31} & R_{32} & R_{33} & q_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Now,

$$ q_x = \frac{73 * C_1C_2}{5} + 23 * S_1S_3 + 23 * C_1C_2C_3 $$

$$ q_y = \frac{73 * C_2S_1}{5} - 23 * C_1S_3 + 23 * C_2C_3S_1 $$

$$ q_z = \frac{73 * S_2}{5} + 23 * C_3S_2 + 25/8 $$

By solving above equations simultaneously we can find values of θ_1, θ_2, θ_3 for required coordinates q_x, q_y, q_z as follows.

$$ \theta_3 = cos^{-1}\left(\frac{14.6^2 + 23^2 - q_x^2 - q_y^2 - (q_z - 3.125)^2}{2 * 14.6 * 23}\right) $$

$$ \theta_2 = sin^{-1}\left(\frac{q_z - 3.125}{14.6 + 23 * (\theta_3)}\right) $$

$$ \theta_1 = tan^{-1}\left(\frac{q_y}{q_x}\right) - tan^{-1}\left(\frac{23 * \sin(\theta_3)}{\cos(\theta_2) * (14.6 + 23 * \cos(\theta_3))}\right) $$

We can use above values of θ_1, θ_2, θ_3 for required coordinates q_x, q_y, q_z.

4.2 Kinematic Study on Right Leg

As we know legs are important part of a humanoid robot to make it mobile and capable to perform various assigned tasks, performing kinematic study on leg will provide us necessary parameters for path generation.

4.2.1 Modelling of Leg Assembly

First, we modelled our leg in CAD software and gave required amount of DOFs at required positions so that it will be dexterous and can mimic human motion.

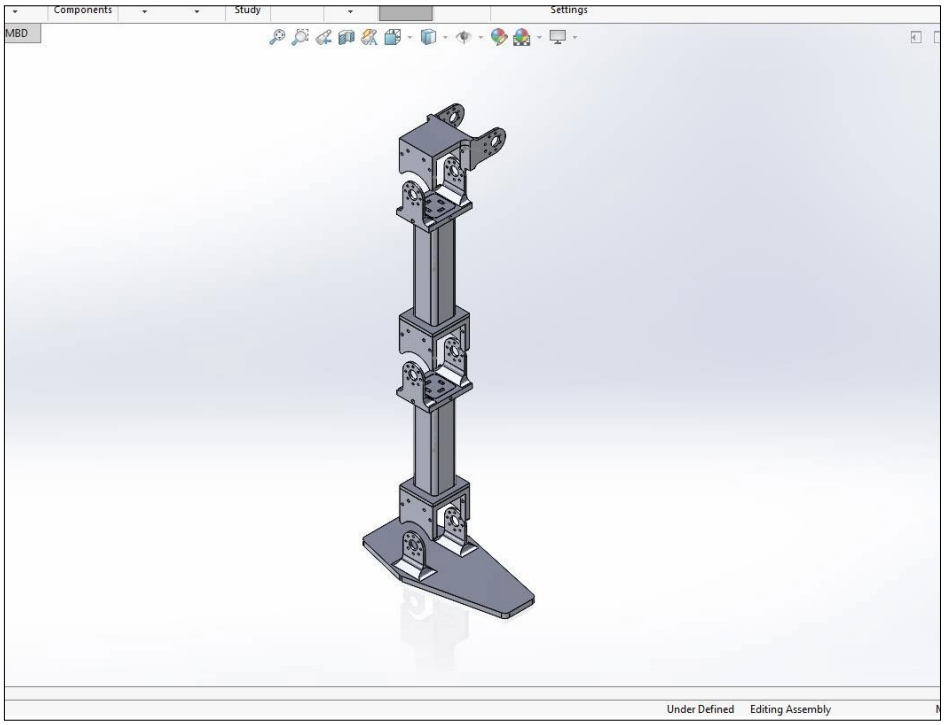


Figure 4.3: CAD Modelling of Right Leg

4.2.2 Kinematic Diagram

Then we converted that model into simpler kinematic diagram to perform kinematic analysis. We chose to consider only motion in sagittal plane as motion of leg will mostly will be in that plane and it will make calculations much easier.

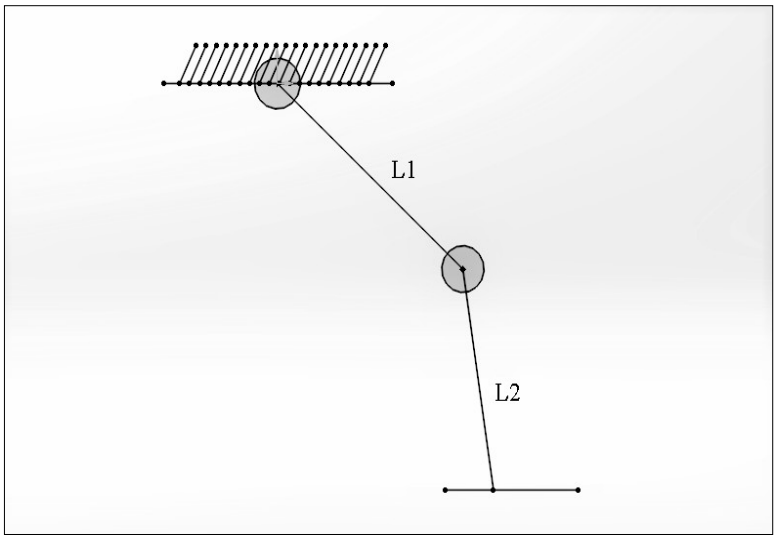


Figure 4.4: Kinematic Diagram of Right Leg

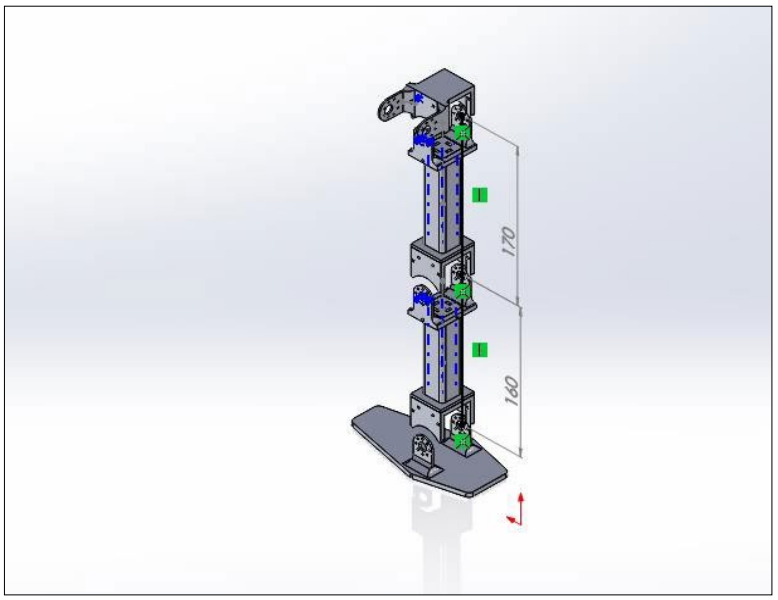


Figure 4.5: Length of links

Here in this leg Assembly, Due to revolute joints Joint angle (θ_i) is variable. And its value is θ_1 for first revolute joint. And θ_2 for second Revolute Joint.

Further Coordinate systems where assigned according to Denavit-Hartenberg notations [34].

Now parameter Table For given assembly are:

Frame	θ_i	d_i	α_i	a_i
1	θ_1	0	0	0.17
2	θ_2	0	0	0.16

Table 4.2: DH Parameters for Leg

4.2.3 Forward Kinematics

Now for Forward Kinematics, we know

$$^{Base}_{2}T = {}^{Base}_{1}T * {}^{1}_{2}T$$

Now,

$$^{Base}_{1}T = (\bar{Z}, \theta_1)Trans(\bar{X}.17)$$

$$= \begin{bmatrix} C_1 & -S_1 & 0 & .17 * C_1 \\ S_1 & C_1 & 0 & .17 * S_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Also,

$$^{1}_{2}T = (\bar{Z}, \theta_2)Trans(\bar{X}.16)$$

$$= \begin{bmatrix} C_2 & -S_2 & 0 & .16 * C_2 \\ S_2 & C_2 & 0 & .16 * S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now,

$$^{Base}_{2}T = {}^{Base}_{1}T * {}^{1}_{2}T$$

$$= \begin{bmatrix} C_{12} & -S_{12} & 0 & .17 * C_1 + .16 * C_{12} \\ S_{12} & C_{12} & 0 & .17 * S_1 + .16 * S_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This is final equation for forward Kinematics of position of foot corresponding to the Hip joint.

4.2.4 Inverse Kinematics

Now for Inverse Kinematics,

$$^{Base}_{2}T = \begin{bmatrix} C_\phi & -S_\phi & 0 & q_x \\ S_\phi & C_\phi & 0 & q_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now,

$$q_x = .17C_1 + .16C_{12}$$

$$q_y = .17S_1 + .16S_{12}$$

Squaring both the sides and adding,

$$q_x^2 + q_y^2 = .17^2 + .16^2 + 2 * .17 * 16 * C_1 * C_{12} + 2 * .17 * .16 * S_1 * S_{12}$$

Computing we get,

$$C_2 = \frac{q_x^2 + q_y^2 - .17^2 - .16^2}{2 * .17 * .16}$$

i.e.

$$C_2 = \frac{q_x^2 + q_y^2 - .0545}{0.0544}$$

Taking Inverse, we get,

$$\theta_2 = \arccos\left(\frac{q_x^2 + q_y^2 - .0545}{0.0544}\right)$$

This is the value of θ_2

Now,

$$q_x = .17C_1 + .16C_1C_2 - .16S_1S_2$$

$$= C_1(.17 + .16C_2) - S_1(.16S_2)$$

Let,

$$\rho = \sqrt{(.17 + .16C_2)^2 + (.16S_2)^2}$$

$$T = arctan\left(\frac{.17 + .16C_2}{.16S_2}\right)$$

Therefore,

$$q_x = \rho C_1 sinT - \rho S_1 cosT = \rho \sin(T - \theta_1)$$

Similarly,

$$q_y = \rho \cos(T - \theta_1)$$

Therefore,

$$\theta_1 = T - arctan\left(\frac{q_x}{q_y}\right)$$

By this equation we can find value of θ_1.

Using Equations of θ_1& θ_2 from Inverse kinematics, we can find orientation of link 1 and link 2 for given position of foot.

4.3 Kinematic Simulation Using Numerical computing environment (MATLAB)

As we can see, equations of kinematics are very complicated and requires a lot of calculations in terms of matrices. So Numerical computing software (MATLAB) was used to perform kinematics.

4.3.1 MATLAB

MATLAB (*matrix laboratory*) is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.

4.3.2 Modelling in MATLAB

Robotics toolbox is used for plotting and calculation of robot in MATLAB. It uses dh parameters to form serial link robots and we can perform calculations on them.

Modelling of Arm

Following program is an example for plotting serial robot i.e. right arm of humanoid in MATLAB.

```
%Program to plot and model of right arm

startup_rvc

dh=[
    0 3.125 0 1.571
    0 0 14.6 1.571
     0 0 23 1.571
    ]
k = SerialLink(dh,'name','Right arm');
k.plot([0 0 0],'tilesize',1,'basecolor',[1 1
1],'linkcolor',[0 0 0],'basewidth',1)
view([0 1 0]); k.teach;
```

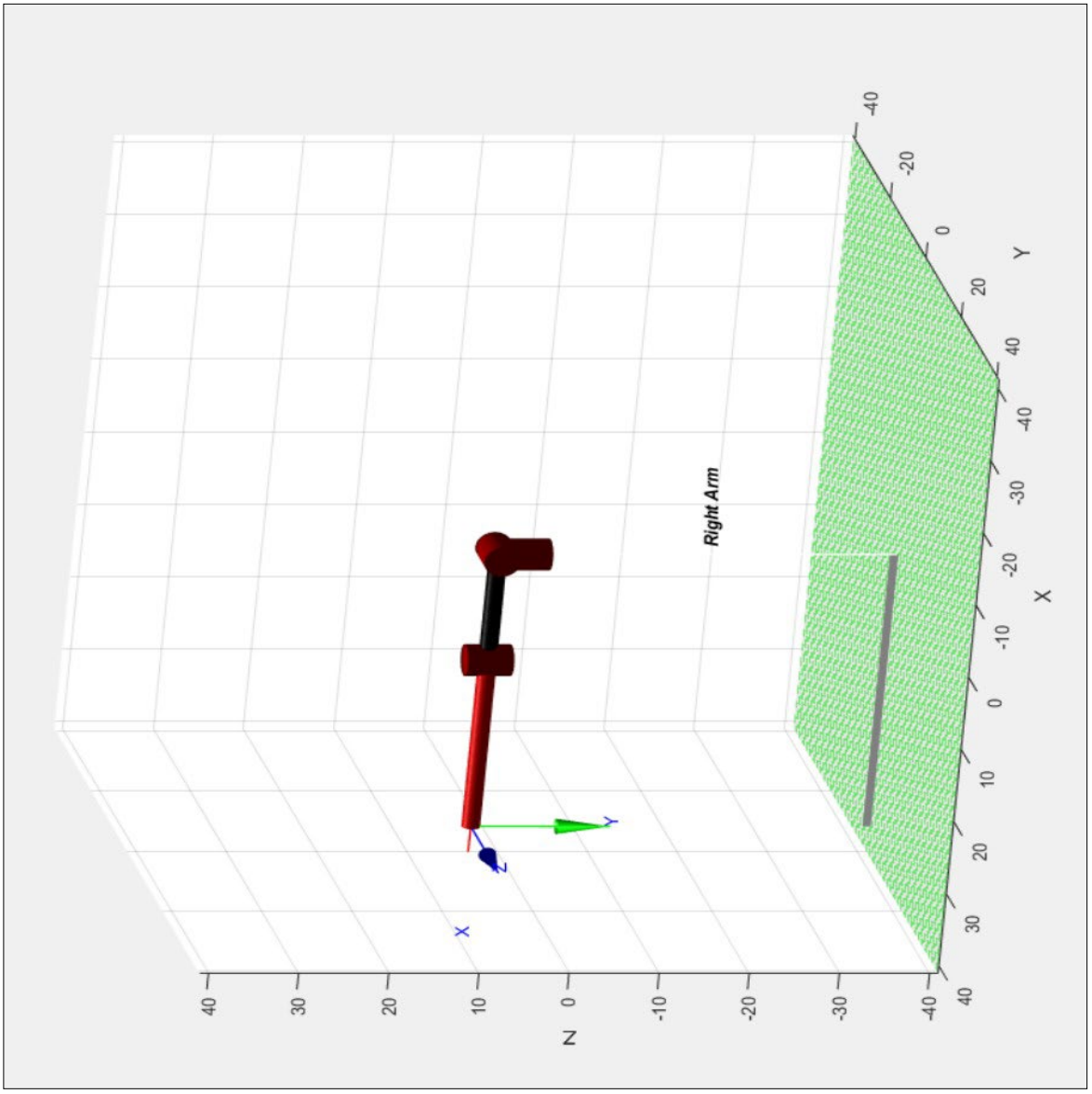


Figure 4.6: Modelling of Arm in MATLAB

4.3.3 Transformation matrix of dh parameters function

As we know transformation matrix for Dh parameter is complex matrix of trigonometric functions. we wrote a function to calculate transformation matrix to simplify calculations in forward kinematics.

```
%transformation matrix for dh notations where theta is
variable and should be of syms data type.
function [matrix]=Trans_mat(theta_deg,d,a,alpha_deg)
theta=theta_deg*pi/180;   %to convert degree into radians

alpha=alpha_deg*pi/180;   %to convert degree into radians

matrix=[cos(theta) -sin(theta)*cos(alpha) sin(theta)
a*cos(theta)
    sin(theta) cos(theta)*cos(alpha) -
cos(theta)*sin(alpha) a*sin(theta)
    0 sin(alpha) cos(alpha) d
    0 0 0 1];
end
```

this function returns the transformation matrix of given DH parameters.

4.3.4 Forward Kinematics

As we know matrix calculations are easier on MATLAB so, we wrote a MATLAB program to find final transformation matrix in forward kinematics to know final position of end effector for given set of values of θ_i.

this is program for finding out Final transformation matrix of forward kinematics of arm of humanoid.

```
%Program for finding out final transformation matrix of
forward kinematics
clc
symstheta1;
assume(in(theta1,'real') & theta1>(-pi) & theta1<pi);
symstheta2;
assume(in(theta2,'real') & theta2>(-pi) & theta2<pi);
symstheta3;
assume(in(theta3,'real') & theta3>(-pi) & theta3<pi);
t1=Trans_mat(theta1,3.125,0,90)
t2=Trans_mat(theta2,0,14.6,90)
t3=Trans_mat(theta3,0,23,90)
t=t1*t2*t3       %final transformation matrix
```

4.3.5 Inverse kinematic program

Forward kinematics of robot arm results in complex equations of trigonometric functions so following MATLAB program is used for finding out values of various joint parameters corresponding to different values of position of end effector.

```
%program to find values theta1 theta2 theta3 for a
particular value of x y %z coordinates
clc
symstheta1;
assume(in(theta1,'real') & theta1>(-180) & theta1<180);
symstheta2;
assume(in(theta2,'real') & theta2>(-180) & theta2<180);
symstheta3;
assume(in(theta3,'real') & theta3>(-180) & theta3<180);

t1=Trans_mat(theta1,3.125,0,90)
t2=Trans_mat(theta2,0,14.6,90)
t3=Trans_mat(theta3,0,23,90)
t=t1*t2*t3
x=35;    %X co-ordinate
y=5;    %Y co-ordinate
z=3.25; %Z co-ordinate
equ_for_x = t(1,4)== x
equ_for_y = t(2,4)== y
equ_for_z = t(3,4)== z
sol=solve([equ_for_x,equ_for_y,equ_for_z],[theta1,theta2,t
heta3]);
sol
```

this solution gives all possible combination values of θ_1, θ_2, θ_3 for particular position of end effector.

4.3.6 Workspace Calculations

Workspace is important criteria for inspecting effectiveness of Robot (Manipulator). We have done workspace analysis on arm of robot to know which region is accessible for manipulator to perform pick and drop motion. Knowing workspace is important so that robot can autonomously decide whether the object is within reachable zone or not.

Normally, workspace is calculated by manually considering inner most and outermost regions in 2d views. But in this case, we found it difficult to find workspace just using 2d views as hand consists 3 degrees of freedom for motion in 3 dimensions [35].

So instead of general manual method we decided to find workspace using iterative method. In this method, we used transformation matrix of arm and using random values of θ_1, θ_2, θ_3 so that we can see point which can be reached by our arm. We iterated this process thousands of time so that we can find approximate volume which can be swept by arm [36].

```
%program to find workspace using iterative method
clc;
clear;
startup_rvc
dh=[
    0 3.125 0 1.571
    0 0 14.6 1.571
     0 0 23 1.571
    ]
k = SerialLink(dh,'name','Right Arm')
k.plot([0 0 0],'tilesize',1,'basecolor',[1 1 1],'linkcolor',[0 0
0],'basewidth',1)
hold on
k.teach
hold  on
syms theta1;
assume(in(theta1,'real') & theta1>(-pi) & theta1<pi);
syms theta2;
assume(in(theta2,'real') & theta2>(-pi) & theta2<pi);
syms theta3;
assume(in(theta3,'real') & theta3>(-pi) & theta3<pi);
```

```
t1=Trans_mat(theta1,3.125,0)
t2=Trans_mat(theta2,0,14.6)
t3=Trans_mat(theta3,0,23)
t=t1*t2*t3
x = t(1,4);
y = t(2,4);
z = t(3,4);
th1_min=(-pi);
th1_max=pi;
th2_min=0;
th2_max=(pi);
th3_min=(-pi/2);
th3_max=(pi/2);
iteration=0;
total_itr=5000;
for i=1:1:2000
    th1=th1_min+rand*(th1_max-th1_min);
    th2=th2_min+rand*(th2_max-th2_min);
    th3=th3_min+rand*(th3_max-th3_min);
    theta1=th1;
    theta2=th2;
    theta3=th3;
    xplot=subs(x);
    yplot=subs(y);
    zplot=subs(z);
    scatter3(xplot,yplot,zplot,50,'MarkerFaceColor',[1                1
0],'MarkerEdgeColor','k');
    hold on
    iteration=iteration+1;
    fprintf('iteration %8.2f out of %8.2f \n',iteration,total_itr);
end
```

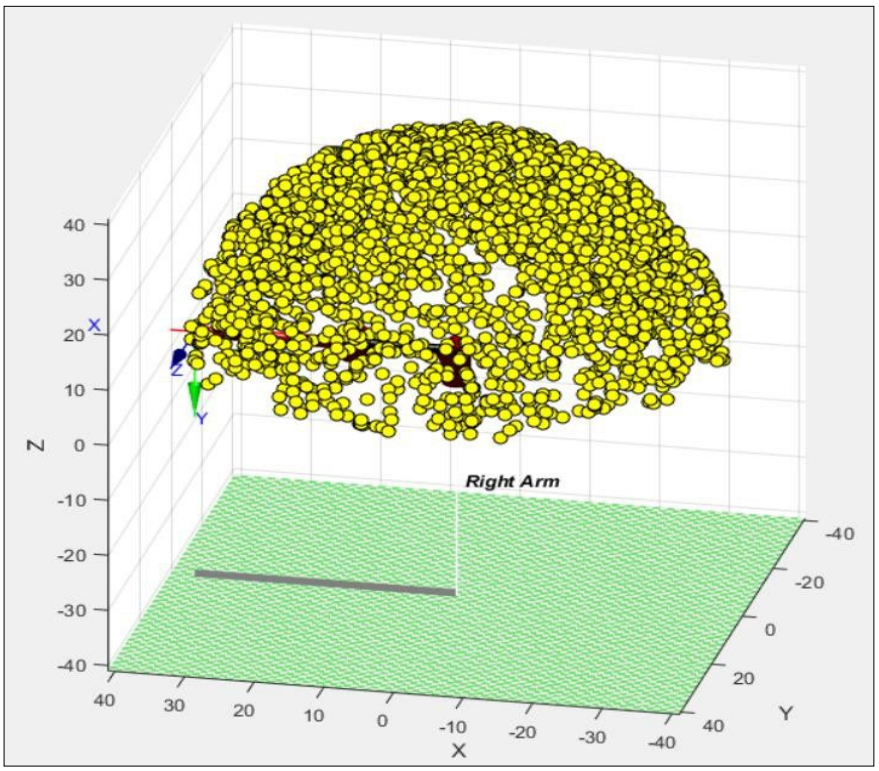


Figure 4.7: Workspace using Iterative Method in MATLAB

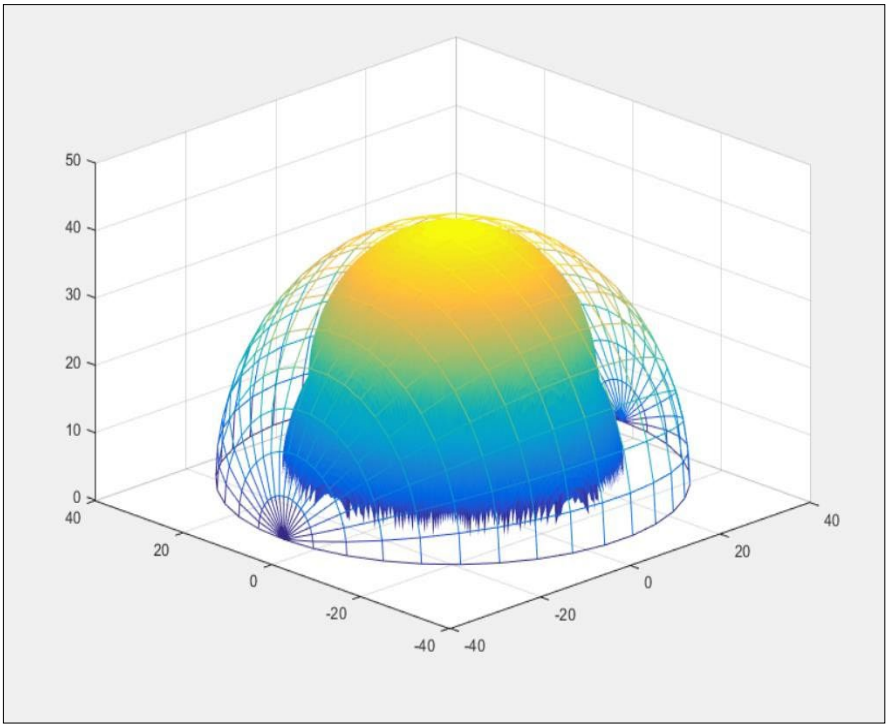


Figure 4.8: Workspace in MATLAB: Outer limit (Lined region) and Inner limit (Solid region)

4.4 Structural Analysis

After designing Humanoid Part, we performed mechanical structural simulation on various parts. 3d Printed parts are not single piece of material but a layer by layer deposition of plastic. Due to this, it exhibits different mechanical properties for different orientation, Shell thickness, Fill density, and fill structure. So, it is Difficult to find exact strength of 3d printed parts. Also, there is possibility of manufacturing defects and errors in 3d printing which also could result in less strength so we decided to keep huge factor of safety for those parts.

We had different types of parts to be 3D printed in our project, so we used different types of setting to manufacture those parts. Also, we kept orientation such that Force on Part was perpendicular to the layers of 3d printing. As it will have more strength [37].

Mostly we divide our parts in two different categories according to their structure. First is Skeleton Joints and other is skeleton Bones. Skeleton joints were Complex parts with very thin cross section and less cross section area. Skeleton bones were comparatively simple structures with large cross section areas.

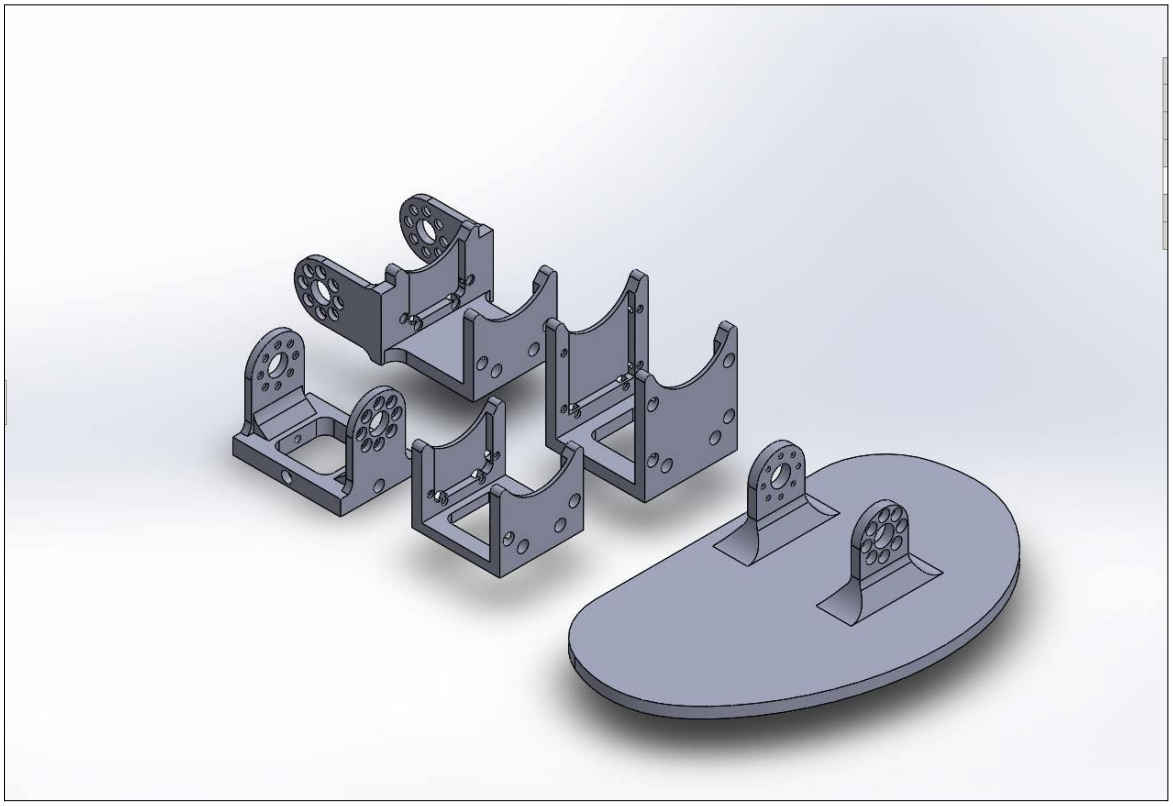


Figure 4.9: Skeleton Joint Parts

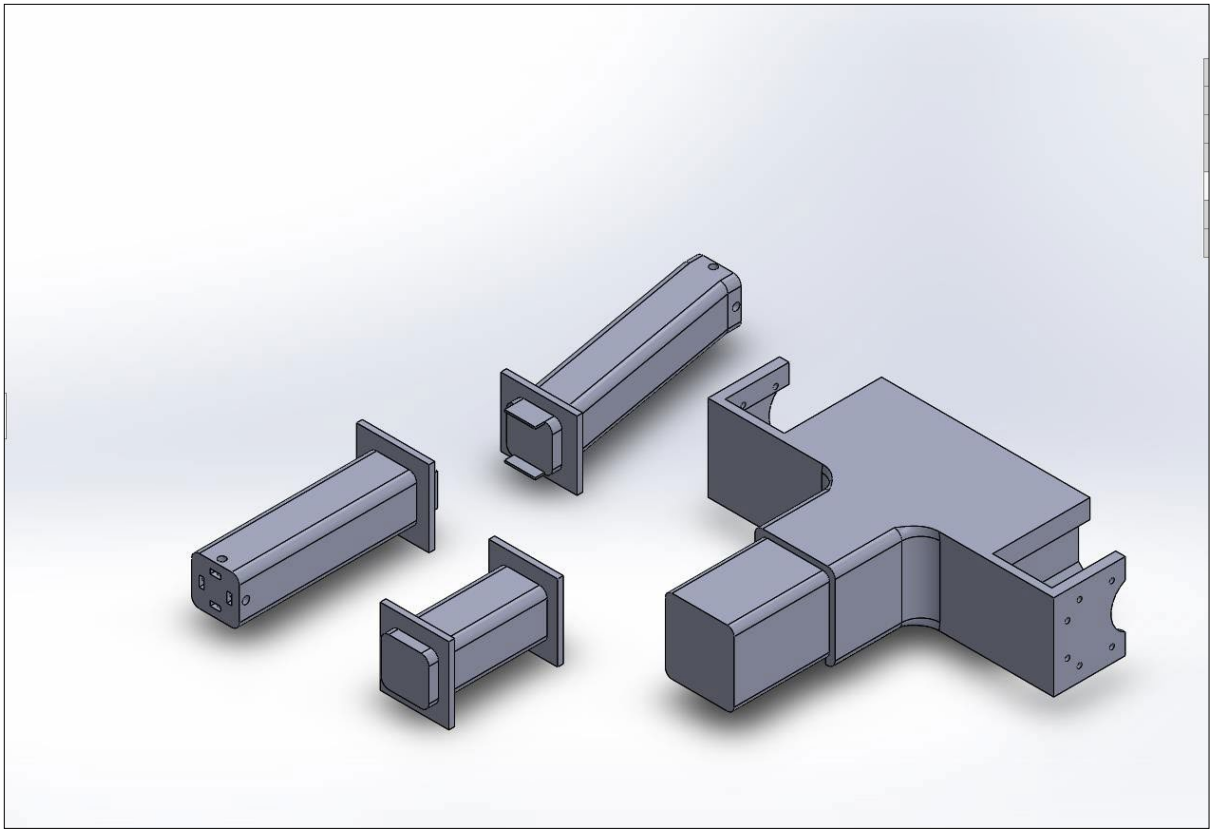


Figure 4.10: Skeleton Bone Parts

Now to increase strength of Joint parts we kept shell thickness of those parts of 2mm and fill density of 20%. For Bone parts we kept Shell thickness 0.5 mm and fill density of 15% to reduce weight.

In our humanoid major force is self-weight due to weight of motors battery and other 3d Printed parts. So, we performed structural analysis on parts with compressive force. Our esteemed weight is maximum 3Kg considering weight of battery. Which results in 30N compressive Force. which was applied on those parts.

4.4.1 Structural Analysis of Foot

We decided to perform structural analysis on foot part as was the joint part which was supposed to take weight of entire body. So, we applied force on hole of shaft which is going to transfer force on part. We fixed base of foot as it is going to rest on ground.

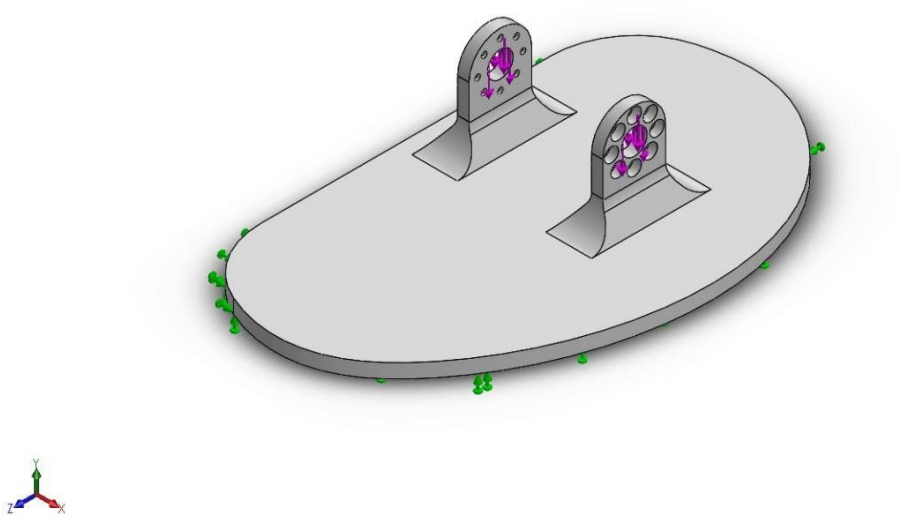


Figure 4.11: Force Diagram of Foot

We applied all material properties of PLA while considering fill density of 80% due to small cross sections having with 2 mm shell thickness resulting in approximately 100% fill. Compressive strength for PLA 3D printed parts with 80% fill density is 43.19 MPa [38]. Which is used for analysis.

Results were as follows:

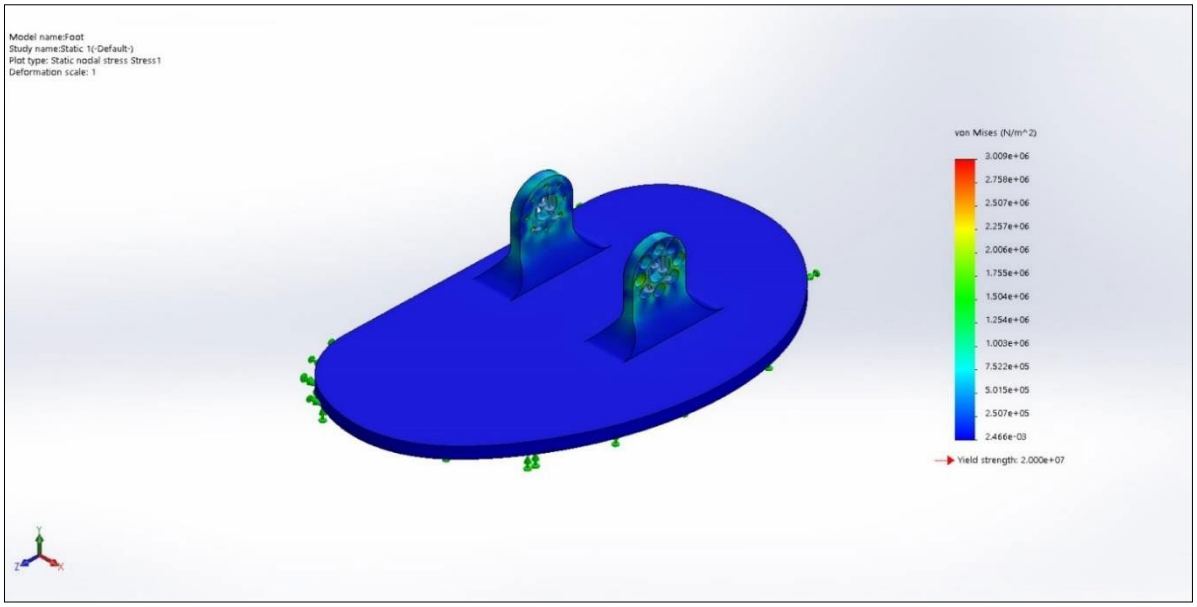


Figure 4.12: Stress Plot for Foot

Maximum stress was found to be 3.009 MPa which is much lower than permissible strength.

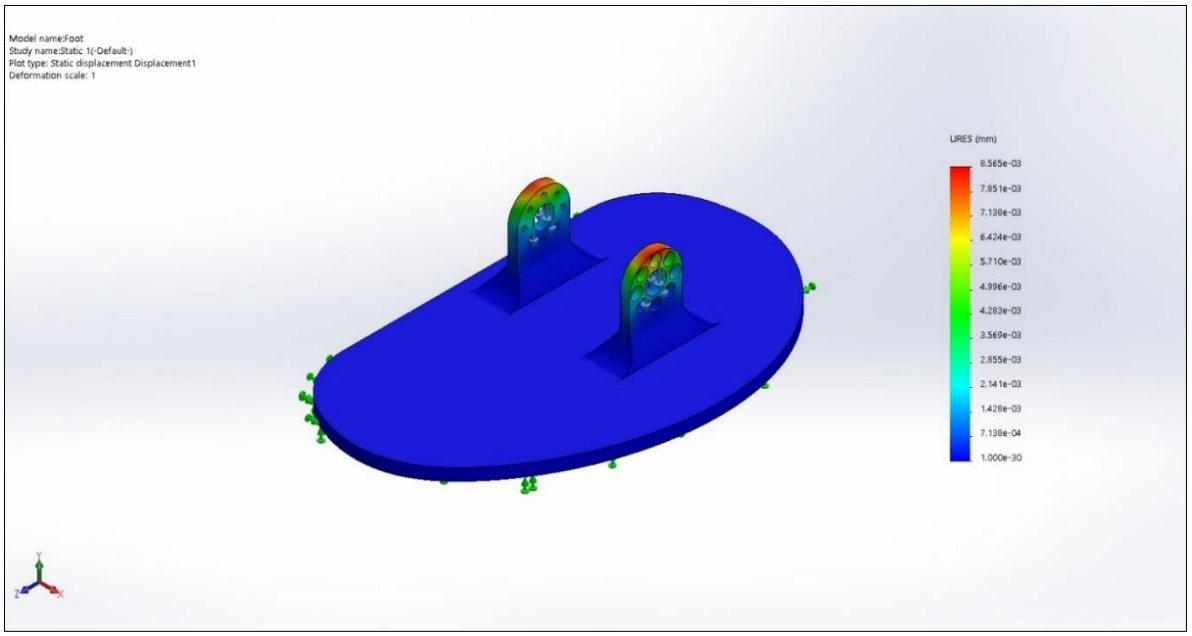


Figure 4.13: Displacement Plot for Foot

Maximum Displacement was Found to be 8.565 micron which is almost negligible.

4.4.2 Structural Analysis on Leg Bone

We also performed structural analysis on leg bone as it is one of the bones of lowest parts having weight of whole body. For safety we took Compressive force of 30N on Upper side which is transmitting force on body and fixed lower part of bone.[2]

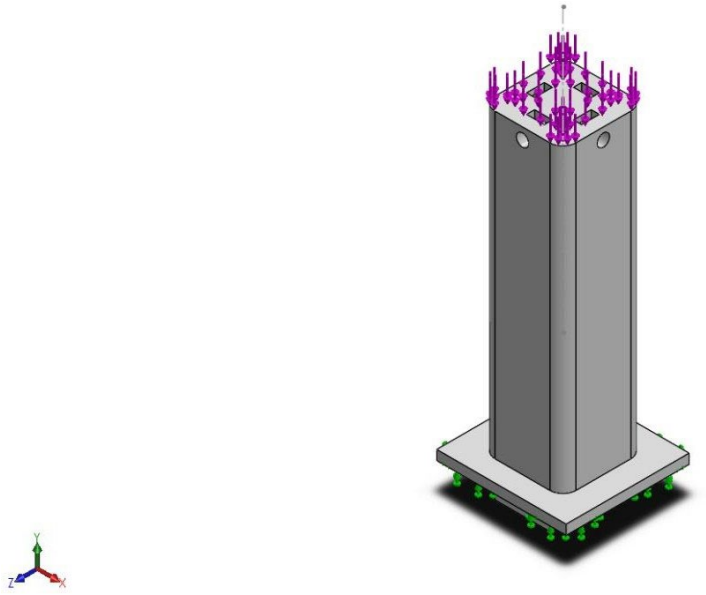


Figure 4.14: Force Diagram for Leg Bone

After applying Forces, we applied all material properties of PLA while considering fill density of 20% due to large cross sections having with 0.5 mm shell thickness and 20% fill Density resulting in approximately 20% fill. Compressive strength for PLA 3D printed parts with 20% fill density is 12.63 MPa [38]. Which is used for analysis.

Results were as follows:

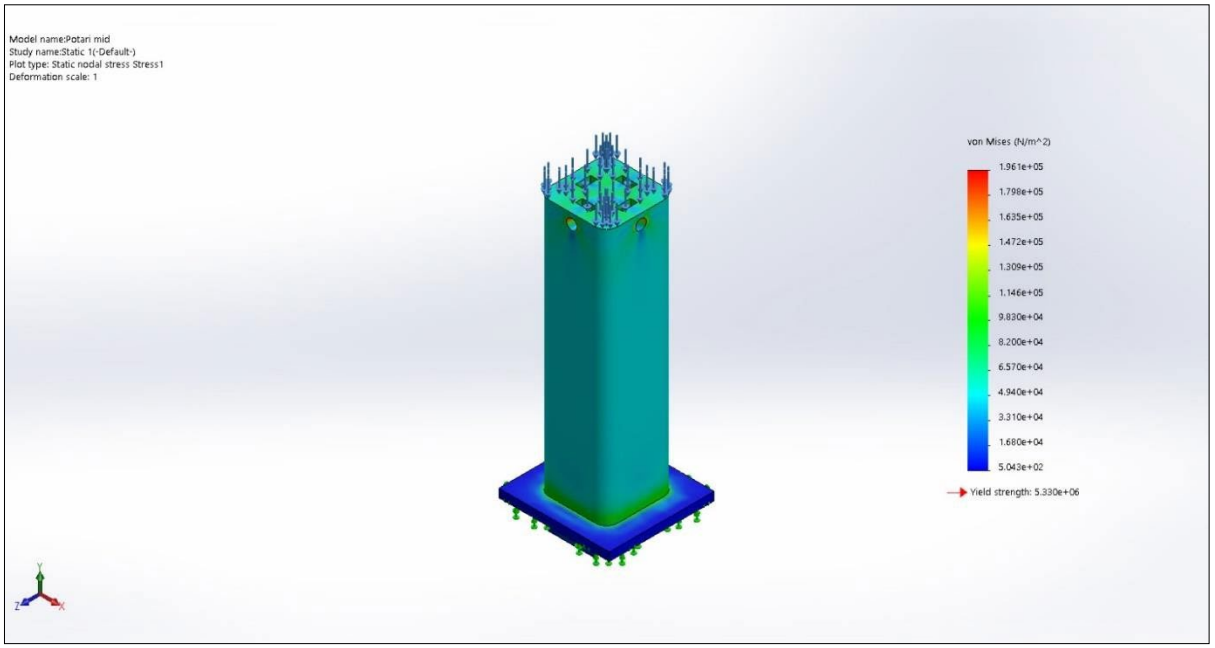


Figure 4.15: Stress Plot for Leg Bone

Maximum stress was found to be 0.1961 MPa which is much lower than permissible strength.

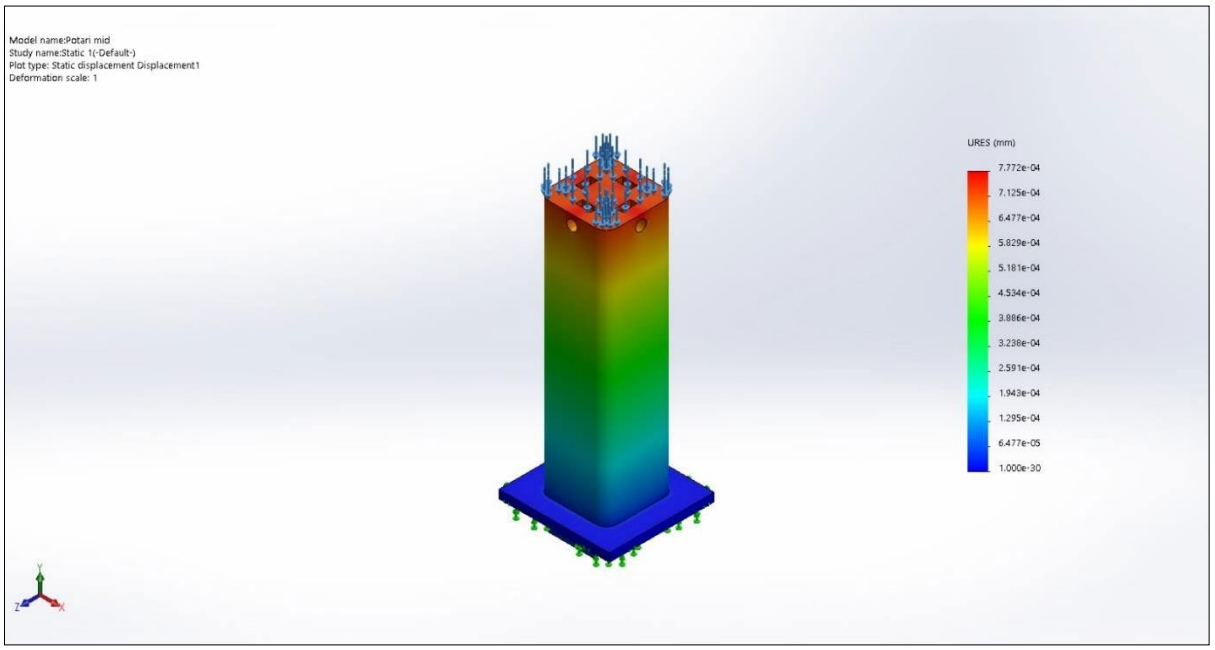


Figure 4.16: Displacement Plot for Leg Bone

Maximum Displacement was Found to be 0.7772 micron which is almost negligible.

Considering above results, we concluded that our design is working satisfactorily with respect mechanical structural and Kinematical aspect so proceeded with Manufacturing of parts of Humanoid.

5. WORKING OF HUMANOID

5.1 Manufacturing

After Designing and completing Structural analysis with satisfactory results, our next step towards making Humanoid functional was manufacturing process. We have used 3D printing technology to manufacture our parts. LulzBot Taz 6 3D printer based on FDM (Fused Deposition modelling) is used to serve the purpose.

We have used PLA (Polylactic Acid) material having diameter 2.85mm which has given results in feasible range in structural analysis.

We have divided manufacturing in two parts according to different manufacturing settings required into Skeleton Joints and skeleton Bones. Skeleton joints were Complex parts with very thin cross section and less cross section area. Skeleton bones were comparatively simple structures with large cross section areas.

Considering results of structural analysis, we decided to use different types of setting to manufacture our parts. The Important settings are as follows

1) Keeping orientation such, that Force on Part was perpendicular to the layers of 3d printing. As it will have more strength.
2) To increase strength of Joint parts which have less cross section we kept shell thickness of those parts of 2mm and fill density of 20%.
3) Bone parts having large cross section are printed with Shell thickness 0.5 mm and fill density of 15% to reduce weight.
4) Other settings Such as nozzle temperature, bed temperature, speed, are also considered for better print strength and quality.

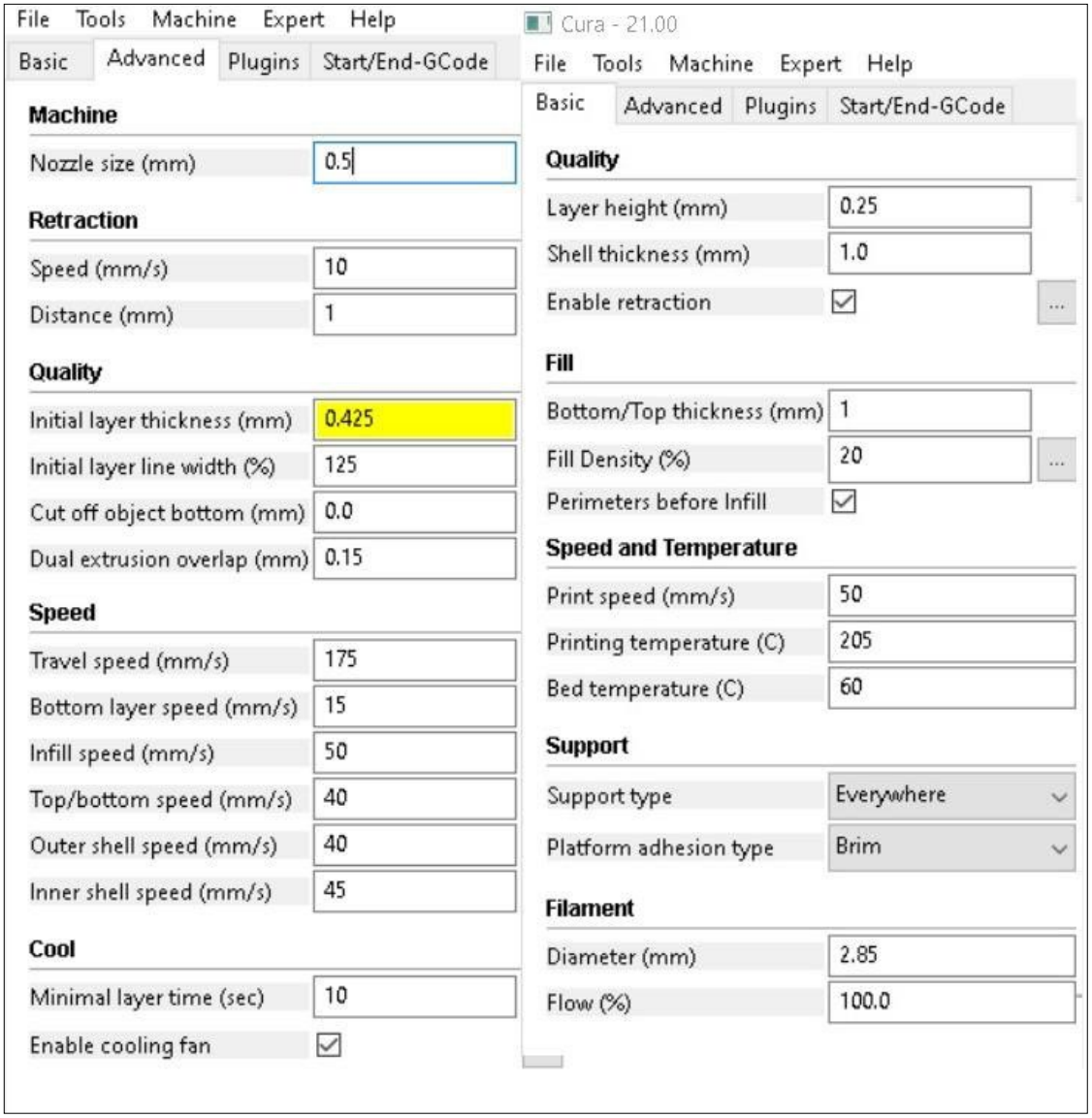


Figure 5.1: Cura Settings

Above image show different settings and their values in Cura software which we have used to reduce error and manufacture parts with maximum strength.

5.1.1 Manufactured parts

Figure 5.2: Upper half and leg assembly

Above fig. shows upper half leg parts which are manufactured with good surface finish and sufficient strength.

The Clamp printed with 20% density is able to sustain the weight of the body.

The bones having comparatively lager sections then clamp manufactured with 15% density and 0.5mm shell thickness are forming links between joints and have helped to reduce overall weight of the lower half.

5.1.2 Assembly with Actuators

Figure 5.3: Full assembly with actuators

Above figure shows combined full assembly with actuators of all manufactured clamp, holders, bones and actuators. Lower body consists of 8 MX28AT actuators and a MX64AT motor. All themotors are connected in series and are powered using 12v supply.

5.2 Interfacing of Electronic Components

After complete manufacturing process we decide to assemble all actuators that include Dynamixels (MX28AT, MX64AT, and AX12), SS90 servos and sensors (Ultrasonic) that will drive and give feedback to control motion of our humanoid.

5.2.1 Battery

The whole circuit is powered by means of lithium polymer (liPo) battery whose capacity is 5200mAh and gives 12V voltage. The battery is selected using following criteria.

Selection of Battery

Battery was one of the most important components to be selected while designing electronic circuit. We had different components which had different power requirement. The main two component which required power was Raspberry Pi 3 and Dynamixel Actuators.

Dynamixel Servos
12 Volts
100 mA (Standby)

We decided to use 12 Volt battery and step-down Voltage Regulator so that we can get all required voltages. We decided to use 3cell 12-volt Lithium Polymer battery due to its low weight and high reliability.

Next important decision was to select capacity of battery as it is going directly going to affect Battery life of humanoid. For this we considered 3 modes of Operations of humanoid

i) Only Controller ON
ii) Controller ON and Servos on Standby
iii) Controller ON and Servos Under Load

These modes had their different Current requirements. After doing Calculations we selected LiPo 5200 mAh Battery with 40C Capacity. Its maximum continues current discharge is 208A which is more than enough. Battery Life with different modes were as shown further.

Controller ON Mode

With only controller on we can save power in idle Condition ready to perform actions when command comes. In this Mode We only required 5 Volts and 2.5 A.

$$Output\ Power\ Required = 5\ Volts * 2.5\ Ampere$$

$$Output\ Power\ Required = 12.5\ Watts$$

Efficiency of Voltage regulator we selected, is 92% (maximum). Considering Efficiency of 80% we get,

$$Input\ Power\ Required = Output + Losses$$

For, 12.5 watts Output, we get loss of 3.125 at 80% efficiency, substituting we get

$$Input\ Power\ Required = 12.5 + 3.125 = 15.625\ Watts$$

Now, Our Battery has average voltage of 11.1 volts which Gives continues current requirement of,

$$Current\ Drawn\ From\ Battery = \frac{15.625\ Watts}{11.1\ Volts}$$

$$Current\ Drawn\ From\ Battery = 1.4077\ Ampere$$

For this Current requirement we can find Battery Life Time as,

$$Battery\ Life\ Time = \frac{5200\ mAh}{1407.7\ mA}$$

$$Battery\ Life\ Time = 3.694\ hrs$$

Thus, we get a Battery Life Time of 3 Hrs and 42 mins.

Controller ON and Servos on Standby

in this mode servos are on standby mode and will perform any motion if command is given. In standby mode servos consumes 100 mA current and total no. of servos is 17. Which results in

$$Total\ Current = 1.4077 + 1.700 = 3.1077\ A$$

For this Current requirement we can find Battery Life Time as,

$$Battery\ Life\ Time = \frac{5200\ mAh}{3107.7\ mA}$$

$$Battery\ Life\ Time = 1.673\ hrs$$

Thus, we get a Battery Life Time of 1 Hrs and 41 mins.

Controller ON and Servos Under Load

In this mode servos are performing actions. In this Mode current Calculations are fairly difficult as current consume by servos is proportional to Load on servo which is continuously changing. Assuming Continuous discharge of 10.2 Ampere we get,

$$Total\ Current = 1.4077 + 9.3500 = 10.7577\ A$$

For this Current requirement we can find Battery Life Time as,

$$Battery\ Life\ Time = \frac{5200\ mAh}{10757.7\ mA}$$

$$Battery\ Life\ Time = 0.4834\ hrs$$

Thus, we get a Battery Life Time of 29 mins when robot is continuously performing motions.

5.2.2 Managing Dynamixel

Dynamixels are equipped with their own microcontrollers meaning they are mini Computers in themselves which requires setting and managing parameters for their optimum performance. Dynamixels are managed using Dynamixel wizard. DYNAMIXEL Wizard 2.0 is an optimized tool for managing DYNAMIXEL's from various operating systems. The following features are provided with DYNAMIXEL Wizard 2.0.

- DYNAMIXEL Firmware Update
- DYNAMIXEL Diagnosis
- DYNAMIXEL Configuration and Test
- DYNAMIXEL Data Plotting in Real-Time
- Generate & Monitor DYNAMIXEL Packets

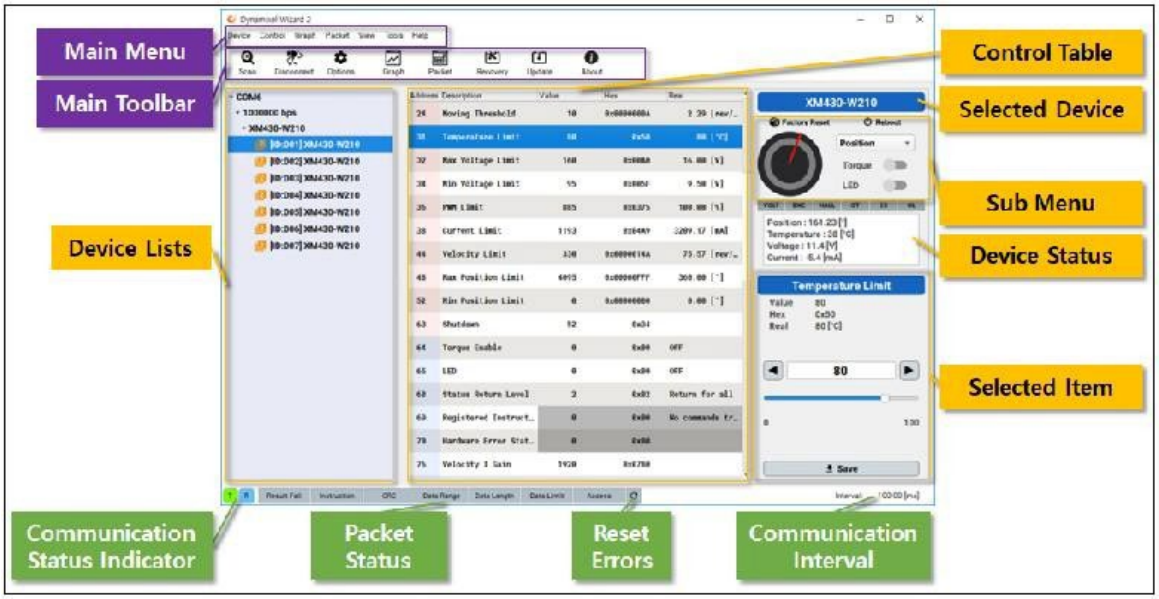


Figure 5.4: Configuration of Parameters via Dynamixel Wizard

Control table

The Control Table is a structure of data implemented in the device. Users can read a specific Data to get status of the device with Read Instruction Packets, and modify Data as well to control the device with WRITE Instruction Packets.

Address	Size (Byte)	Data Name	Description	Access	Initial Value
2	1	Firmware Version	Firmware Version	R	-
3	1	ID	DYNAMIXEL ID	RW	1
4	1	Baud rate	Communication speed	RW	34
24	1	Torque enable	Motor Torque on\off	RW	0
30	2	Goal position	Desired position	RW	-
32	2	Moving speed	Moving velocity	RW	-
36	2	Present Position	Present position	R	-
40	2	Present load	Present load	R	-
42	1	Present voltage	Present voltage	R	-
43	1	Present temperature	Present temperature	R	-
73	1	Goal acceleration	Goal acceleration	RW	0

Table 5.1: Important Control Table Parameters

Dynamixel Parameters

Some parameters listed in control table are to be changed according to our requirement. To change Dynamixel parameters, one can use DYNAMIXEL Wizard or RoboPlus Manager. We have used Dynamixel wizard to set the required parameters. Mode has been set to joint mode.

1) Firmware

   Firmware is a fundamental program to operate the hardware device. It is recommended to keep the firmware updated because it contains various communication regulations to exchange data with other devices such as computers or smart phones. New firmwares are released for updated features or bug fixes. So, it is necessary to update firmware.

2) ID

The ID is a unique value in the network to identify each DYNAMIXEL with an Instruction Packet. 0~252 (0xFC) values can be used as an ID. We have given unique ID's to each motor so that there is no communication failure and each motor can be easily detected and controlled.

3) Baud rate

Baud Rate determines serial communication speed between a controller and DYNAMIXEL. Available value range is 0 ~ 254(0xFE). The default value is 57600 bps which gives 0.794% error. We had set the Baud rate to 1mbps so there is 0% error in data transmission.

5.3 Programming

Programming of humanoid is also important task to make it functional. As This Programming included many new and different aspects, we decided to follow a step wise approach in programming to reduce complications and mistake.

We had many actuators and sensors to interface in this Humanoid. So, we designed our electronic circuit and decided our final controlling structure.

We decided to interface all Dynamixel servos via USB ports to reduce complications in Huge data transmissions as there were many actuators needed to be controlled simultaneously. Other remaining actuators and sensors was to be controlled by GPIO pins of Raspberry PI.

Interfacing Actuators and sensors via GPIO pins was comparatively easy task as we can easily read and write Information from GPIO Pins of Raspberry Pi. Controlling Dynamixel Servos was difficult task as Raspberry Pi has a Raspbian (Linux Debian based) Operating System. To reduce Complications, we followed following step wise approach to interface Dynamixel.

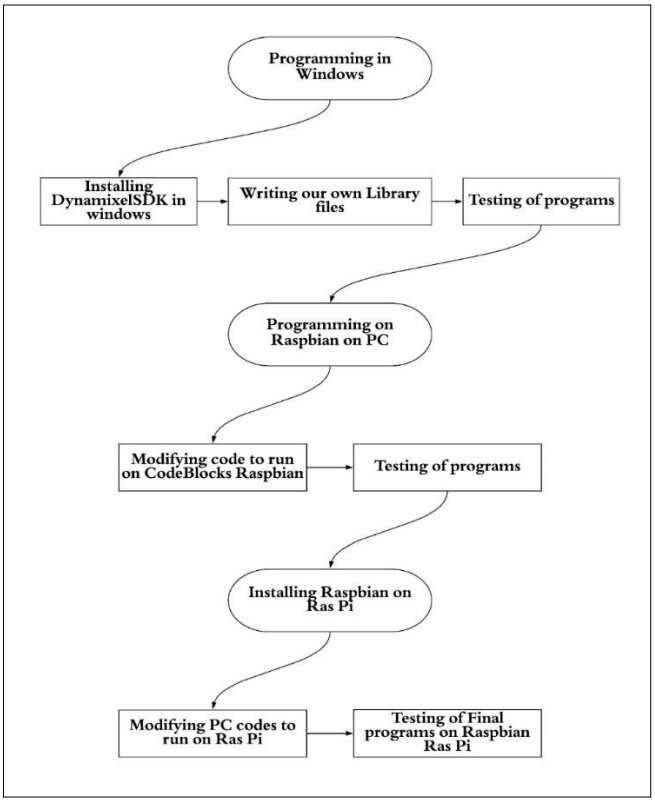


Figure 5.8: Flowchart of Programming approach

Programming in Windows

To Start with programming, we decided to start coding in windows using C++ Programming in Visual Studio. Visual Studio provided a easy environment for coding. It is also fairly easy to include and use library Functions in visual Studio.

Installing DynamixelSDK in Windows

We decided to interface Dynamixel using DynamixelSDK (Dynamixel Software Development Kit) [39]. It is provided by ROBOTIS (manufacturer of Dynamixel). It provides many functions to interface Dynamixel. But It was very complicated and resulted in almost 1500 lines of code to just Make Humanoid stand.

Our main aim was to read and learn how it worked and use it to interface Dynamixel according to our requirements.

Writing Asteroid.h

Due to complexity and length of programs due to DynamixelSDK, we decided to write our own Library file Asteroid.h and make it easy to interface Dynamixel using our library file. we tried to use all functions and features provided by Dynamixel in this Library file.

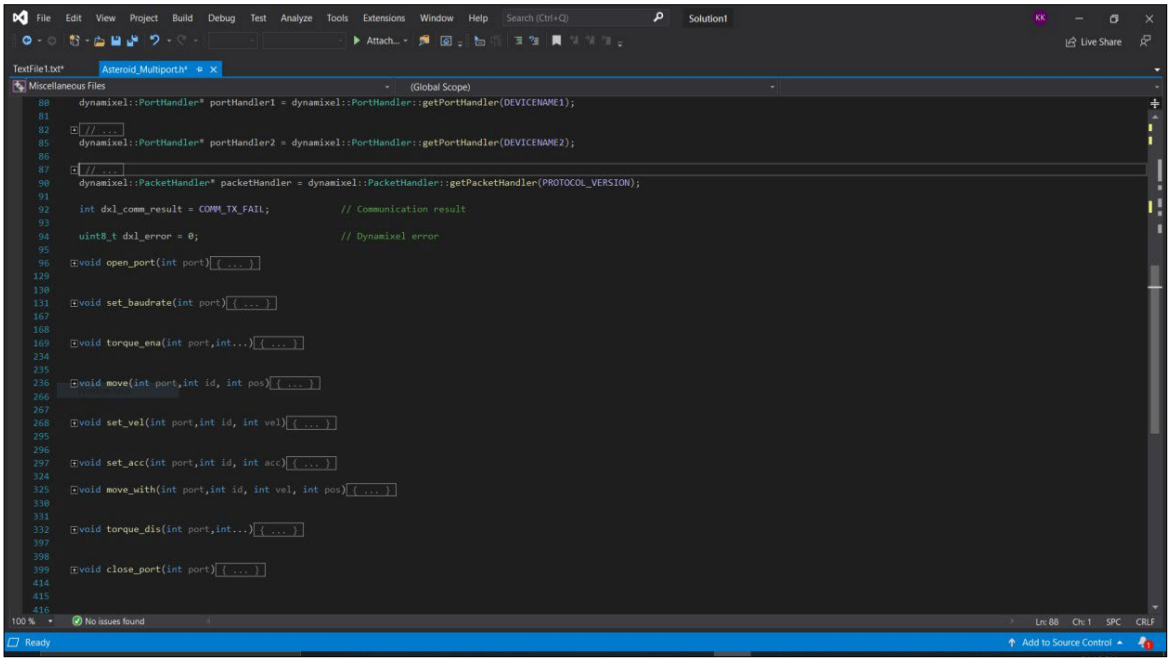


Figure 5.9: Functions in Asteroid.h

Testing Programs on Windows

Using above programs, we were able to make asteroid successfully walk single step and perform hand grasping motion. Also, we were able to interface any number of servos simultaneously.

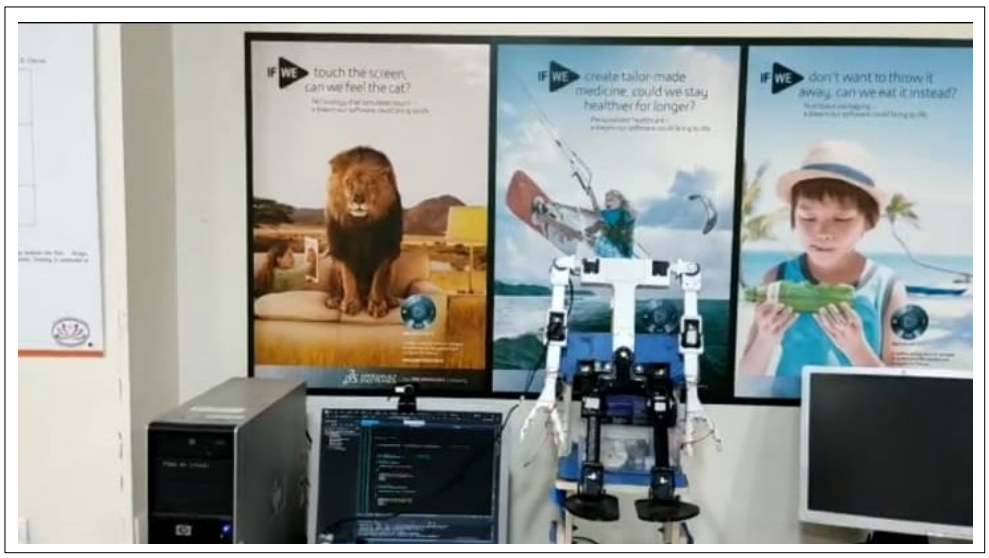


Figure 5.10: Asteroid with Windows interfacing

Programming for Inverse Kinematics

After Interfacing all Dynamixels we wrote another library file to use inverse kinematics framework We learned in MATLAB to be used in C++.

Kinematics.h

We wrote a library file named *kinematics.h* to use kinematics calculations to control Humanoid. We wrote this kinematic framework for hands of asteroid. It performs various kinematical Calculations. It can give Joint angles i.e values of rotation of Dynamixels according to points to be reached in environment. It has many functions like

i) ***is_within_ws() :*** this function returns if point is within arm's reach (Workspace) or not. If yes then which arm can reach it

ii) ***inv_kine() :*** This function returns Joint rotation corresponding to default position of arm to reach a certain point.

iii) ***for_kine() :*** This function returns final position of arm if input angles are used.

Using Dynamixel as Gyroscopic Sensor

We also had one creative idea about using Dynamixel as gyroscopic sensor in our humanoid. We used Dynamixel Load sensor's (Torque sensor) value and used it as information about position of leg on ground and using value of this sensor wrote a program which itself corrects the angle of leg with respect to ground inclination. So, without using gyroscopic sensor we were able to correct angle of legs.

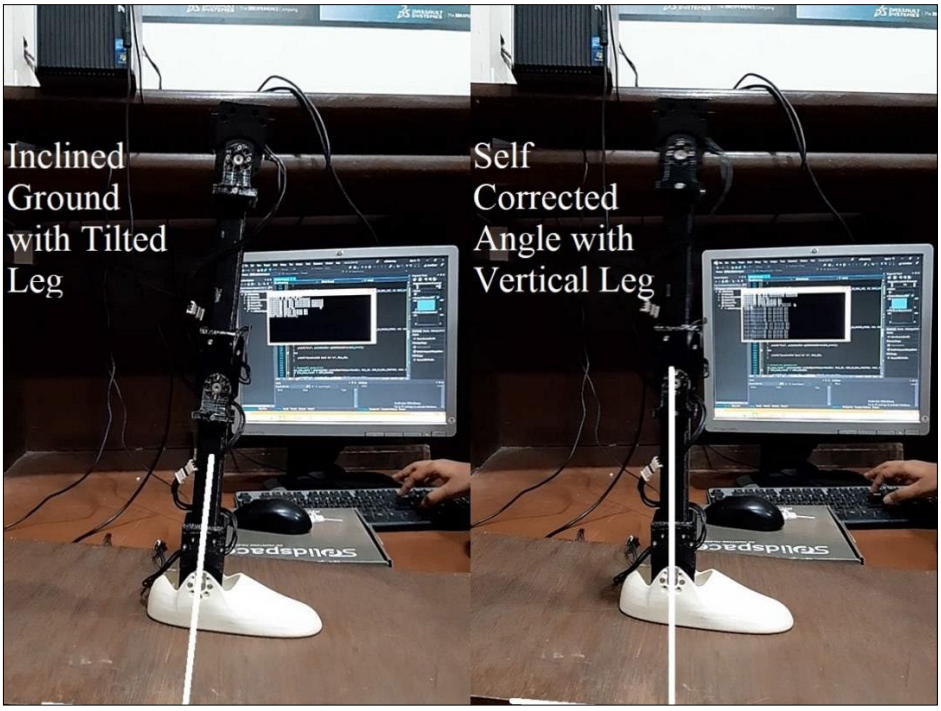


Figure 5.11: Self Balancing Program Outcome

5.4 Walking of Asteroid

We performed static walking on Asteroid Legs. We used CAD (Solidworks Mate controller) for realising static walk virtually. We applied all material and mass to parts of Humanoid parts. Using various joints, we controlled humanoid in mate controller and gave him required motion to walk forward. We used our knowledge of Static walking and constantly kept ZMP of humanoid within Support polygon while walking and derived complete Static walking Gait similar to humans.

To replicate Upper Body of Humanoid we used Mild Steel Cylinder Blocks of 570 grams and another block of 1.1 Kg. we gave it height such that it will match the cg of upper body. We used this model to test our static walk of asteroid.

Next task was to maintain cg of whole body above support polygon. We used upper body weight to change center cg gravity of whole body. We used MX64 from spine to move cg in frontal plane of humanoid. To move cg in sagittal plane we used two MX28 servos from hip joint. Normally when we actuate those motors, leg is raised from hip joint but when we actuate those two Dynamixels simultaneously we get spine moved in sagittal plane as resistance to leg motion is much more than resistance to spine motion and spine gets rotated in sagittal plane. Using these 2 motions we can get maintain cg above support polygon.

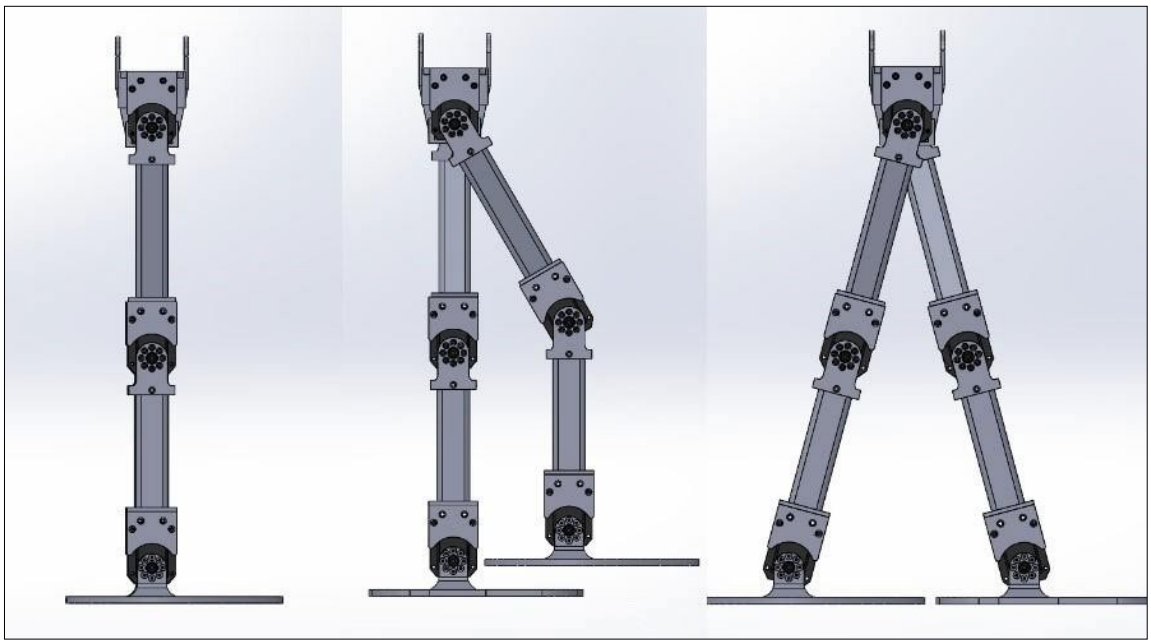


Figure 5.12: 1st Walking Step

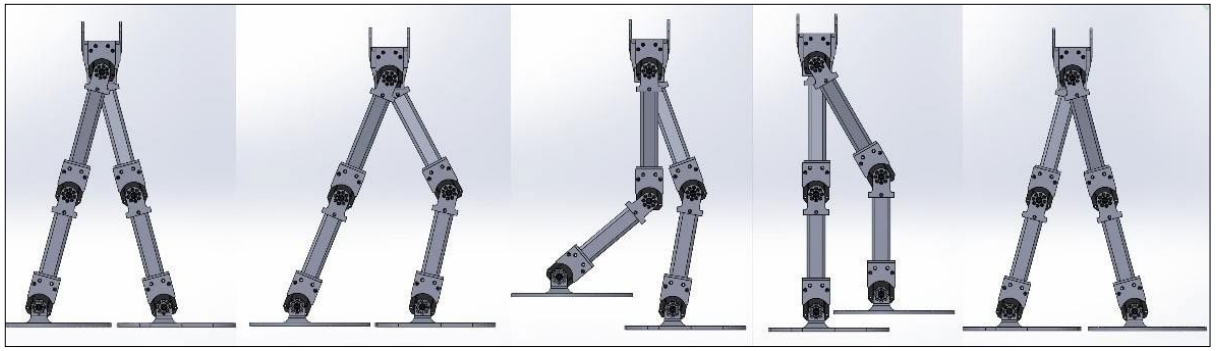


Figure 5.13: 2nd Walking Step

After Finding Out perfect walking gait for humanoid we calculated final position and speed for each and every joint. And set those parameters to Dynamixels. We also used *Time.h/Dos.h* library file to add delay in motions to get exact results as expected.

We can use 1st step to initiate and terminate walking of humanoid any time and we can use 2nd step with alternate legs to make humanoid walk as long as we want.

5.5 Pick and drop

After walking, we performed pick and drop action using Arm of humanoid. We used inverse kinematic framework from our kinematical analysis and motion controlling Asteroid.h library from our programming work. We desired that after entering a certain coordinate of space arm should automatically reach that point and grasp that object. We considered this as an important ability in for future development in humanoid. If image processing module installed in our raspberry pi, we can use it to know position of various objects in space and this module of pick and drop can use coordinates from image processing and use those to handle those objects.

This module is also of important consideration in remote control applications where it hard for human to reach or survive.

Currently we are using manual input i.e. we will enter the coordinates we need to reach and our humanoid can make its own calculations of desired motion and perform them using its own microcontroller.

Our hand was completely symmetrical and was able to grasp various objects so Initially we decided to use only inverse kinematics framework and programming to pick various object but it resulted in interference with objects as we did not give control over path it should follow. this was major flaw in direct grasp technique.

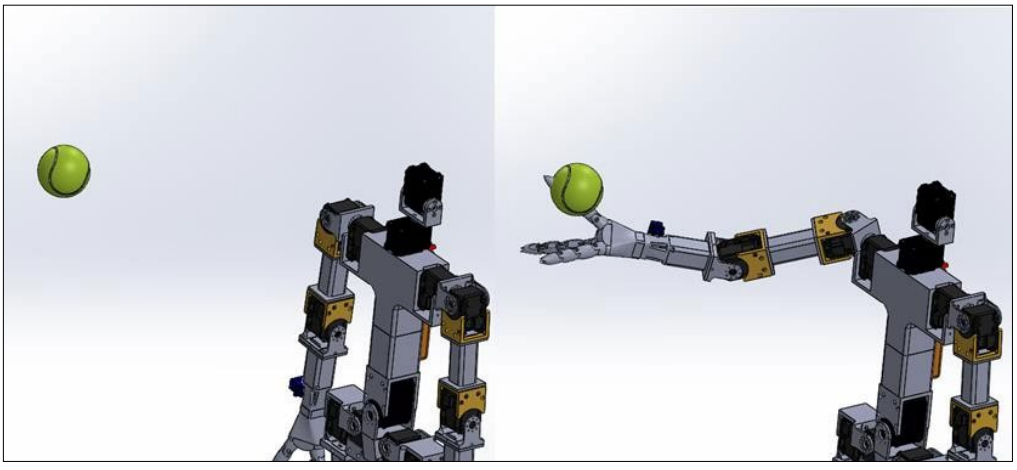


Figure 5.14: Interference in Direct grasp

To remove this error in direct grasp, we developed a technique of fast reach and slow grasp to approach objects without interference. Fast reach resulted in reducing time of motion n slow grasp ensured no interference with object.

For Fast reach we first calculated the orientation in which hand will grasp the object and considering that orientation we used forward kinematics to find a point preceding the final position as safe point to fast reach without worrying about interference. After fast reach we can use slow grasp so that we can pick object firmly without much interference.

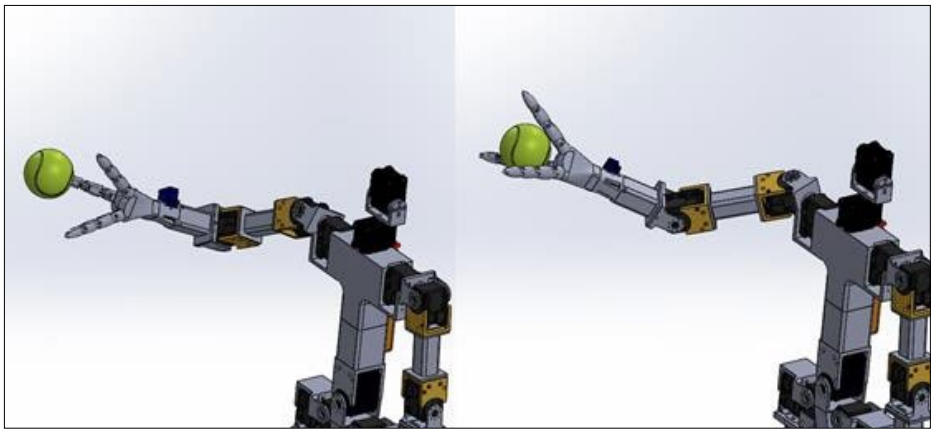


Figure 5.15: Fast reach and slow grasp

After enabling to pick objects, dropping was comparatively easy task which only required inverse kinematics and motion control of humanoid.

6. CONCLUSION AND FUTUREWORK

6.1 Conclusion

- After studying the Human and Various Humanoids, Final Humanoids design gave a closer approach to the structural design of human body.

- All Electronic components were neatly studied and resulted in successful electronic interfacing of all actuators and sensors.

- Kinematics Study showed neat approach to realise motion of humanoid. Forward and inverse kinematics analysis helped in programming of Humanoid.

- Structural Analysis validated mechanical modelling of Humanoid.

- The lower body of the skeleton structure of this project has been manufactured and assembly of the all required electronics equipment to mechanical structure successfully completed.

- The conclusions and result achieved from the project is a light and effective weight Humanoid robot, which delivers a Stable Humanoid Locomotion on plane surface.

6.2 Future Scope

- More sensors like pressure pads, ultrasonic sensors, cameras, speakers, microphone can be introduced in the circuit to get more knowledge about surrounding and make humanoid more interactive and independent.
- Casing can be design for aesthetic purpose and more human like appearance casing will also help to cover all motors, electronic circuits keeping them safe from sudden changes in work area.
- More compact circuit can be built to reduce weight and increase ease of operation.
- Design can be modified to reduce overall weight to favour balancing of humanoid.
- More DOF can be added to increase functionality of the robot.

6.3 Applications

- Industrial Automations.
- Surveillance & Disaster Management.
- Research in Medical & Prosthesis Application.
- Human assistance in various professions & also as emotional support
- Technology Demonstration**.**

❖ REFERENCES

1. https://en.wikipedia.org/wiki/Human_body#Anatomy
2. https://www.ifittraining.co.uk/insights/movement-variability-degrees-freedom/
3. https://en.wikipedia.org/wiki/Gait_(human)
4. V. V. Hafner and F. Bachmann, "Human-Humanoid walking gait recognition", Humanoids 2008 8th IEEE-RAS International Conference on Humanoid Robots, Daejeon, 2008, pp. 598-602
5. https://en.m.wikipedia.org/wiki/Humanoid_robot
6. Kimura, Tokio. Waga Waseda: Ōkuma Shigenobu to sono kengaku seishin, Tokyo, Kobunsha, 1997.
7. https://en.m.wikipedia.org/wiki/Waseda_University
8. https://global.honda/innovation/robotics/robot-development-history.html
9. https://global.honda/innovation/robotics/ASIMO.html
10. https://www.softbankrobotics.com/emea/en/nao
11. http://robotglobe.org/robonaut-2-lauded-as-2014-government-invention-of-the-year/nasa-robonout-2/
12. https://mitpress.mit.edu/books/semantic-information-processing
13. http://fortune.com/2017/10/27/sophia-the-robot-artificial-intelligence/
14. https://www.poppy-project.org/en/
15. http://www.hansonrobotics.com/
16. https://www.popularmechanics.com/technology/robots/a30646158/india-humanoid-robot-vyommitra/
17. https://www.livemint.com/Industry/rc86Iu7h3rb44087oDts1H/Meet-Manav-Indias-first-3Dprinted-humanoid-robot.html
18. Shannon Angel D'costa and Royson Donate D'Souza;Humanoid Robots – Past, Present and the Future;European Journal of Advances in Engineering and Technology, 2016;1-5,7-9;
19. Alec, MANAV, India's first 3D printed humanoid robot, unveiled at IIT Mumbai TechFest, 3D Printing Applications, Jan 7, 2015
20. thieu Lapeyre, Pierre Rouanet, Pierre-Yves;Poppy Humanoid Platform: Experimental Evaluation ofthe Role of a Bio-inspired Thigh Shape;Humanoids 2013, Oct 2013,Atlanta, United States. . <hal-00861110>;2013

21. J.J. Craig, Introduction to Robotics, Addison-Wesley Publishing Company, 1986
22. Shuuji Kajita, Hirohisa Hirukawa, Kensuke Harada, Kazuhito Yokoi, Introduction to Humanoid Robotics, Springer, Berlin, Heidelberg, 2014, pp. 19-67,105-158.
23. https://swayam.gov.in/nd1_noc19_me74/preview
24. http://emanual.robotis.com/docs/en/dxl/mx/mx-28/
25. http://emanual.robotis.com/docs/en/dxl/mx/mx-64/
26. http://emanual.robotis.com/docs/en/dxl/ax/ax-12a/
27. http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf
28. http://emanual.robotis.com/docs/en/parts/interface/u2d2/
29. http://www.robotis.us/smps2dynamixel/
30. https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf
31. W. W. Tai, Study of Ultrasonic Sensor Capability in Human Following Robot System, IOP Conf. Ser.: Mater. Sci. Eng. 705 012045, 2019
32. https://www.3dhubs.com/knowledge-base/selecting-optimal-shell-and-infill-parameters-fdm-3d-printing/
33. D. A. Winter, Biomechanics and Motor Control of Human Movement, John Wiley & Sons, Inc., ed. 4, 2009.
34. https://robotacademy.net.au/lesson/denavit-hartenberg-notation/
35. Goyal, Khushdeep & Sethi, Davinder, An analytical method to find workspace of a robotic manipulator, Journal of Mechanical Engineering, Vol. ME 41, No. 1, 2010, pp. 25-30
36. Gudla, Arun Gowtham, "A methodology to determine the functional workspace of a 6R robot using forward kinematics and geometrical methods", Electronic Theses and Dissertations. 4809, 2012
37. Keleş, Özgür, Blevins, Caleb, Bowman, Keith, Effect of build orientation on the mechanical reliability of 3D printed ABS, Rapid Prototyping Journal, 10.1108/RPJ-09-2015-0122, 2016/03/31.
38. Lalit Singh Mehta, Dr. Priam Pillai, Compression Testing of PLA in 3D Printing, al System IJEECS, ISSN 2348-117X, Volume 6, Issue 8, August 2017, pp. 466-470.

39. https://github.com/ROBOTIS-GIT/DynamixelSDK

❖ ANNEXURE

Asteroid.h

```
#pragma once
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include<stdarg.h>
// Uses Dynamixel SDK library
#include "dynamixel_sdk.h"


// Defining Control table address
#define ADDR_MX_TORQUE_ENABLE           24
#define ADDR_MX_GOAL_POSITION           30
#define ADDR_MX_PRESENT_POSITION        36
#define ADDR_MX_MOVING_SPEED            32
#define ADDR_MX_MOVING_ACC              73

// Defining Protocol version (which protocol version is used in
the Dynamixel)
#define PROTOCOL_VERSION                1.0

//**************** Default setting *********************
// Check Baud rate of Dynamixel
#define BAUDRATE                        1000000
// Check which port is being used on your controller
#define DEVICENAME1                     "COM3"
#define DEVICENAME2                     "COM4"

 // Value for enabling the torque
#define TORQUE_ENABLE                    1
// Value for disabling the torque
#define TORQUE_DISABLE                   0


// Functions to use keyboard
//Defined in DynamixelSDK
int getch(){. }

int kbhit(void){.  }
```

```
// Initialize PortHandler1 instance
// Set the port path
// Get methods and members of PortHandlerLinux or
PortHandlerWindows
dynamixel::PortHandler* portHandler1 =
dynamixel::PortHandler::getPortHandler(DEVICENAME1);

// Initialize PortHandler2 instance
// Set the port path
// Get methods and members of PortHandlerLinux or
PortHandlerWindows
dynamixel::PortHandler* portHandler2 =
dynamixel::PortHandler::getPortHandler(DEVICENAME2);

// Initialize Packet Handler instance
// Set the protocol version
// Get methods and members of Protocol1PacketHandler or
Protocol2PacketHandler
dynamixel::PacketHandler* packetHandler =
dynamixel::PacketHandler::getPacketHandler(PROTOCOL_VERSION);


// defining int to store Communication result
int dxl_comm_result = COMM_TX_FAIL;
// defining int to store Dynamixel error
uint8_t dxl_error = 0;


//**Function to open port**
// no. of port is required as argument
void open_port(int port)
{
    if (port == 1)
    {
        // Open port
        if (portHandler1->openPort())
        {
            printf("Succeeded to open the port %d!\n",port);
        }
        else
        {
            printf("Failed to open the port %d!\n",port);
```

```
            printf("Press any key to terminate...\n");
            getch();
            return (exit(0));
        }
    }
    else if (port == 2) {...}
}


//**Function to set Baudrate**
//Baudrate is defined above and can be changes
//No. of port whose baudrate is to be set is reuired as
argument
void set_baudrate(int port)
{
    if (port == 1)
    {
        // Set port baudrate
        if (portHandler1->setBaudRate(BAUDRATE))
        {
            printf("Succeeded to change the baudrate %d to
%d!\n",port,BAUDRATE);
        }
        else
        {
            printf("Failed to change the baudrate of
%d!\n",port);
            printf("Press any key to terminate...\n");
            getch();
            return (exit(0));
        }
    }
    else if (port == 2) {...}

}


//**Function to Enable torque**
//No. of port to which dxls are connected and their ids are
reuired as argument
//Upto 25 dynamixel servoes can be connected
void torque_ena(int port,int...)
```

```
{
    va_list ids;
    va_start(ids, 25);
    int a[25];

    if (port == 1)
    {
        for (int i = 0; i < 25; i++)
        {
            a[i] = va_arg(ids, int);
            if (abs(a[i]) > 50)
                break;
            else
            {

                // Enable Dynamixel Torque
                dxl_comm_result = packetHandler-
>write1ByteTxRx(portHandler1, a[i], ADDR_MX_TORQUE_ENABLE,
TORQUE_ENABLE, &dxl_error);
                if (dxl_comm_result != COMM_SUCCESS)
                {
                    printf("Torque not Enabled of ID%d due to
%s\n", a[i], packetHandler->getTxRxResult(dxl_comm_result));
                }
                else if (dxl_error != 0)
                {
                    printf("Torque not Enabled of ID%d due to
%s\n", a[i], packetHandler->getRxPacketError(dxl_error));
                }
                else
                {
                    printf("Dynamixel %d has been successfully
connected \n", a[i]);
                }
            }

        }
        va_end(ids);
    }
    else if (port == 2) {...}
}
```

```
//**Function to move dxl to specific position**
//No. of port, id, position are reuired as argument
void move(int port,int id, int pos)
{
    if (port == 1)
    {
        // Write goal position
        dxl_comm_result = packetHandler-
>write2ByteTxRx(portHandler1, id, ADDR_MX_GOAL_POSITION, pos,
&dxl_error);
        if (dxl_comm_result != COMM_SUCCESS)
        {
            printf("move failed of ID %d due to %s\n", id,
packetHandler->getTxRxResult(dxl_comm_result));
        }
        else if (dxl_error != 0)
        {
            printf("move failed of ID %d due to %s\n", id,
packetHandler->getRxPacketError(dxl_error));
        }
    }
    else if (port == 2) {...}

}


//**Function to set velocity**
//No. of port, id, velocity are reuired as argument
void set_vel(int port,int id, int vel)
{
    if (port == 1)
    {
        dxl_comm_result = packetHandler-
>write2ByteTxRx(portHandler1, id, ADDR_MX_MOVING_SPEED, vel,
&dxl_error);
        if (dxl_comm_result != COMM_SUCCESS)
        {
            printf("set vel failed of ID %d due to %s\n", id,
packetHandler->getTxRxResult(dxl_comm_result));
        }
        else if (dxl_error != 0)
```

```
        {
            printf("set vel failed of ID %d due to %s\n", id,
packetHandler->getRxPacketError(dxl_error));
        }
    }
    else if (port==2) {...}
}


//**Function to set accelaration**
//No. of port, id, accelaration are reuired as argument
void set_acc(int port,int id, int acc)
{
    if (port == 1)
    {
        dxl_comm_result = packetHandler-
>write1ByteTxRx(portHandler1, id, ADDR_MX_MOVING_ACC, acc,
&dxl_error);
        if (dxl_comm_result != COMM_SUCCESS)
        {
            printf("set acc failed of ID %d due to %s\n", id,
packetHandler->getTxRxResult(dxl_comm_result));
        }
        else if (dxl_error != 0)
        {
            printf("set acc failed of ID %d due to %s\n", id,
packetHandler->getRxPacketError(dxl_error));
        }
    }
    else if (port == 2) {...}
}


//**Function to move dxl to specific position with specific
velocity**
//No. of port, id, velocity, position are reuired as argument
void move_with(int port,int id, int vel, int pos)
{
    set_vel(port,id, vel);
    move(port,id,pos);
}
```

```
//**Function to read current position of dxl**
//No. of port, id are reuired as argument
//returns present position as output
int read_pos(int port, int id)
{
    // Variable to store Present position
    uint16_t dxl_present_position = 0;

    if (port == 1)
    {
        // Read Present Position
        dxl_comm_result = packetHandler-
>read2ByteTxRx(portHandler1, id, 36 , &dxl_present_position,
&dxl_error);
        if (dxl_comm_result != COMM_SUCCESS)
        {
            printf("Read position failed of ID %d due to %s\n",
id, packetHandler->getTxRxResult(dxl_comm_result));
            return 0;
        }
        else if (dxl_error != 0)
        {
            printf("Read position failed of ID %d due to %s\n",
id, packetHandler->getRxPacketError(dxl_error));
            return 0;
        }
        else
        {
            printf("Present position of ID %d is %d\n", id,
dxl_present_position);
            return dxl_present_position;
        }
    }
    else if (port == 2) {...}

}
```

```
//**Function to read current load of dxl**
//No. of port, id are reuired as argument
//returns present load as output
int read_load(int port, int id)
{
    // Variable to store Present load
    uint16_t dxl_present_load = 0;

    if (port == 1)
    {
        // Read Present load
        dxl_comm_result = packetHandler-
>read2ByteTxRx(portHandler1, id, 40 , &dxl_present_load,
&dxl_error);
        if (dxl_comm_result != COMM_SUCCESS)
        {
            printf("Read load failed of ID %d due to %s\n", id,
packetHandler->getTxRxResult(dxl_comm_result));
            return 0;
        }
        else if (dxl_error != 0)
        {
            printf("Read load failed of ID %d due to %s\n", id,
packetHandler->getRxPacketError(dxl_error));
            return 0;
        }
        else
        {
            printf("Present load of ID %d is %d\n", id,
dxl_present_load);
            return dxl_present_load;
        }
    }
    else if (port == 2) {...}

}


//**Function to read current voltage of dxl**
//No. of port, id are reuired as argument
//returns present voltage as output
int read_volt(int port, int id)
```

```
{
    // Variable to store Present Voltage of battery
    uint8_t dxl_present_volt = 0;

    if (port == 1)
    {
        // Read Present Voltage
        dxl_comm_result = packetHandler-
>read1ByteTxRx(portHandler1, id, 42, &dxl_present_volt,
&dxl_error);
        if (dxl_comm_result != COMM_SUCCESS)
        {
            printf("Read volt failed of ID %d due to %s\n", id,
packetHandler->getTxRxResult(dxl_comm_result));
            return 0;
        }
        else if (dxl_error != 0)
        {
            printf("Read volt failed of ID %d due to %s\n", id,
packetHandler->getRxPacketError(dxl_error));
            return 0;
        }
        else
        {
            printf("Present volt of ID %d is %d\n", id,
dxl_present_volt);
            return dxl_present_volt;
        }
    }
    else if (port == 2) {...}

}


//**Function to disable torque**
//No. of port to which dxls are connected and their ids are
reuired as argument
//Upto 25 dynamixel servoes can be disconnected
void torque_dis(int port,int...)
{
    va_list ids;
    va_start(ids, 25);
```

```
    int a[25];

    if (port == 1)
    {
        for (int i = 0; i < 25; i++)
        {
            a[i] = va_arg(ids, int);
            if (abs(a[i]) > 50)
                break;
            else
            {

                // Enable Dynamixel Torque
                dxl_comm_result = packetHandler-
>write1ByteTxRx(portHandler1, a[i], ADDR_MX_TORQUE_ENABLE,
TORQUE_DISABLE, &dxl_error);
                if (dxl_comm_result != COMM_SUCCESS)
                {
                    printf("torque dis failed of ID%d due to
%s\n", a[i], packetHandler->getTxRxResult(dxl_comm_result));
                }
                else if (dxl_error != 0)
                {
                    printf("torque dis failed of ID%d due to
%s\n", a[i], packetHandler->getRxPacketError(dxl_error));
                }
                else
                {
                    printf("Dynamixel %d has been successfully
disconnected \n", a[i]);
                }
            }

        }
        va_end(ids);
    }
    else if (port == 2) {...}
}
```

```
//**Function to close port**
// no. of port is reuired as argument
void close_port(int port)
{
    if (port == 1)
    {
        // Close port
        portHandler1->closePort();
        printf("port %d closed",port);
    }
    else if(port == 2) {...}
}
```

Kinematics.h

```
#pragma once
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <cmath>



//Function to know if given point is within the workspace of
arm or not
//Function returns 3 when point is in limit, less if not..
int is_within_ws(float x_mid, float y_mid, float z_mid)
{

	int Count = 0;

	// z Coordinate of kinemaatic ref is 9.065cm away from
	float x_sho = x_mid;
	float y_sho = y_mid;
	float z_sho = z_mid-9.065-3.125;

	float r_sho = sqrt((x_sho * x_sho) + (y_sho * y_sho) +
(z_sho * z_sho));

	if (z_sho > 0 && r_sho < 37.6)
	{
		Count++;			// within outer limit
	}

	if (z_sho > 0 && r_sho > 27.2)
	{
		Count++;		// within inner limit
	}

	float x_elb = x_mid;
	float y_elb = y_mid;
	float z_elb = z_mid - 9.065 - 3.125 -14.6;

	float r_elb = sqrt((x_elb * x_elb) + (y_elb * y_elb) +
(z_sho * z_elb));
```

```
      if (r_elb > 23)
      {
            Count++;      // within inner limit
      }

      return Count;
      }



//Function to know Joint angles to reach given point in space
//Function returns 3 joint angles in given result array
void inv_kine(float x_mid, float y_mid, float z_mid, float
theta_deg[])
{
      // z Coordinate of kinemaatic ref is 9.065cm away from
      float x_sho = x_mid;
      float y_sho = y_mid;
      float z_sho = z_mid - 9.065 -3.125 ;

      float theta1, theta2, theta3;

      theta3 = acos(-((14.6 * 14.6) + (23 * 23) - (x_sho *
x_sho) - (y_sho * y_sho) - (z_sho * z_sho)) / (2 * 14.6 * 23));
      float a = cos(theta3);
      float b = z_sho / (14.6 + (23 * a));
      theta2 = asin(b);
      theta1 = atan(y_sho / x_sho) - atan((23 * sin(theta3)) /
(cos(theta2) * (14.6 + 23 * cos(theta3))));

      theta_deg[0] = theta1 * (180 / 3.14159265);
      theta_deg[1] = theta2 * (180 / 3.14159265);
      theta_deg[2] = theta3 * (180 / 3.14159265);

}
```

```
//Function to know point which will be reached if given joint
angles are used
//Function returns 3 space coordinates in given result array
void for_kine(float theta1, float theta2, float theta3, float
position[])
{
      float t1 = theta1 * (3.14159265 / 180);
      float t2 = theta2 * (3.14159265 / 180);
      float t3 = -theta3 * (3.14159265 / 180);

      float x, y, z;

      x = 14.6 * cos(t1) * cos(t2) + 23 * sin(t1) * sin(t3) +
23 * cos(t1) * cos(t2) * cos(t3);
      y = 14.6 * cos(t2) * sin(t1) - 23 * cos(t1) * sin(t3) +
23 * cos(t2) * cos(t3) * sin(t1);
      z = 14.6 * sin(t2) + 23 * cos(t3) * sin(t2) + 3.125 +
9.065;

      position[0] = x;
      position[1] = y;
      position[2] = z;

}



//Function to know Joint angles to grasp a object
//Function returns 3 joint angles in given result array
void for_approach(float x_mid, float y_mid, float z_mid, float
result[])
{
      float thetas[3];
      float t1, t2, t3;

      inv_kine(x_mid, y_mid, z_mid,thetas);

      t1 = thetas[0] * (3.14159265 / 180);
      t2 = thetas[1] * (3.14159265 / 180);
      t3 = -thetas[2] * (3.14159265 / 180);
```

```
      float x, y, z;

      x = 14.6 * cos(t1) * cos(t2) + 17 * sin(t1) * sin(t3) +
17 * cos(t1) * cos(t2) * cos(t3);
      y = 14.6 * cos(t2) * sin(t1) - 17 * cos(t1) * sin(t3) +
17 * cos(t2) * cos(t3) * sin(t1);
      z = 14.6 * sin(t2) + 17 * cos(t3) * sin(t2) + 3.125 +
9.065;

      inv_kine(x, y, z,result);

}
```