LISA - The Quadruped Robot

A

Major Project Report

Submitted in partial fulfillment of the requirements
for the award of the degree of

Bachelor of Technology
in
Computer Science And Engineering

Submitted by

A. RAJ KUMAR	(17SS1A0504 CSE)
T. VAMSHI	(17SS1A0546 CSE)
V. HARISATHWIK	(17SS1A0550 CSE)
N. ARYAN KUMAR	(17SS1A2905 ME)

Under the guidance of

Ms. Faiza Iram

Department of Computer Science and Engineering
JNTUH COLLEGE OF ENGINEERING, SULTANPUR
Sultanpur (V), Pulkal (M), Sangareddy District, Telangana - 502273.

June 2021.

JNTUH COLLEGE OF ENGINEERING SULTANPUR

Sulthanpur(V),Pulkal(M),Sangareddy-502273 ,Telangana

Department of Computer Science and Engineering

Certificate

This is to certify that the Major Project report work entitled **"LISA THE QUADRUPED ROBOT"** is a bonafide work carried out by **A.RAJKUMAR (17SS1A0504 CSE), T.VAMSHI (17SS1A0546 CSE), V.HARISATHWIK (17SS1A0550 CSE), ARYAN KUMAR.N (17SS1A2905 ME)** in partial fulfillment of the requirements of the degree of BACHELOR of TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING discipline to Jawaharlal Nehru Technological University,Hyderabad during the academic year 2020-21. The results embodied in this report have not been submitted to any other University or Institution for the award of any degree or diploma.

Project Guide
Ms. FAIZA IRAM
Assistant Professor(C)

Head of the Department
Sri. JOSHI SHRIPAD
Associate Professor

PRINCIPAL

Declaration

We hereby declaring that our Major Project report work is being entitled as **"LISA THE QUADRUPED ROBOT"** is the work done by our teammates A.RAJKUMAR (17SS1A0504 CSE), T.VAMSHI (17SS1A0546 CSE), V.HARISATHWIK(17SS1A0550 CSE), ARYAN KUMAR.N (17SS1A2905 ME) and is submitted in the partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering from Jawaharlal Nehru Technological University Hyderabad College of Engineering Sultanpur.The results embodied in this report have not been submitted to any other University or Institution for the award of any degree or diploma.

A.RAJKUMAR (**17SS1A0504 CSE**)
T.VAMSHI (**17SS1A0546 CSE**)
V.HARISATHWIK (**17SS1A0550 CSE**)
ARYAN KUMAR.N (**17SS1A2905 ME**)

Acknowledgement

We wish to take this opportunity to express our deep gratitude to all those who helped us in various ways during our Major Project report work. It is our pleasure to acknowledge the help of all those individuals who where responsible for foreseeing the successful completion of our documentation report. We express our sincere gratitude to **Prof.B.Balu Naik**, Principal of JNTUHCES for his support during the course period. We sincerely thank **Prof.V.Venkateswara Reddy**, Vice Principal of JNTUHCES for his kind help and Cooperation. We thankful to **Sri.Joshi Shripad**, Associate Professor and Head of the Department of Computer Science and Engineering of JNTUHCES for his effective suggestions during the course period. We very much thankful to **Sri S.Ravinder**, Assistant Professor and **Mrs.B.Sangeetha**, Assistant Professor for their support and guidance through out the course period. Finally, We express our gratitude with great admiration and respect to our faculty for their moral support and encourage throughout the course.

A.RAJKUMAR (**17SS1A0504 CSE**)
T.VAMSHI (**17SS1A0546 CSE**)
V.HARISATHWIK (**17SS1A0550 CSE**)
ARYAN KUMAR.N (**17SS1A2905 ME**)

Abstract

Mobile robots have an extensive area of applications in various fields like space exploration, military application, industrial use, and many more. Hence, the design and development of a mobile robot is a crucial part of the above applications.Among all the mobile robots, the quadrupedal robot is a legged robot, which is superior to wheeled and tracked robots due to its potential to explore in all the terrain like the human and animal. Quadruped Robots have four legs or limbs and follow the gait patterns of quadruped animals. They benefit from increased stability over bipedal robots, especially during movement. They also benefit from lower center of gravity than two-legged systems and require dynamic walking control.This project introduces a quadruped robot which was inspired from Boston Dynamic's Spot.This project aims to develop controlling and gaiting mechanisms (Inverse Kinematics) for this robot.We present custom, voice control systems, and simulated environments in which this robot can be operated.

List of Figures

Contents

Chapter 1

INTRODUCTION

1.1 An Overview on Mobile Robots

Autonomous robots can be roughly classified into two categories, stationary robots and mobile robots. One type of stationary robotic systems is the manipulator, which places surface-mounted components with extremely high precision, i.e., in the assembly line, a robot arm is able to move with high speed and accuracy to perform repetitive tasks, such as spot welding and painting . Another important type of stationary robotic systems is the grasping device , which is commonly used to grasp the desired objects.Nowadays, a variety of robotic hand and finger mechanics have been developed and have been widely applied in agriculture, etc. Unfortunately, these stationary robots are suffering from the lack of mobility and limited range of motion.

In contrast, mobile robots are able to move over rough terrains. Due to the flexibility of the mobile robots, they are widely applied in many areas such as planetary exploration, medical care, and construction, etc.

In these application scenarios, mobile robots can be classified into land-based robots, aerial robots, and underwater robots. Furthermore, according to different mechanical structures, the land-based robots can be divided into wheeled robots, tracked robots, and legged robots, respectively.

1.1.1 Wheeled and Tracked Robots

As one of the most important systems for robot locomotion, wheeled robots play significant roles in transportation and logistics. Generally, wheeled robots move faster and consume less energy than tracked vehicles. Meanwhile, they also tend to be much cheaper and do not present difficulties in terms of balance issues, in comparison with their legged counterparts. Therefore, the control strategies for wheeled robots are less complex than other solutions. Compared to conventional wheeled robots, tracked robots feature larger ground contact patches and less ground impact . However, they are less precise in maneuverability and more difficult to get repaired.

1.1.2 Legged Robots

In nature, there are some specific places, such as planetary surfaces, construction, and disaster salvation, etc. These circumstances are mostly uneven terrains, which limit the application of wheeled robots and tracked robots. However, legged robots are more suitable in these situations for locomotion implementation, by virtue of their terrain adaptation ability. In addition, they can also be applied in plenty of fields, such as medical applications and education . Therefore, the legged robot has become a hot spot in robotics research, due to their advantages over wheeled and tracked robots.

Firstly, legged robots present superior mobility in natural terrains. Unlike wheeled robots, they can avoid small obstacles by making discrete footholds without the necessity of continuous support surface. Moreover, the use of multiple degrees of freedom 3 (DOFs) in the leg joints adjusts the leg extension to position the center of gravity (COG) of the body, and further varies the body height. Hence, legged robots are able to move over irregular terrains by adjusting their leg configurations.

The other advantage is that legged robots have redundant legs, and therefore they can maintain balance and continue walking even with one leg mutilated during static locomotion, which improves failure tolerance. Additionally, the legs can work as an active support base to flexibly actuate the robot body when feet are fixed to the ground.

Although legged locomotion is advantageous compared with traditional wheeled and tracked vehicles, legged robots also have drawbacks. They require complex mechan-

ical design and control algorithms. Furthermore, they are heavier and consume larger energy since a large number of actuators are required to manipulate multiple DOFs legs. There are many types of walking robots depending on the number of legs.

1.1.3 Monopod

The Monopod robots have only one type of locomotion gait, i.e., hopping, which represents a highly nonlinear dynamical behavior, consisting of alternating flight and stance phases.

A Toyota Monopod robot(figure-1.1) developed by Tajima and Suga in 2006 has a total of seven DOFs, three at the hip, one at the knee, two at the ankle, and one at the toe, with one extra DOF for 3D motion. This redundancy is beneficial for reducing the excessive joint speed when it is hopping.

Figure 1.1: Toyota Monopod

Another distinctive jumping robot, named as **Salto-1p** , was created by UC Berkeley's Biomimetic Millisystems Lab in 2017. It is designed to mimic saltatorial animals like galago or bushbaby, which has a strong vertical jumping agility.

1.1.4 Bipedal robot

Bipedal robots make use of sensors to reproduce human capabilities, such as going up and down stairs, jumping, or even doing somersaults.The bipedal **"REEM-C"** is a 100%

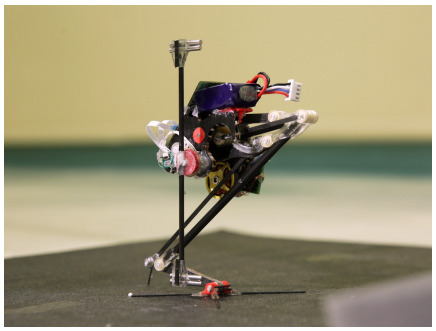


Figure 1.2: Jumping Robot SALTO on platform

Robot Operating System (ROS) based bipedal robotics platform, which was launched by PAL Robotics in 2013 . It is controlled by two computers with Intel Core i7-2710QE CPUs. The sonars and cameras help to map its environment, perform path planning, and achieve obstacle avoidance. The MTi-30 AHRS sensor provides the real-time attitude information and improves walking stability. In addition, all the motors and sensors communicate with the PC via CAN network.

Figure 1.3: Reem-c

In 2017, a new humanoid robot "TALOS", also launched by PAL Robotics, is mainly used for artificial intelligence and human-robot interaction, as shown in Figure 1.4. It is a fully electrical actuated and torque-controllable robot, assembled with torque sensors and EtherCAT communication network to manipulate up to 6 kg payload for each stretched arm.

Figure 1.4: Talos

1.1.5 Quadruped robot

Quadruped robots have the advantage of being statically stable when they are not moving. However, dynamic walking control for leg coordination becomes more complicated and high computational speed is required. Moreover, according to different types of actuators, the application scenarios and control performance are different.

The **"HyQReal"** (figure-1.5) is developed to support humans in emergency scenarios, presented by IIT (Istituto Italiano di Tecnologia) in 2019 . It is an electro-hydraulic powered quadruped robot that features high ruggedness, reliability and energy efficiency. Combined with large output driving force, fast response speed of the hydraulic actuator and compact structure of the electric motors, it can pull a small passenger airplane weighing more than 3 tons. Additionally, it is controlled by two computers on board: One is dedicated to vision and the other is for control.

Figure 1.5: Hyqreal

The **"Cheetah 3"** (figure-1.6) is designed to move across rough terrain and through obstacles without relying on vision, built by MIT in 2018. It is driven by self-made Proprioceptive Actuators (PA's), which have high torque density to improve the load-carrying ability and low-speed efficiency. The Actuation 6 system is designed to achieve certain tasks, such as power plant inspection and blind locomotion, which makes the robot suited for reconnaissance and rescue missions. The locomotion control and state estimation are handled by an embedded computer with a second GEN Core i7 CPU, while the leg controller is performed by an ARM-A8 processor.

Figure 1.6: Cheetah 3

"Laikago" released by Unitree Robotics in 2017, is a lightweight quadruped robot research platform and actuated by 12 Unitree brushless DC motors. It is capable of maintaining stability as the terrain changes and adjusts the posture automatically. Therefore, Laikago is applied to perform tasks such as security patrolling, deliveries or research studies.

Figure 1.7: Laikago

"Anymal" is designed for autonomous operation in challenging environments, created by ETH Zurich and ANYbotics in 2016. Driven by special compliant and highly integrated Series Elastic Actuators (SEA's), the robot is capable of a wide range of motion and high-mobile climbing. The robot navigation is achieved by using onboard thermal cameras, microphones and Lidar scanners. Furthermore, it can open doors autonomously, walk upstairs in any terrain, and crawl through tight spaces by changing the leg configurations.

Figure 1.8: ANYMAL

Spot is the quadruped robot that climbs stairs and traverses rough terrain with unprecedented ease, yet is small enough to use indoors. Built to be a rugged and customizable platform, Spot has a proven track record of supporting remote operation and autonomous sensing across a variety of industries, and is remarkably intuitive, enabling you to focus on the job you do best.

Whether you are a jobsite manager, sensor developer, performer, or anything in between, Spot is the adaptable platform you need to inspect, sense, perform, and more:

- Inspect dangerous, inaccessible, and remote environments
- Automate data collection on your site.
- Carry payloads on unstructured or unknown terrain.

Figure 1.9: SPOT

Spot is intended for industrial and commercial use, by individuals trained to operate it in accordance with its user guide. This version is not intended for use in the home, or by children or others who cannot operate it responsibly. Spot is capable of incredibly robust locomotion; however, Spot should always be operated at least two meters away from people and shouldn't be used in situations where a fall could result in injury to the operator or bystanders. It is your responsibility to ensure that Spot is used safely.

Chapter 2

INSPIRATION - BOSTON DYNAMICS SPOT

2.1 Boston Dynamics :

Boston Dynamics is an American engineering and robotics design company founded in 1992 as a spin-off from the Massachusetts Institute of Technology. Headquartered in Waltham, Massachusetts, Boston Dynamics has been owned by the Hyundai Motor Group since December 2020.

Boston Dynamics is best known for the development of a series of dynamic highly mobile robots, including Big Dog, Spot, Atlas, and Handle. Since 2019, Spot has been made commercially available, making it the first commercially available robot from Boston Dynamics. The company stated its intent to commercialize other robots, including Handle.

2.2 Marc Raibert :

Marc Raibert (born December 22, 1949) is the founder, former CEO, and now Chairman of Boston Dynamics, a robotics company known for creating BigDog, Atlas, Spot, and Handle. Before starting Boston Dynamics, Raibert was a Professor of Electrical Engineering and Computer Science at MIT and an associate professor of Computer Science and Robotics at Carnegie Mellon University. At CMU he founded the Leg Laboratory

Figure 2.1: Boston Dynamics

(1980), a lab that helped establish the scientific basis for highly dynamic robots. Raibert developed the first self-balancing hopping robots, a significant step forward in robotics. Raibert earned an Electrical Engineering, BSEE from Northeastern University in 1973 and a PhD from MIT in 1977.

Figure 2.2: Marc Raibert

2.3 SPOT :

On June 23, 2016, Boston Dynamics revealed the four-legged canine-inspired Spot. On May 11, 2018 CEO of Boston Dynamics Marc Raibert on Tech Crunch Robotics Session 2018 announced that the Spot robot was in pre-production and preparing for commercial availability in 2019.On June 23, 2020, a lone Spot named 'Zeus' was used by SpaceX at their Boca Chica Starship Test Site to help contain sub-cooled liquid nitrogen and to inspect 'Potentially Dangerous' sites at and around the Launchpad.

Figure 2.3: Boston Dynamics Spot

On July 9, 2020, a team of Spot robots performed as cheerleaders in the stands at a baseball match between the Fukuoka SoftBank Hawks and the Rakuten Eagles, backed by a team of SoftBank Pepper Robot Spot performed inspection tasks on the Skarv floating production storage and offloading vessel in November 2020.

Chapter 3

LISA

3.1 What is LISA ?

LISA stands for **"Legged Inter Servo Articulation"**. It is a quadrupedal robot.

3.2 Why LISA ?

As we know that, the value for a product depends upon the market. And as the competition in the market increases the products value decreases.
Currently SPOT costs around 55,00,000 Lakhs in Indian rupees for the general public. And the technology behind the SPOT is kept a secret.
LISA is 1/4 the size of SPOT, which will have all the basic features of SPOT and it will be open source. This will help students and researchers to conduct experiments in the field of legged robotics. And it will be an example that, with low budget and open source we can achieve industrial technology.

Chapter 4

COMPONENTS IN LISA

4.1 Raspberry Pi 4 Model B

Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcomm.The Main Aim of the foundation is to promote teaching basic computer science in schools and developing countries. Later, the raspberry pi is used in target markets such as Robotics and now widely in use for weather monitoring for it's low cost and design. It is typically used by computer and electronic hobbyists, due to its adoption of HDMI and USB devices.

The Raspberry Pi Foundation provides Raspberry Pi OS (formerly called Raspbian), a Debian-based (32-bit) Linux distribution for download, as well as third-party Ubuntu, Windows 10 IoT Core, and specialized distributions for Kodi media centre. It promotes Python and Scratch as the main programming languages. It is the best replacement for desktops with a smaller size. The Size of Raspberry Pi is as of a credit card.

Raspberry Pi also supports external modules that are configurable with GPIO pins for much more functionalities. Few of the modules for Pi are GSM modules, Camera modules, IR modules, Display modules, motors and sensors like Temperature sensor , Moisture sensor , Barometric Sensor.

Raspberry Pi 4 Model B was released in June 2019 with a 1.5 GHz 64-bit quad core ARM Cortex-A72 processor, on-board 802.11ac Wi-Fi, Bluetooth 5.0 , Gigabit Ethernet , two USB 2.0 ports, two USB 3.0 ports, and dual-monitor support via a pair

Figure 4.1: Raspberry Pi 4 Model B

of micro HDMI ports for up to 4K resolution. The Raspberry Pi 4 is also powered via a USB-C port, enabling additional power to be provided to downstream peripherals, when used with an appropriate PSU.

Raspberry Pi for it's computational power and the same features as the regular PC replaces many automated functions in various fields. Also Raspberry Pi can be operable wireless hence, it is used in projects and research fields.

4.2 Metal Gear MG996R Motor

A Servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

Servomotors are not a specific class of motor, although the term servomotor is often used to refer to a motor suitable for use in a closed-loop control system. Servomotors are used in applications such as robotics, automated manufacturing.

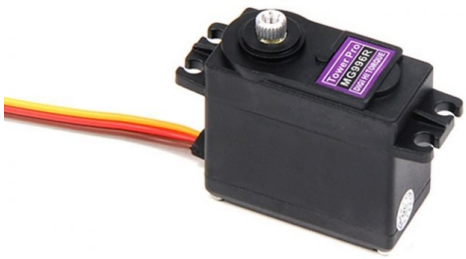


Figure 4.2: MG996R Servo Motor

The MG996R is a metal gear servo motor with a maximum stall torque of 11 kg/cm. Like other RC servos the motor rotates from 0 to 180 degree based on the duty cycle of the PWM wave supplied to its signal pin.

The motor is paired with some type of position encoder to provide position and speed feedback. In the simplest case, only the position is measured. The measured position of the output is compared to the command position, the external input to the controller. If the output position differs from that required, an error signal is generated which then causes the motor to rotate in either direction, as needed to bring the output shaft to the appropriate position. As the positions approach, the error signal reduces to zero and the motor stops.

MG996R Features

- Operating Voltage is +5V typically.

- Current: 2.5A (6V).
- Stall Torque: 9.4 kg/cm (at 4.8V).
- Maximum Stall Torque: 11 kg/cm (6V).
- Operating speed is 0.17 s/60°.
- Gear Type: Metal.
- Rotation : 0°-180°.
- Weight of motor : 55gm.

A servomotor is a closed-loop servomechanism that uses position feedback to control its motion and final position. The input to its control is a signal (either analogue or digital) representing the position commanded for the output shaft.

Next comes the most important parameter, which is the torque at which the motor operates. Again there are many choices here but let us assume the one with 2.5kg/cm torque which comes with the MG996R Motor. This 2.5kg/cm torque means that the motor can pull a weight of 2.5kg when it is suspended at a distance of 1cm. So if you suspend the load at 0.5cm then the motor can pull a load of 5kg similarly if you suspend the load at 2cm then can pull only 1.25. Based on the load which you use in the project you can select the motor with proper torque.

4.3 PCA9685 Servo Driver

Driving servo motors with the Servo library is pretty easy, but each one consumes a precious pin - not to mention some Raspberry Pi processing power. The 16-Channel 12-bit Pulse Width Modulation (PWM) /Servo Driver will drive up to 16 servos over I2C with only 2 pins. The On-board PWM controller will drive all 16 channels simultaneously with no additional processing overhead. You can chain up to 62 of them to control up to 992 servos - all with the same 2 pins. The PWM / Servo Driver is the perfect solution for any project that requires a lot of servos.

Control Pins

- Serial Clock (SCL) - I2C clock pin, connect to your microcontrollers I2C clock line.Can use 3V or 5V logic, and has a weak pullup to Voltage Common Collector (VCC).
- Serial Data (SDA) - I2C data pin, connect to your microcontrollers I2C data line. Can use 3V or 5V logic, and has a weak pullup to VCC.
- Output enable (OE) - Can be used to quickly disable all outputs. When this pin is low all pins are enabled. When the pin is high the outputs are disabled. Pulled low by default so it's an optional pin.

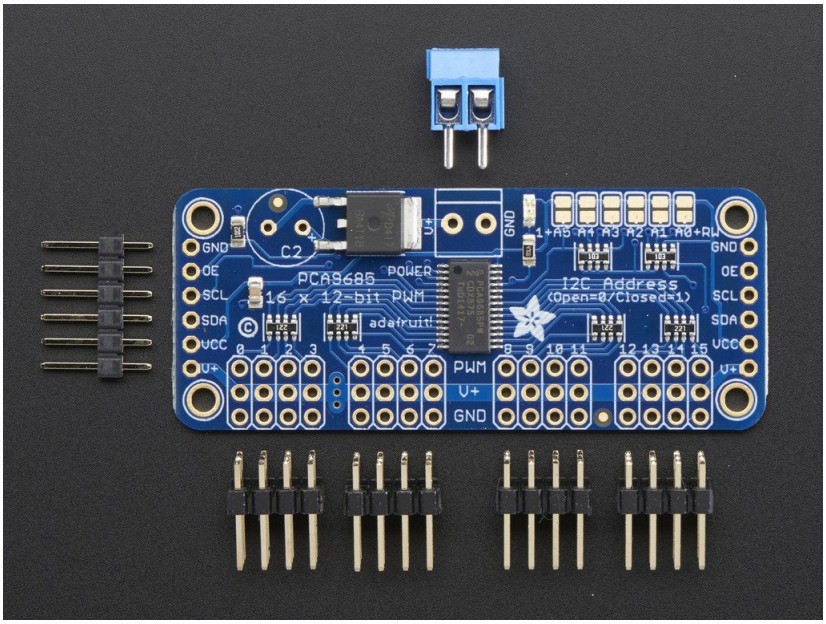


Figure 4.3: PCA9685 Servo Driver

4.3.1 Output Ports

There are 16 output ports. Each port has 3 pins: V+, GND and the PWM output. Each PWM runs completely independently but they must all have the same PWM frequency. That is, for LEDs you probably want 1.0 KHz but servos need 60 Hz - so you cannot use half for LEDs @ 1.0 KHz and half @ 60 Hz.

They're set up for servos but you can use them for LEDs! Max current per pin is 25mA. There are 220 ohm resistors in series with all PWM Pins and the output

logic is the same as VCC.

Power Pins

- Ground (GND) - This is the power and signal ground pin, must be connected.
- Voltage Common Collector (VCC) - This is the logic power pin, connect this to the logic level you want to use for the PCA9685 output, should be 3 - 5V max! It's also used for the 10K pullups on SCL/SDA so unless you have your own pullups, have it match the microcontroller's logic level too.
- Volt (V) - This is an optional power pin that will supply distributed power to the servos. If you are not using for servos you can leave disconnected. It is not used at all by the chip. You can also inject power from the 2-pin terminal block at the top of the board. You should provide 5-6V DC if you are using servos. If you have to, you can go higher to 12V DC, but if you mess up and connect VCC to V+ you could damage your board!.

4.4 Buck Converter

A buck converter (step-down converter) is a DC-to-DC power converter which steps down voltage, while drawing less average current from its input supply to its output load. It is a class of Switched Mode Power Supply (SMPS) typically containing at least two semiconductors, a diode and a transistor, although modern buck converters frequently replace the diode with a second transistor used for synchronous rectification and at least one energy storage element, a capacitor, inductor, or the two in combination. To reduce voltage ripple, filters made of capacitors, sometimes in combination with inductors are normally added to such a converter's output load-side filter and input supply-side filter.

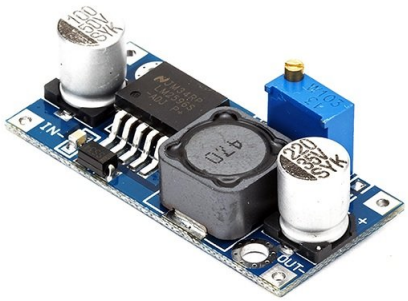


Figure 4.4: Buck Converter

Switching converters (such as buck converters) provide much greater power efficiency as DC-to-DC converters than linear regulators, which are simpler circuits that lower voltages by dissipating power as heat, but do not step up output current. It is intended to buck (or lower) the input voltage of an unregulated DC supply to a stabilized lower output voltage.

Buck converters can be highly efficient often higher than 90 percent, making them useful for tasks such as converting a computer's main bulk supply voltage often 12 V down to lower voltages needed by USB, DRAM and the CPU (1.8 V or less). Buck converters are, especially compared to traditional voltage regulators, widely valued for their extremely high efficiencies. Buck converters are often used in lieu of traditional, non-efficient linear regulators to provide low-voltage on-board power in a variety of applications such as microprocessors, communication equipment, control systems, and more.

4.5 Switched-Mode Power Supply

A switched-mode power supply is an electronic power supply that incorporates a switching regulator to convert electrical power efficiently. Like other power supplies, an SMPS

transfers power from a DC or AC source (often mains power, see AC adapter) to DC loads, such as a personal computer, while converting voltage and current characteristics. Unlike a linear power supply, the pass transistor of a switching-mode supply continually switches between low-dissipation, full-on and full-off states, and spends very little time in the high dissipation transitions, which minimizes wasted energy. A hypothetical ideal switched-mode power supply dissipates no power. Voltage regulation is achieved by varying the ratio of on-to-off time (also known as duty cycles). In contrast, a linear power supply regulates the output voltage by continually dissipating power in the pass transistor. This higher power conversion efficiency is an important advantage of a switched-mode power supply.

Figure 4.5: Switched Mode Power Supply

Switched-mode power supplies can also be substantially smaller and lighter than a linear supply because the transformer can be much smaller. This is because it operates on the switching frequency which ranges from several 100KHz to several MHz in contrast to the 50-60Hz which is typical for the mains AC frequency. Despite the reduction in size, the power supply topology itself and the requirement for electromagnetic interfer-

ence suppression in commercial designs result in a usually much greater component count and corresponding circuit complexity.Switching regulators are used as replacements for linear regulators when higher efficiency, smaller size or lighter weight are required. They are, however, more complicated; switching currents can cause electrical noise problems if not carefully suppressed, and simple designs may have a poor power factor.

Chapter 5

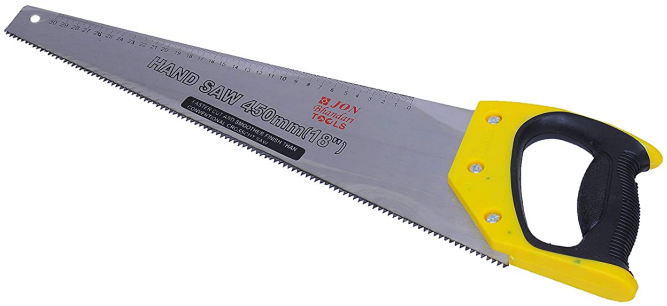
TOOLS REQUIRED

5.1 Bosch Impact Drill Kit

5.2 3D printer

5.3 Cutting Pliers

5.4 Hand Saw

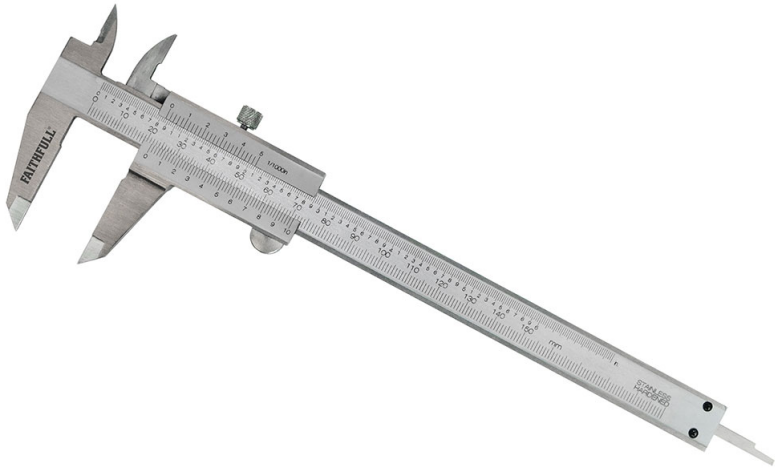


5.5 Multimeter

5.6 Screwdriver Kit

5.7 Steel Ruler

5.8 Vernier Caliper



5.9 Wire Cutter

5.10 Zip Tie

5.11 Glue Gun

Chapter 6

DOG ANATOMY

NOTE : It is essential to understand the anatomy of a dog to develop an appropriate gaiting mechanism for the proposed robot dog.

Dog movement can be viewed from many different aspects. In the working dog world or athletic dog world, optimal movement results in optimal performance and a lengthy operational career. In the pet world, movement is taken for granted until it is not normal.it is important to have an understanding of the components of movement or locomotion. Musculoskeletal movement and its actions are the core focus when trying to understand lameness, rehabilitation and performance of the dog. Locomotion and segmental movement are terms that can be used to describe the musculoskeletal actions involved when an animal is moving or performing different related actions.

Locomotion is described as the movements of the body that allows it to get from one location to another. The body is made of components or segments, so it is the sum of the segmental movements that produce locomotion.

Discerning the difference between body locomotion and segmental movement, allows for relative descriptors. Gait and stride are two terms that can be used to describe body locomotion. Limb movement, step cycle, stance phase, swing phase, flexion, extension, angular displacement and range-of-motion are terms used when describing segmental movement.

6.1 Complexity of the Movement Actions:

Each leg is made of components (segments) which can be defined in different manners. One way is anatomically by the various tissue types, i.e, bones, muscles, etc. Another way to define the limb is geographically from the proximity to the body core.

These would include (from proximal to distal) the humeral/femoral, radial/tibial, carpal/tarsal and distal extremities (paws). These are defined as the areas between the joints. Assessing segmental movement allows for an understanding of how they interrelate with each other.

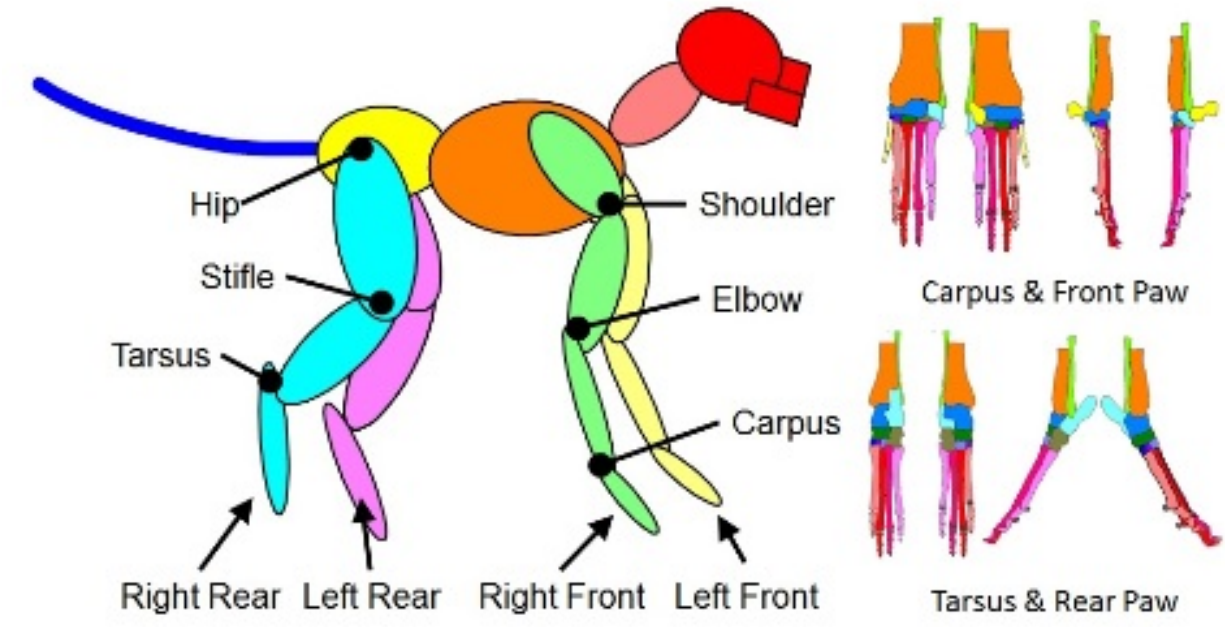


Figure 6.1: Geographical segments of the dog's limbs

6.2 Step Cycle

Locomotion is made of the combined actions of the legs balanced with the core efforts of the abdomen, thorax and neck. Each leg will go through a step cycle. Therefore a biped (human) will have two step cycles, a quadruped (dog) will have four step cycles and a centipede will have 100 step cycles. It is typically defined as the actions the leg goes through from one point in the cycle until the leg again reaches that point of the cycle.

It is generally defined as the leg actions that occur starting at the first touchdown of the paw until that paw again touches down.

Step cycle consists following two phases :

- Stance Phase
- Swing Phase

6.3 Leg Actions in the Step Cycle: The Basis of Locomotion:

The step cycle is the term that defines the cycle of actions of the tissues and structures that produce repetitive movement of the leg. As previously described, this involves a complex series of stimuli and cellular actions. Each leg plays its own part in the full action of locomotion. The forces of propulsion occur during the stance phase. Then the leg raises off the ground during the swing phase as it prepares to repeat the action.

6.3.1 Step cycle of front leg:

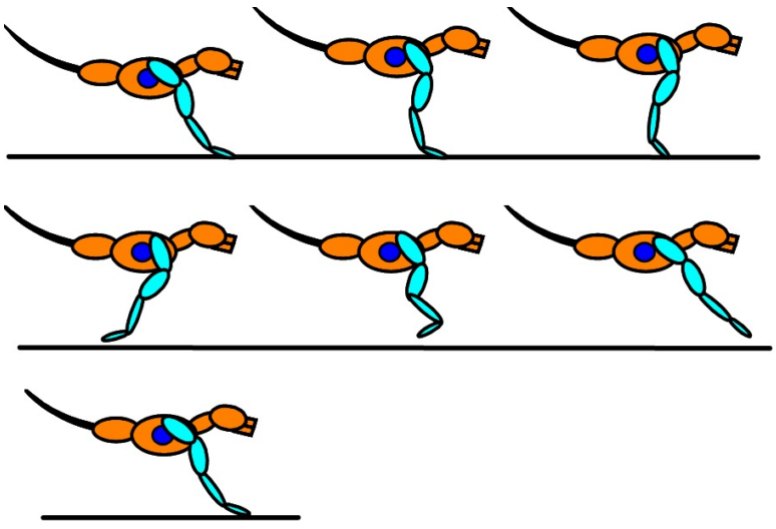


Figure 6.2: Step cycle of front leg

6.3.2 Step cycle of rear leg:

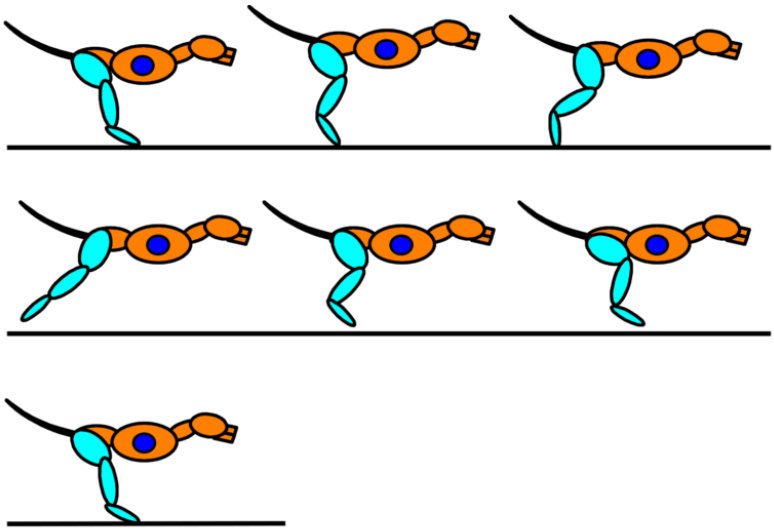


Figure 6.3: Step cycle of rear leg

6.4 Actions of step cycle and body weight balancing:

The step cycle of a dog's leg, or limb, is typically defined as beginning when the paw first touches down and ends when that paw again touches down. Each leg will go through a step cycle but how the forces are received during the cycle are dependent on the anatomical design of the dog. Because of this, each leg will have different forces applied to it. The dog's design is similar to a four legged table. The four legs support the body but there are two appendages that are attached to the front (the head) and back (the tail).

In the case of most any quadruped (a four legged animal) the weight of the head balances the weight towards the front end. The head weighs more than the tail, so the front legs end up bearing more weight than the back legs. The right legs will bear weight different than the left legs. The legs of the right side will have to offset the weight of the body being on their left side, while the legs on the left have to offset the weight of the body being on their right side.

The front legs typically bear more weight than the rear legs in the normal dog. The weight of the head and neck shift the weight forward.

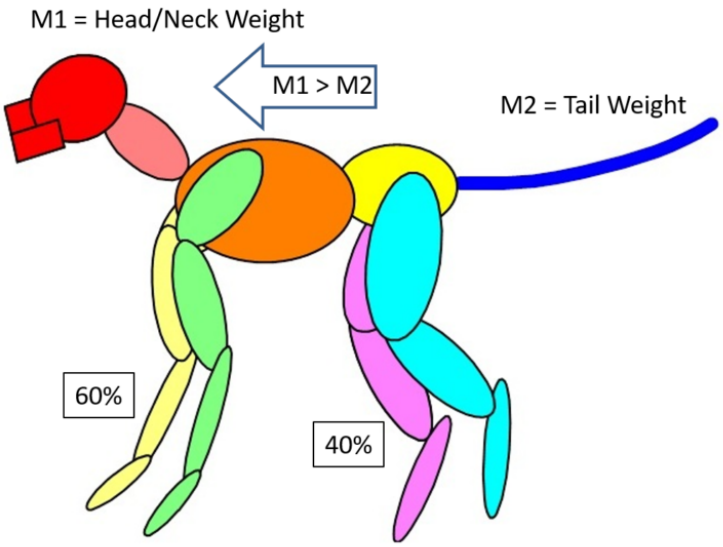


Figure 6.4: Body Balance

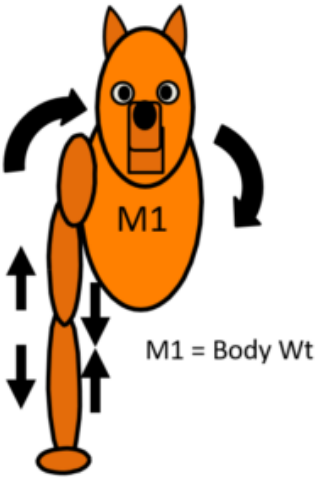
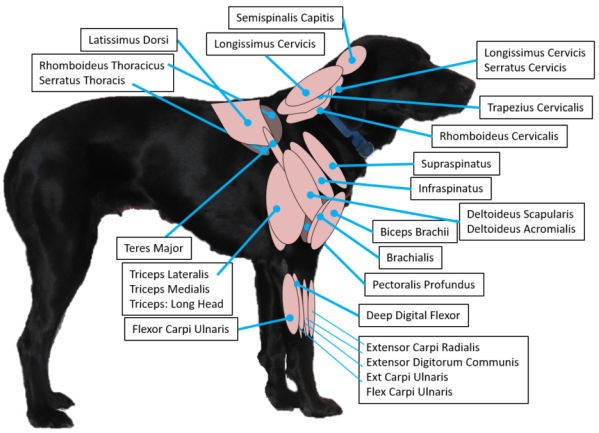


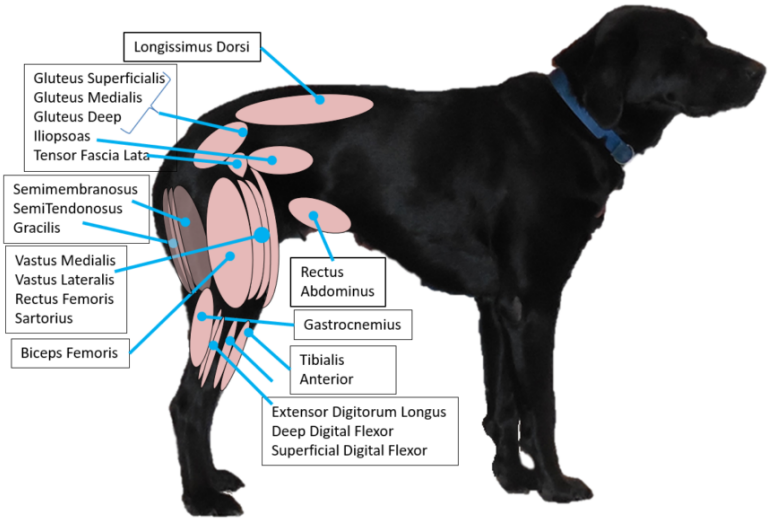
Figure 6.5: Body Balance

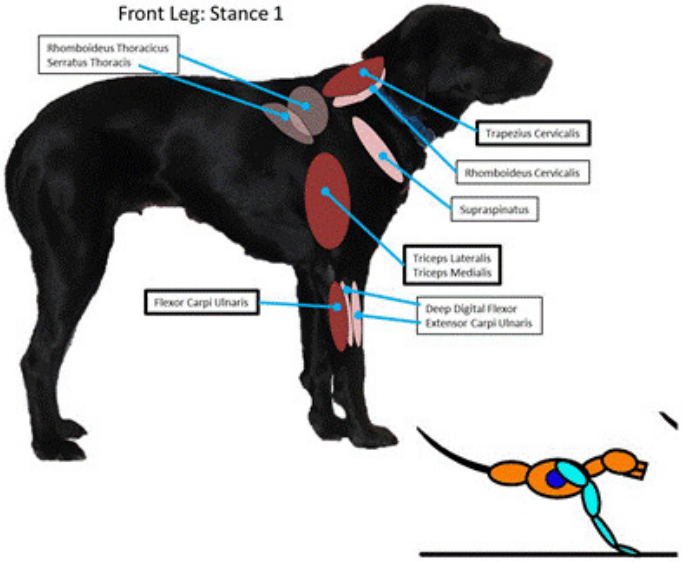
Each leg of the dog will bear the forces differently. When the right front is in the stance phase, the weight of the body will act on its medial aspect. This creates distractive forces on the outside and compressive forces on the inside of the leg.

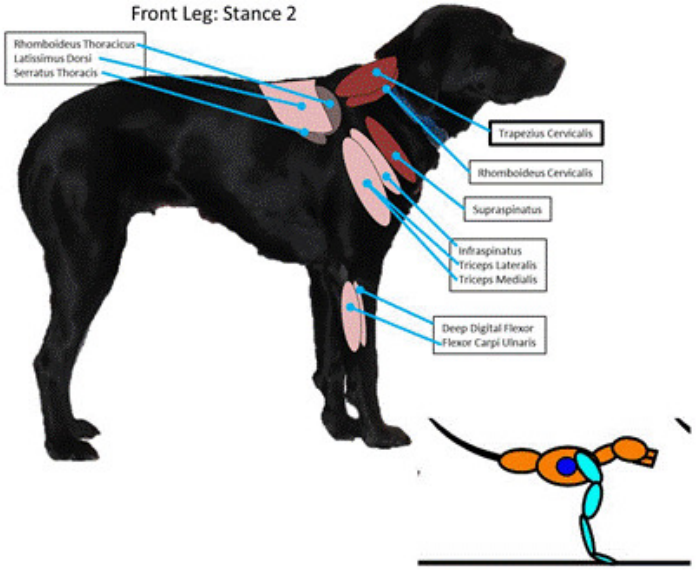
6.5 Muscles used in front leg movement:

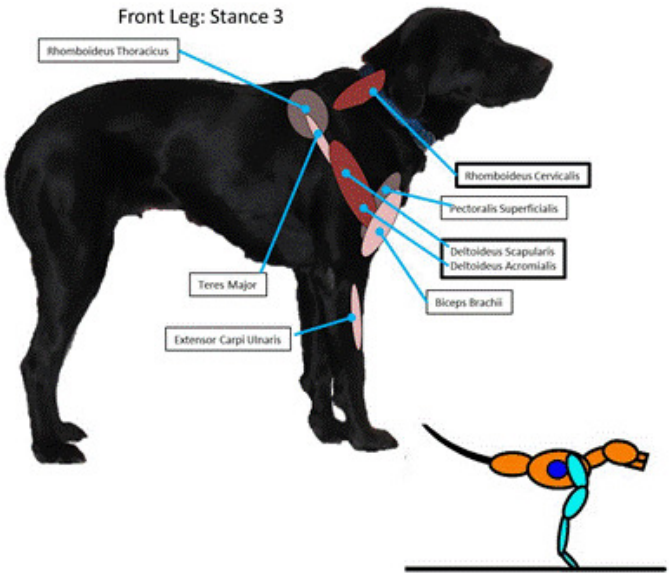


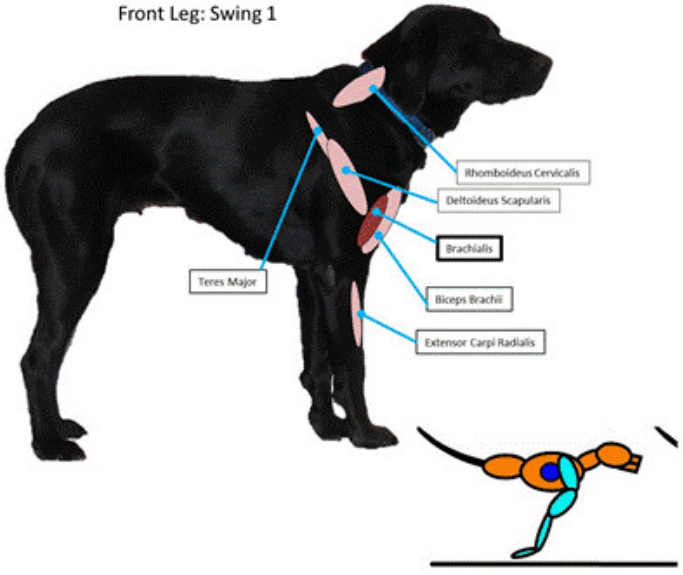
6.6 Muscles used in rear leg movement:

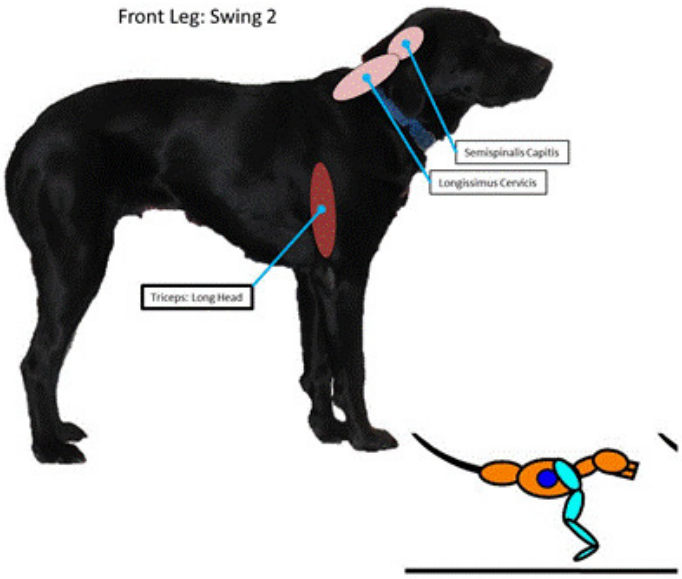


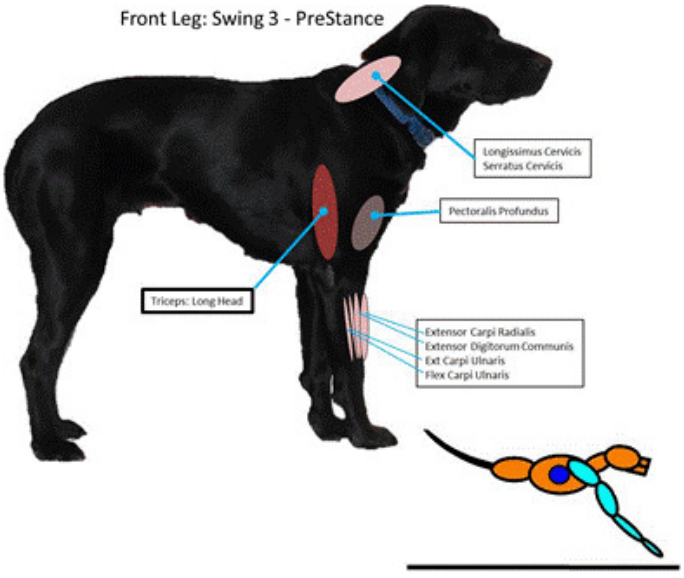

Front Leg: Stance 1
Rhomboideus Thoracicus
Serratus Thoracis
Trapezius Cervicalis
Rhomboideus Cervicalis
Supraspinatus
Triceps Lateralis
Triceps Medialis
Flexor Carpi Ulnaris
Deep Digital Flexor
Extensor Carpi Ulnaris

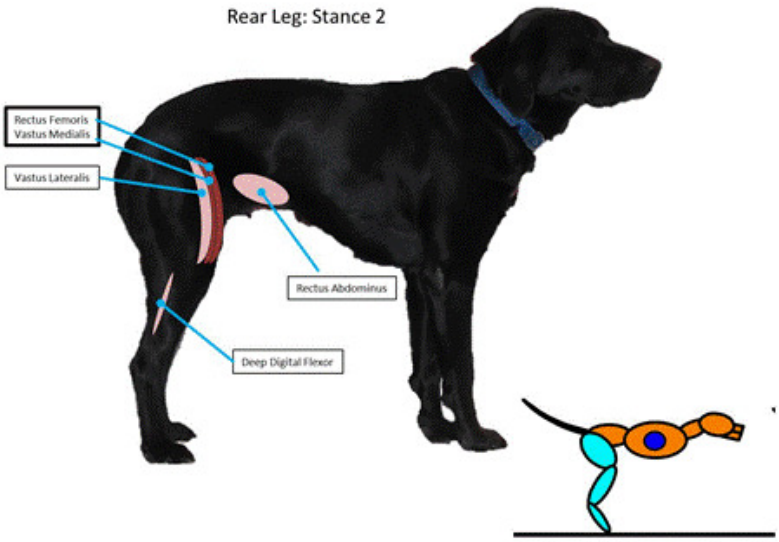

Front Leg: Stance 2
Rhomboideus Thoracicus
Latissimus Dorsi
Serratus Thoracis
Trapezius Cervicalis
Rhomboideus Cervicalis
Supraspinatus
Infraspinatus
Triceps Lateralis
Triceps Medialis
Deep Digital Flexor
Flexor Carpi Ulnaris

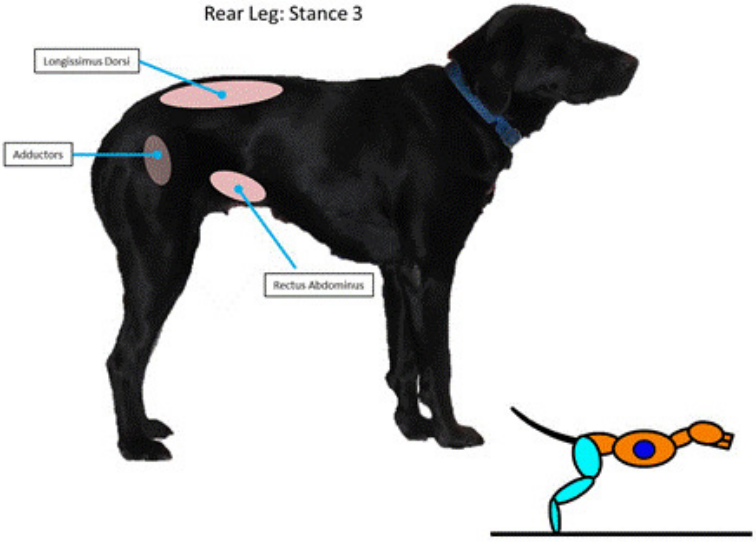

Front Leg: Stance 3
Rhomboideus Thoracicus
Rhomboideus Cervicalis
Pectoralis Superficialis
Deltoideus Scapularis
Deltoideus Acromialis
Teres Major
Biceps Brachii
Extensor Carpi Ulnaris

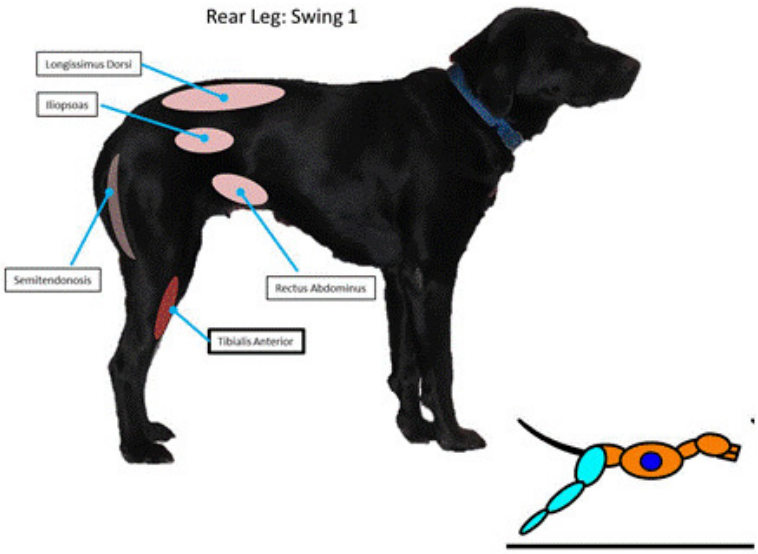

Front Leg: Swing 1
Rhomboideus Cervicalis
Deltoideus Scapularis
Brachialis
Teres Major
Biceps Brachii
Extensor Carpi Radialis

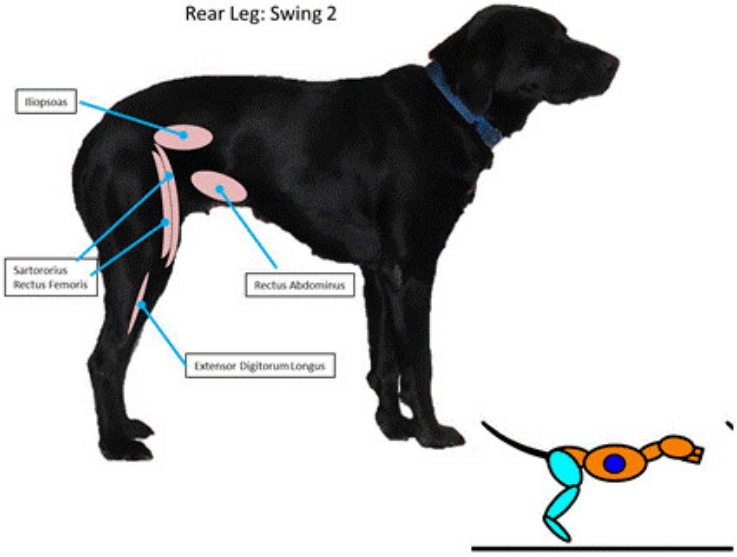

Front Leg: Swing 2
Semispinalis Capitis
Longissimus Cervicis
Triceps: Long Head

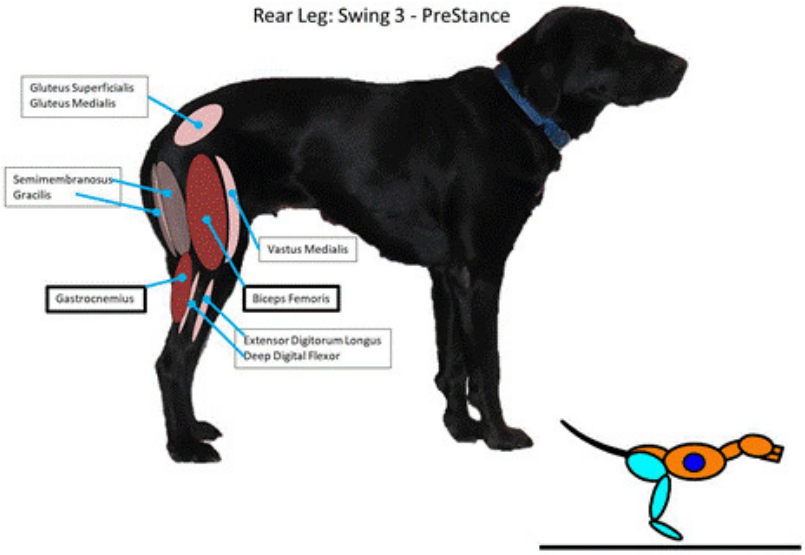

Front Leg: Swing 3 - PreStance
Longissimus Cervicis
Serratus Cervicis
Pectoralis Profundus
Triceps: Long Head
Extensor Carpi Radialis
Extensor Digitorum Communis
Ext Carpi Ulnaris
Flex Carpi Ulnaris


Rear Leg: Stance 2
Rectus Femoris
Vastus Medialis
Vastus Lateralis
Rectus Abdominus
Deep Digital Flexor


Rear Leg: Stance 3
Longissimus Dorsi
Adductors
Rectus Abdominus


Rear Leg: Swing 1
Longissimus Dorsi
Iliopsoas
Semitendonosis
Rectus Abdominus
Tibialis Anterior


Rear Leg: Swing 2
Iliopsoas
Sartororius
Rectus Femoris
Rectus Abdominus
Extensor Digitorum Longus


Rear Leg: Swing 3 - PreStance
Gluteus Superficialis
Gluteus Medialis
Semimembranosus
Gracilis
Vastus Medialis
Gastrocnemius
Biceps Femoris
Extensor Digitorum Longus
Deep Digital Flexor

Chapter 7

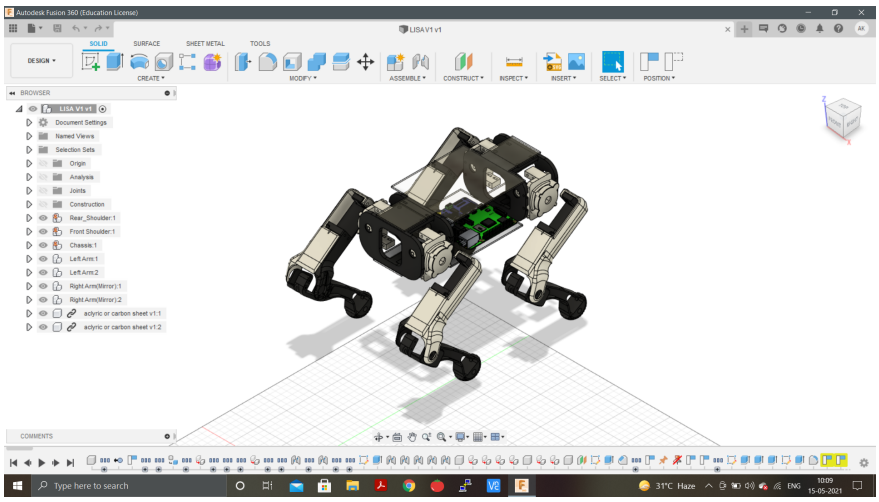
DESIGN

7.1 3-D MODELING

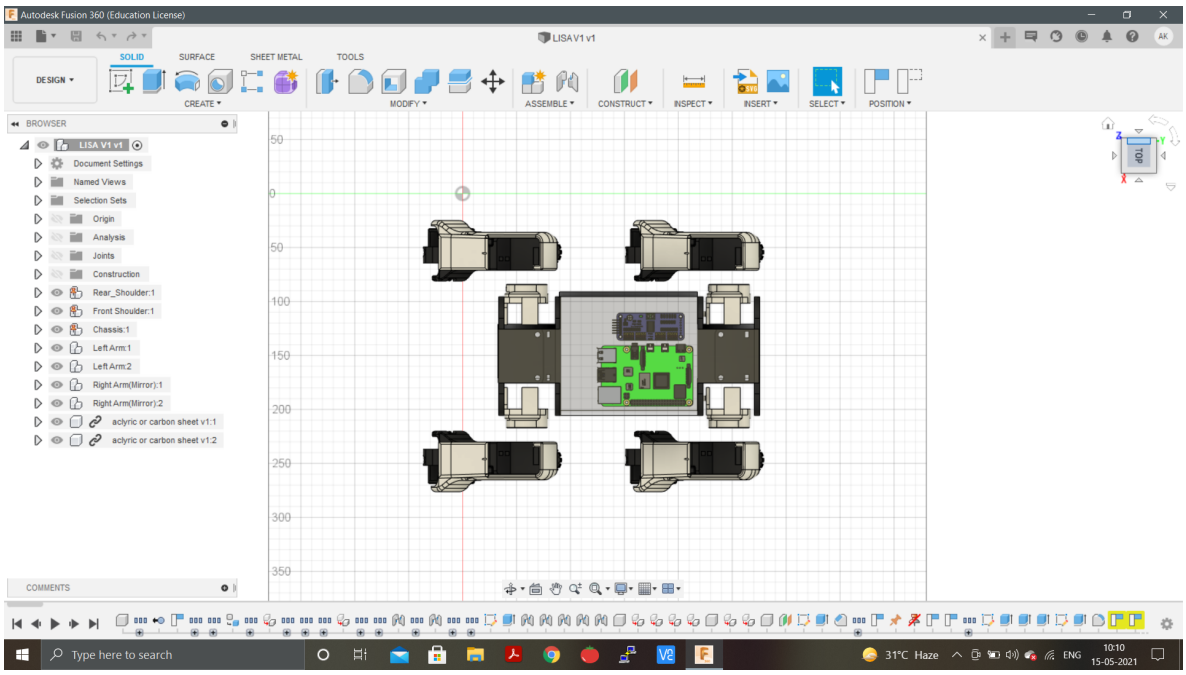
7.1.1 Fusion 360

Fusion 360 is a cloud-based CAD/CAM tool for collaborative product development. Fusion 360 enables exploration and iteration on product ideas and collaboration within distributed product development team. Fusion 360 combines organic shapes modelling, mechanical design and manufacturing in one comprehensive package.

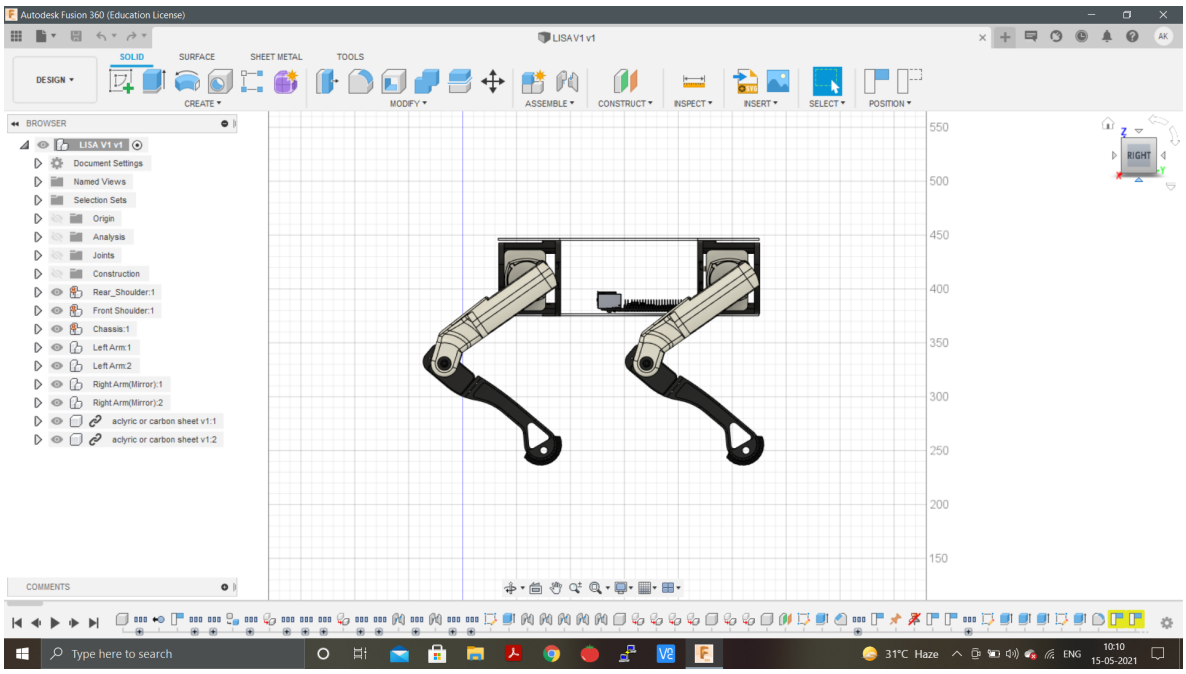
Fusion 360 is free software for educators, academic institutions, and students as a 1-year subscription.



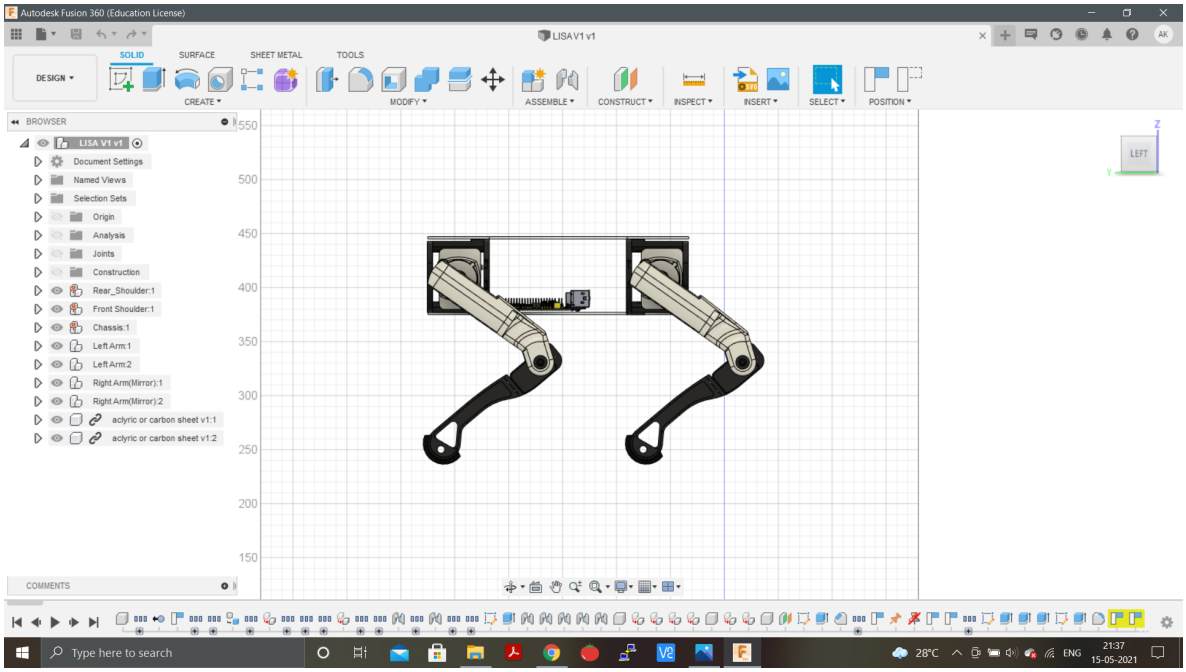
7.1.2 Lisa top view



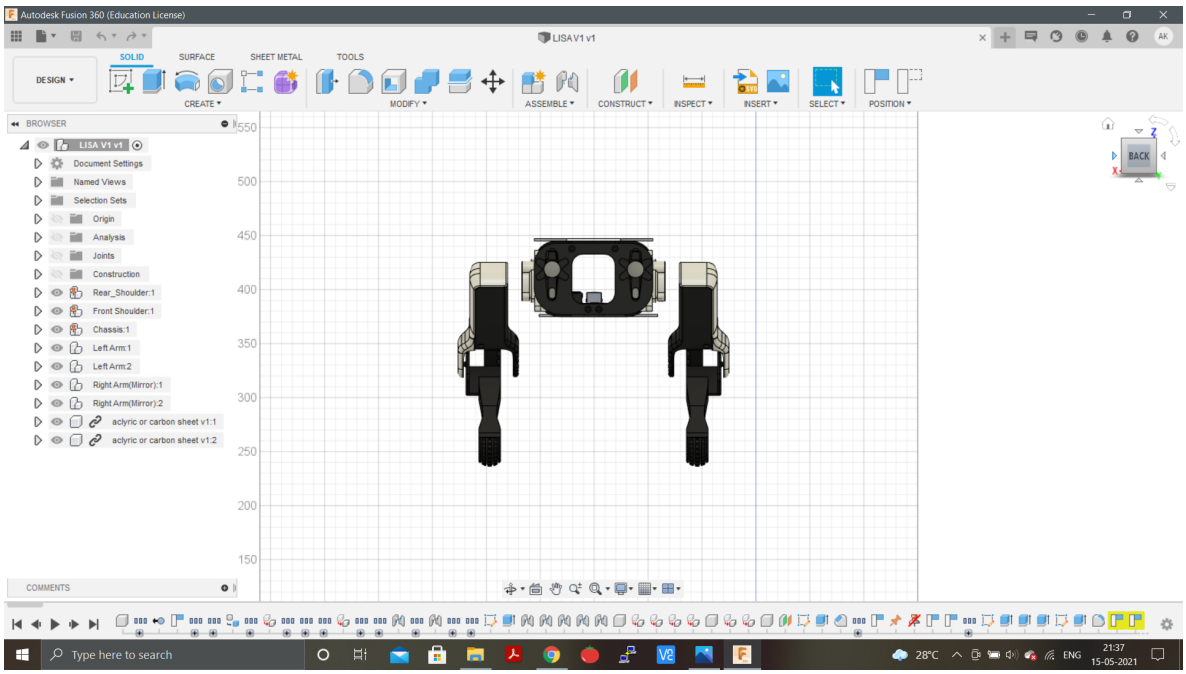
7.1.3 Lisa right side view



7.1.4 Lisa left side view



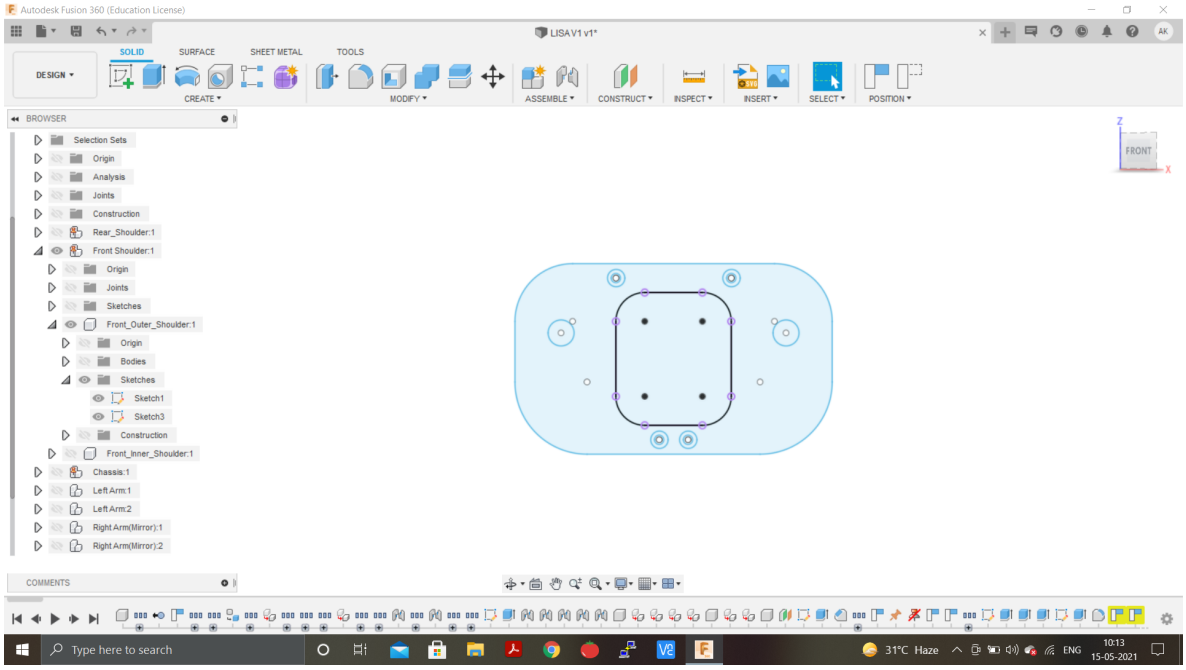
7.1.5 Lisa front view



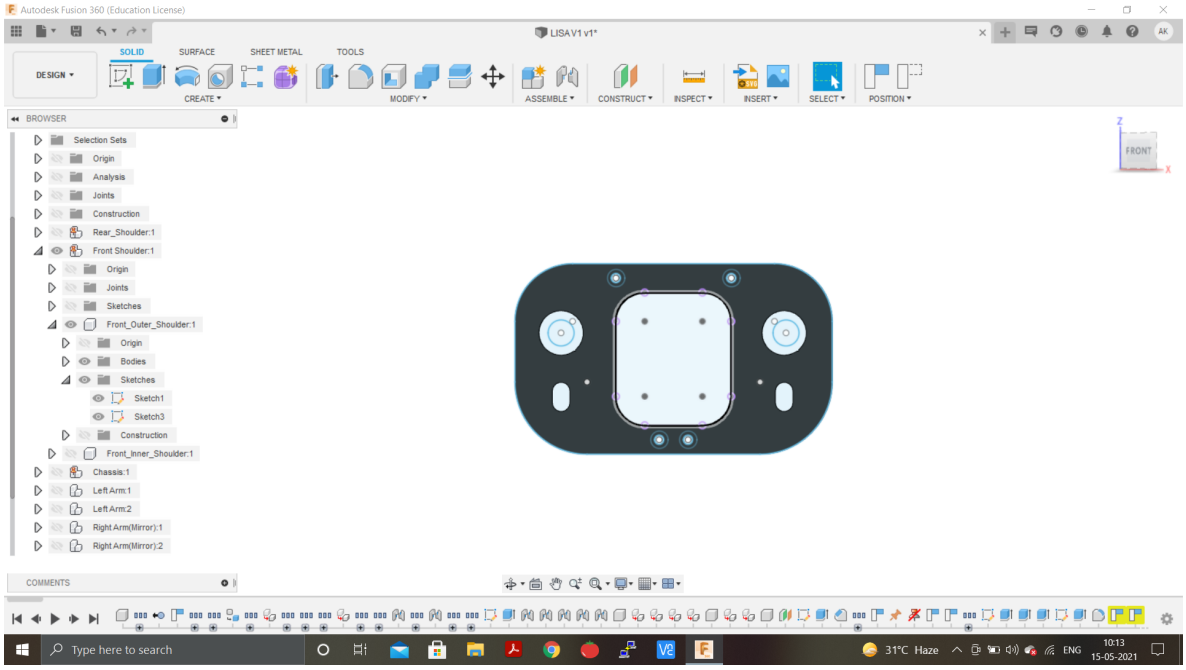
7.1.6 Components

Front shoulder

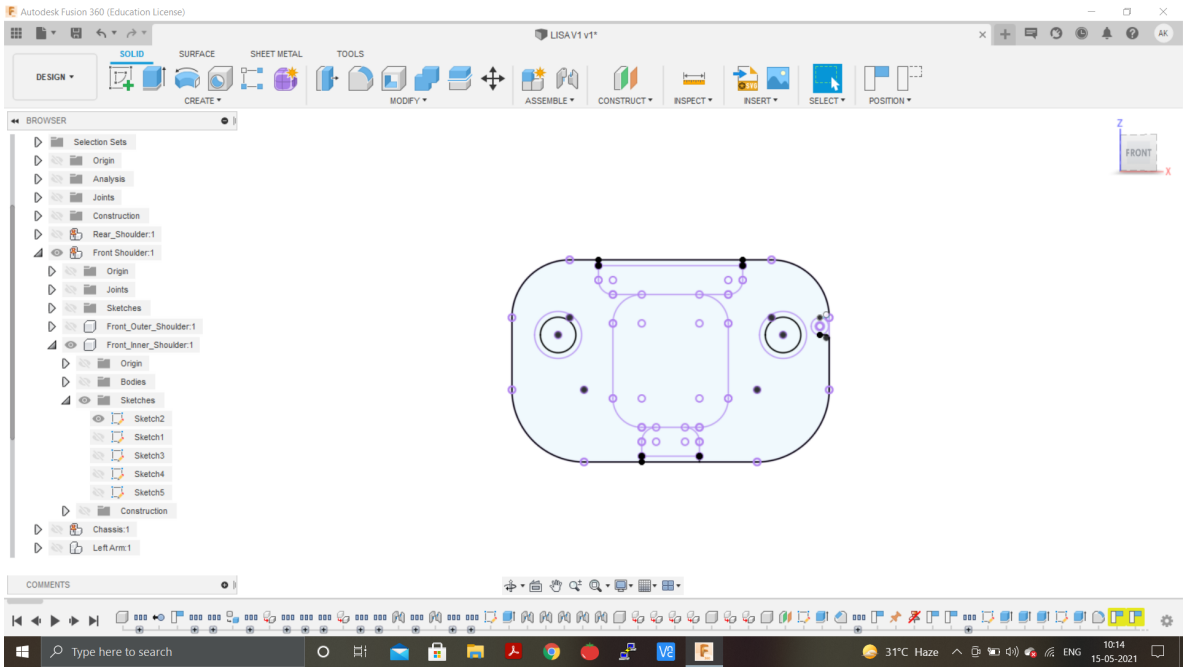
Using line, arc and circle tools in sketch section create the following sketch considering nuts and screw dimensions.



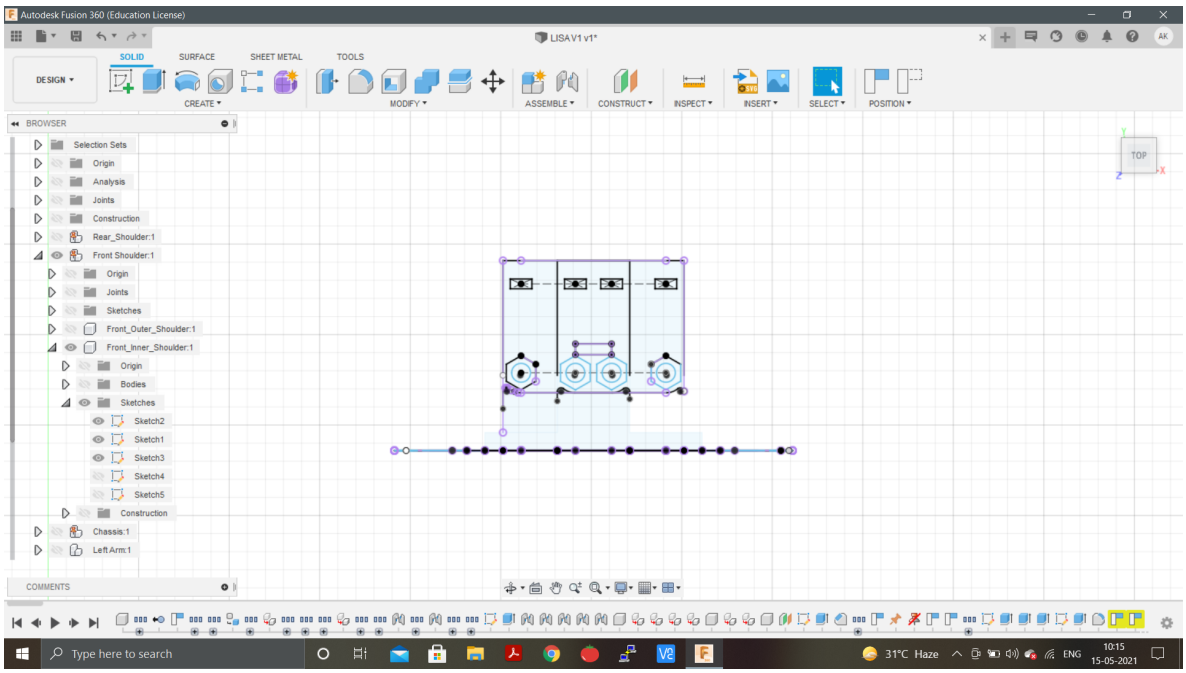
Finish the sketch and extrude the sketch.

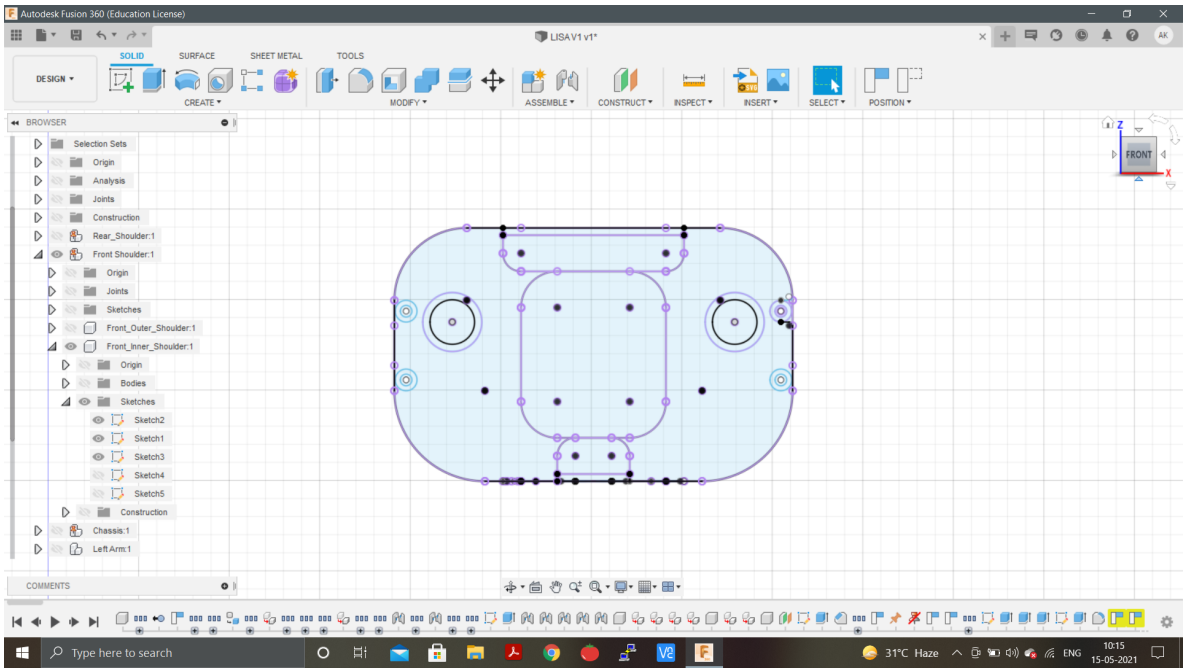


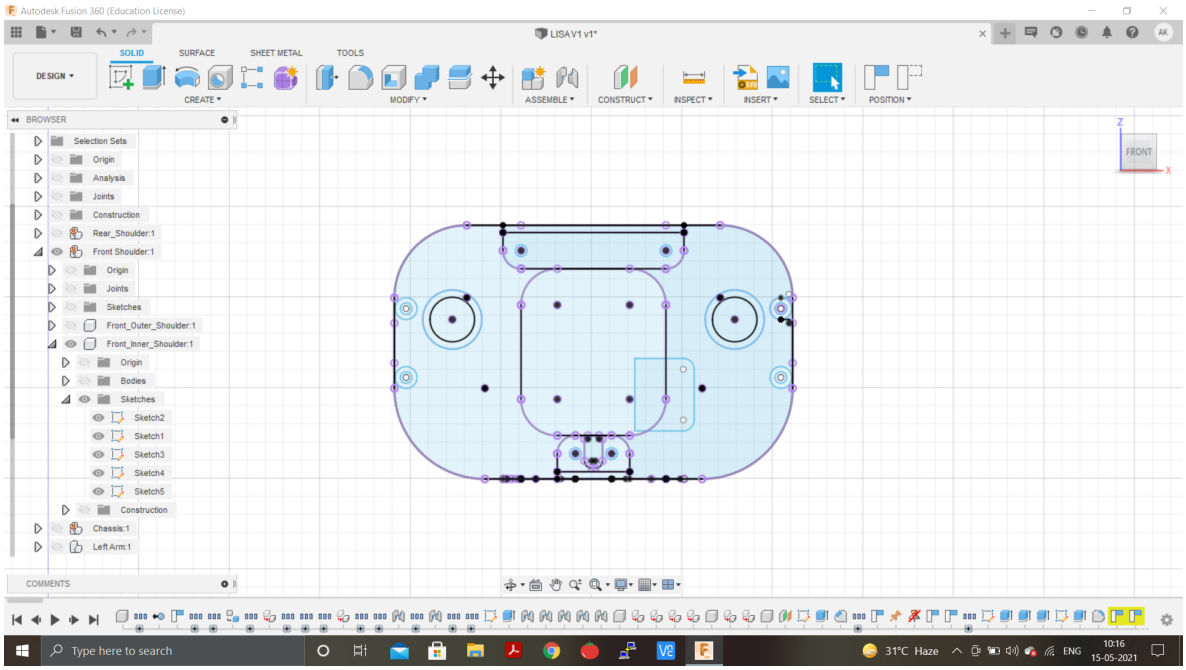
Creating the second part of front shoulder.



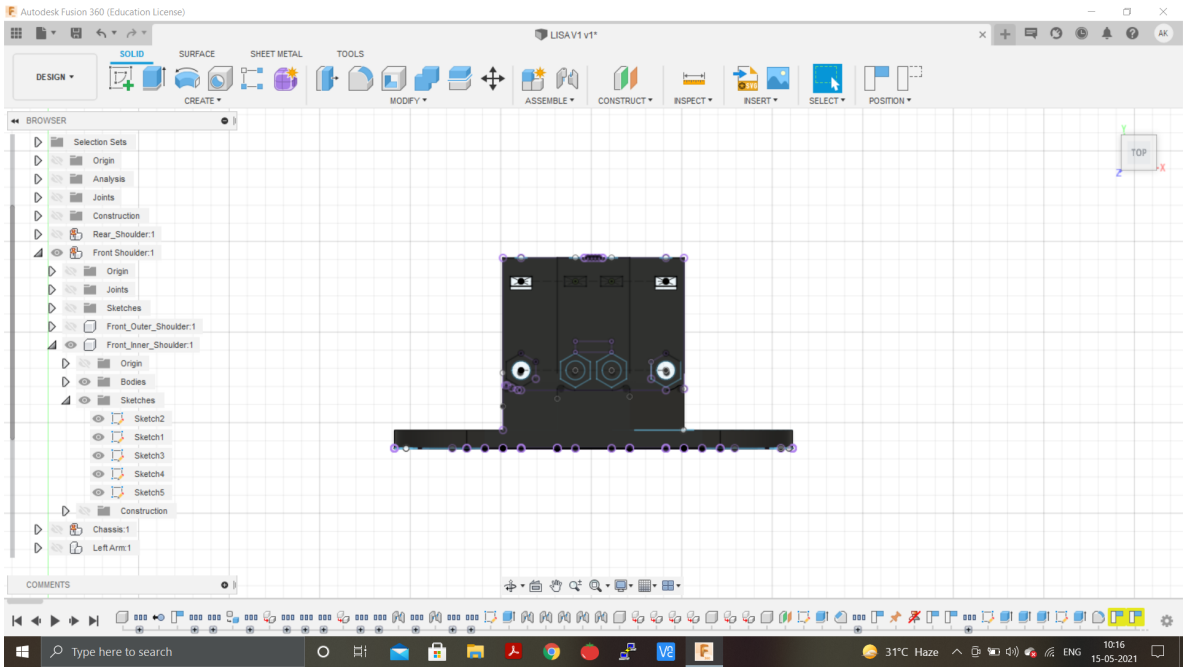
considering the screw and nuts dimensions for combining upper and lower acrylic sheet.

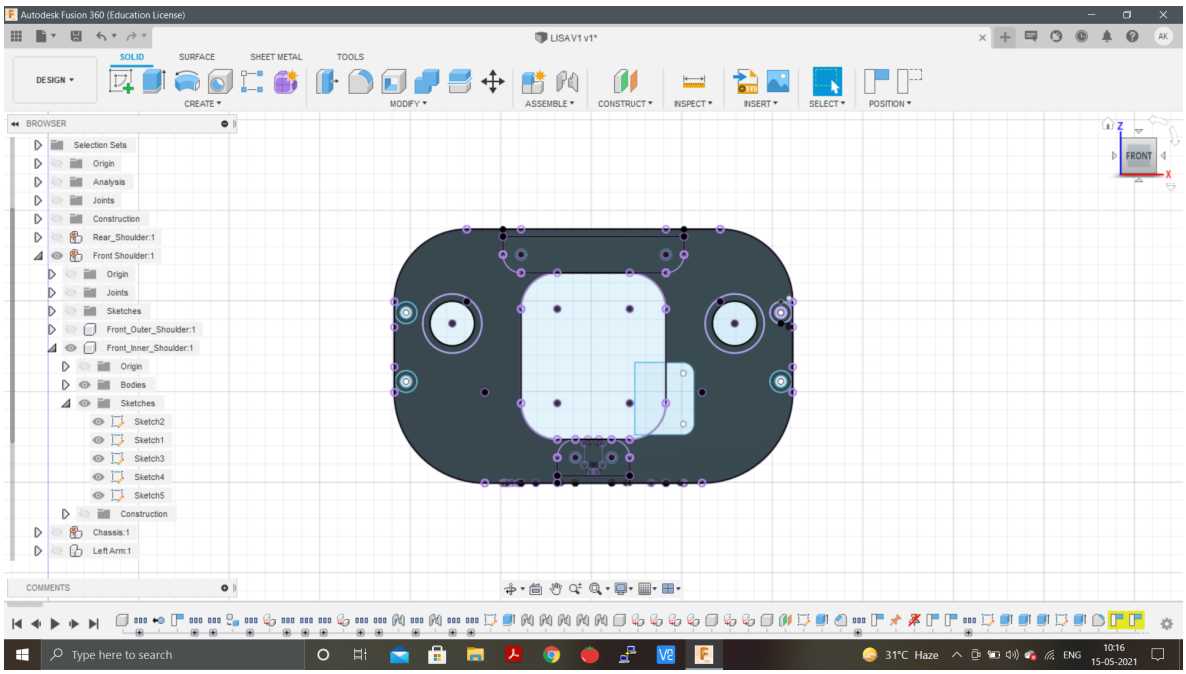


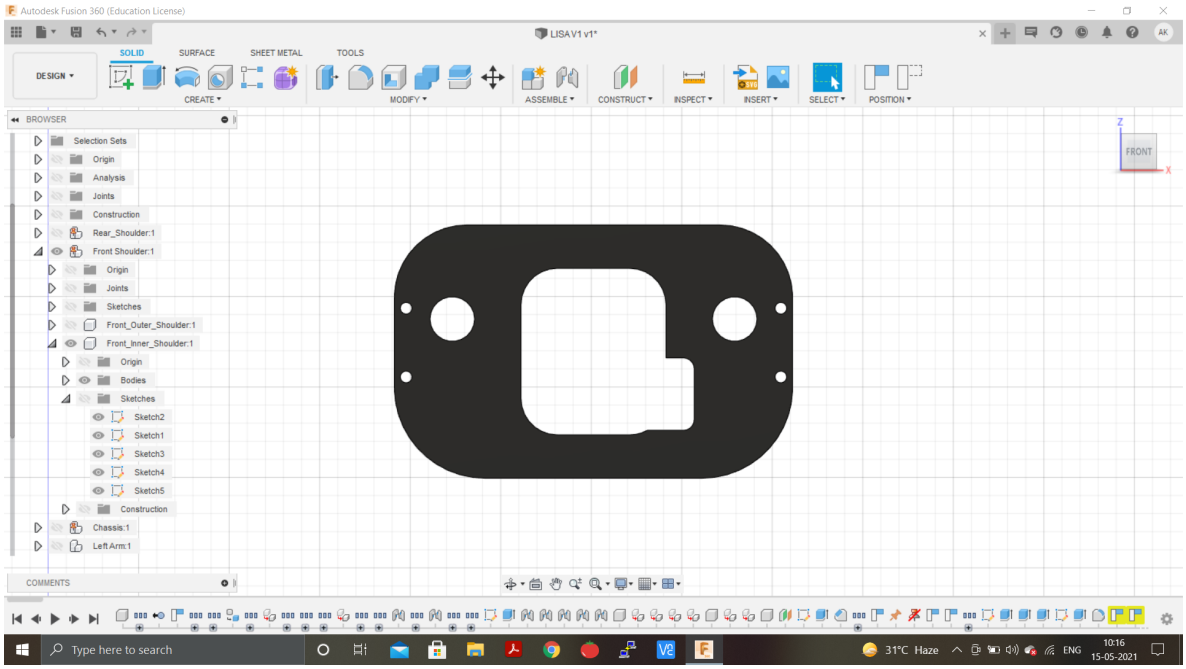

Autodesk Fusion 360 (Education License)
LISA V1 v1*
SOLID
SURFACE
SHEET METAL
TOOLS
DESIGN
CREATE
MODIFY
ASSEMBLE
CONSTRUCT
INSPECT
INSERT
SELECT
POSITION
BROWSER
Selection Sets
Origin
Analysis
Joints
Construction
Rear_Shoulder:1
Front Shoulder:1
Origin
Joints
Sketches
Front_Outer_Shoulder:1
Front_Inner_Shoulder:1
Origin
Bodies
Sketches
Sketch2
Sketch1
Sketch3
Sketch4
Sketch5
Construction
Chassis:1
Left Arm:1
COMMENTS
FRONT
Type here to search
31°C Haze
ENG
10:15
15-05-2021

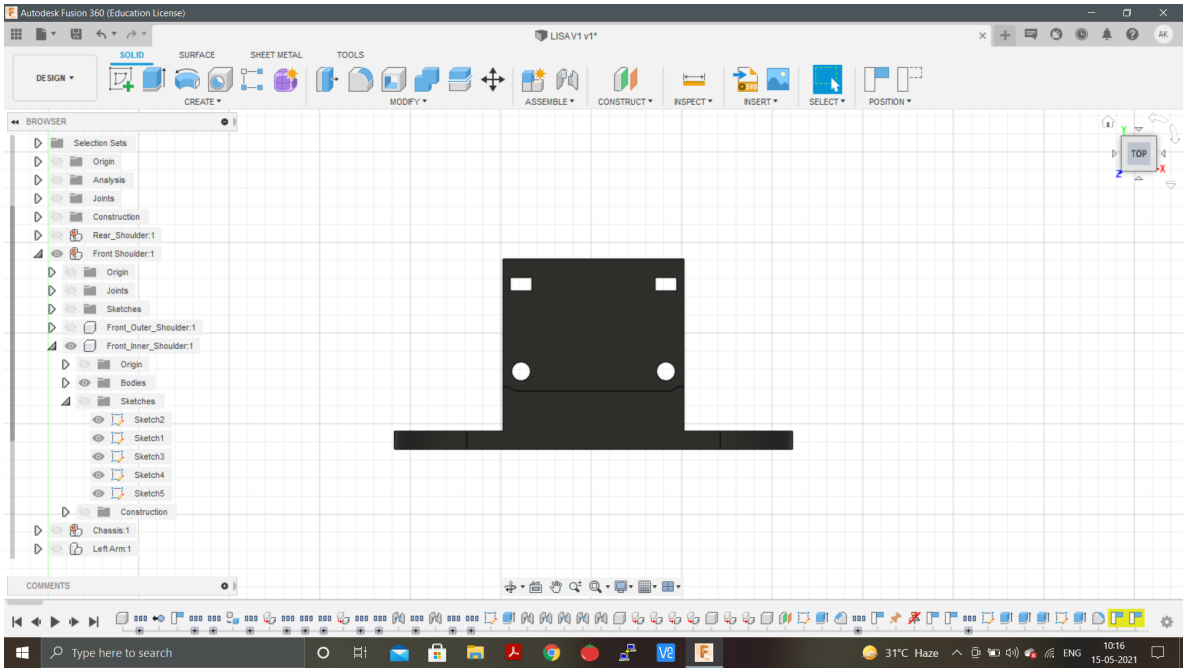

Autodesk Fusion 360 (Education License)
LISA V1 v1*
SOLID
SURFACE
SHEET METAL
TOOLS
DESIGN
CREATE
MODIFY
ASSEMBLE
CONSTRUCT
INSPECT
INSERT
SELECT
POSITION
BROWSER
Selection Sets
Origin
Analysis
Joints
Construction
Rear_Shoulder:1
Front Shoulder:1
Origin
Joints
Sketches
Front_Outer_Shoulder:1
Front_Inner_Shoulder:1
Origin
Bodies
Sketches
Sketch2
Sketch1
Sketch3
Sketch4
Sketch5
Construction
Chassis:1
Left Arm:1
COMMENTS
FRONT
Type here to search
31°C Haze
ENG
10:16
15-05-2021

Extrude by selecting planes.

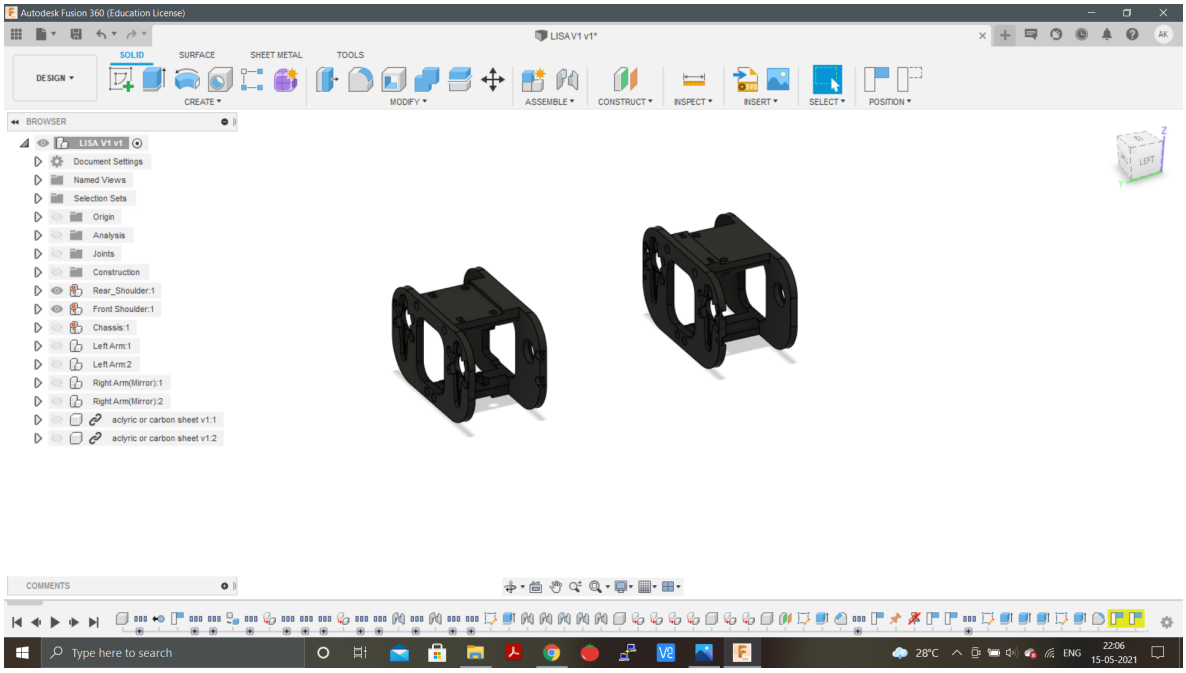




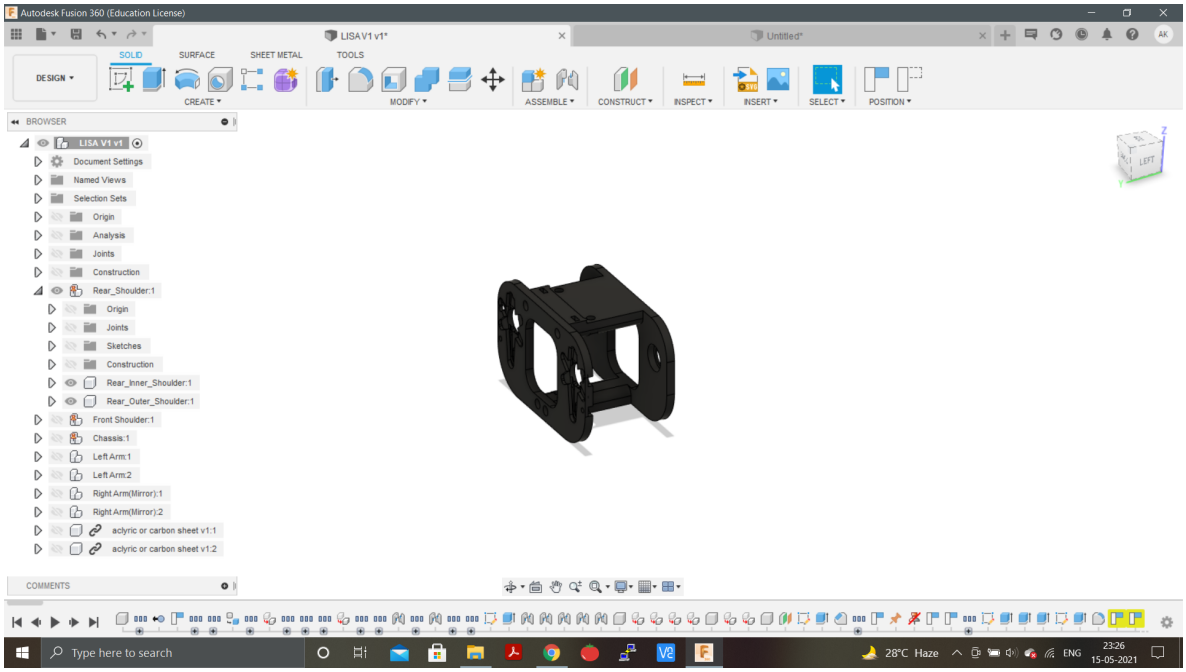

Autodesk Fusion 360 (Education License)
LISA V1 v1*
DESIGN
SOLID
SURFACE
SHEET METAL
TOOLS
CREATE
MODIFY
ASSEMBLE
CONSTRUCT
INSPECT
INSERT
SELECT
POSITION
BROWSER
Selection Sets
Origin
Analysis
Joints
Construction
Rear_Shoulder:1
Front Shoulder:1
Origin
Joints
Sketches
Front_Outer_Shoulder:1
Front_Inner_Shoulder:1
Origin
Bodies
Sketches
Sketch2
Sketch1
Sketch3
Sketch4
Sketch5
Construction
Chassis:1
Left Arm:1
COMMENTS
FRONT
Type here to search
31°C Haze
ENG
10:16
15-05-2021


Autodesk Fusion 360 (Education License)
LISA V1 v1*
DESIGN
SOLID
SURFACE
SHEET METAL
TOOLS
CREATE
MODIFY
ASSEMBLE
CONSTRUCT
INSPECT
INSERT
SELECT
POSITION
BROWSER
Selection Sets
Origin
Analysis
Joints
Construction
Rear_Shoulder:1
Front Shoulder:1
Origin
Joints
Sketches
Front_Outer_Shoulder:1
Front_Inner_Shoulder:1
Origin
Bodies
Sketches
Sketch2
Sketch1
Sketch3
Sketch4
Sketch5
Construction
Chassis:1
Left Arm:1
COMMENTS
TOP
Type here to search
31°C Haze
ENG
10:16
15-05-2021

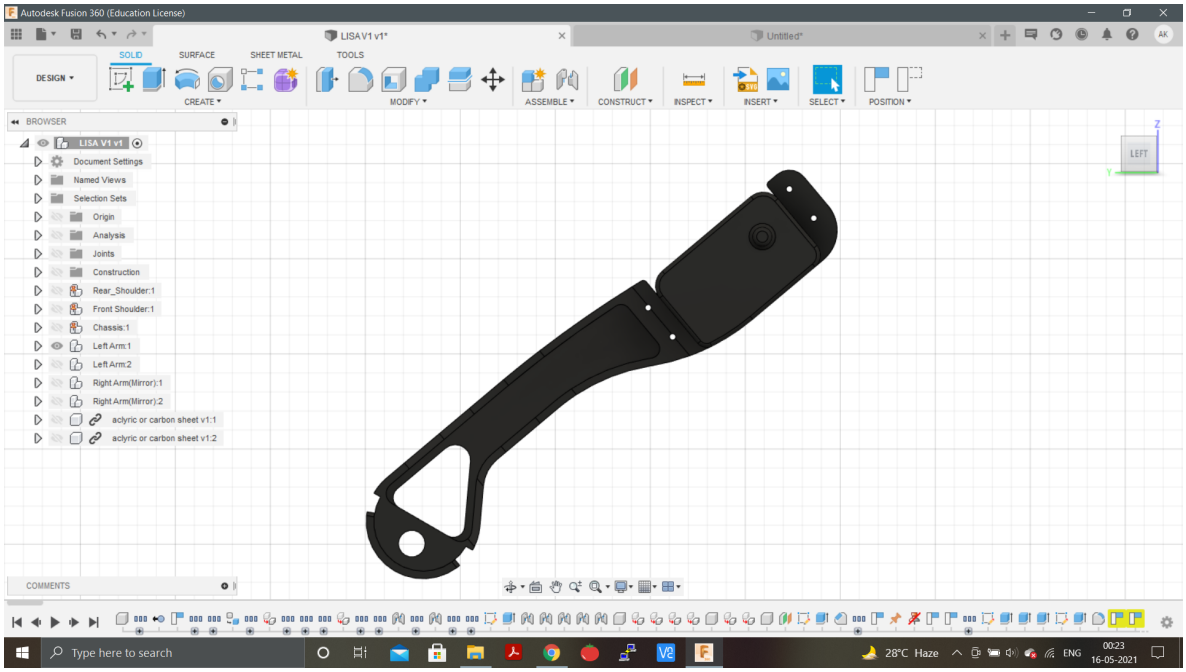
Take a offset plane and mirror the front shoulder to obtain back shoulder.

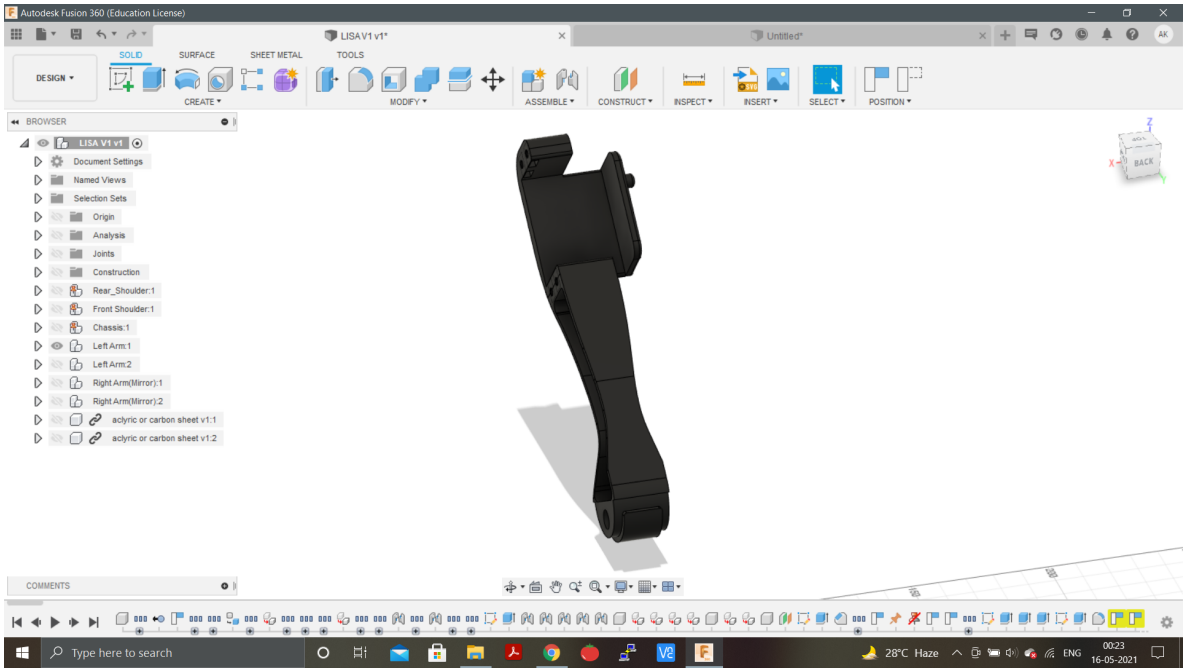


Back shoulder

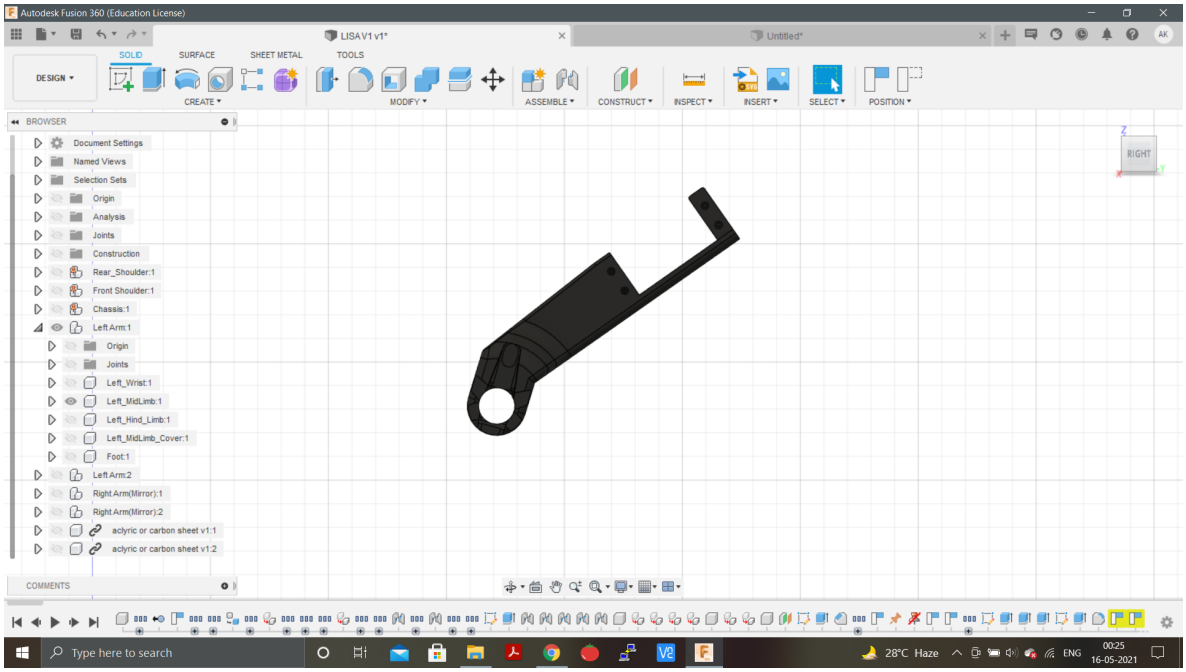


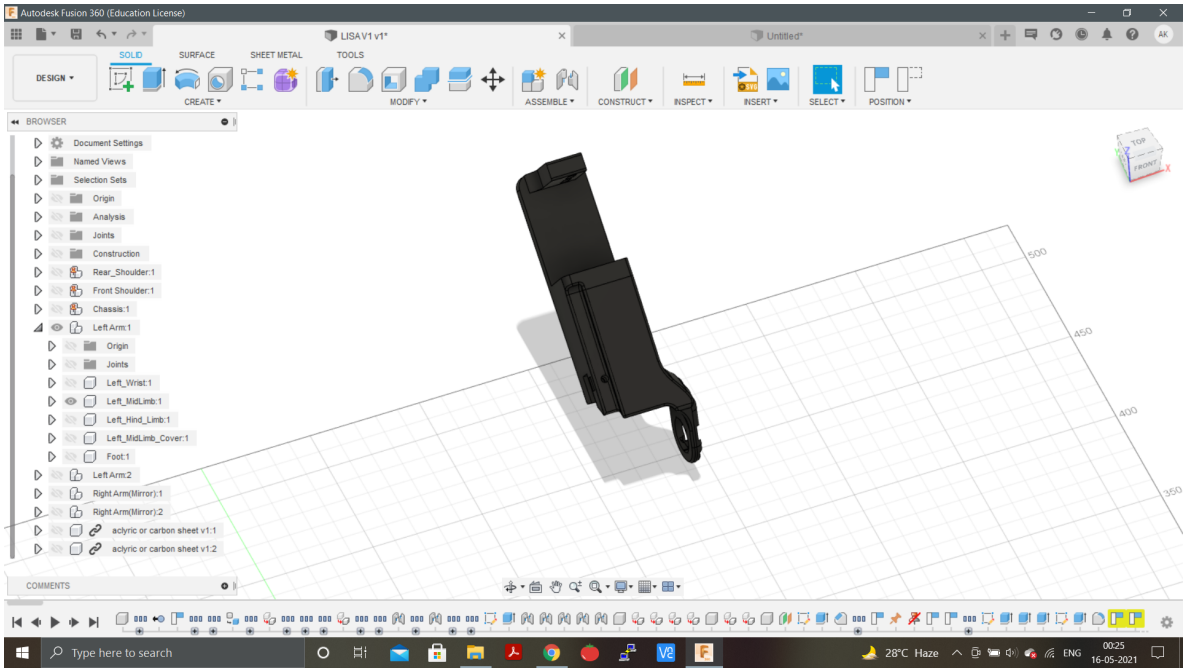
Left wrist


Autodesk Fusion 360 (Education License)
LISA V1 v1*
Untitled*
DESIGN
SOLID
SURFACE
SHEET METAL
TOOLS
CREATE
MODIFY
ASSEMBLE
CONSTRUCT
INSPECT
INSERT
SELECT
POSITION
BROWSER
LISA V1 v1
Document Settings
Named Views
Selection Sets
Origin
Analysis
Joints
Construction
Rear_Shoulder:1
Front Shoulder:1
Chassis:1
Left Arm:1
Left Arm:2
Right Arm(Mirror):1
Right Arm(Mirror):2
aclyric or carbon sheet v1:1
aclyric or carbon sheet v1:2
COMMENTS
LEFT
Type here to search
28°C Haze
ENG
00:23
16-05-2021

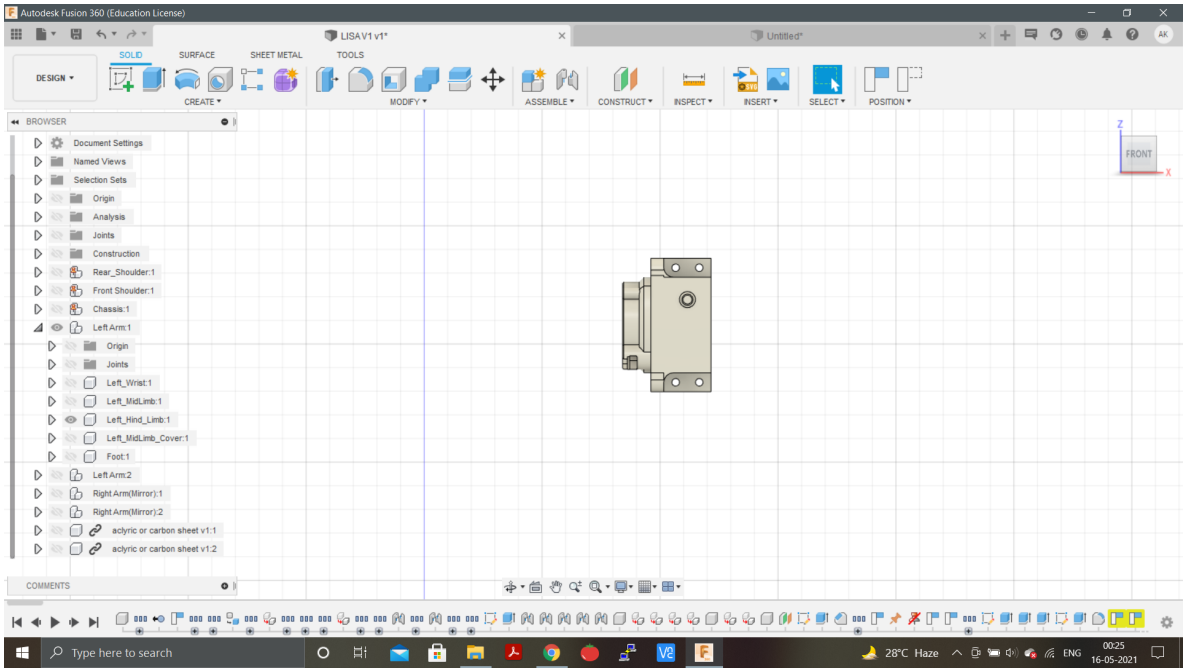

Autodesk Fusion 360 (Education License)
LISA V1 v1*
Untitled*
DESIGN
SOLID
SURFACE
SHEET METAL
TOOLS
CREATE
MODIFY
ASSEMBLE
CONSTRUCT
INSPECT
INSERT
SELECT
POSITION
BROWSER
LISA V1 v1
Document Settings
Named Views
Selection Sets
Origin
Analysis
Joints
Construction
Rear_Shoulder:1
Front Shoulder:1
Chassis:1
Left Arm:1
Left Arm:2
Right Arm(Mirror):1
Right Arm(Mirror):2
aclyric or carbon sheet v1:1
aclyric or carbon sheet v1:2
COMMENTS
BACK
Type here to search
28°C Haze
ENG
00:23
16-05-2021

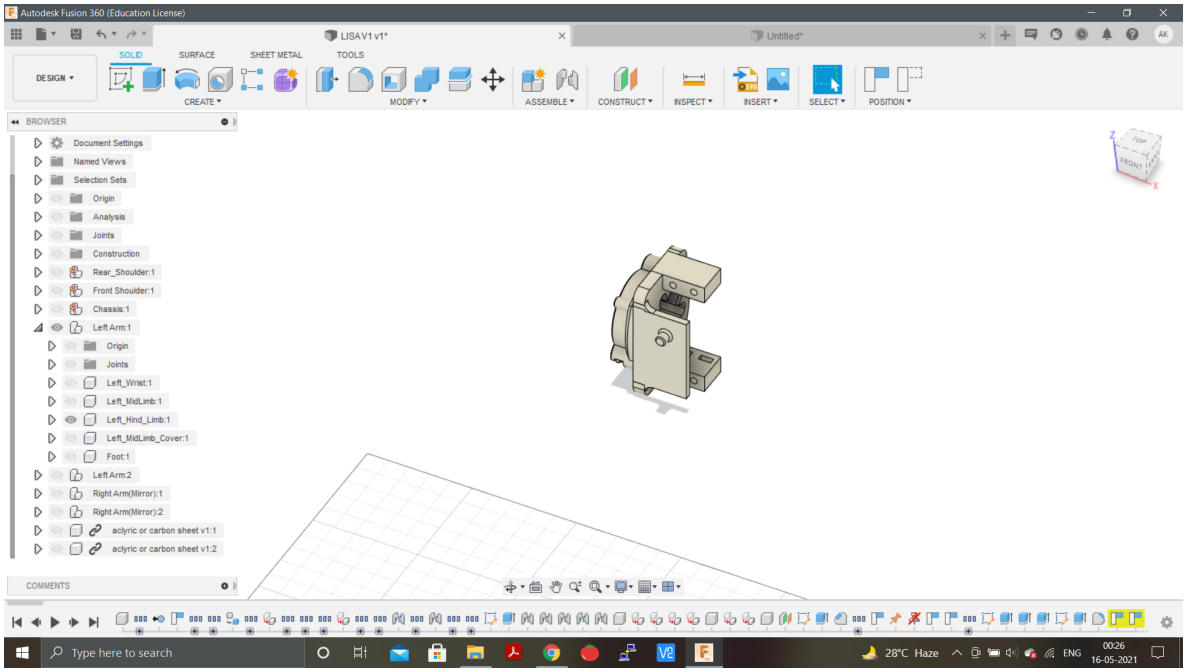
Left mid limb



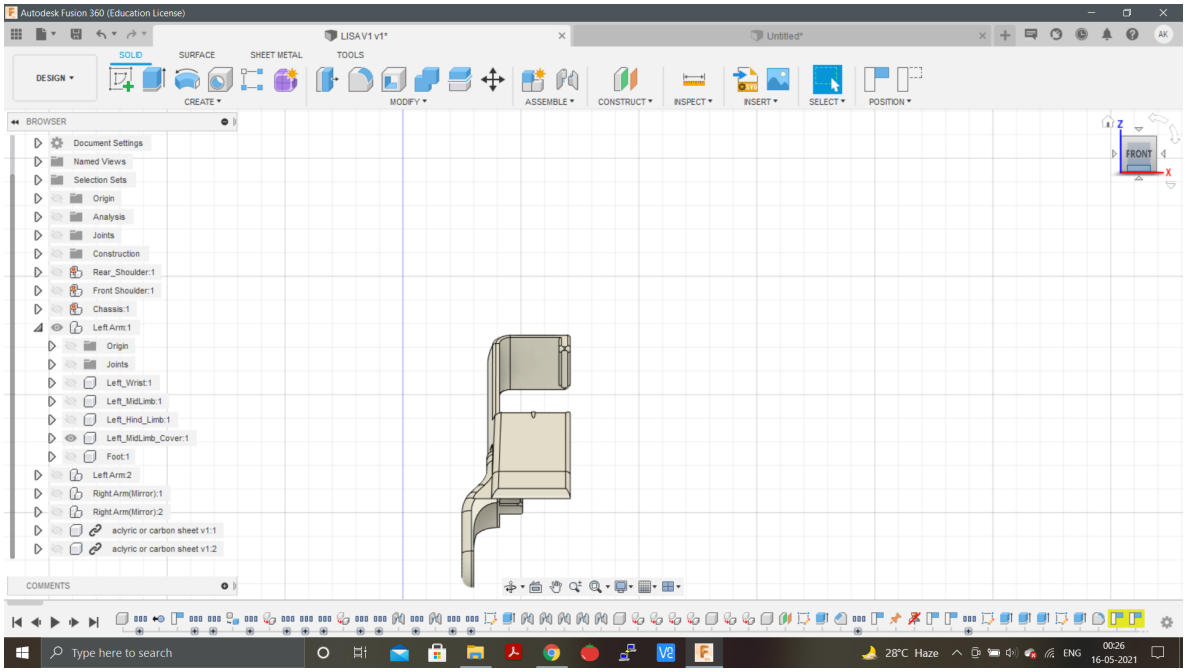


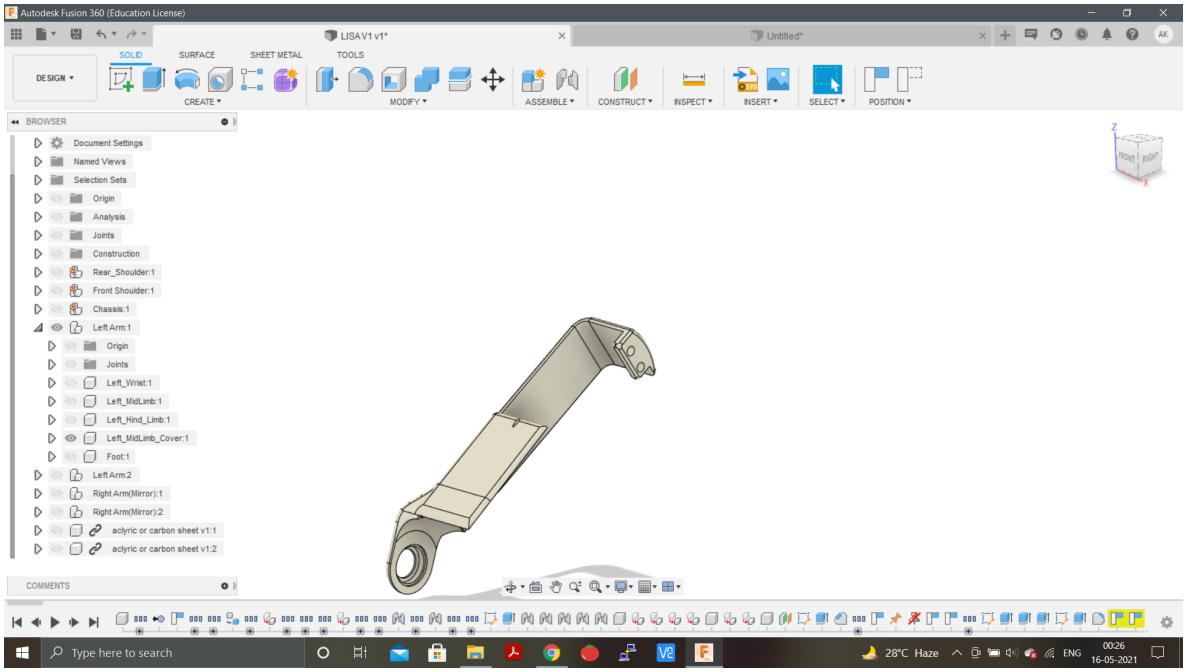
Left hind limb



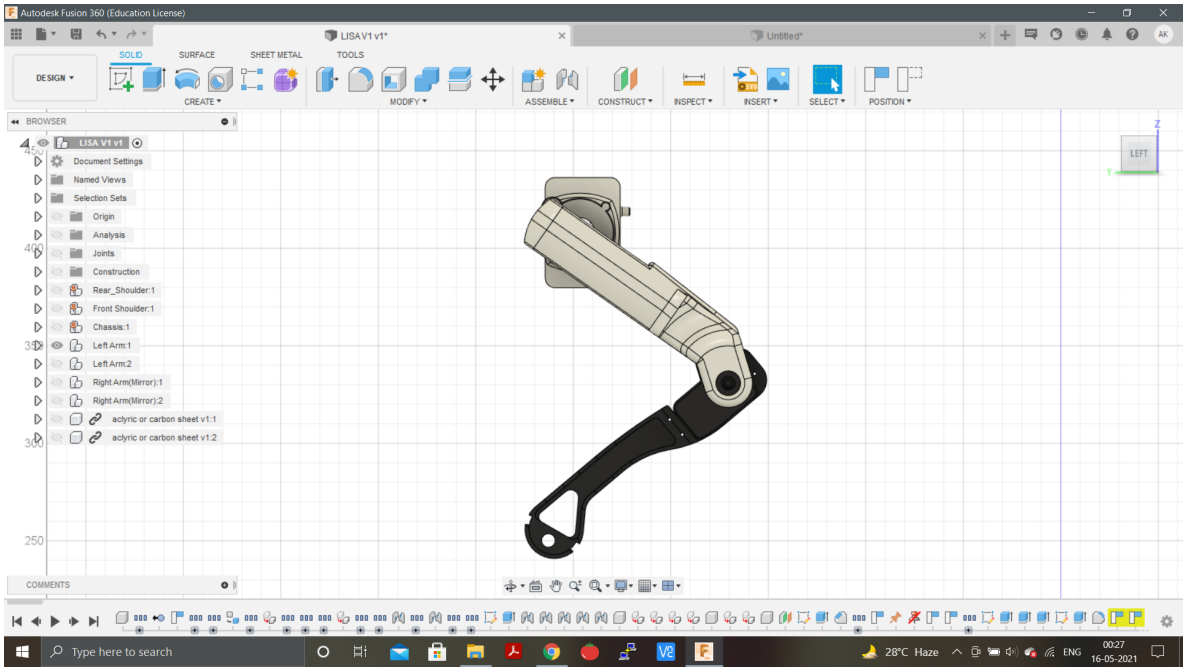


Left mid limb cover

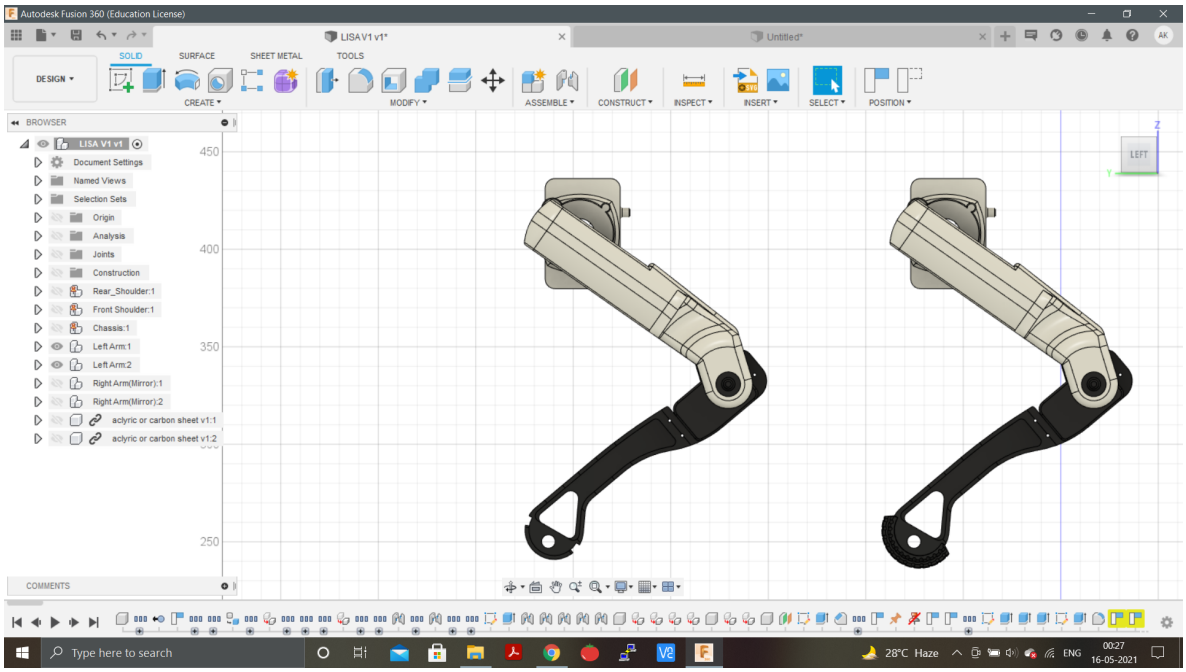




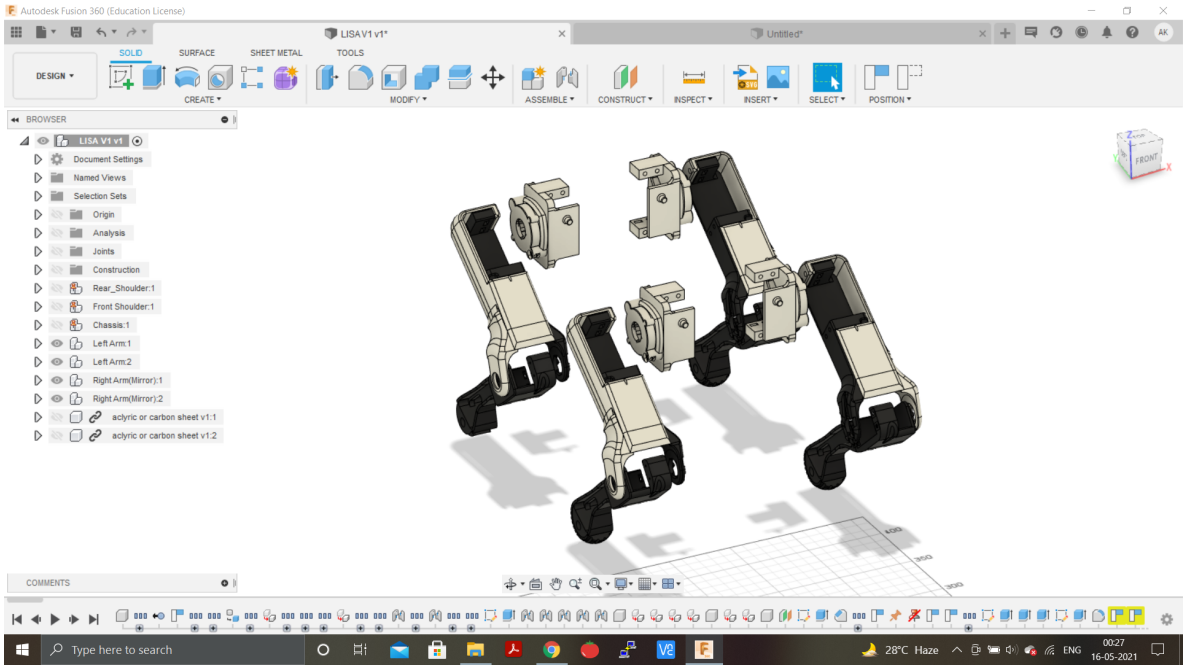
Left leg assembled

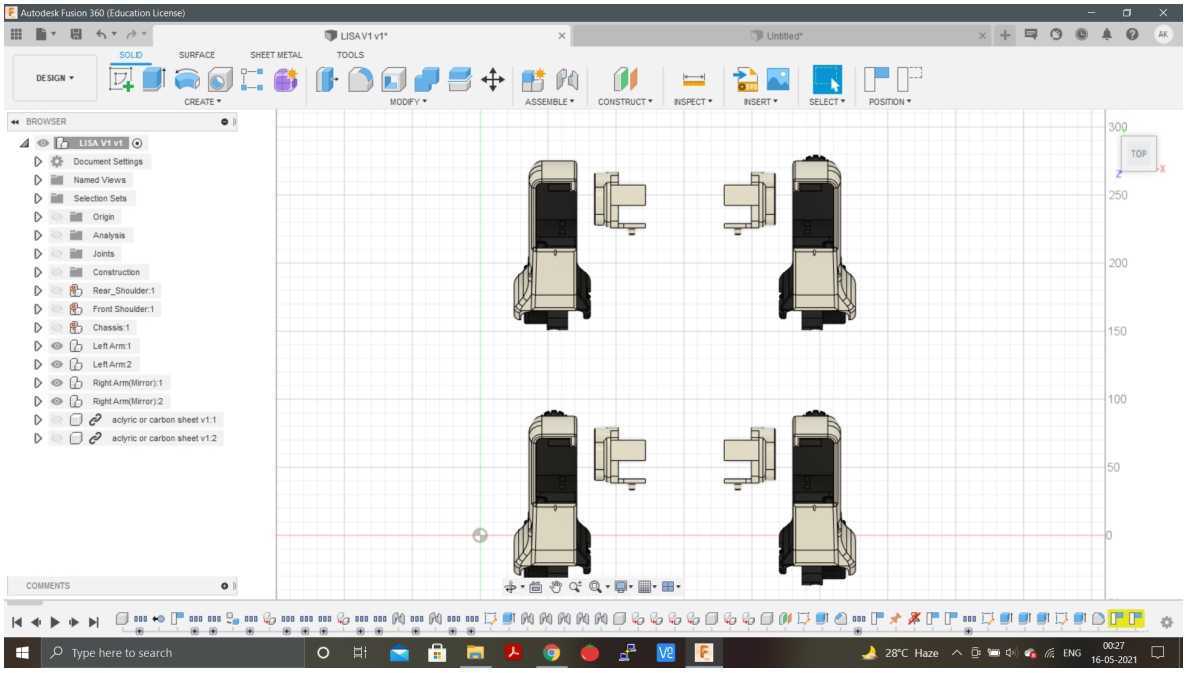


Copy the left leg.

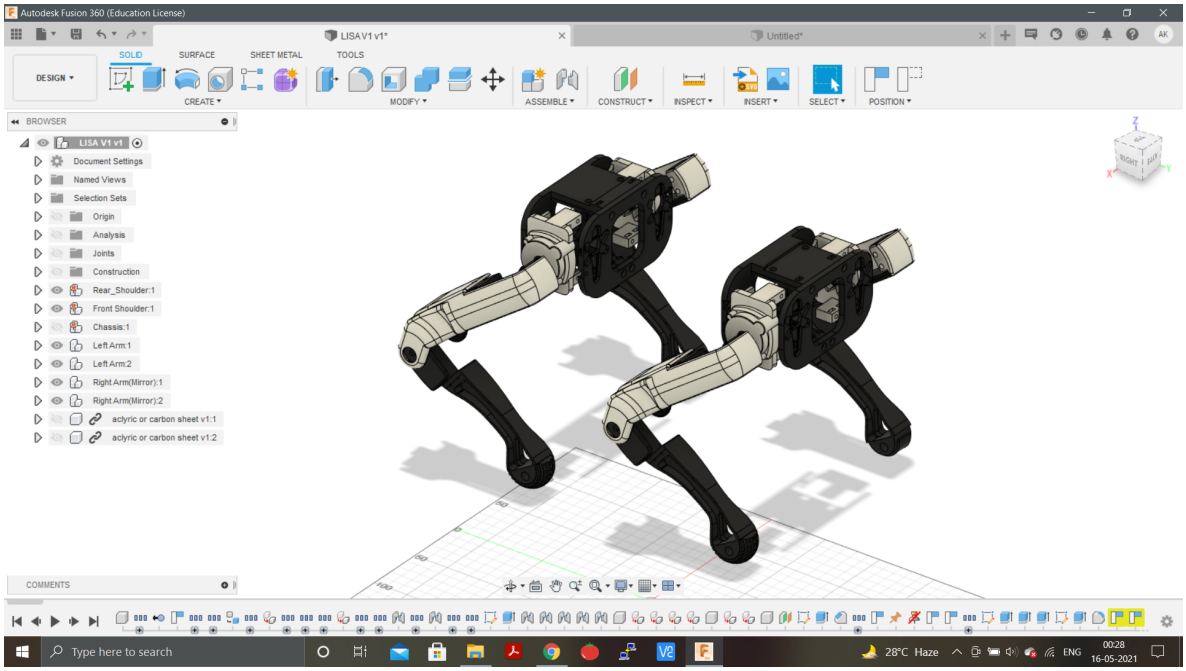


Mirrored legs

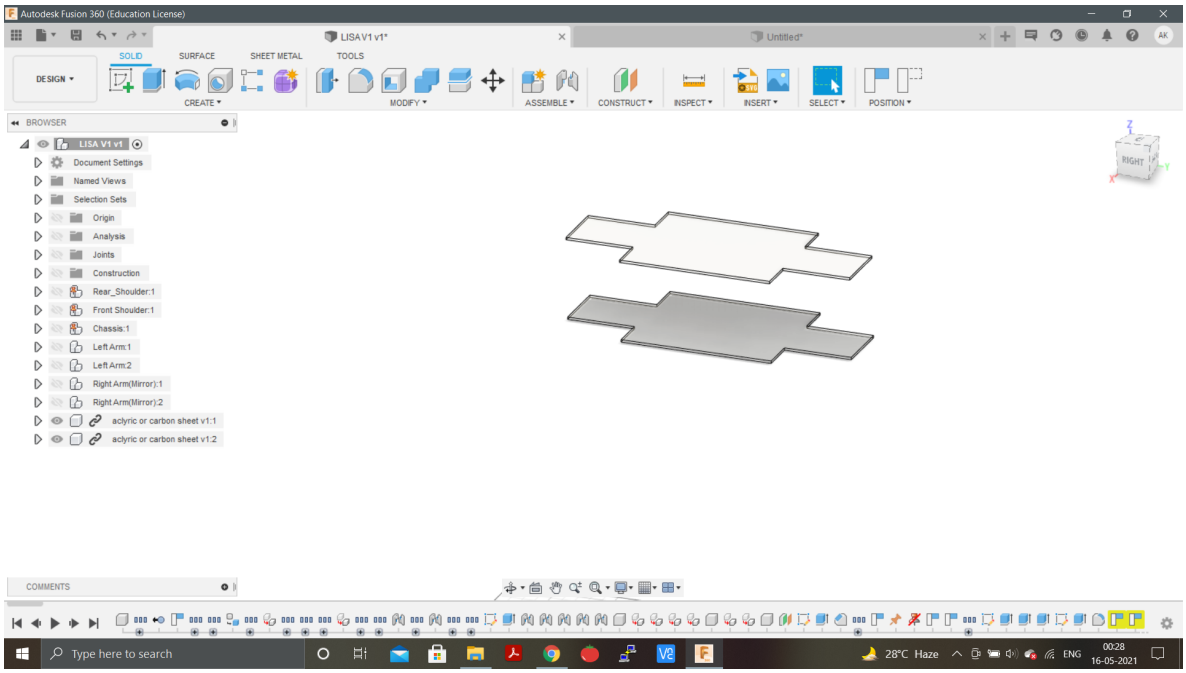




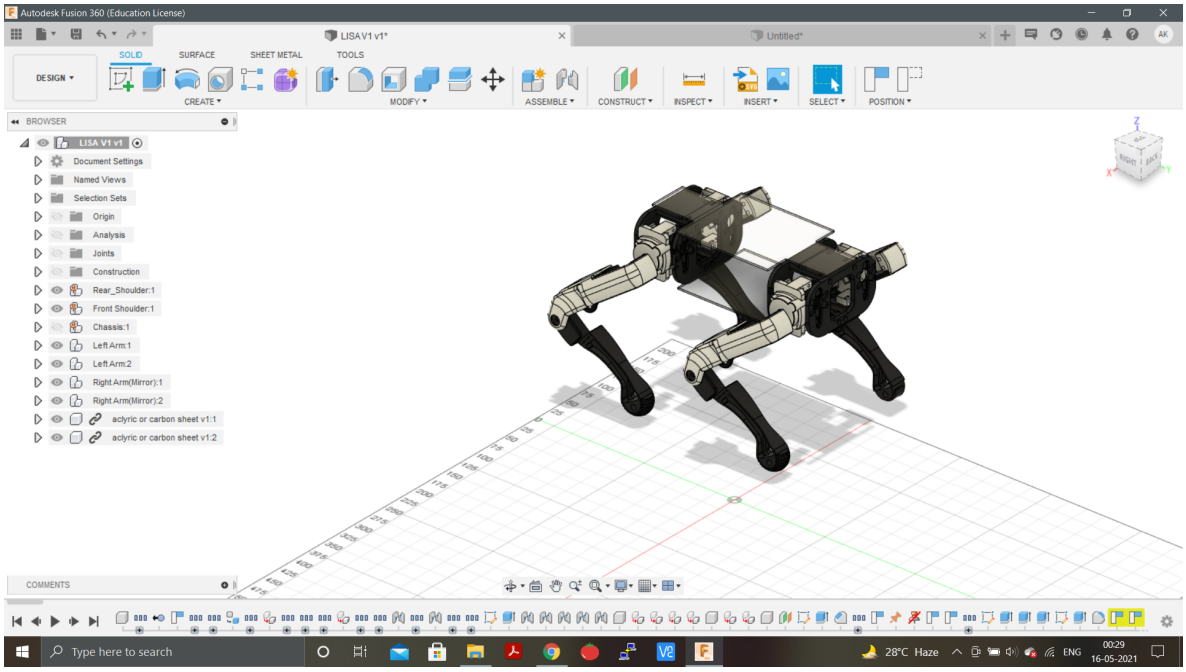
Legs assembled with shoulder parts

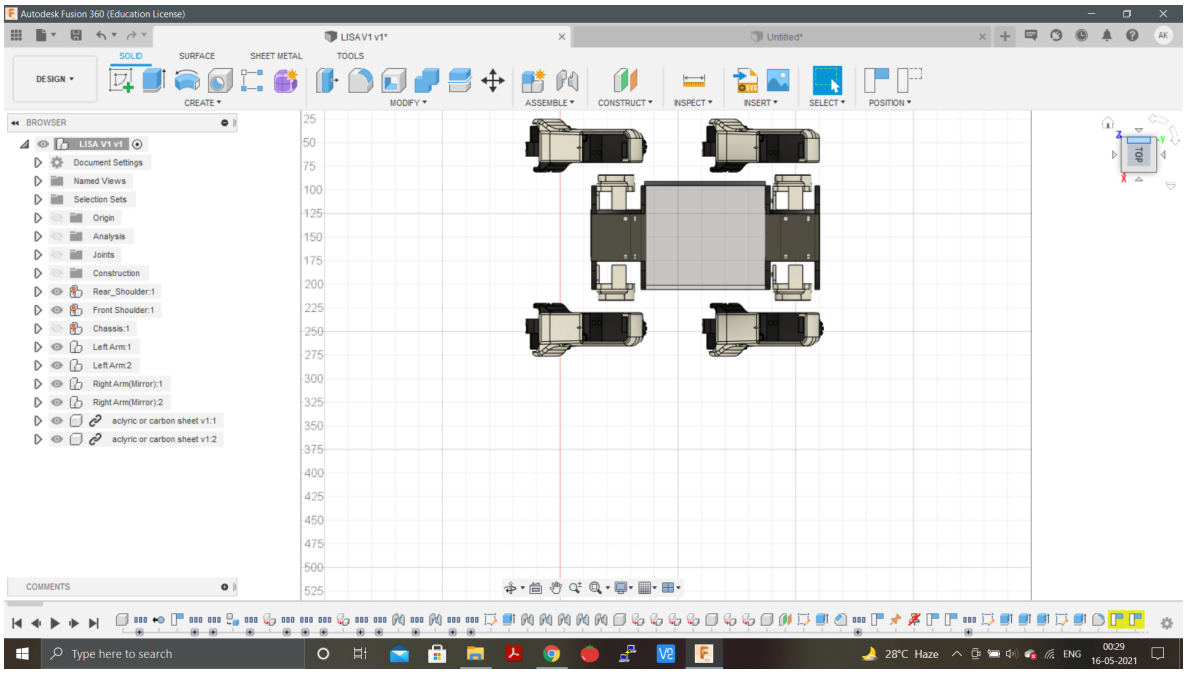


Acrylic sheets

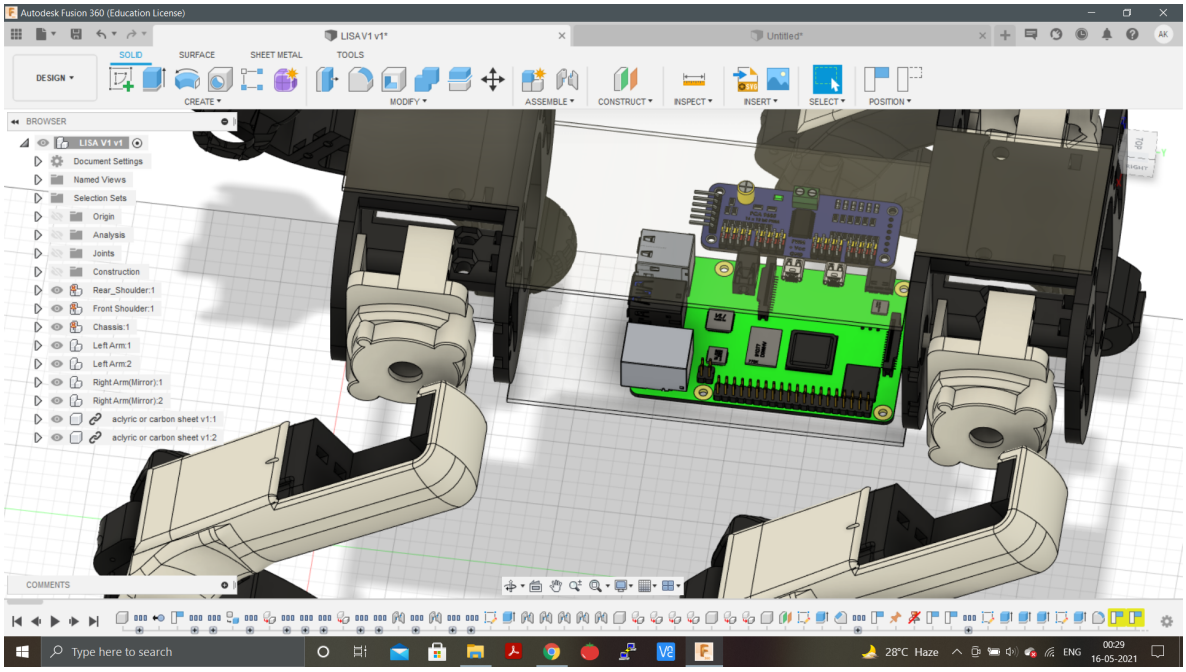


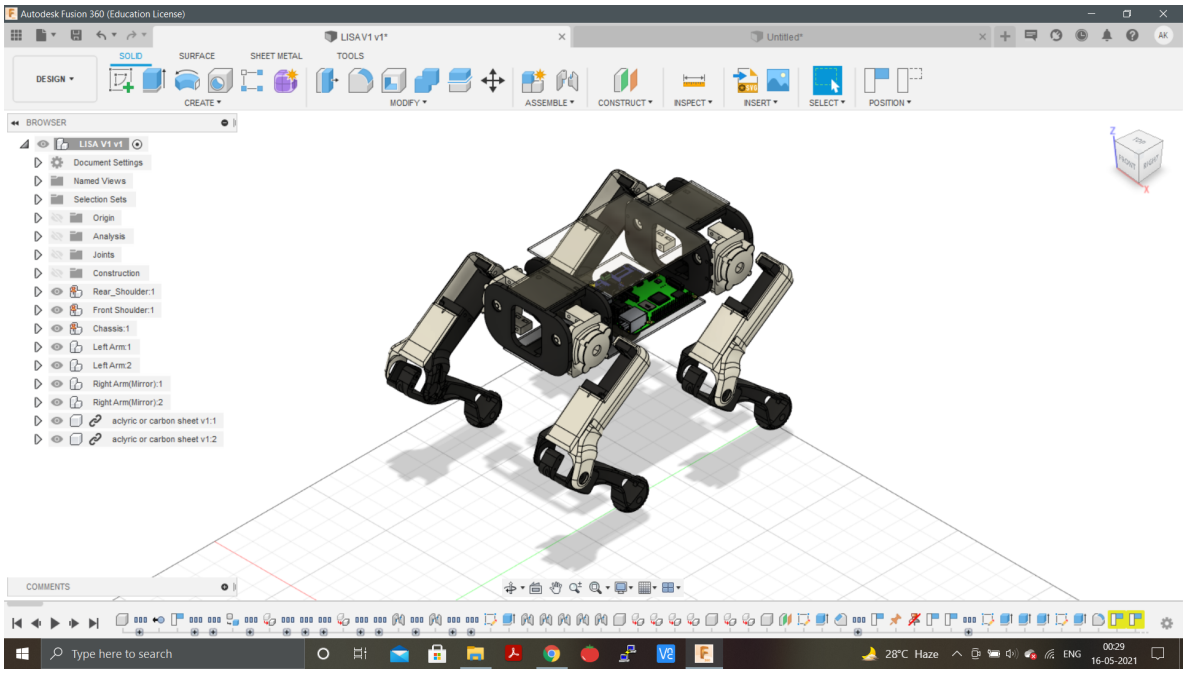
Assembled with acrylic sheet



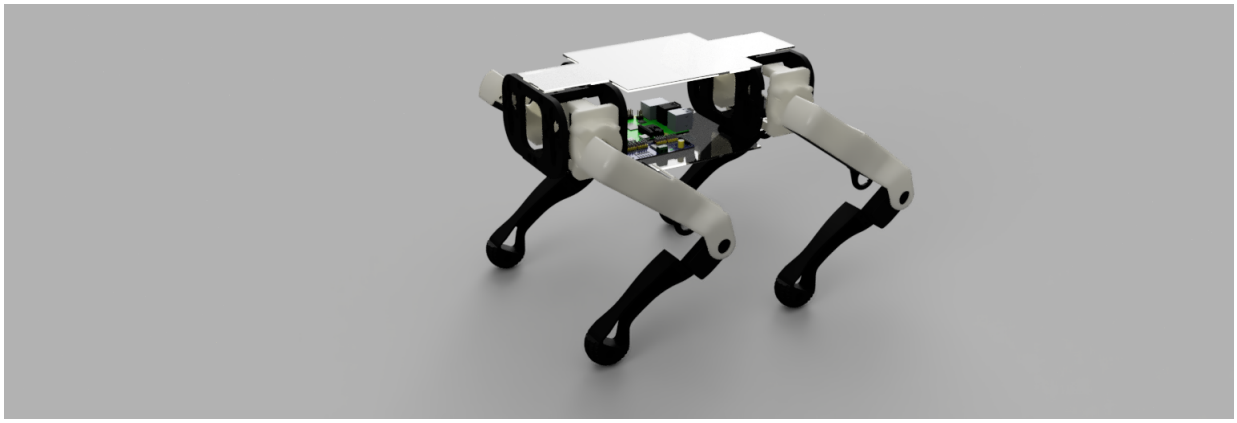


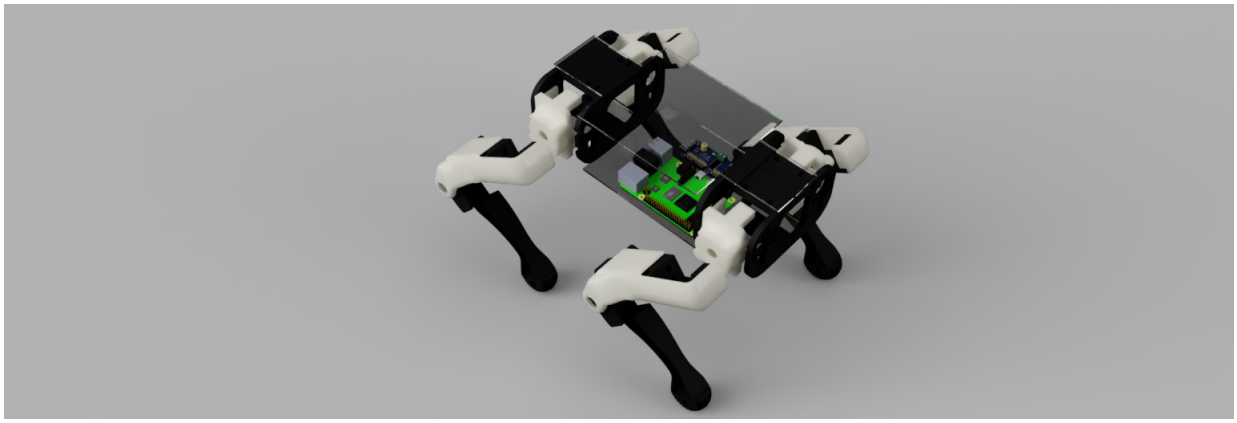
Importing RPi and PCA board for assembly





Lisa renders in fusion 360

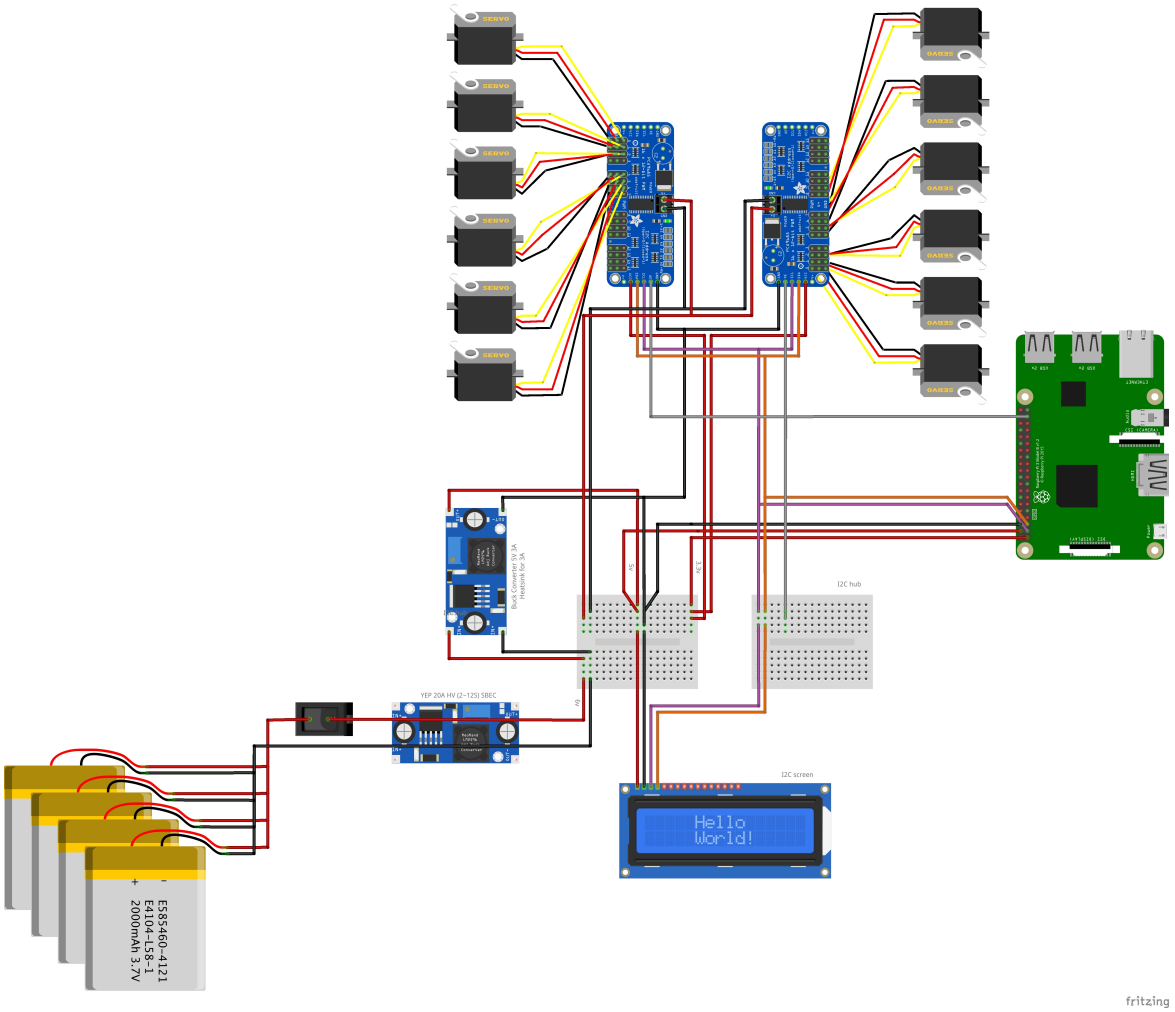




Chapter 8

CIRCUIT DIAGRAM

8.1 Circuit



Chapter 9

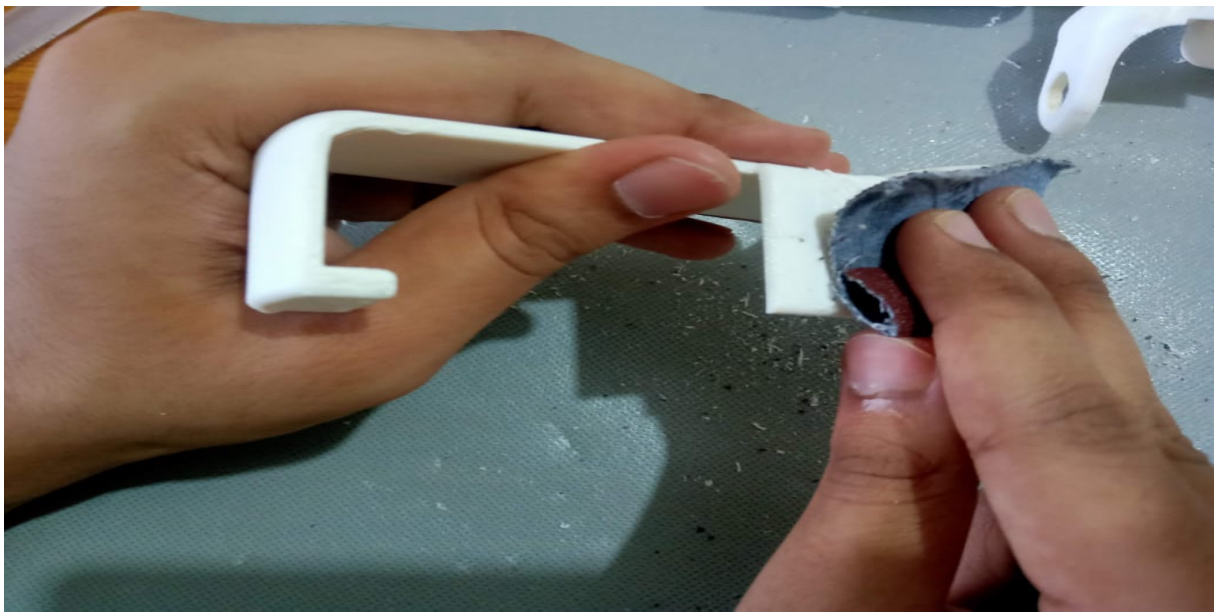
ASSEMBLY

9.1 Gathering all parts and tool:

Gather all the necessary parts and tools

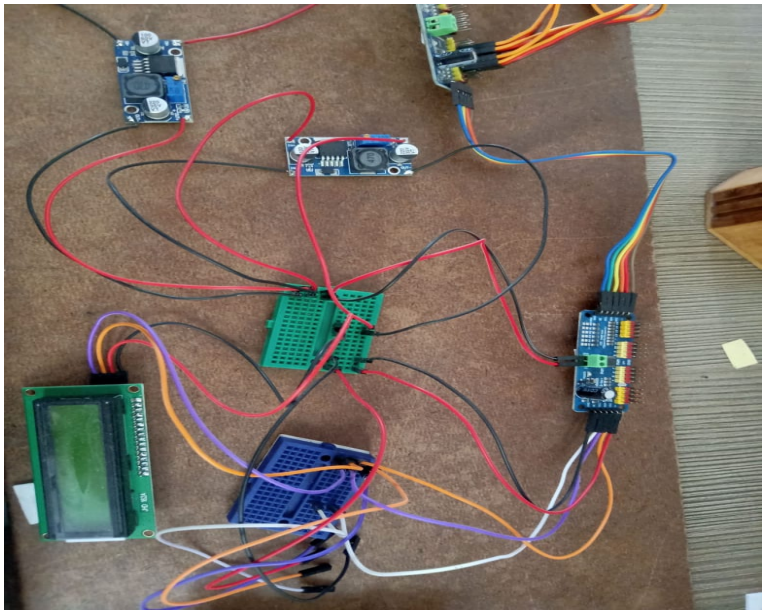
9.2 Sanding:

Using a 60 grit Sandpaper remove the supports from 3d printed parts.



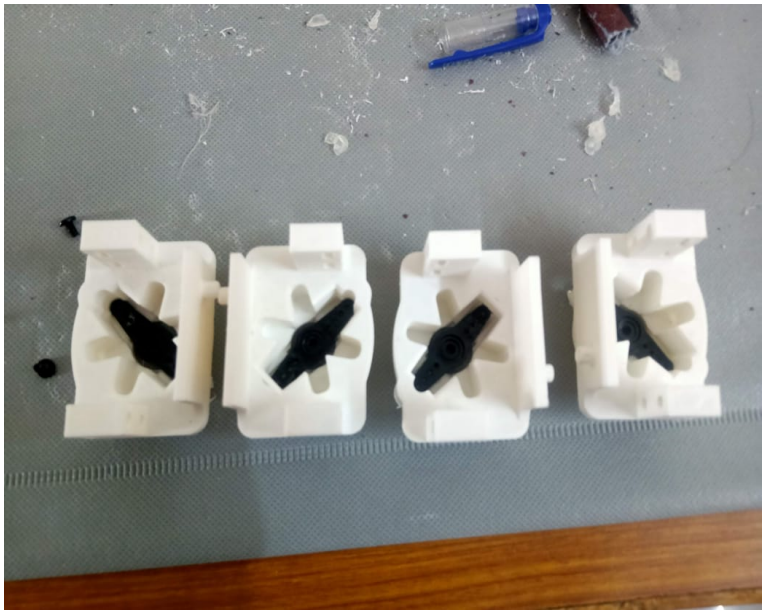
9.3 Testing electronics:

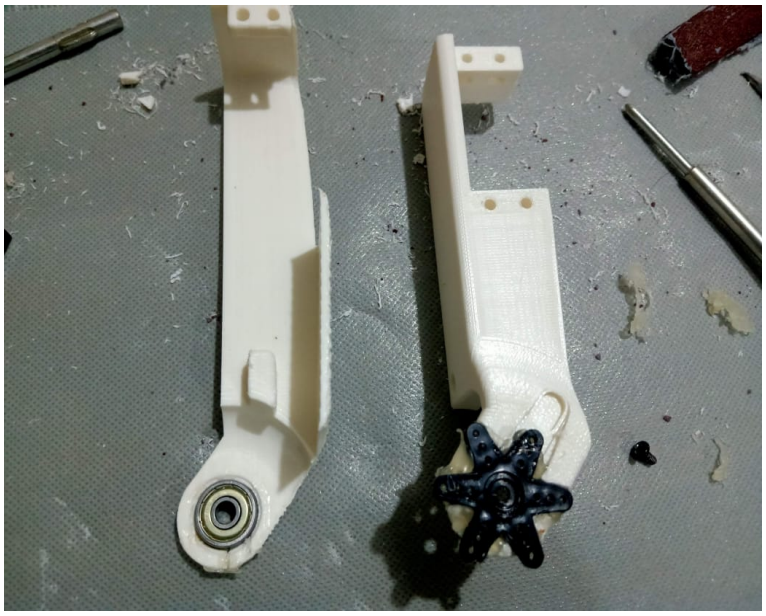
Before assembling the electronics and hardware parts together, connect the electronics part as per the circuit diagram and test each part.



9.4 Building Legs:

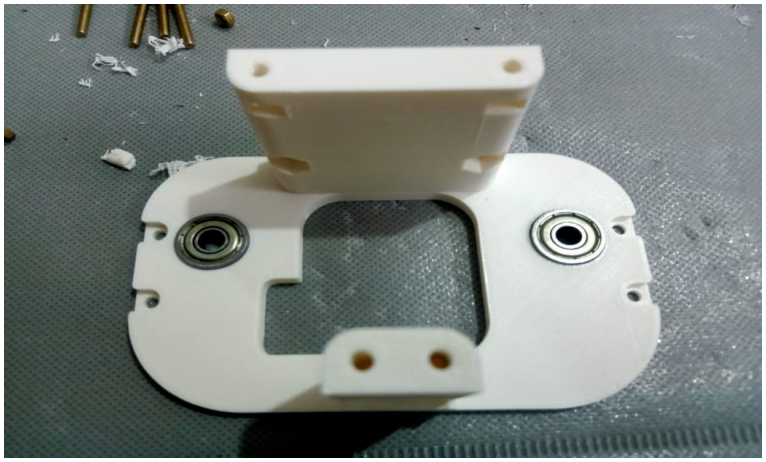
Using Glue gun, glue the servo horns and 626ZZ bearings in the respective 3d printed leg parts and let the glue harden for few minutes. After that screw Mg996R motors to the respective slots in the 3d printed parts using M3 screws.Repeat this process for 4 legs.





9.5 Assembling Hips:

Using Glue gun, glue the 626zz in 3d printed parts and attach the legs to hips. Repeat the same process for other hip.

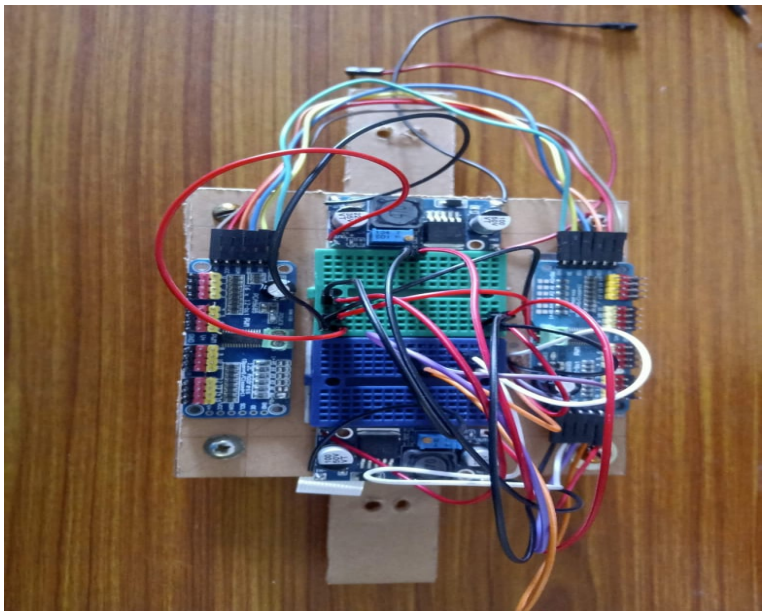


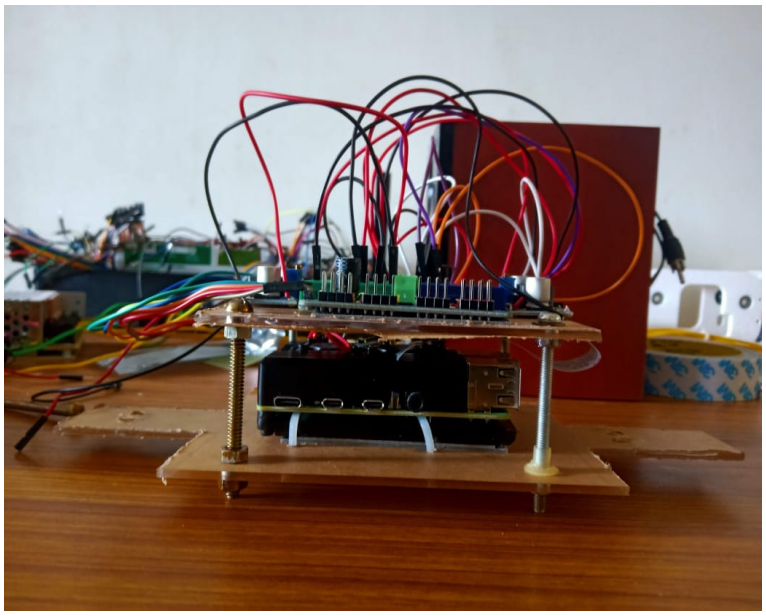
9.6 Cutting and Drilling the Acrylic Sheet:

Cut the acrylic sheet using Hand Saw as per the dimensions mentioned in the 3d modelling section and drill 4.5mm holes using impact drill machine and sand the edges.

9.7 Assembling the electronics:

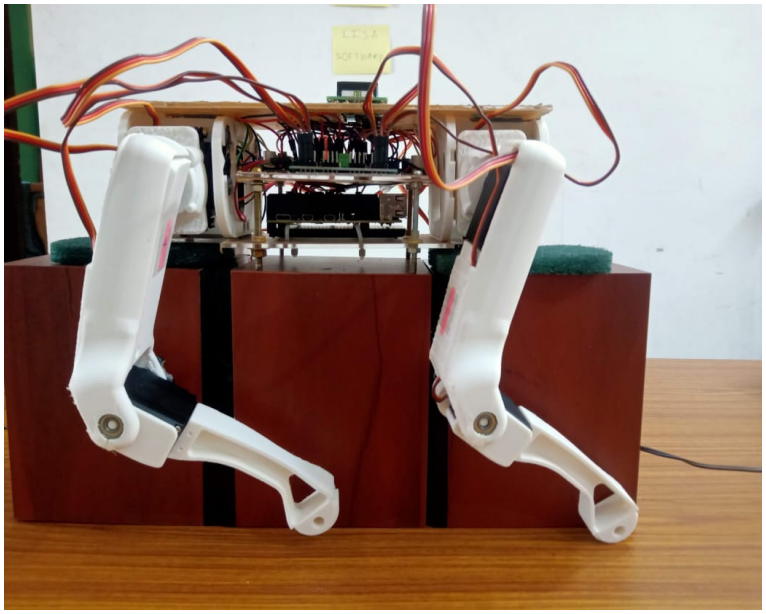
Assemble all the electronics on the acrylic sheet.





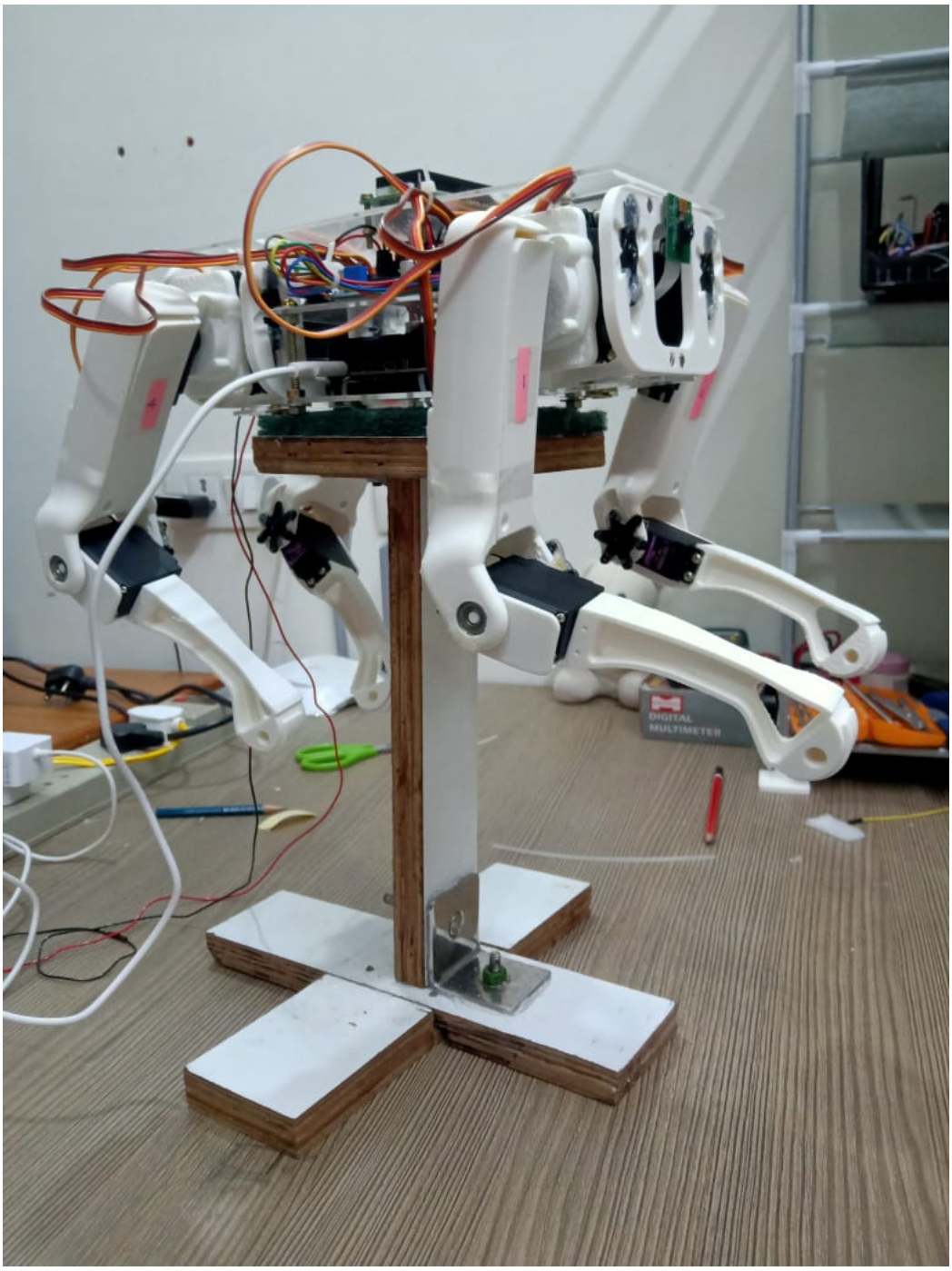
9.8 Assembling hip and electronic assembly:

Assemble the hip section and electronic section using a top acrylic sheet.



9.9 Fully assembled LISA:

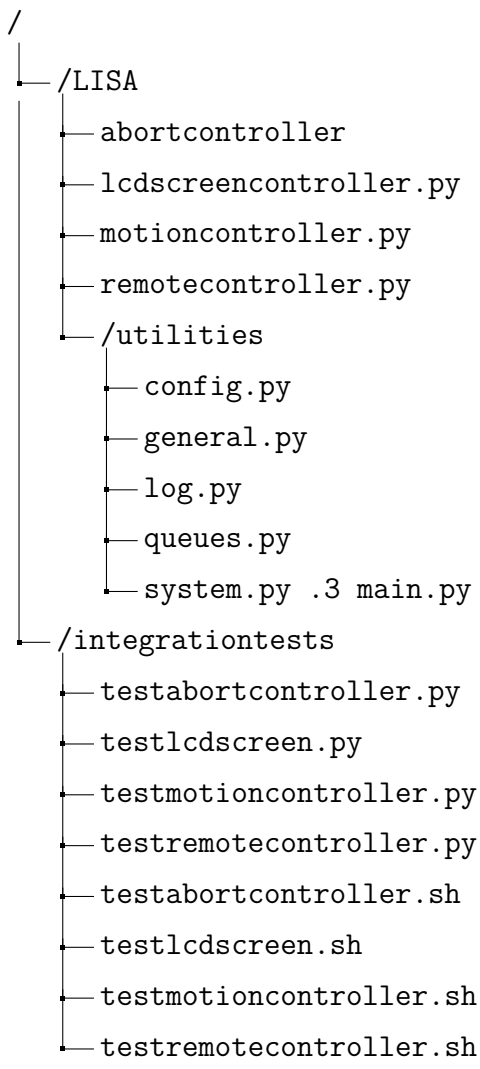
LISA on a training stand.

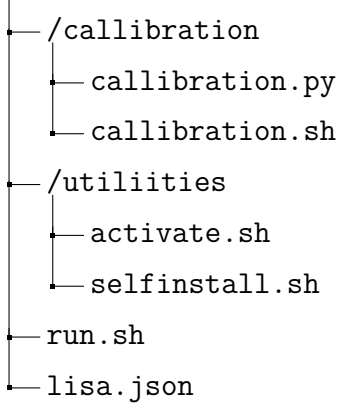

DIGITAL
MULTIMETER

Chapter 10

IMPLEMENTATION(CODE SNIPPETS)

10.1 Entire directory structure:




/callibration
callibration.py
callibration.sh
/utiliities
activate.sh
selfinstall.sh
run.sh
lisa.json

10.2 Lisa.json

The following file maps hardware components used in making the robot to available i2c addresses.

```
{
        "abort_controller": [{
                "gpio_port": 17
        }],
        "lcd_screen_controller": [{
                "lcd_screen": [{
                        "address": "0x27"
                }]
        }],
        "remote_controller_controller": [{
                "remote_controller": [{
                        "device": "js0"
                }]
        }],
        "motion_controller": [{
                "boards": [{
                        "pca9685_1": [{
                                "address": "0x40",
                                "reference_clock_speed": 25000000,
                                "frequency": 50
                        }],
                        "pca9685_2": [{
                                "address": "0x42",
                                "reference_clock_speed": 25000000,
                                "frequency": 50
                        }]
                }],
                "servos": [{
                        "rear_shoulder_left": [{
                                "pca9685": 1,
                                "channel": 0,
                                "min_pulse": 500,
                                "max_pulse": 2500,
                                "rest_angle": 90
                        }],
                        "rear_leg_left": [{
                                "pca9685": 1,
                                "channel": 1,
                                "min_pulse": 500,
                                "max_pulse": 2500,
                                "rest_angle": 90
                        }],
                        "rear_feet_left": [{
                                "pca9685": 1,
                                "channel": 2,
                                "min_pulse": 500,
                                "max_pulse": 2500,
                                "rest_angle": 90
                        }],
                        "rear_shoulder_right": [{
                                "pca9685": 1,
```

```
        "channel": 3,
        "min_pulse": 500,
        "max_pulse": 2500,
        "rest_angle": 90
}],
"rear_leg_right": [{
        "pca9685": 1,
        "channel": 4,
        "min_pulse": 500,
        "max_pulse": 2500,
        "rest_angle": 90
}],
"rear_feet_right": [{
        "pca9685": 1,
        "channel": 5,
        "min_pulse": 500,
        "max_pulse": 2500,
        "rest_angle": 90
}],
"front_shoulder_left": [{
        "pca9685": 2,
        "channel": 0,
        "min_pulse": 500,
        "max_pulse": 2500,
        "rest_angle": 90
}],
"front_leg_left": [{
        "pca9685": 2,
        "channel": 1,
        "min_pulse": 500,
        "max_pulse": 2500,
        "rest_angle": 90
}],
"front_feet_left": [{
        "pca9685": 2,
        "channel": 2,
        "min_pulse": 500,
        "max_pulse": 2500,
        "rest_angle": 90
}],
"front_shoulder_right": [{
        "pca9685": 2,
        "channel": 3,
        "min_pulse": 500,
        "max_pulse": 2500,
        "rest_angle": 90
}],
"front_leg_right": [{
        "pca9685": 2,
        "channel": 4,
        "min_pulse": 500,
        "max_pulse": 2500,
        "rest_angle": 90
}],
"front_feet_right": [{
        "pca9685": 2,
        "channel": 5,
```

```
                        "min_pulse": 500,
                        "max_pulse": 2500,
                        "rest_angle": 90
                    }],
                    "arm_rotation": [{
                        "pca9685": 1,
                            "channel": 6,
                            "min_pulse": 500,
                            "max_pulse": 2500,
                            "rest_angle": 90
                    }],
                    "arm_lift": [{
                        "pca9685": 1,
                            "channel": 7,
                            "min_pulse": 500,
                            "max_pulse": 2500,
                            "rest_angle": 90
                    }],
                    "arm_range": [{
                        "pca9685": 1,
                            "channel": 8,
                            "min_pulse": 500,
                            "max_pulse": 2500,
                            "rest_angle": 90
                    }],
                    "arm_cam_tilt": [{
                        "pca9685": 1,
                            "channel": 9,
                            "min_pulse": 500,
                            "max_pulse": 2500,
                            "rest_angle": 90
                    }]
                }]
        }]
}
```

10.3 AbortController.py

The following file is used to cut power to the servo board. To protect robot from accidental damages.

```
import signal
import RPi.GPIO as GPIO
import sys
from spotmicroai.utilities.log import Logger
from spotmicroai.utilities.config import Config
import spotmicroai.utilities.queues as queues

log = Logger().setup_logger('Abort controller')


class AbortController:
    gpio_port = None

    def __init__(self, communication_queues):

        try:

            log.debug('Starting controller...')

            signal.signal(signal.SIGINT, self.exit_gracefully)
            signal.signal(signal.SIGTERM, self.exit_gracefully)

            self.gpio_port =
                Config().get(Config.ABORT_CONTROLLER_GPIO_PORT)

            GPIO.setmode(GPIO.BCM)
            GPIO.setup(self.gpio_port, GPIO.OUT)

            self._abort_queue =
                communication_queues[queues.ABORT_CONTROLLER]
            self._lcd_screen_queue =
                communication_queues[queues.LCD_SCREEN_CONTROLLER]

            self.abort()

            self._lcd_screen_queue.put(queues.LCD_SCREEN_SHOW_ABORT_CONTROLLER_OK_ON)

        except Exception as e:
            log.error('Abort controller initialization problem', e)
            self._lcd_screen_queue.put(queues.LCD_SCREEN_SHOW_ABORT_CONTROLLER_NOK)
            try:
                self.abort()
            finally:
                sys.exit(1)

    def exit_gracefully(self, signum, frame):
        try:
            self.abort()
        finally:
            log.info('Terminated')
            sys.exit(0)
```

```
def do_process_events_from_queue(self):

    try:
        while True:
            event = self._abort_queue.get()

            if event == queues.ABORT_CONTROLLER_ACTION_ACTIVATE:
                self.activate_servos()

            if event == queues.ABORT_CONTROLLER_ACTION_ABORT:
                self.abort()

    except Exception as e:
        log.error('''Unknown problem while processing the
            queue of the abort controller''', e)
        sys.exit(1)

def activate_servos(self):
    self._lcd_screen_queue.put(queues.LCD_SCREEN_SHOW_ABORT_CONTROLLER_OK_ON)
    GPIO.output(self.gpio_port, GPIO.LOW)

def abort(self):
    self._lcd_screen_queue.put(queues.LCD_SCREEN_SHOW_ABORT_CONTROLLER_OK_OFF)
    GPIO.output(self.gpio_port, GPIO.HIGH)
```

10.4 LcdScreenController.py

The following file depicts the status of the robot in the lcd screen.

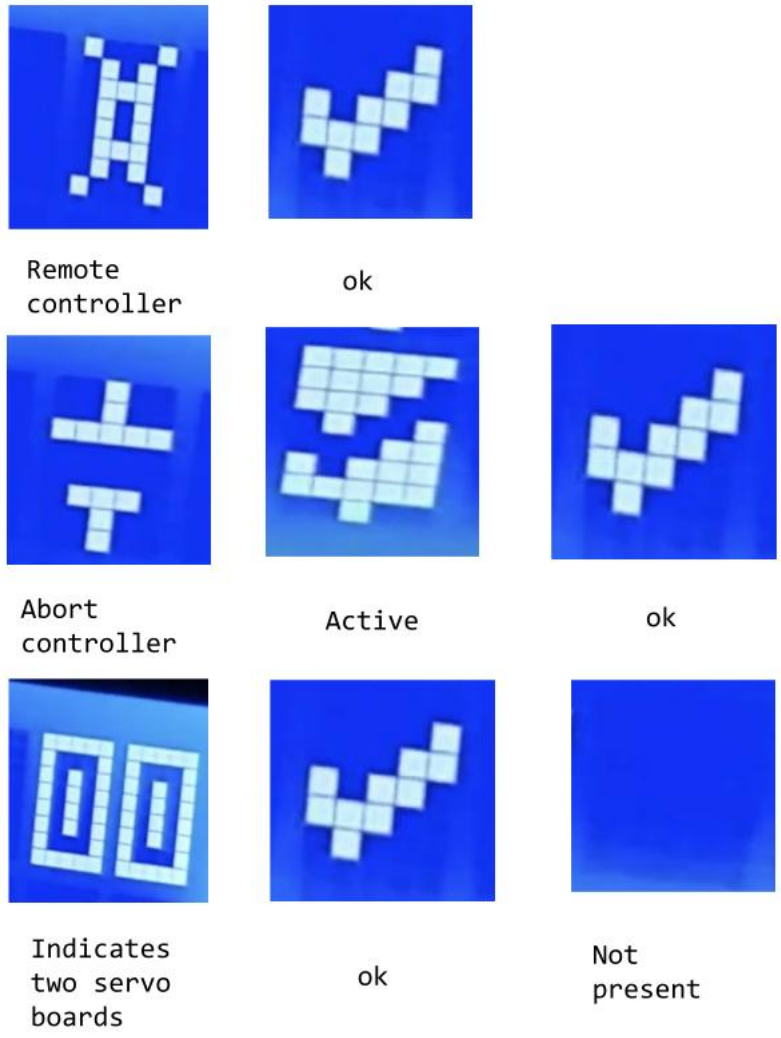


Figure 10.1: Status icons

```
import signal
import sys
import time
import queue

from spotmicroai.utilities.log import Logger
from spotmicroai.lcd_screen_controller import LCD_16x2_I2C_driver
from spotmicroai.utilities.config import Config
from spotmicroai.utilities.system import System

import spotmicroai.utilities.queues as queues

log = Logger().setup_logger('LCD Screen controller')


class LCDScreenController:
    is_alive = False

    lcd_screen_controller = None
    abort_controller = None
    remote_controller_controller = None
    motion_controller_1 = None
    motion_controller_2 = None

    def __init__(self, communication_queues):
        try:

            log.debug('Starting controller...')

            signal.signal(signal.SIGINT, self.exit_gracefully)
            signal.signal(signal.SIGTERM, self.exit_gracefully)

            i2c_address =
                int(Config().get(Config.LCD_SCREEN_CONTROLLER_I2C_ADDRESS), 0)

            self.screen = LCD_16x2_I2C_driver.lcd(address=i2c_address)

            self._lcd_screen_queue =
                communication_queues[queues.LCD_SCREEN_CONTROLLER]

            self.screen.lcd_clear()
            self.update_lcd_creen()
            self.turn_on()

            self.is_alive = True

        except Exception as e:
            self.is_alive = False
            log.error('''LCD Screen controller
                initialization problem, module
                not critical, skipping''', e)

    def exit_gracefully(self, signum, frame):
        try:
            self.turn_off()
        finally:
            log.info('Terminated')
```

```
            sys.exit(0)

    def do_process_events_from_queue(self):

        if not self.is_alive:
            log.error("SpotMicro␣is␣working␣without␣LCD␣Screen")
            return

        try:
            while True:

                try:
                    event =
                        self._lcd_screen_queue.get(block=True, timeout=1)

                    if event.startswith(queues.LCD_SCREEN_CONTROLLER + '␣'):
                        self.lcd_screen_controller =
                        event[len(queues.LCD_SCREEN_CONTROLLER) + 1:]

                    if event.startswith(queues.ABORT_CONTROLLER + ''):
                        self.abort_controller =
                        event[len(queues.ABORT_CONTROLLER) + 1:]

                    if event.startswith(queues.REMOTE_CONTROLLER_CONTROLLER + '␣'):
                        self.remote_controller_controller =
                        event[len(queues.REMOTE_CONTROLLER_CONTROLLER + '␣'):]

                    if event.startswith('motion_controller_1␣'):
                        self.motion_controller_1 =
                        event[len('motion_controller_1␣'):]

                    if event.startswith('motion_controller_2␣'):
                        self.motion_controller_2 =
                        event[len('motion_controller_2␣'):]

                except queue.Empty as e:
                    self.update_lcd_creen()
                    time.sleep(1)

        except Exception as e:
            log.error('''Unknown problem while processing the
                queue of the lcd screen controller''', e)

    def turn_off(self):
        self.screen.lcd_clear()
        time.sleep(0.1)
        self.screen.backlight(0)

    def turn_on(self):
        self.screen.backlight(1)

    def update_lcd_creen(self):  # https://www.quinapalus.com/hd44780udg.html

        if self.lcd_screen_controller == 'ON':
            self.turn_on()
        elif self.lcd_screen_controller == 'OFF':
            self.turn_off()
```

```
temperature = System().temperature()

custom_icons = []

icon_empty = [0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0]
icon_success = [0x0, 0x1, 0x3, 0x16, 0x1c, 0x8, 0x0, 0x0]
icon_pca9685 = [0x1f, 0x11, 0x15, 0x15, 0x15, 0x15, 0x11, 0x1f]
icon_gpio = [0x4, 0x4, 0x1f, 0x0, 0x0, 0xe, 0x4, 0x4]
icon_remote_controller = [0x11, 0xa, 0xe, 0xa, 0xa, 0xe, 0xa, 0x11]
icon_temperature = [0x18, 0x18, 0x3, 0x4, 0x4, 0x4, 0x3, 0x0]
icon_problem = [0x0, 0x1b, 0xe, 0x4, 0xe, 0x1b, 0x0, 0x0]
icon_success_reverse = [0x1f, 0x1e, 0x1c, 0x9, 0x3, 0x17, 0x1f]

# There is only memory for 7 in the lcd screen controller
custom_icons.insert(0, icon_empty)
custom_icons.insert(1, icon_success)
custom_icons.insert(2, icon_pca9685)
custom_icons.insert(3, icon_gpio)
custom_icons.insert(4, icon_remote_controller)
custom_icons.insert(5, icon_temperature)
custom_icons.insert(6, icon_problem)
custom_icons.insert(7, icon_success_reverse)

self.screen.lcd_load_custom_chars(custom_icons)

self.screen.lcd_write(0x80)  # First line

for char in 'SpotMicro':
    self.screen.lcd_write(ord(char), 0b00000001)

self.screen.lcd_write_char(0)
self.screen.lcd_write_char(4)
self.screen.lcd_write_char(0)
self.screen.lcd_write_char(3)
self.screen.lcd_write_char(0)
self.screen.lcd_write_char(2)
self.screen.lcd_write_char(2)

# Write next three chars to row 2 directly
self.screen.lcd_write(0xC0)  # Second line
self.screen.lcd_write_char(0)
self.screen.lcd_write_char(0)
self.screen.lcd_write_char(0)
self.screen.lcd_write_char(0)
self.screen.lcd_write_char(0)

if temperature:
    for char in temperature.rjust(3, '␣'):
        self.screen.lcd_write(ord(char), 0b00000001)
    self.screen.lcd_write_char(5)
else:
    self.screen.lcd_write_char(0)
    self.screen.lcd_write_char(0)
    self.screen.lcd_write_char(0)
    self.screen.lcd_write_char(0)
```

```
self.screen.lcd_write_char(0)

if self.remote_controller_controller == 'OK':
    self.screen.lcd_write_char(1)
elif self.remote_controller_controller == 'SEARCHING':
    self.screen.lcd_write_char(7)
else:
    self.screen.lcd_write_char(6)

self.screen.lcd_write_char(0)

if self.abort_controller == 'OK␣ON':
    self.screen.lcd_write_char(1)
elif self.abort_controller == 'OK␣OFF':
    self.screen.lcd_write_char(7)
else:
    self.screen.lcd_write_char(6)

self.screen.lcd_write_char(0)

if self.motion_controller_1 == 'OK':
    self.screen.lcd_write_char(1)
else:
    self.screen.lcd_write_char(6)

if self.motion_controller_2 == 'OK':
    self.screen.lcd_write_char(1)
else:
    self.screen.lcd_write_char(6)
```

10.5 Motioncontroller.py

The following script is the entire heart of the controlling part.Where joystick inputs are mapped to several functions Such as

Figure 10.2: Joy Stick

```
import signal
import sys

import queue
import busio
from board import SCL, SDA
from adafruit_pca9685 import PCA9685
from adafruit_motor import servo
import time

from spotmicroai.utilities.log import Logger
from spotmicroai.utilities.config import Config
import spotmicroai.utilities.queues as queues
from spotmicroai.utilities.general import General

log = Logger().setup_logger('Motion controller')


class MotionController:
    boards = 1

    is_activated = False

    i2c = None
    pca9685_1 = None
    pca9685_2 = None

    pca9685_1_address = None
    pca9685_1_reference_clock_speed = None
    pca9685_1_frequency = None
    pca9685_2_address = None
    pca9685_2_reference_clock_speed = None
    pca9685_2_frequency = None

    servo_rear_shoulder_left = None
    servo_rear_shoulder_left_pca9685 = None
    servo_rear_shoulder_left_channel = None
    servo_rear_shoulder_left_min_pulse = None
    servo_rear_shoulder_left_max_pulse = None
    servo_rear_shoulder_left_rest_angle = None

    servo_rear_leg_left = None
    servo_rear_leg_left_pca9685 = None
    servo_rear_leg_left_channel = None
    servo_rear_leg_left_min_pulse = None
    servo_rear_leg_left_max_pulse = None
    servo_rear_leg_left_rest_angle = None

    servo_rear_feet_left = None
    servo_rear_feet_left_pca9685 = None
    servo_rear_feet_left_channel = None
    servo_rear_feet_left_min_pulse = None
    servo_rear_feet_left_max_pulse = None
    servo_rear_feet_left_rest_angle = None

    servo_rear_shoulder_right = None
    servo_rear_shoulder_right_pca9685 = None
```

```
servo_rear_shoulder_right_channel = None
servo_rear_shoulder_right_min_pulse = None
servo_rear_shoulder_right_max_pulse = None
servo_rear_shoulder_right_rest_angle = None

servo_rear_leg_right = None
servo_rear_leg_right_pca9685 = None
servo_rear_leg_right_channel = None
servo_rear_leg_right_min_pulse = None
servo_rear_leg_right_max_pulse = None
servo_rear_leg_right_rest_angle = None

servo_rear_feet_right = None
servo_rear_feet_right_pca9685 = None
servo_rear_feet_right_channel = None
servo_rear_feet_right_min_pulse = None
servo_rear_feet_right_max_pulse = None
servo_rear_feet_right_rest_angle = None

servo_front_shoulder_left = None
servo_front_shoulder_left_pca9685 = None
servo_front_shoulder_left_channel = None
servo_front_shoulder_left_min_pulse = None
servo_front_shoulder_left_max_pulse = None
servo_front_shoulder_left_rest_angle = None

servo_front_leg_left = None
servo_front_leg_left_pca9685 = None
servo_front_leg_left_channel = None
servo_front_leg_left_min_pulse = None
servo_front_leg_left_max_pulse = None
servo_front_leg_left_rest_angle = None

servo_front_feet_left = None
servo_front_feet_left_pca9685 = None
servo_front_feet_left_channel = None
servo_front_feet_left_min_pulse = None
servo_front_feet_left_max_pulse = None
servo_front_feet_left_rest_angle = None

servo_front_shoulder_right = None
servo_front_shoulder_right_pca9685 = None
servo_front_shoulder_right_channel = None
servo_front_shoulder_right_min_pulse = None
servo_front_shoulder_right_max_pulse = None
servo_front_shoulder_right_rest_angle = None

servo_front_leg_right = None
servo_front_leg_right_pca9685 = None
servo_front_leg_right_channel = None
servo_front_leg_right_min_pulse = None
servo_front_leg_right_max_pulse = None
servo_front_leg_right_rest_angle = None

servo_front_feet_right = None
servo_front_feet_right_pca9685 = None
servo_front_feet_right_channel = None
```

```
servo_front_feet_right_min_pulse = None
servo_front_feet_right_max_pulse = None
servo_front_feet_right_rest_angle = None

servo_arm_rotation = None
servo_arm_rotation_pca9685 = None
servo_arm_rotation_channel = None
servo_arm_rotation_min_pulse = None
servo_arm_rotation_max_pulse = None
servo_arm_rotation_rest_angle = None

servo_arm_lift = None
servo_arm_lift_pca9685 = None
servo_arm_lift_channel = None
servo_arm_lift_min_pulse = None
servo_arm_lift_max_pulse = None
servo_arm_lift_rest_angle = None

servo_arm_range = None
servo_arm_range_pca9685 = None
servo_arm_range_channel = None
servo_arm_range_min_pulse = None
servo_arm_range_max_pulse = None
servo_arm_range_rest_angle = None

servo_arm_cam_tilt = None
servo_arm_cam_tilt_pca9685 = None
servo_arm_cam_tilt_channel = None
servo_arm_cam_tilt_min_pulse = None
servo_arm_cam_tilt_max_pulse = None
servo_arm_cam_tilt_rest_angle = None

def __init__(self, communication_queues):

    try:

        log.debug('Starting␣controller...')

        signal.signal(signal.SIGINT, self.exit_gracefully)
        signal.signal(signal.SIGTERM, self.exit_gracefully)

        self.i2c = busio.I2C(SCL, SDA)
        self.load_pca9685_boards_configuration()
        self.load_servos_configuration()

        self._abort_queue =
            communication_queues[queues.ABORT_CONTROLLER]
        self._motion_queue =
            communication_queues[queues.MOTION_CONTROLLER]
        self._lcd_screen_queue =
            communication_queues[queues.LCD_SCREEN_CONTROLLER]

        if self.pca9685_2_address:
            self._lcd_screen_queue.put('motion_controller_1␣OK')
            self._lcd_screen_queue.put('motion_controller_2␣OK')
        else:
            self._lcd_screen_queue.put('motion_controller_1␣OK')
```

```
                self._lcd_screen_queue.put('motion_controller_2 NOK')

            self._previous_event = {}

        except Exception as e:
            log.error('Motion controller initialization problem', e)
            self._lcd_screen_queue.put('motion_controller_1 NOK')
            self._lcd_screen_queue.put('motion_controller_2 NOK')
            try:
                self.pca9685_1.deinit()
            finally:
                try:
                    if self.boards == 2:
                        self.pca9685_2.deinit()
                finally:
                    sys.exit(1)

    def exit_gracefully(self, signum, frame):
        try:
            self.pca9685_1.deinit()
        finally:
            try:
                if self.boards == 2:
                    self.pca9685_2.deinit()
            finally:
                log.info('Terminated')
                sys.exit(0)

    def do_process_events_from_queues(self):

        while True:

            try:

                event = self._motion_queue.get(block=True, timeout=60)

                # log.debug(event)

                if event['start']:
                    if self.is_activated:
                        self.rest_position()
                        time.sleep(0.5)
                        self.deactivate_pca9685_boards()
                        self._abort_queue.
                        put(queues.ABORT_CONTROLLER_ACTION_ABORT)
                    else:
                        self._abort_queue.
                        put(queues.ABORT_CONTROLLER_ACTION_ACTIVATE)
                        self.activate_pca9685_boards()
                        self.activate_servos()
                        self.rest_position()

                if not self.is_activated:
                    log.info('Press START/OPTIONS to enable the servos')
                    continue

                if event['a']:
```

```
                self.rest_position()

            if event['hat0y']:
                self.body_move_body_up_and_down(event['hat0y'])

            if event['hat0x']:
                self.body_move_body_left_right(event['hat0x'])

            if event['ry']:
                self.body_move_body_up_and_down_analog(event['ry'])

            if event['rx']:
                self.body_move_body_left_right_analog(event['rx'])

            if event['hat0x'] and event['tl2']:
                # 2 buttons example
                pass

            if event['y']:
                self.standing_position()

            if event['b']:
                self.body_move_position_right()

            if event['x']:
                self.body_move_position_left()

            if event['tl']:
                self.arm_set_rotation(event['lx'])

            if event['tl']:
                self.arm_set_lift(event['ly'])

            if event['tr']:
                self.arm_set_range(event['ly'])

            if event['tr']:
                self.arm_set_cam_tilt(event['ry'])

            self.move()

        except queue.Empty as e:
            log.info('''Inactivity lasted 60 seconds,
                shutting down the servos, '
                     'press start to reactivate''')
            if self.is_activated:
                self.rest_position()
                time.sleep(0.5)
                self.deactivate_pca9685_boards()

        except Exception as e:
            log.error('''Unknown problem while processing the
             queue of the motion controller''')
            log.error(''' - Most likely a servo is not able
             to get to the assigned position''')

def load_pca9685_boards_configuration(self):
```

```
        self.pca9685_1_address =
        int(Config().get(Config.MOTION_CONTROLLER_BOARDS_PCA9685_1_ADDRESS), 0)
        self.pca9685_1_reference_clock_speed =
        int(Config().get(Config.MOTION_CONTROLLER_BOARDS_PCA9685_1_REFERENCE_CLOCK_SPEED))
        self.pca9685_1_frequency =
        int(Config().get(Config.MOTION_CONTROLLER_BOARDS_PCA9685_1_FREQUENCY))

        self.pca9685_2_address = False
        try:
            self.pca9685_2_address =
            int(Config().get(Config.MOTION_CONTROLLER_BOARDS_PCA9685_2_ADDRESS), 0)

            if self.pca9685_2_address:
                self.pca9685_2_reference_clock_speed =
                int(Config().get(Config.MOTION_CONTROLLER_BOARDS_PCA9685_2_REFERENCE_CLOCK_SPEED))
                self.pca9685_2_frequency =
                int(Config().get(Config.MOTION_CONTROLLER_BOARDS_PCA9685_2_FREQUENCY))

        except Exception as e:
            log.debug('''Only 1 PCA9685 is present in the configuration''')

    def activate_pca9685_boards(self):

        self.pca9685_1 =
        PCA9685(self.i2c, address=self.pca9685_1_address,
                    reference_clock_speed=self.pca9685_1_reference_clock_speed)
        self.pca9685_1.frequency = self.pca9685_1_frequency

        if self.pca9685_2_address:
            self.pca9685_2 =
            PCA9685(self.i2c, address=self.pca9685_2_address,
                    reference_clock_speed=self.pca9685_2_reference_clock_speed)
            self.pca9685_2.frequency = self.pca9685_2_frequency
            self.boards = 2

        self.is_activated = True
        log.debug(str(self.boards) + ' PCA9685 board(s) activated')

    def deactivate_pca9685_boards(self):

        try:
            if self.pca9685_1:
                self.pca9685_1.deinit()
        finally:
            try:
                if self.boards == 2 and self.pca9685_2:
                    self.pca9685_2.deinit()
            finally:
                # self._abort_queue.put(queues.ABORT_CONTROLLER_ACTION_ABORT)
                self.is_activated = False

        log.debug(str(self.boards) + ' PCA9685 board(s) deactivated')

    def load_servos_configuration(self):

        self.servo_rear_shoulder_left_pca9685 =
        Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_SHOULDER_LEFT_PCA9685)
```

```
self.servo_rear_shoulder_left_channel =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_SHOULDER_LEFT_CHANNEL)
self.servo_rear_shoulder_left_min_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_SHOULDER_LEFT_MIN_PULSE)
self.servo_rear_shoulder_left_max_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_SHOULDER_LEFT_MAX_PULSE)
self.servo_rear_shoulder_left_rest_angle =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_SHOULDER_LEFT_REST_ANGLE)

self.servo_rear_leg_left_pca9685 =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_LEG_LEFT_PCA9685)
self.servo_rear_leg_left_channel =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_LEG_LEFT_CHANNEL)
self.servo_rear_leg_left_min_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_LEG_LEFT_MIN_PULSE)
self.servo_rear_leg_left_max_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_LEG_LEFT_MAX_PULSE)
self.servo_rear_leg_left_rest_angle =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_LEG_LEFT_REST_ANGLE)

self.servo_rear_feet_left_pca9685 =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_FEET_LEFT_PCA9685)
self.servo_rear_feet_left_channel =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_FEET_LEFT_CHANNEL)
self.servo_rear_feet_left_min_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_FEET_LEFT_MIN_PULSE)
self.servo_rear_feet_left_max_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_FEET_LEFT_MAX_PULSE)
self.servo_rear_feet_left_rest_angle =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_FEET_LEFT_REST_ANGLE)

self.servo_rear_shoulder_right_pca9685 =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_SHOULDER_RIGHT_PCA9685)
self.servo_rear_shoulder_right_channel =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_SHOULDER_RIGHT_CHANNEL)
self.servo_rear_shoulder_right_min_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_SHOULDER_RIGHT_MIN_PULSE)
self.servo_rear_shoulder_right_max_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_SHOULDER_RIGHT_MAX_PULSE)
self.servo_rear_shoulder_right_rest_angle =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_SHOULDER_RIGHT_REST_ANGLE)

self.servo_rear_leg_right_pca9685 =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_LEG_RIGHT_PCA9685)
self.servo_rear_leg_right_channel =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_LEG_RIGHT_CHANNEL)
self.servo_rear_leg_right_min_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_LEG_RIGHT_MIN_PULSE)
self.servo_rear_leg_right_max_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_LEG_RIGHT_MAX_PULSE)
self.servo_rear_leg_right_rest_angle =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_LEG_RIGHT_REST_ANGLE)

self.servo_rear_feet_right_pca9685 =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_FEET_RIGHT_PCA9685)
self.servo_rear_feet_right_channel =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_FEET_RIGHT_CHANNEL)
```

```
self.servo_rear_feet_right_min_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_FEET_RIGHT_MIN_PULSE)
self.servo_rear_feet_right_max_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_FEET_RIGHT_MAX_PULSE)
self.servo_rear_feet_right_rest_angle =
Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_FEET_RIGHT_REST_ANGLE)

self.servo_front_shoulder_left_pca9685 =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_SHOULDER_LEFT_PCA9685)
self.servo_front_shoulder_left_channel =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_SHOULDER_LEFT_CHANNEL)
self.servo_front_shoulder_left_min_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_SHOULDER_LEFT_MIN_PULSE)
self.servo_front_shoulder_left_max_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_SHOULDER_LEFT_MAX_PULSE)
self.servo_front_shoulder_left_rest_angle =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_SHOULDER_LEFT_REST_ANGLE)

self.servo_front_leg_left_pca9685 =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_LEG_LEFT_PCA9685)
self.servo_front_leg_left_channel =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_LEG_LEFT_CHANNEL)
self.servo_front_leg_left_min_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_LEG_LEFT_MIN_PULSE)
self.servo_front_leg_left_max_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_LEG_LEFT_MAX_PULSE)
self.servo_front_leg_left_rest_angle =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_LEG_LEFT_REST_ANGLE)

self.servo_front_feet_left_pca9685 =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_FEET_LEFT_PCA9685)
self.servo_front_feet_left_channel =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_FEET_LEFT_CHANNEL)
self.servo_front_feet_left_min_pulse =
 Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_FEET_LEFT_MIN_PULSE)
self.servo_front_feet_left_max_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_FEET_LEFT_MAX_PULSE)
self.servo_front_feet_left_rest_angle =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_FEET_LEFT_REST_ANGLE)

self.servo_front_shoulder_right_pca9685 =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_SHOULDER_RIGHT_PCA9685)
self.servo_front_shoulder_right_channel =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_SHOULDER_RIGHT_CHANNEL)
self.servo_front_shoulder_right_min_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_SHOULDER_RIGHT_MIN_PULSE)
self.servo_front_shoulder_right_max_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_SHOULDER_RIGHT_MAX_PULSE)
self.servo_front_shoulder_right_rest_angle =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_SHOULDER_RIGHT_REST_ANGLE)

self.servo_front_leg_right_pca9685 =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_LEG_RIGHT_PCA9685)
self.servo_front_leg_right_channel =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_LEG_RIGHT_CHANNEL)
self.servo_front_leg_right_min_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_LEG_RIGHT_MIN_PULSE)
```

```
self.servo_front_leg_right_max_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_LEG_RIGHT_MAX_PULSE)
self.servo_front_leg_right_rest_angle =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_LEG_RIGHT_REST_ANGLE)

self.servo_front_feet_right_pca9685 =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_FEET_RIGHT_PCA9685)
self.servo_front_feet_right_channel =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_FEET_RIGHT_CHANNEL)
self.servo_front_feet_right_min_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_FEET_RIGHT_MIN_PULSE)
self.servo_front_feet_right_max_pulse =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_FEET_RIGHT_MAX_PULSE)
self.servo_front_feet_right_rest_angle =
Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_FEET_RIGHT_REST_ANGLE)

if self.servo_arm_rotation_pca9685:
    self.servo_arm_rotation_pca9685 =
    Config().get(Config.ARM_CONTROLLER_SERVOS_ARM_ROTATION_PCA9685)
    self.servo_arm_rotation_channel =
    Config().get(Config.ARM_CONTROLLER_SERVOS_ARM_ROTATION_CHANNEL)
    self.servo_arm_rotation_min_pulse =
    Config().get(Config.ARM_CONTROLLER_SERVOS_ARM_ROTATION_MIN_PULSE)
    self.servo_arm_rotation_max_pulse =
    Config().get(Config.ARM_CONTROLLER_SERVOS_ARM_ROTATION_MAX_PULSE)
    self.servo_arm_rotation_rest_angle =
    Config().get(Config.MOTION_CONTROLLER_SERVOS_ARM_ROTATION_REST_ANGLE)

    self.servo_arm_lift_pca9685 =
    Config().get(Config.ARM_CONTROLLER_SERVOS_ARM_LIFT_PCA9685)
    self.servo_arm_lift_channel =
    Config().get(Config.ARM_CONTROLLER_SERVOS_ARM_LIFT_CHANNEL)
    self.servo_arm_lift_min_pulse =
    Config().get(Config.ARM_CONTROLLER_SERVOS_ARM_LIFT_MIN_PULSE)
    self.servo_arm_lift_max_pulse =
    Config().get(Config.ARM_CONTROLLER_SERVOS_ARM_LIFT_MAX_PULSE)
    self.servo_arm_lift_rest_angle =
    Config().get(Config.MOTION_CONTROLLER_SERVOS_ARM_LIFT_REST_ANGLE)

    self.servo_arm_range_pca9685 =
    Config().get(Config.ARM_CONTROLLER_SERVOS_ARM_RANGE_PCA9685)
    self.servo_arm_range_channel =
    Config().get(Config.ARM_CONTROLLER_SERVOS_ARM_RANGE_CHANNEL)
    self.servo_arm_range_min_pulse =
    Config().get(Config.ARM_CONTROLLER_SERVOS_ARM_RANGE_MIN_PULSE)
    self.servo_arm_range_max_pulse =
    Config().get(Config.ARM_CONTROLLER_SERVOS_ARM_RANGE_MAX_PULSE)
    self.servo_arm_range_rest_angle =
    Config().get(Config.MOTION_CONTROLLER_SERVOS_ARM_RANGE_REST_ANGLE)

    self.servo_arm_cam_tilt_pca9685 =
    Config().get(Config.ARM_CONTROLLER_SERVOS_ARM_CAM_TILT_PCA9685)
    self.servo_arm_cam_tilt_channel =
    Config().get(Config.ARM_CONTROLLER_SERVOS_ARM_CAM_TILT_CHANNEL)
    self.servo_arm_cam_tilt_min_pulse =
    Config().get(Config.ARM_CONTROLLER_SERVOS_ARM_CAM_TILT_MIN_PULSE)
    self.servo_arm_cam_tilt_max_pulse =
```

```
            Config().get(Config.ARM_CONTROLLER_SERVOS_ARM_CAM_TILT_MAX_PULSE)
            self.servo_arm_cam_tilt_rest_angle =
            Config().get(Config.MOTION_CONTROLLER_SERVOS_ARM_CAM_TILT_REST_ANGLE)

    def activate_servos(self):

        if self.servo_rear_shoulder_left_pca9685 == 1:
            self.servo_rear_shoulder_left =
            servo.Servo(self.pca9685_1.channels[self.servo_rear_shoulder_left_channel])
        else:
            self.servo_rear_shoulder_left =
            servo.Servo(self.pca9685_2.channels[self.servo_rear_shoulder_left_channel])
        self.servo_rear_shoulder_left.
        set_pulse_width_range(min_pulse=self.servo_rear_shoulder_left_min_pulse,
            max_pulse=self.servo_rear_shoulder_left_max_pulse)

        if self.servo_rear_leg_left_pca9685 == 1:
            self.servo_rear_leg_left =
            servo.Servo(self.pca9685_1.channels[self.servo_rear_leg_left_channel])
        else:
            self.servo_rear_leg_left =
            servo.Servo(self.pca9685_2.channels[self.servo_rear_leg_left_channel])
        self.servo_rear_leg_left.
        set_pulse_width_range(min_pulse=self.servo_rear_leg_left_min_pulse,
            max_pulse=self.servo_rear_leg_left_max_pulse)

        if self.servo_rear_feet_left_pca9685 == 1:
            self.servo_rear_feet_left =
            servo.Servo(self.pca9685_1.channels[self.servo_rear_feet_left_channel])
        else:
            self.servo_rear_feet_left =
            servo.Servo(self.pca9685_2.channels[self.servo_rear_feet_left_channel])
        self.servo_rear_feet_left.
        set_pulse_width_range(min_pulse=self.servo_rear_feet_left_min_pulse,
            max_pulse=self.servo_rear_feet_left_max_pulse)

        if self.servo_rear_shoulder_right_pca9685 == 1:
            self.servo_rear_shoulder_right =
            servo.Servo(self.pca9685_1.channels[self.servo_rear_shoulder_right_channel])
        else:
            self.servo_rear_shoulder_right =
            servo.Servo(self.pca9685_2.channels[self.servo_rear_shoulder_right_channel])
        self.servo_rear_shoulder_right.
        set_pulse_width_range(min_pulse=self.servo_rear_shoulder_right_min_pulse,
            max_pulse=self.servo_rear_shoulder_right_max_pulse)

        if self.servo_rear_leg_right_pca9685 == 1:
            self.servo_rear_leg_right =
            servo.Servo(self.pca9685_1.channels[self.servo_rear_leg_right_channel])
        else:
            self.servo_rear_leg_right =
            servo.Servo(self.pca9685_2.channels[self.servo_rear_leg_right_channel])
        self.servo_rear_leg_right.
        set_pulse_width_range(min_pulse=self.servo_rear_leg_right_min_pulse,
            max_pulse=self.servo_rear_leg_right_max_pulse)

        if self.servo_rear_feet_right_pca9685 == 1:
```

```
        self.servo_rear_feet_right =
        servo.Servo(self.pca9685_1.channels[self.servo_rear_feet_right_channel])
    else:
        self.servo_rear_feet_right =
        servo.Servo(self.pca9685_2.channels[self.servo_rear_feet_right_channel])
    self.servo_rear_feet_right.
    set_pulse_width_range(min_pulse=self.servo_rear_feet_right_min_pulse,
        max_pulse=self.servo_rear_feet_right_max_pulse)

    if self.servo_front_shoulder_left_pca9685 == 1:
        self.servo_front_shoulder_left =
        servo.Servo(self.pca9685_1.channels[self.servo_front_shoulder_left_channel])
    else:
        self.servo_front_shoulder_left =
        servo.Servo(self.pca9685_2.channels[self.servo_front_shoulder_left_channel])
    self.servo_front_shoulder_left.
    set_pulse_width_range(min_pulse=self.servo_front_shoulder_left_min_pulse,
        max_pulse=self.servo_front_shoulder_left_max_pulse)

    if self.servo_front_leg_left_pca9685 == 1:
        self.servo_front_leg_left =
        servo.Servo(self.pca9685_1.channels[self.servo_front_leg_left_channel])
    else:
        self.servo_front_leg_left =
        servo.Servo(self.pca9685_2.channels[self.servo_front_leg_left_channel])
    self.servo_front_leg_left.
    set_pulse_width_range(min_pulse=self.servo_front_leg_left_min_pulse,
        max_pulse=self.servo_front_leg_left_max_pulse)

    if self.servo_front_feet_left_pca9685 == 1:
        self.servo_front_feet_left =
        servo.Servo(self.pca9685_1.channels[self.servo_front_feet_left_channel])
    else:
        self.servo_front_feet_left =
        servo.Servo(self.pca9685_2.channels[self.servo_front_feet_left_channel])
    self.servo_front_feet_left.
    set_pulse_width_range(min_pulse=self.servo_front_feet_left_min_pulse,
        max_pulse=self.servo_front_feet_left_max_pulse)

    if self.servo_front_shoulder_right_pca9685 == 1:
        self.servo_front_shoulder_right =
        servo.Servo(self.pca9685_1.channels[self.servo_front_shoulder_right_channel])
    else:
        self.servo_front_shoulder_right = servo.Servo(
            self.pca9685_2.channels[self.servo_front_shoulder_right_channel])
    self.servo_front_shoulder_right.
    set_pulse_width_range(min_pulse=self.servo_front_shoulder_right_min_pulse,
        max_pulse=self.servo_front_shoulder_right_max_pulse)

    if self.servo_front_leg_right_pca9685 == 1:
        self.servo_front_leg_right =
        servo.Servo(self.pca9685_1.channels[self.servo_front_leg_right_channel])
    else:
        self.servo_front_leg_right = servo.Servo(
            self.pca9685_2.channels[self.servo_front_leg_right_channel])
    self.servo_front_leg_right.
    set_pulse_width_range(min_pulse=self.servo_front_leg_right_min_pulse,
```

```
            max_pulse=self.servo_front_leg_right_max_pulse)

        if self.servo_front_feet_right_pca9685 == 1:
            self.servo_front_feet_right =
            servo.Servo(self.pca9685_1.channels[self.servo_front_feet_right_channel])
        else:
            self.servo_front_feet_right =
            servo.Servo(self.pca9685_2.channels[self.servo_front_feet_right_channel])
        self.servo_front_feet_right.
        set_pulse_width_range(min_pulse=self.servo_front_feet_right_min_pulse,
            max_pulse=self.servo_front_feet_right_max_pulse)

        if self.servo_arm_rotation_pca9685:

            if self.servo_arm_rotation_pca9685 == 1:
                self.servo_arm_rotation =
                servo.Servo(self.pca9685_1.channels[self.servo_arm_rotation_channel])
            else:
                self.servo_arm_rotation =
                servo.Servo(self.pca9685_2.channels[self.servo_arm_rotation_channel])
            self.servo_arm_rotation.
            set_pulse_width_range(min_pulse=self.servo_arm_rotation_min_pulse,
                max_pulse=self.servo_arm_rotation_max_pulse)

            if self.servo_arm_lift_pca9685 == 1:
                self.servo_arm_lift =
                servo.Servo(self.pca9685_1.channels[self.servo_arm_lift_channel])
            else:
                self.servo_arm_lift =
                servo.Servo(self.pca9685_2.channels[self.servo_arm_lift_channel])
            self.servo_arm_lift.
            set_pulse_width_range(min_pulse=self.servo_arm_lift_min_pulse,
                max_pulse=self.servo_arm_lift_max_pulse)

            if self.servo_arm_range_pca9685 == 1:
                self.servo_arm_range =
                servo.Servo(self.pca9685_1.channels[self.servo_arm_range_channel])
            else:
                self.servo_arm_range =
                servo.Servo(self.pca9685_2.channels[self.servo_arm_range_channel])
            self.servo_arm_range.
            set_pulse_width_range(min_pulse=self.servo_arm_range_min_pulse,
                max_pulse=self.servo_arm_range_max_pulse)

            if self.servo_arm_cam_tilt_pca9685 == 1:
                self.servo_arm_cam_tilt =
                servo.Servo(self.pca9685_1.channels[self.servo_arm_cam_tilt_channel])
            else:
                self.servo_arm_cam_tilt =
                servo.Servo(self.pca9685_2.channels[self.servo_arm_cam_tilt_channel])
            self.servo_arm_cam_tilt.
            set_pulse_width_range(min_pulse=self.servo_arm_cam_tilt_min_pulse,
                max_pulse=self.servo_arm_cam_tilt_max_pulse)

    def move(self):

        try:
```

```
        self.servo_rear_shoulder_left.angle =
        self.servo_rear_shoulder_left_rest_angle
    except ValueError as e:
        log.error('Impossible servo_rear_shoulder_left angle requested')

    try:
        self.servo_rear_leg_left.angle =
        self.servo_rear_leg_left_rest_angle
    except ValueError as e:
        log.error('Impossible servo_rear_leg_left angle requested')

    try:
        self.servo_rear_feet_left.angle =
        self.servo_rear_feet_left_rest_angle
    except ValueError as e:
        log.error('Impossible servo_rear_feet_left angle requested')

    try:
        self.servo_rear_shoulder_right.angle =
        self.servo_rear_shoulder_right_rest_angle
    except ValueError as e:
        log.error('Impossible servo_rear_shoulder_right angle requested')

    try:
        self.servo_rear_leg_right.angle =
        self.servo_rear_leg_right_rest_angle
    except ValueError as e:
        log.error('Impossible servo_rear_leg_right angle requested')

    try:
        self.servo_rear_feet_right.angle =
        self.servo_rear_feet_right_rest_angle
    except ValueError as e:
        log.error('Impossible servo_rear_feet_right angle requested')

    try:
        self.servo_front_shoulder_left.angle =
        self.servo_front_shoulder_left_rest_angle
    except ValueError as e:
        log.error('Impossible servo_front_shoulder_left angle requested')

    try:
        self.servo_front_leg_left.angle =
            self.servo_front_leg_left_rest_angle
    except ValueError as e:
        log.error('Impossible servo_front_leg_left angle requested')

    try:
        self.servo_front_feet_left.angle =
            self.servo_front_feet_left_rest_angle
    except ValueError as e:
        log.error('Impossible servo_front_feet_left angle requested')

    try:
        self.servo_front_shoulder_right.angle =
            self.servo_front_shoulder_right_rest_angle
    except ValueError as e:
```

```
            log.error('Impossible servo_front_shoulder_right angle requested')

        try:
            self.servo_front_leg_right.angle =
                self.servo_front_leg_right_rest_angle
        except ValueError as e:
            log.error('Impossible servo_front_leg_right angle requested')

        try:
            self.servo_front_feet_right.angle =
                self.servo_front_feet_right_rest_angle
        except ValueError as e:
            log.error('Impossible servo_front_feet_right angle requested')

        if self.servo_arm_rotation_pca9685:
            try:
                self.servo_arm_rotation.angle =
                 self.servo_arm_rotation_rest_angle
            except ValueError as e:
                log.error('Impossible servo_arm_rotation angle requested')

            try:
                self.servo_arm_lift.angle =
                    self.servo_arm_lift_rest_angle
            except ValueError as e:
                log.error('Impossible arm_lift angle requested')

            try:
                self.servo_arm_range.angle =
                self.servo_arm_range_rest_angle
            except ValueError as e:
                log.error('Impossible servo_arm_range angle requested')

            try:
                self.servo_arm_cam_tilt.angle =
                self.servo_arm_cam_tilt_rest_angle
            except ValueError as e:
                log.error('Impossible servo_arm_cam_tilt angle requested')

    def rest_position(self):

        self.servo_rear_shoulder_left_rest_angle =
        Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_SHOULDER_LEFT_REST_ANGLE)
        self.servo_rear_leg_left_rest_angle =
        Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_LEG_LEFT_REST_ANGLE)
        self.servo_rear_feet_left_rest_angle =
        Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_FEET_LEFT_REST_ANGLE)
        self.servo_rear_shoulder_right_rest_angle =
        Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_SHOULDER_RIGHT_REST_ANGLE)
        self.servo_rear_leg_right_rest_angle =
        Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_LEG_RIGHT_REST_ANGLE)
        self.servo_rear_feet_right_rest_angle =
        Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_FEET_RIGHT_REST_ANGLE)
        self.servo_front_shoulder_left_rest_angle =
        Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_SHOULDER_LEFT_REST_ANGLE)
        self.servo_front_leg_left_rest_angle =
        Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_LEG_LEFT_REST_ANGLE)
```

```
        self.servo_front_feet_left_rest_angle =
        Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_FEET_LEFT_REST_ANGLE)
        self.servo_front_shoulder_right_rest_angle =
        Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_SHOULDER_RIGHT_REST_ANGLE)
        self.servo_front_leg_right_rest_angle =
        Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_LEG_RIGHT_REST_ANGLE)
        self.servo_front_feet_right_rest_angle =
        Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_FEET_RIGHT_REST_ANGLE)

        if self.servo_arm_rotation_pca9685:
            self.servo_arm_rotation.angle =
            Config().get(Config.MOTION_CONTROLLER_SERVOS_ARM_ROTATION_REST_ANGLE)
            self.servo_arm_lift.angle =
            Config().get(Config.MOTION_CONTROLLER_SERVOS_ARM_LIFT_REST_ANGLE)
            self.servo_arm_range.angle =
            Config().get(Config.MOTION_CONTROLLER_SERVOS_ARM_RANGE_REST_ANGLE)
            self.servo_arm_cam_tilt.angle =
            Config().get(Config.MOTION_CONTROLLER_SERVOS_ARM_CAM_TILT_REST_ANGLE)

    def body_move_body_up_and_down(self, raw_value):

        range = 10
        range2 = 15

        if raw_value < 0:
            self.servo_rear_leg_left_rest_angle -= range
            self.servo_rear_feet_left_rest_angle += range2
            self.servo_rear_leg_right_rest_angle += range
            self.servo_rear_feet_right_rest_angle -= range2
            self.servo_front_leg_left_rest_angle -= range
            self.servo_front_feet_left_rest_angle += range2
            self.servo_front_leg_right_rest_angle += range
            self.servo_front_feet_right_rest_angle -= range2

        elif raw_value > 0:
            self.servo_rear_leg_left_rest_angle += range
            self.servo_rear_feet_left_rest_angle -= range2
            self.servo_rear_leg_right_rest_angle -= range
            self.servo_rear_feet_right_rest_angle += range2
            self.servo_front_leg_left_rest_angle += range
            self.servo_front_feet_left_rest_angle -= range2
            self.servo_front_leg_right_rest_angle -= range
            self.servo_front_feet_right_rest_angle += range2

        else:
            self.rest_position()

        print(str(self.servo_rear_leg_left_rest_angle) + ', ' +
            str(self.servo_rear_feet_left_rest_angle) + ', ' +
            str(self.servo_rear_leg_right_rest_angle) + ', ' +
            str(self.servo_rear_feet_right_rest_angle) + ', ' +
             str(self.servo_front_leg_left_rest_angle) + ', ' +
             str(self.servo_front_feet_left_rest_angle) + ', ' +
              str(self.servo_front_leg_right_rest_angle) + ', ' +
              str(self.servo_front_feet_right_rest_angle))

    def body_move_body_up_and_down_analog(self, raw_value):
```

```
servo_rear_leg_left_max_angle = 38
servo_rear_feet_left_max_angle = 70
servo_rear_leg_right_max_angle = 126
servo_rear_feet_right_max_angle = 102
servo_front_leg_left_max_angle = 57
servo_front_feet_left_max_angle = 85
servo_front_leg_right_max_angle = 130
servo_front_feet_right_max_angle = 120

delta_rear_leg_left =
int(General().maprange((1, -1),
    (Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_LEG_LEFT_REST_ANGLE),
        servo_rear_leg_left_max_angle), raw_value))

delta_rear_feet_left =
int(General().maprange((1, -1),
    (Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_FEET_LEFT_REST_ANGLE),
        servo_rear_feet_left_max_angle), raw_value))

delta_rear_leg_right =
int(General().maprange((1, -1),
    (Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_LEG_RIGHT_REST_ANGLE),
        servo_rear_leg_right_max_angle), raw_value))
delta_rear_feet_right =
int(General().maprange((1, -1),
    (Config().get(Config.MOTION_CONTROLLER_SERVOS_REAR_FEET_RIGHT_REST_ANGLE),
        servo_rear_feet_right_max_angle), raw_value))

delta_front_leg_left =
int(General().maprange((1, -1),
    (Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_LEG_LEFT_REST_ANGLE),
        servo_front_leg_left_max_angle), raw_value))

delta_front_feet_left =
int(General().maprange((1, -1),
    (Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_FEET_LEFT_REST_ANGLE),
        servo_front_feet_left_max_angle), raw_value))

delta_front_leg_right =
int(General().maprange((1, -1),
    (Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_LEG_RIGHT_REST_ANGLE),
        servo_front_leg_right_max_angle), raw_value))

delta_front_feet_right =
int(General().maprange((1, -1),
    (Config().get(Config.MOTION_CONTROLLER_SERVOS_FRONT_FEET_RIGHT_REST_ANGLE),
        servo_front_feet_right_max_angle), raw_value))

self.servo_rear_leg_left_rest_angle = delta_rear_leg_left
self.servo_rear_feet_left_rest_angle = delta_rear_feet_left
self.servo_rear_leg_right_rest_angle = delta_rear_leg_right
self.servo_rear_feet_right_rest_angle = delta_rear_feet_right
self.servo_front_leg_left_rest_angle = delta_front_leg_left
self.servo_front_feet_left_rest_angle = delta_front_feet_left
self.servo_front_leg_right_rest_angle = delta_front_leg_right
self.servo_front_feet_right_rest_angle = delta_front_feet_right
```

```
def body_move_body_left_right(self, raw_value):

    range = 5

    if raw_value < 0:
        self.servo_rear_shoulder_left_rest_angle -= range
        self.servo_rear_shoulder_right_rest_angle -= range
        self.servo_front_shoulder_left_rest_angle += range
        self.servo_front_shoulder_right_rest_angle += range

    elif raw_value > 0:
        self.servo_rear_shoulder_left_rest_angle += range
        self.servo_rear_shoulder_right_rest_angle += range
        self.servo_front_shoulder_left_rest_angle -= range
        self.servo_front_shoulder_right_rest_angle -= range

    else:
        self.rest_position()

def body_move_body_left_right_analog(self, raw_value):

    delta_a = int(General().maprange((-1, 1), (30, 150), raw_value))
    delta_b = int(General().maprange((-1, 1), (150, 30), raw_value))

    self.servo_rear_shoulder_left_rest_angle = delta_a
    self.servo_rear_shoulder_right_rest_angle = delta_a
    self.servo_front_shoulder_left_rest_angle = delta_b
    self.servo_front_shoulder_right_rest_angle = delta_b

def standing_position(self):

    variation_leg = 50
    variation_feet = 70

    self.servo_rear_shoulder_left.angle =
        self.servo_rear_shoulder_left_rest_angle + 10
    self.servo_rear_leg_left.angle =
        self.servo_rear_leg_left_rest_angle - variation_leg
    self.servo_rear_feet_left.angle =
        self.servo_rear_feet_left_rest_angle + variation_feet

    self.servo_rear_shoulder_right.angle =
        self.servo_rear_shoulder_right_rest_angle - 10
    self.servo_rear_leg_right.angle =
        self.servo_rear_leg_right_rest_angle + variation_leg
    self.servo_rear_feet_right.angle =
        self.servo_rear_feet_right_rest_angle - variation_feet

    time.sleep(0.05)

    self.servo_front_shoulder_left.angle =
        self.servo_front_shoulder_left_rest_angle - 10
    self.servo_front_leg_left.angle =
        self.servo_front_leg_left_rest_angle - variation_leg + 5
    self.servo_front_feet_left.angle =
        self.servo_front_feet_left_rest_angle + variation_feet - 5
```

```
        self.servo_front_shoulder_right.angle =
            self.servo_front_shoulder_right_rest_angle + 10
        self.servo_front_leg_right.angle =
            self.servo_front_leg_right_rest_angle + variation_leg - 5
        self.servo_front_feet_right.angle =
            self.servo_front_feet_right_rest_angle - variation_feet + 5

    def body_move_position_right(self):

        move = 20

        variation_leg = 50
        variation_feet = 70

        self.servo_rear_shoulder_left.angle =
            self.servo_rear_shoulder_left_rest_angle + 10 + move
        self.servo_rear_leg_left.angle =
            self.servo_rear_leg_left_rest_angle - variation_leg
        self.servo_rear_feet_left.angle =
            self.servo_rear_feet_left_rest_angle + variation_feet

        self.servo_rear_shoulder_right.angle =
            self.servo_rear_shoulder_right_rest_angle - 10 + move
        self.servo_rear_leg_right.angle =
            self.servo_rear_leg_right_rest_angle + variation_leg
        self.servo_rear_feet_right.angle =
            self.servo_rear_feet_right_rest_angle - variation_feet

        time.sleep(0.05)

        self.servo_front_shoulder_left.angle =
            self.servo_front_shoulder_left_rest_angle - 10 - move
        self.servo_front_leg_left.angle =
            self.servo_front_leg_left_rest_angle - variation_leg + 5
        self.servo_front_feet_left.angle =
            self.servo_front_feet_left_rest_angle + variation_feet - 5

        self.servo_front_shoulder_right.angle =
            self.servo_front_shoulder_right_rest_angle + 10 - move
        self.servo_front_leg_right.angle =
            self.servo_front_leg_right_rest_angle + variation_leg - 5
        self.servo_front_feet_right.angle =
            self.servo_front_feet_right_rest_angle - variation_feet + 5

    def body_move_position_left(self):

        move = 20

        variation_leg = 50
        variation_feet = 70

        self.servo_rear_shoulder_left.angle =
            self.servo_rear_shoulder_left_rest_angle + 10 - move
        self.servo_rear_leg_left.angle =
            self.servo_rear_leg_left_rest_angle - variation_leg
        self.servo_rear_feet_left.angle =
```

```
            self.servo_rear_feet_left_rest_angle + variation_feet

        self.servo_rear_shoulder_right.angle =
            self.servo_rear_shoulder_right_rest_angle - 10 - move
        self.servo_rear_leg_right.angle =
            self.servo_rear_leg_right_rest_angle + variation_leg
        self.servo_rear_feet_right.angle =
            self.servo_rear_feet_right_rest_angle - variation_feet

        time.sleep(0.05)

        self.servo_front_shoulder_left.angle =
            self.servo_front_shoulder_left_rest_angle - 10 + move
        self.servo_front_leg_left.angle =
            self.servo_front_leg_left_rest_angle - variation_leg + 5
        self.servo_front_feet_left.angle =
            self.servo_front_feet_left_rest_angle + variation_feet - 5

        self.servo_front_shoulder_right.angle =
            self.servo_front_shoulder_right_rest_angle + 10 + move
        self.servo_front_leg_right.angle =
            self.servo_front_leg_right_rest_angle + variation_leg - 5
        self.servo_front_feet_right.angle =
            self.servo_front_feet_right_rest_angle - variation_feet + 5

    def arm_set_rotation(self, raw_value):

        if not self.servo_arm_rotation_pca9685:
            return

        left_position =
            int(General().maprange((-1, 1), (0, 180), raw_value / 2))

        if int(self.servo_arm_rotation.angle) != int(left_position):
            self.servo_arm_rotation.angle = left_position

    def arm_set_lift(self, raw_value):

        if not self.servo_arm_rotation_pca9685:
            return

        lift_position =
            int(General().maprange((-1, 1), (180, 0), raw_value / 2))

        if int(self.servo_arm_lift.angle) != int(lift_position):
            self.servo_arm_lift.angle = lift_position

    def arm_set_range(self, raw_value):

        if not self.servo_arm_rotation_pca9685:
            return

        range_position =
            int(General().maprange((-1, 1), (180, 0), raw_value / 2))

        if int(self.servo_arm_range.angle) != int(range_position):
            self.servo_arm_range.angle = range_position
```

```
def arm_set_cam_tilt(self, raw_value):

    if not self.servo_arm_rotation_pca9685:
        return

    tilt_position =
        int(General().maprange((-1, 1), (100, 150), raw_value))

    if int(self.servo_arm_cam_tilt.angle) != int(tilt_position):
        self.servo_arm_cam_tilt.angel = tilt_position
```

10.6 Remotecontroller.py

The following script ensures wireless connectivity of PS4controller.

```
import time
import os
import struct
import array
from fcntl import ioctl
import signal
import sys
from spotmicroai.utilities.log import Logger
from spotmicroai.utilities.config import Config
import spotmicroai.utilities.queues as queues

log = Logger().setup_logger('Remote controller')


class RemoteControllerController:

    def __init__(self, communication_queues):

        try:

            log.debug('Starting controller...')

            signal.signal(signal.SIGINT, self.exit_gracefully)
            signal.signal(signal.SIGTERM, self.exit_gracefully)

            # We'll store the states here.
            self.connected_device = False
            self.axis_states = {}
            self.button_states = {}
            self.button_map = []
            self.axis_map = []
            self.jsdev = None
            self.previous_fvalue = 0

            self._abort_queue =
                communication_queues[queues.ABORT_CONTROLLER]
            self._motion_queue =
                communication_queues[queues.MOTION_CONTROLLER]
            self._lcd_screen_queue =
                communication_queues[queues.LCD_SCREEN_CONTROLLER]

        except Exception as e:
            self._lcd_screen_queue.
                put(queues.LCD_SCREEN_SHOW_REMOTE_CONTROLLER_CONTROLLER_NOK)
            log.error('Remote controller controller initialization problem', e)
            sys.exit(1)

    def exit_gracefully(self, signum, frame):
        log.info('Terminated')
        sys.exit(0)

    def do_process_events_from_queues(self):
```

```
remote_controller_connected_already = False

while True:

    if self.connected_device and not remote_controller_connected_already:
        self._lcd_screen_queue.
            put(queues.LCD_SCREEN_CONTROLLER_ACTION_ON)
        self._lcd_screen_queue.
            put(queues.LCD_SCREEN_SHOW_REMOTE_CONTROLLER_CONTROLLER_OK)
        remote_controller_connected_already = True
    else:
        time.sleep(2.5)
        self._abort_queue.
            put(queues.ABORT_CONTROLLER_ACTION_ABORT)
        self._lcd_screen_queue.
            put(queues.LCD_SCREEN_SHOW_REMOTE_CONTROLLER_CONTROLLER_SEARCHING)
        remote_controller_connected_already = False
        self.check_for_connected_devices()
        continue

    # Main event loop
    i = 0
    while True:

        try:
            evbuf = self.jsdev.read(8)
            if evbuf:
                buftime, value, type, number = struct.unpack('IhBB', evbuf)

                if type & 0x80:
                    continue

                if type & 0x01:
                    button = self.button_map[number]
                    if button:
                        self.button_states[button] = value

                if type & 0x02:
                    axis = self.axis_map[number]
                    if axis:
                        i += 1
                        fvalue = round(value / 32767.0, 3)

                        #if self.previous_fvalue == fvalue:
                        #    continue

                        self.axis_states[axis] = fvalue
                        self.previous_fvalue = fvalue

                        if axis in ['lx', 'ly', 'lz', 'rx', 'ry', 'rz']:
                            if i >= 6:
                                i = 0
                            else:
                                continue

            states = {}
            states.update(self.button_states)
```

```
                states.update(self.axis_states)

                # log.debug(states)

                self._motion_queue.put(states)

            except Exception as e:
                log.error('''Unknown problem while processing
                    the queue of the remote controller controller''', e)
                self._abort_queue.put(queues.ABORT_CONTROLLER_ACTION_ABORT)
                remote_controller_connected_already = False
                self.check_for_connected_devices()
                break

def check_for_connected_devices(self):

    connected_device =
        Config().get(Config.REMOTE_CONTROLLER_CONTROLLER_DEVICE)

    log.info('''The remote controller is not detected,
            looking for connected devices''')
    self.connected_device = False
    for fn in os.listdir('/dev/input'):
        # if fn.startswith('js'):
        if fn.startswith(str(connected_device)):
            self.connected_device = True

            # These constants were borrowed from linux/input.h
            axis_names = {
                0x00: 'lx',
                0x01: 'ly',
                0x02: 'lz',
                0x03: 'rx',
                0x04: 'ry',
                0x05: 'rz',
                0x06: 'trottle',
                0x07: 'rudder',
                0x08: 'wheel',
                0x09: 'gas',
                0x0a: 'brake',
                0x10: 'hat0x',
                0x11: 'hat0y',
                0x12: 'hat1x',
                0x13: 'hat1y',
                0x14: 'hat2x',
                0x15: 'hat2y',
                0x16: 'hat3x',
                0x17: 'hat3y',
                0x18: 'pressure',
                0x19: 'distance',
                0x1a: 'tilt_x',
                0x1b: 'tilt_y',
                0x1c: 'tool_width',
                0x20: 'volume',
                0x28: 'misc',
            }
```

```
button_names = {
    0x120: 'trigger',
    0x121: 'thumb',
    0x122: 'thumb2',
    0x123: 'top',
    0x124: 'top2',
    0x125: 'pinkie',
    0x126: 'base',
    0x127: 'base2',
    0x128: 'base3',
    0x129: 'base4',
    0x12a: 'base5',
    0x12b: 'base6',
    0x12f: 'dead',
    0x130: 'a',
    0x131: 'b',
    0x132: 'c',
    0x133: 'x',
    0x134: 'y',
    0x135: 'z',
    0x136: 'tl',
    0x137: 'tr',
    0x138: 'tl2',
    0x139: 'tr2',
    0x13a: 'select',
    0x13b: 'start',
    0x13c: 'mode',
    0x13d: 'thumbl',
    0x13e: 'thumbr',

    0x220: 'dpad_up',
    0x221: 'dpad_down',
    0x222: 'dpad_left',
    0x223: 'dpad_right',

    # XBox 360 controller uses these codes.
    0x2c0: 'dpad_left',
    0x2c1: 'dpad_right',
    0x2c2: 'dpad_up',
    0x2c3: 'dpad_down',
}

# Open the joystick device.
fn = '/dev/input/' + str(connected_device)

log.debug(('Opening␣%s...' % fn))
self.jsdev = open(fn, 'rb')

# Get the device name.
# buf = bytearray(63)
buf = array.array('B', [0] * 64)
ioctl(self.jsdev, 0x80006a13 + (0x10000 * len(buf)), buf)
# JSIOCGNAME(len)
js_name = buf.tostring().rstrip(b'\x00').decode('utf-8')
log.info(('Connected␣to␣device:␣%s' % js_name))

# Get number of axes and buttons.
```

```
        buf = array.array('B', [0])
        ioctl(self.jsdev, 0x80016a11, buf)  # JSIOCGAXES
        num_axes = buf[0]

        buf = array.array('B', [0])
        ioctl(self.jsdev, 0x80016a12, buf)  # JSIOCGBUTTONS
        num_buttons = buf[0]

        # Get the axis map.
        buf = array.array('B', [0] * 0x40)
        ioctl(self.jsdev, 0x80406a32, buf)  # JSIOCGAXMAP

        for axis in buf[:num_axes]:
            axis_name = axis_names.get(axis, 'unknown(0x%02x)' % axis)
            self.axis_map.append(axis_name)
            self.axis_states[axis_name] = 0.0

        # Get the button map.
        buf = array.array('H', [0] * 200)
        ioctl(self.jsdev, 0x80406a34, buf)  # JSIOCGBTNMAP

        for btn in buf[:num_buttons]:
            btn_name = button_names.get(btn, 'unknown(0x%03x)' % btn)
            self.button_map.append(btn_name)
            self.button_states[btn_name] = 0

        log.info(('%d axes found: %s' % (num_axes, ', '.join(self.axis_map))))
        log.info(('%d buttons found: %s' % (num_buttons, ', '.join(self.button_map))))

        break
```

The above code is entirely managed by the below program.

10.7 Main.py

```
#!/usr/bin/env python3

import sys

from spotmicroai.utilities.log import Logger

import multiprocessing

from spotmicroai.motion_controller.motion_controller
    import MotionController
from spotmicroai.abort_controller.abort_controller
    import AbortController
from spotmicroai.lcd_screen_controller.lcd_screen_controller
    import LCDScreenController
from spotmicroai.remote_controller.remote_controller
    import RemoteControllerController

log = Logger().setup_logger()


def process_abort_controller(communication_queues):
    abort = AbortController(communication_queues)
    abort.do_process_events_from_queue()


def process_motion_controller(communication_queues):
    motion = MotionController(communication_queues)
    motion.do_process_events_from_queues()


def process_remote_controller_controller(communication_queues):
    remote_controller =
        RemoteControllerController(communication_queues)
    remote_controller.do_process_events_from_queues()


# Optional
def process_output_lcd_screen_controller(communication_queues):
    lcd_screen = LCDScreenController(communication_queues)
    lcd_screen.do_process_events_from_queue()


def create_controllers_queues():
    communication_queues =
        {'abort_controller': multiprocessing.Queue(10),
            'motion_controller': multiprocessing.Queue(1),
                'lcd_screen_controller': multiprocessing.Queue(10)}

    log.info('Created␣the␣communication␣queues:␣' +
        ',␣'.join(communication_queues.keys()))
```

```
    return communication_queues


def close_controllers_queues(communication_queues):
    log.info('Closing␣controller␣queues')

    for queue in communication_queues.items():
        queue.close()
        queue.join_thread()


def main():
    communication_queues = create_controllers_queues()

    # Abort controller
    # Controls the OE port from PCA9685 to cut -
    # the power to the servos conveniently if needed.
    abort_controller =
    multiprocessing.Process(target=process_abort_controller,
        args=(communication_queues,))
    abort_controller.daemon = True  # The daemon dies if the parent process dies

    # Start the motion controller
    # Moves the servos
    motion_controller =
    multiprocessing.Process(target=process_motion_controller,
        args=(communication_queues,))
    motion_controller.daemon = True

    # Activate Bluetooth controller
    # Let you move the dog using the bluetooth paired device
    remote_controller_controller =
    multiprocessing.Process(target=process_remote_controller_controller,
                            args=(communication_queues,))
    remote_controller_controller.daemon = True

    # Screen
    # Show status of the components in the screen
    lcd_screen_controller =
    multiprocessing.Process(target=process_output_lcd_screen_controller,
                                args=(communication_queues,))
    lcd_screen_controller.daemon = True

    # Start the threads, queues messages are produced and consumed in those
    abort_controller.start()
    motion_controller.start()
    remote_controller_controller.start()
    lcd_screen_controller.start()

    if not abort_controller.is_alive():
        log.error("SpotMicro␣can't␣work␣without␣abort_controller")
        sys.exit(1)

    if not motion_controller.is_alive():
        log.error("SpotMicro␣can't␣work␣without␣motion_controller")
        sys.exit(1)
```

```
    if not remote_controller_controller:
        log.error('''SpotMicro can't work without
            remote_controller_controller''')
        sys.exit(1)

    # Make sure the thread/process ends
    abort_controller.join()
    motion_controller.join()
    remote_controller_controller.join()
    lcd_screen_controller.join()

    close_controllers_queues(communication_queues)


if __name__ == '__main__':
    log.info('SpotMicro starting...')

    try:
        main()

    except KeyboardInterrupt:
        log.info('Terminated due Control+C was pressed')

    else:
        log.info('Normal termination')
```

10.8 WebStream.py

The following code implements videostream coming from picamera attached in front of the robot.It provides following functionalities - Recording video - Image capturing - Heatmap generation(Recording video stream) - saving all recorded images and videos to disk

```
from django.conf import Settings
from imutils.video import VideoStream
import imutils
import cv2
import datetime
import time
import os
import numpy as np
import copy

class VideoCamera(object):
        def __init__(self):
                self.video = cv2.VideoCapture(0)
                self.filename="video.mp4"
                self.pic="new_img.jpg"
                self.res="720p"
                self.is_record = False
                self.out = None
                self.recordingThread=None
                self.record=False

        def __del__(self):
                self.video.release()
                cv2.destroyAllWindows()

        def change_res(self,cap,width,height):
                cap.set(3,width)
                cap.set(4,height)

        def get_dims(self,cap,re):
                STD_DIMENSIONS={"720p":(1280,720)}
                width,height=STD_DIMENSIONS[re]
                if re in STD_DIMENSIONS:
                        width,height=STD_DIMENSIONS[re]
                self.change_res(cap,width,height)
                return width,height

        def get_video_type(self,file):
                VIDEO_TYPE =
                        {"avi":cv2.VideoWriter_fourcc(*'mp4v'),
                        "mp4":cv2.VideoWriter_fourcc(*'mp4v')}
                file,ext=os.path.splitext(file)
                if ext in VIDEO_TYPE:
                        return VIDEO_TYPE[ext]
                return VIDEO_TYPE['mp4']

        def file(self,c,f):
                if f:
                        self.filename="video"+str(c)+".mp4"
```

```
                return self.filename
        else:
                return self.filename

def img(self,d,g):
        if g:
                self.pic="new_img"+str(d)+".jpg"
                return self.pic
        else:
                return self.pic


#web streaming
def get_frame(self,record,videocapture,imagecapture,
        hmap=False,c=None,f=False,d=None,g=False):
        if record:
                out =
                cv2.VideoWriter(self.file(c,f),
                        self.get_video_type(self.filename),
                        15, self.get_dims(self.video, self.res))
                newimg=self.img(d,g)
                casx='haarcascade_frontalface_default.xml'
                faceCascade=cv2.CascadeClassifier(casx)
                ret,frame=self.video.read()
                fgbg=cv2.createBackgroundSubtractorMOG2()
                height,width=frame.shape[:2]
                accum_image=np.zeros((height,width),np.uint8)
                while True:
                        ret,image = self.video.read()
                        gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
                        faces = faceCascade.detectMultiScale(gray,1.1,4)
                        font = cv2.FONT_HERSHEY_COMPLEX
                        text = "width:␣"+str(self.video.get(3))+
                                "height:␣"+str(self.video.get(4))
                        dates = str(datetime.datetime.now())
                        frame_flip =
                                cv2.putText(image,dates,(10,50),
                                        font,1,(0,255,255),2,cv2.LINE_AA)
                        for (x,y,w,h) in faces:
                                cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),2)
                        if hmap:
                                fgmask=fgbg.apply(gray)
                                thresh=2
                                maxvalue=2
                                ret,th1 =
                                        cv2.threshold(fgmask,thresh,maxvalue,
                                                        cv2.THRESH_BINARY)
                                accum_image = cv2.add(accum_image,th1)
                                color_image, im_color=
                                        cv2.applyColorMap(accum_image,cv2.COLORMAP_HOT)
                                frame_flip =
                                        cv2.addWeighted(image,0.7,color_image,0.7,0)
                        if videocapture:
                                out.write(frame_flip)
                        if imagecapture:
                                cv2.imwrite(newimg,frame_flip)
                                imagecapture=False
```

```
                ret, jpeg = cv2.imencode('.jpg', frame_flip)
                frame=jpeg.tobytes()
                yield (b'--frame\r\n'b'Content-Type: image/jpeg\r\n\r\n'
                        + frame + b'\r\n\r\n')
else:
        out.release()
```

Chapter 11

SIMULATION

Low-cost hobbyist robots such as Stanford Pupperand OpenQuadruped have gained popularity due to affordable 3D printing, embedded microcontrollers, and hobby servo motors. However, many algorithms require powerful computers and graphics cards to both train and deploy the neural networks that enable robots to exhibit complex behaviors such as walking over rough and uneven terrain or recovering from falls.Oftentimes, the graphics cards used to train these complex networks (e.g., a GeForce RTX 2080 Ti retailing for $1200) exceed the cost of a typical hobbyist robot (e.g.,the OpenQuadruped at $600). Other successful walking methods require a terrain map or torque control.

If we want robotic systems to be more accessible, it is necessary to develop light-weight algorithms that explicitly consider the constraints and limitations of low-cost hobbyist robots . Thus, we focus on learning simple linear policies that augment and improve locomotion skills for a class of low-cost hobby robotic systems subject to sensing and control limitations. We show that such policies lead to sufficient locomotion over unknown rough and uneven terrain.

To bridge the gap between effective but resource-intensive walking methods and inexpensive hardware, we adopt an overall architecture similar to that of models which follow a 12 layer deep neural network to move legs through proprioception and position. However, we show that Bezier curve gaits can be used as a lower-order open-loop model while a learned linear policy (instead of a neural network) modulates the gaits for improved locomotion on rough terrain. The policy can be efficiently trained on a CPU and be deployed on the low-powered embedded systems commonly used with the low-cost robots.

The entire system designing procedure through which simulation can be implemented is depicted in below diagram.

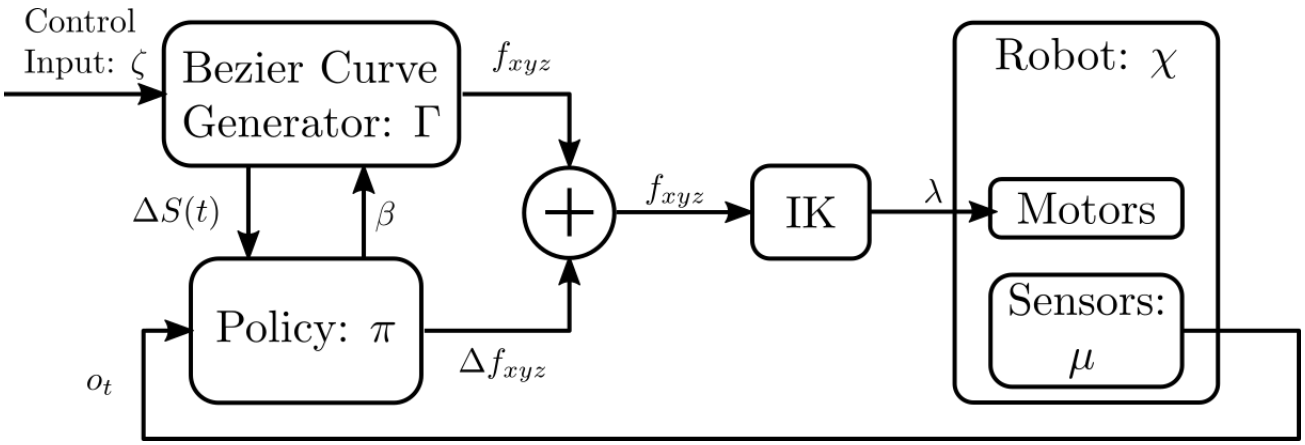


Figure 11.1: System design

Algorithm 1 Gait Modulation with Bezier Curves (GMBC)

Given: Policy π with parameter θ, (Bezier) Curve Generator Γ, External motion command ζ, robot sensor observations o_t, Leg phase $S(t)$

1: obtain gait modulation from π with learned parameter θ
2: $\Delta f_{xyz}, \beta = \pi(o_t, \theta)$
3: calculate (Bezier) gait foot placement
4: $f_{xyz} = \Gamma(S(t), \zeta, \beta)$
5: **return** $f_{xyz} + \Delta f_{xyz}$ to robot for IK joint control

Figure 11.2: Gait Modulation with Bezier Curves

11.1 Domain randomization - Gait modulation through Bezier curves:(DR-GMBC)

Following parameters are randomized at each phase of the training:

Randomized parameter	range
Base mass(Gaussian)	1.1kg $\pm$ 20 %
Leg link masses(Gaussian)	0.15kg $\pm$ 20 %
Foot friction(Uniform)	0.8 to 1.5
XYZ Mesh magnitude(uniform)	0m to 0.08m

Algorithm 2 RL Simulation training for DR-GMBC using Augmented Random Search [15]

Initialize: policy parameters θ_0, domain distribution $\mathbb{P}$, reward function $\mathcal{R}$, GMBC (Algorithm 1), iteration number $k = 0$, construct ARS.

1: **while** training not converged **do**
2: $\sigma \sim \mathbb{P}$ sample domain parameters
3: ARS step of (1) with domain randomization+GMBC
4: $\theta_{k+1} \leftarrow \text{ARS}(\pi, \theta_k, \mathcal{R}, \sigma, k)$
5: $k \leftarrow k + 1$
6: **end while**
7: **return** θ_k

Figure 11.3: Reinforcement learning training algorithm

11.2 Bezier Curve generator algorithm:

Algorithm 3 Bezier Curve Generator Γ - Per Leg ℓ

Inputs: ρ, $\bar{\omega}$, L_{span}

Map t to foot phase $S_\ell(t)$ using (8)

$(f_q^{tr}, f_z^{tr}) = \Upsilon(S_\ell(t), L_{\text{span}})$

$(f_q^{yaw}, f_z^{yaw}) = \Upsilon(S_\ell(t), \bar{\omega})$

$f_{xyz}^{tr} = \begin{bmatrix} f_q^{tr} \cos\rho & f_q^{tr} \sin\rho & f_z^{tr} \end{bmatrix}$

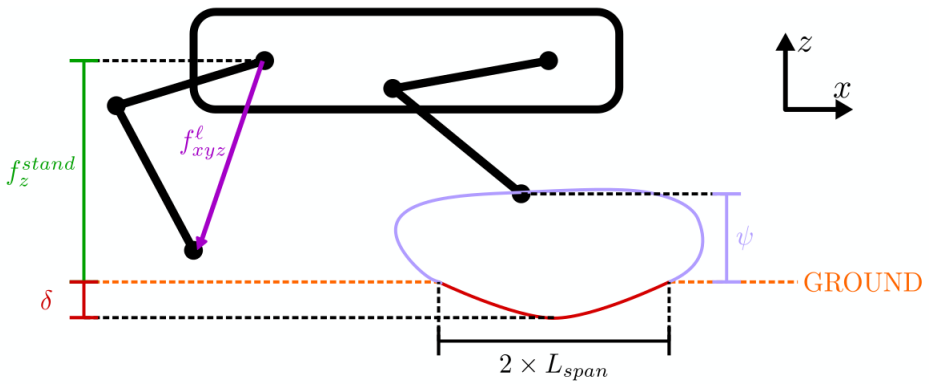
$\phi_{\text{arc}}^{\ell} = g_{\text{ang}}^{\ell} + \phi_{\text{stand}}^{\ell} + \frac{\pi}{2}$

$f_{xyz,\ell}^{yaw} = \begin{bmatrix} f_q^{yaw} \cos\phi_{arc}^{\ell} & f_q \sin\phi_{arc}^{\ell} & f_z^{yaw} \end{bmatrix}$

$f_{xyz}^{\ell} = f_{xyz}^{tr} + f_{xyz,\ell}^{yaw} + f_{xyz}^{\text{stand}}$

return f_{xyz}^{ℓ} to the robot for joint actuation

Following diagrams shows foot placement of robot on Bezier gait generator:



11.3 Environment and training:

An open-source simulation environment with a URDF based on our custom quadruped for training and validation. The simulation features full state randomization, including terrain mesh, body mass, and foot friction resampling at each epoch. The environment contains a simulated IMU, and an optional set of simulated contact sensors. In terms of environment debugging tools, users have the ability to trace the path of each foot in real time, which aids in evaluating agents and in crafting gaits for agents to modulate. The

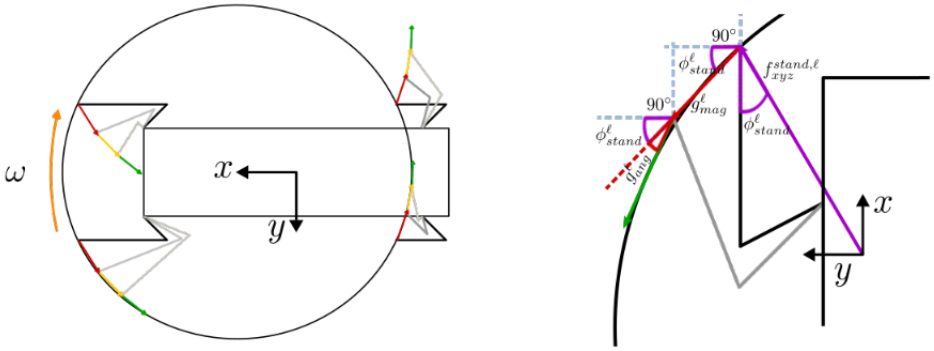
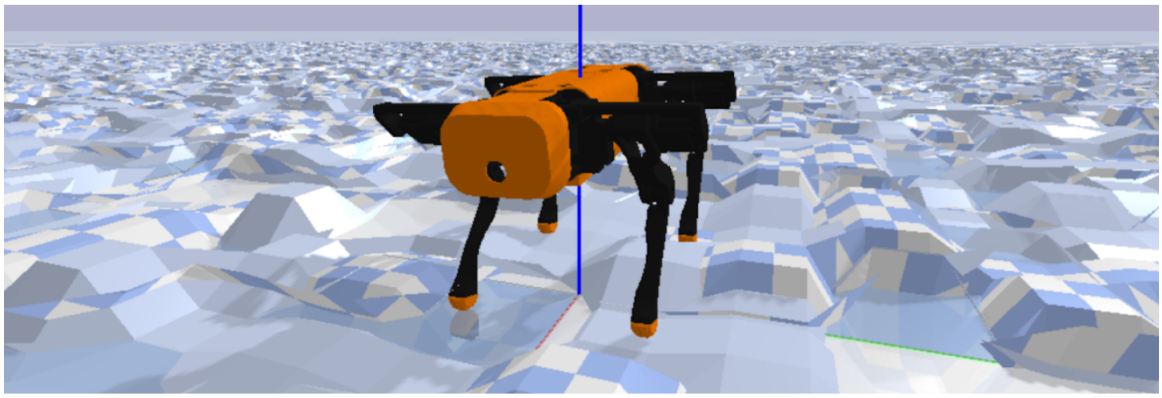
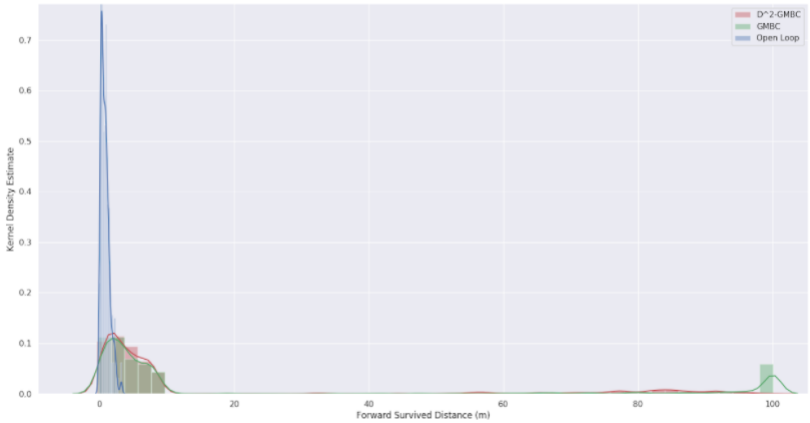


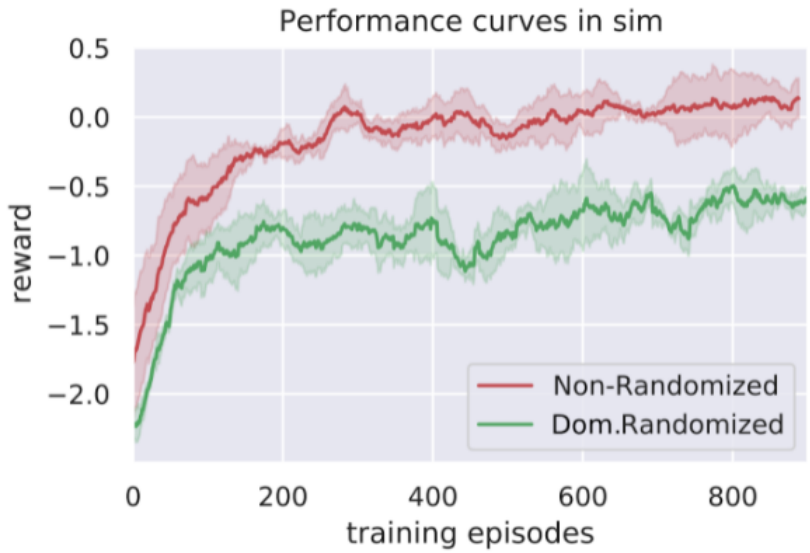
Figure 11.4: Output from Bezier gait generator

terrain mesh is also customizable, with the default terrain height being at up to 40% of body height for our URDF (Robot training happens through the algorithm2).





1000-trial forward motion test results.



50-Epoch Moving Average GMBC Agent Training.

11.4 Validation :

To examine the efficacy of our method, we first plotted the policy output for flat and rough terrain. On flat ground, there are clear repeated patterns that - as expected - disappear once we deploy our trained agent on rough terrain. Finally, we ran a series of sim-to-real experiments with no modification to the agent whatsoever to evaluate performance.

We developed simulation through an open source python library named PyBullet.To simulate it's environment we used an open source software called Bullet physics(SDK) which was developed by Google engineer ErwinCowmans.

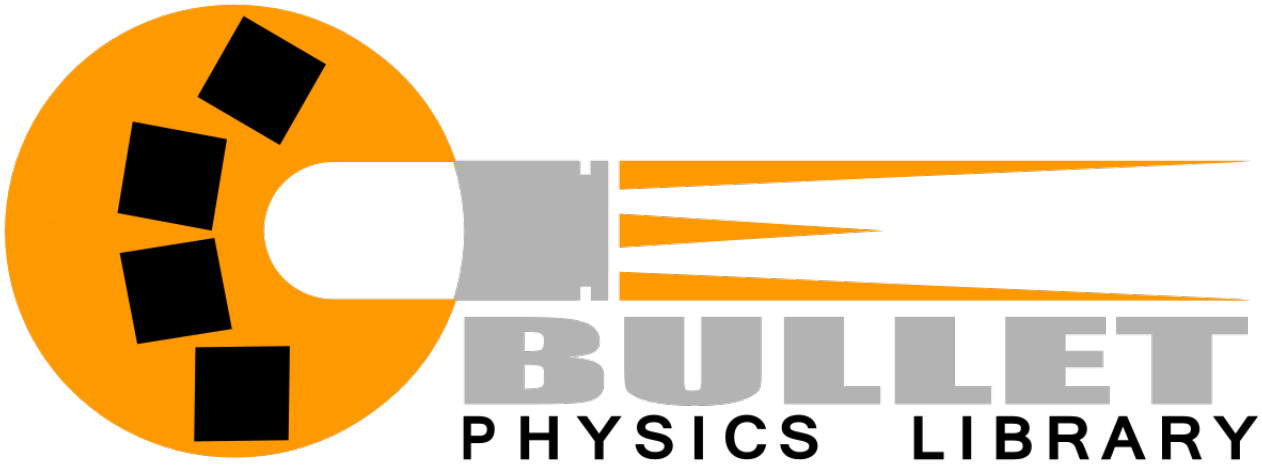
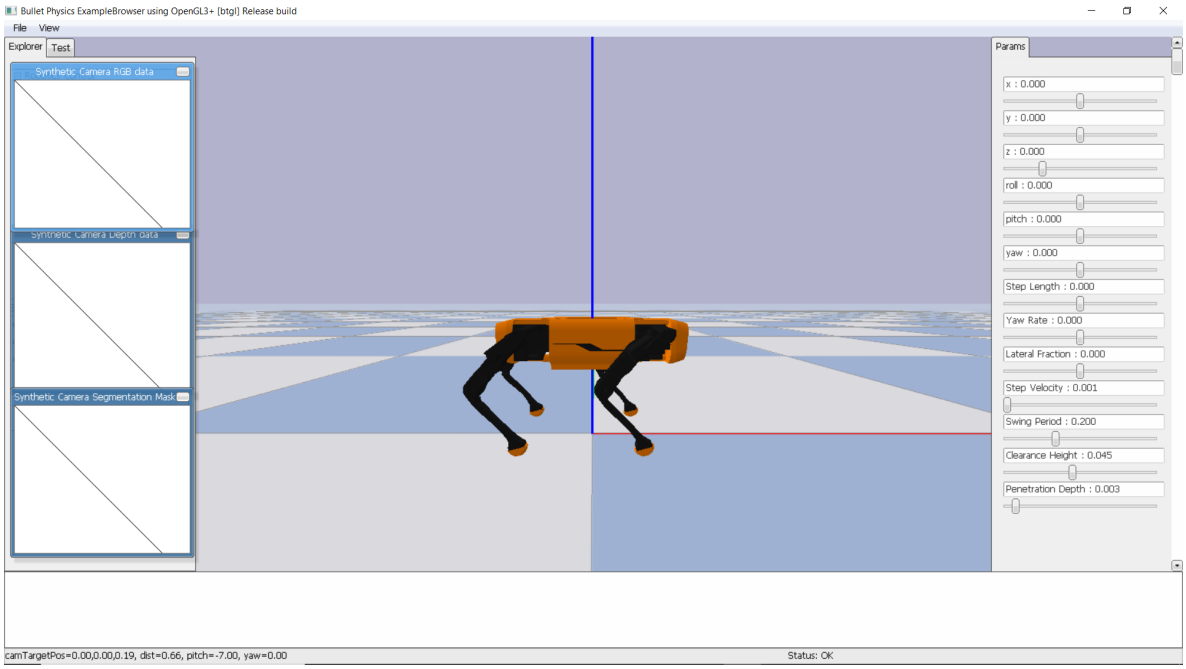
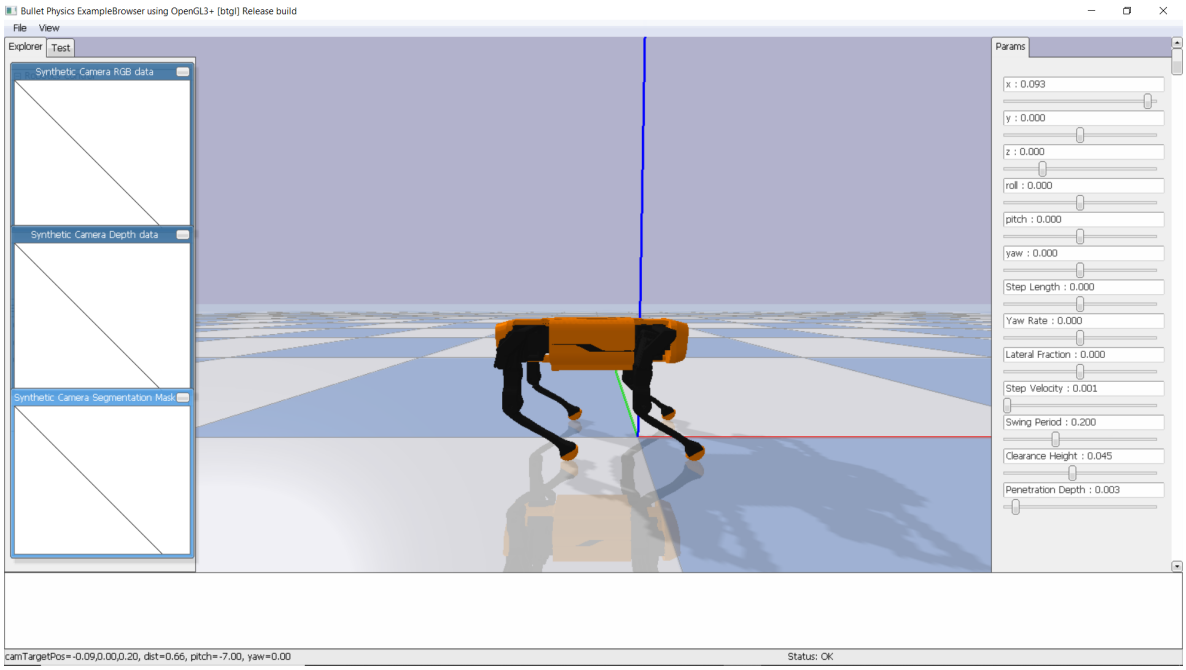
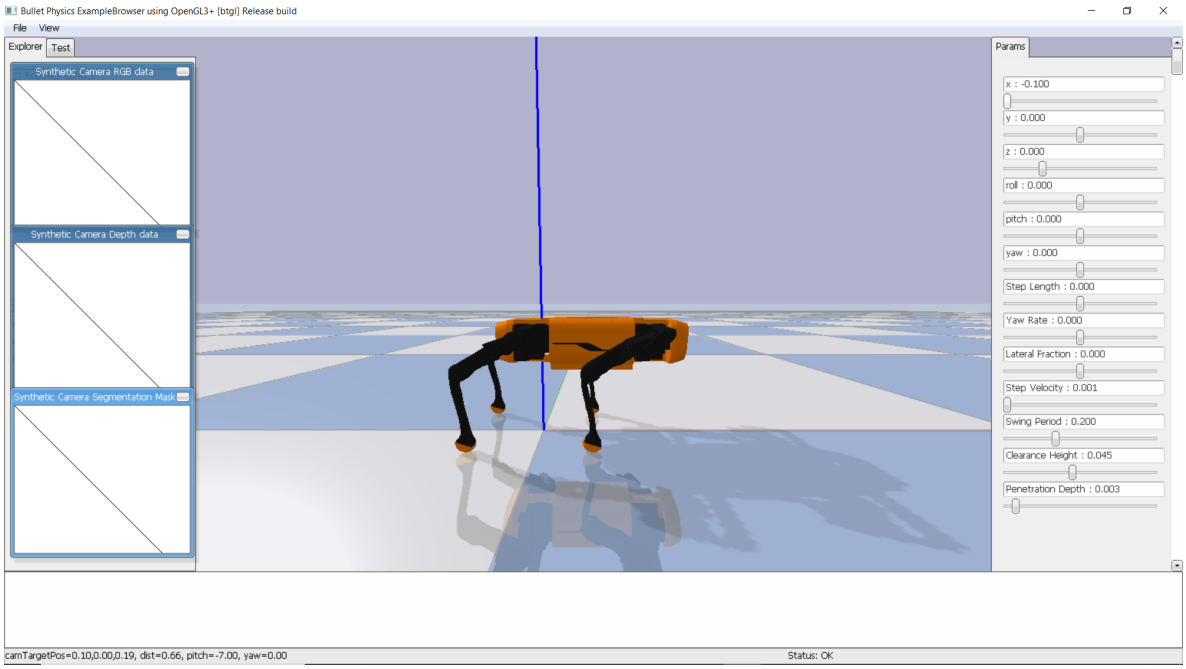


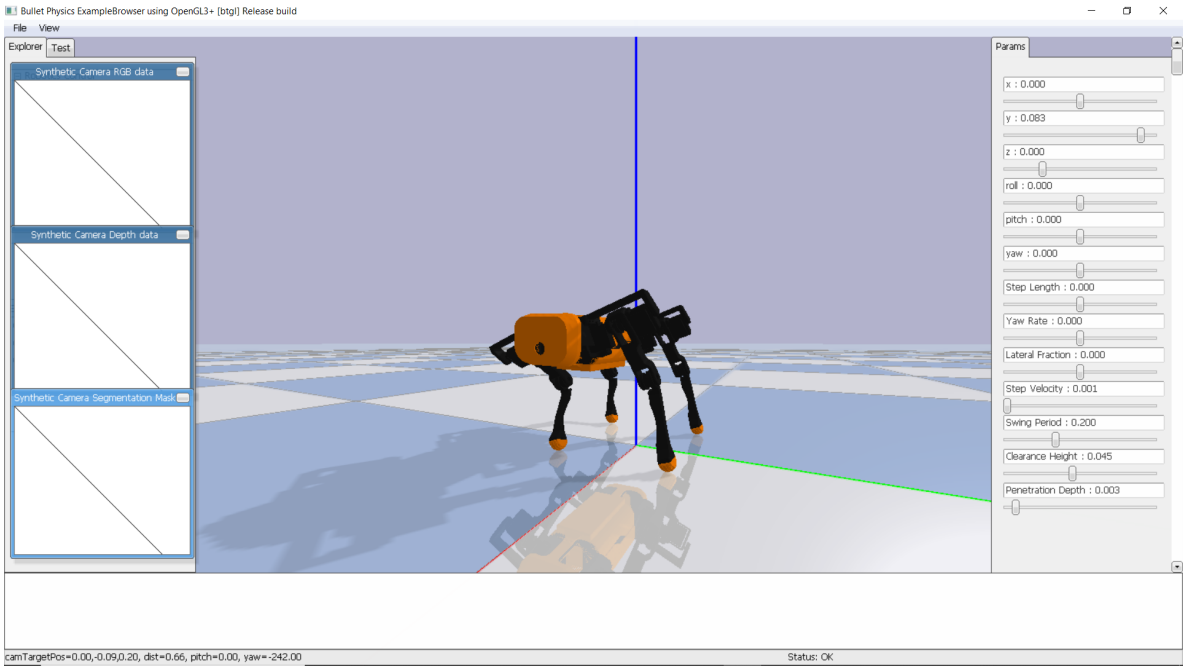
Figure 11.5: Bullet Physics Library

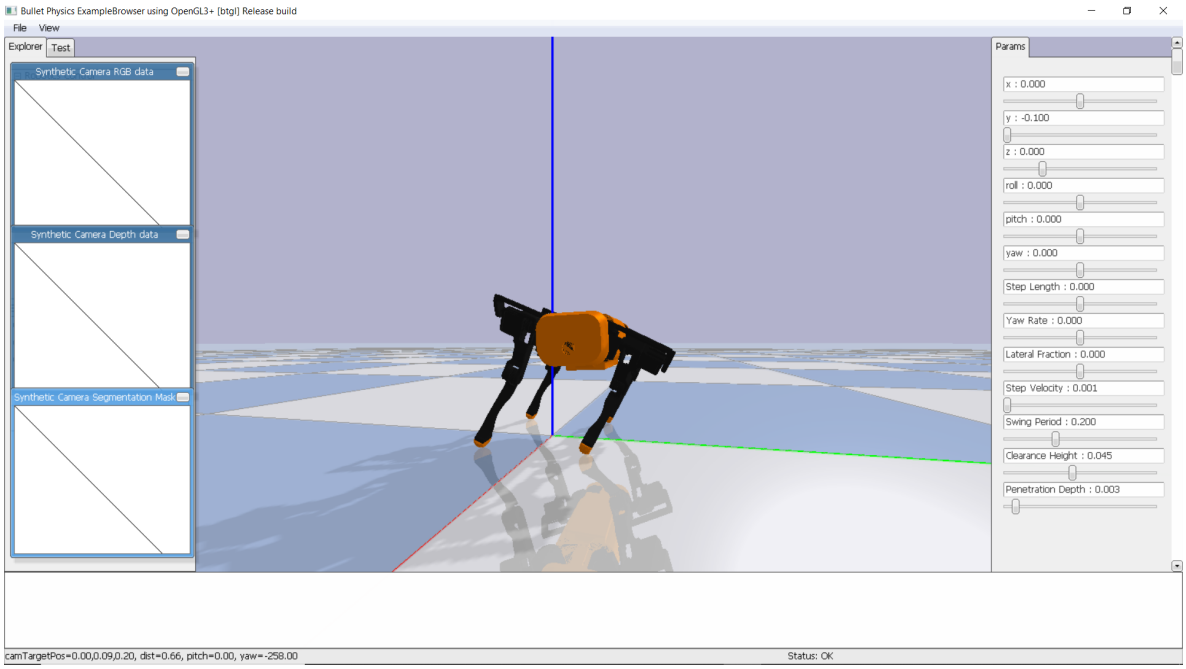
Following pictures illustrate the GUI interface to explore the robot's environment in all possible ways.

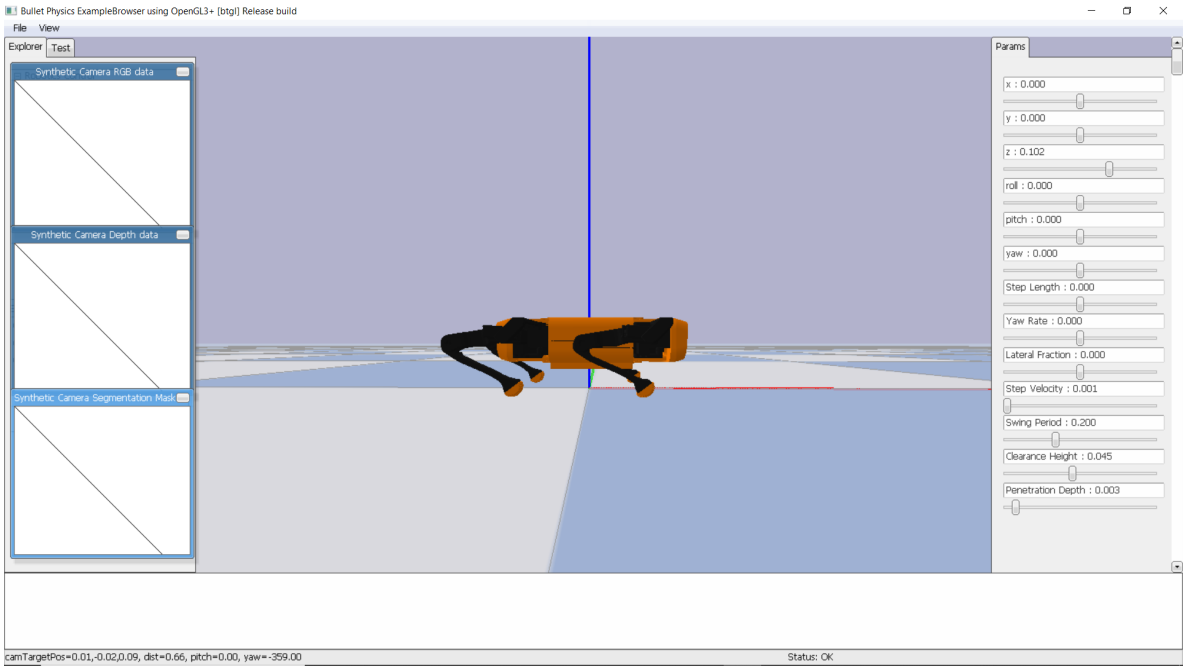

Bullet Physics ExampleBrowser using OpenGL3+ (btgl) Release build
File
View
Explorer
Test
Synthetic Camera RGB data
Synthetic Camera Depth data
Synthetic Camera Segmentation Mask
Params
x : 0.000
y : 0.000
z : 0.000
roll : 0.000
pitch : 0.000
yaw : 0.000
Step Length : 0.000
Yaw Rate : 0.000
Lateral Fraction : 0.000
Step Velocity : 0.001
Swing Period : 0.200
Clearance Height : 0.045
Penetration Depth : 0.003
camTargetPos=0.00,0.00,0.19, dist=0.66, pitch=-7.00, yaw=0.00
Status: OK

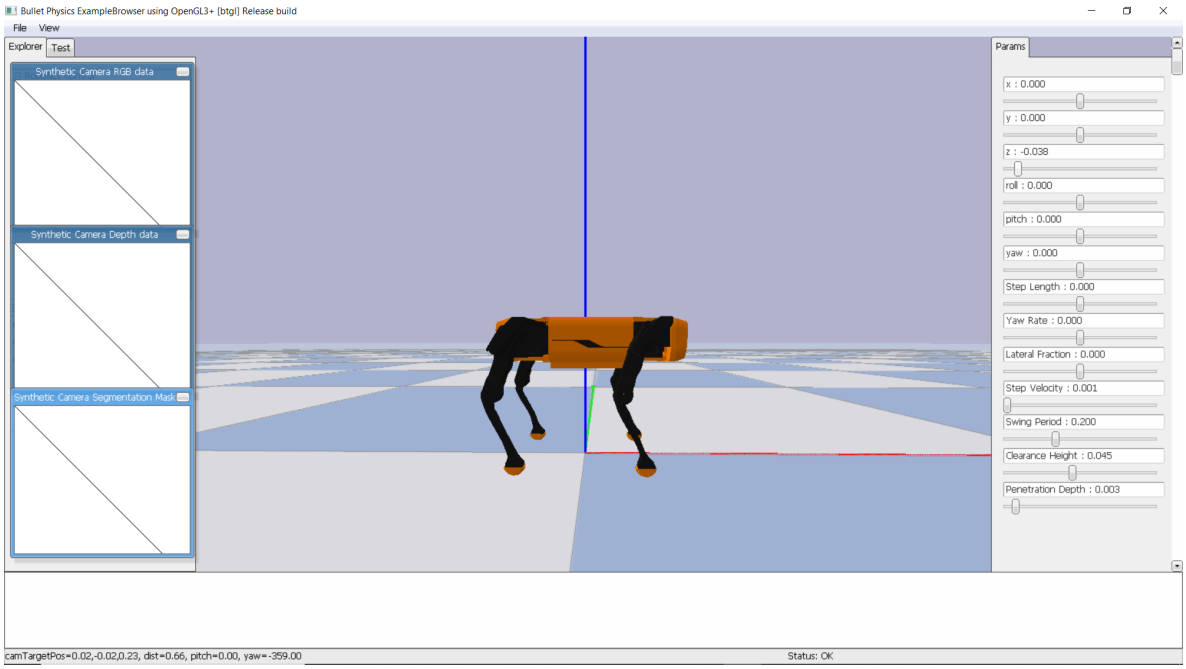

Bullet Physics ExampleBrowser using OpenGL3+ (btgl) Release build
File
View
Explorer
Test
Synthetic Camera RGB data
Synthetic Camera Depth data
Synthetic Camera Segmentation Mask
Params
x : 0.093
y : 0.000
z : 0.000
roll : 0.000
pitch : 0.000
yaw : 0.000
Step Length : 0.000
Yaw Rate : 0.000
Lateral Fraction : 0.000
Step Velocity : 0.001
Swing Period : 0.200
Clearance Height : 0.045
Penetration Depth : 0.003
camTargetPos=-0.09,0.00,0.20, dist=0.66, pitch=-7.00, yaw=0.00
Status: OK

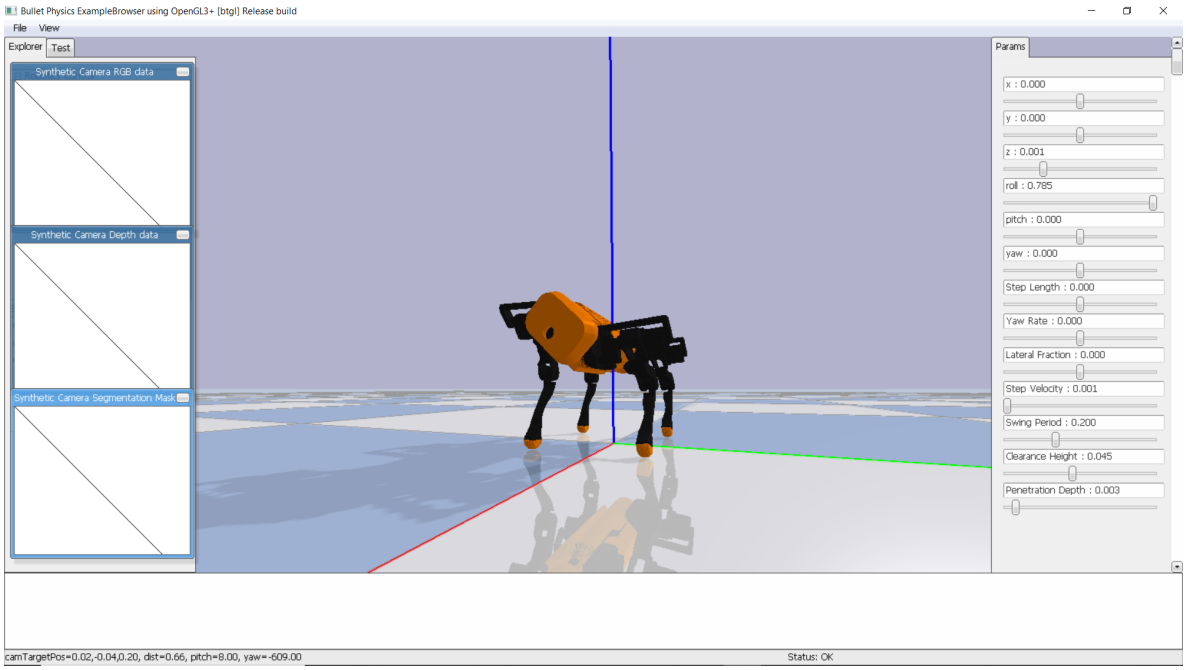

Bullet Physics ExampleBrowser using OpenGL3+ (btgl) Release build
File
View
Explorer
Test
Synthetic Camera RGB data
Synthetic Camera Depth data
Synthetic Camera Segmentation Mask
Params
x : -0.100
y : 0.000
z : 0.000
roll : 0.000
pitch : 0.000
yaw : 0.000
Step Length : 0.000
Yaw Rate : 0.000
Lateral Fraction : 0.000
Step Velocity : 0.001
Swing Period : 0.200
Clearance Height : 0.045
Penetration Depth : 0.003
camTargetPos=0.10,0.00,0.19, dist=0.66, pitch=-7.00, yaw=0.00
Status: OK

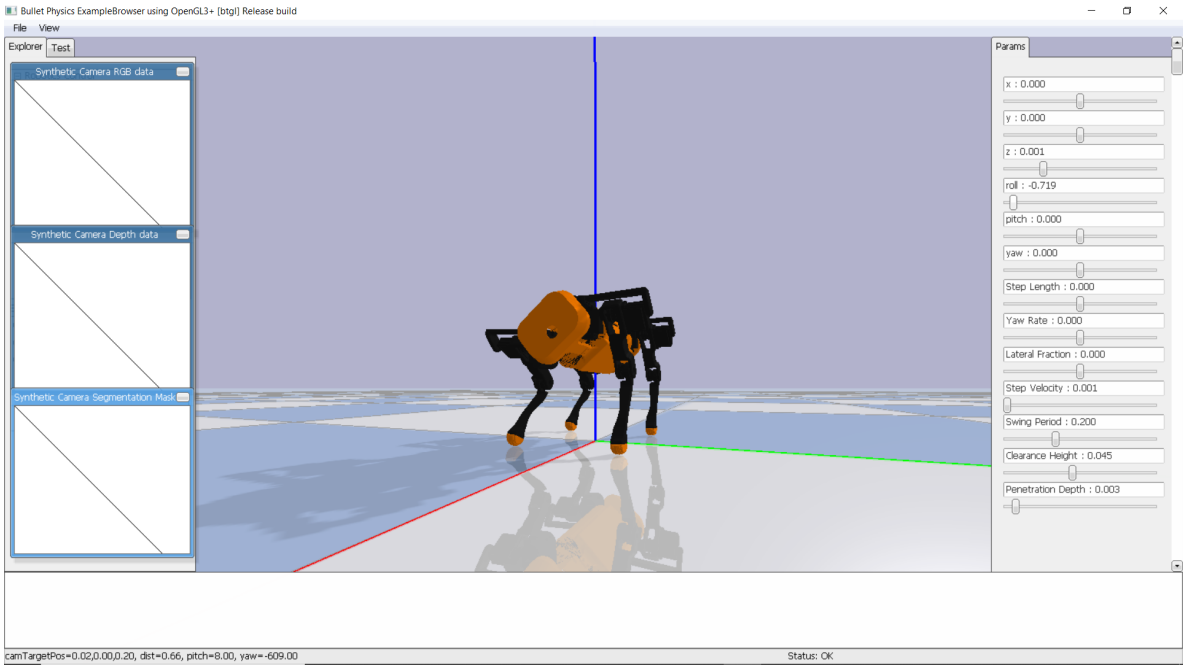

Bullet Physics ExampleBrowser using OpenGL3+ (btgl) Release build
File
View
Explorer
Test
Synthetic Camera RGB data
Synthetic Camera Depth data
Synthetic Camera Segmentation Mask
Params
x : 0.000
y : 0.083
z : 0.000
roll : 0.000
pitch : 0.000
yaw : 0.000
Step Length : 0.000
Yaw Rate : 0.000
Lateral Fraction : 0.000
Step Velocity : 0.001
Swing Period : 0.200
Clearance Height : 0.045
Penetration Depth : 0.003
camTargetPos=0.00,-0.09,0.20, dist=0.66, pitch=0.00, yaw=-242.00
Status: OK

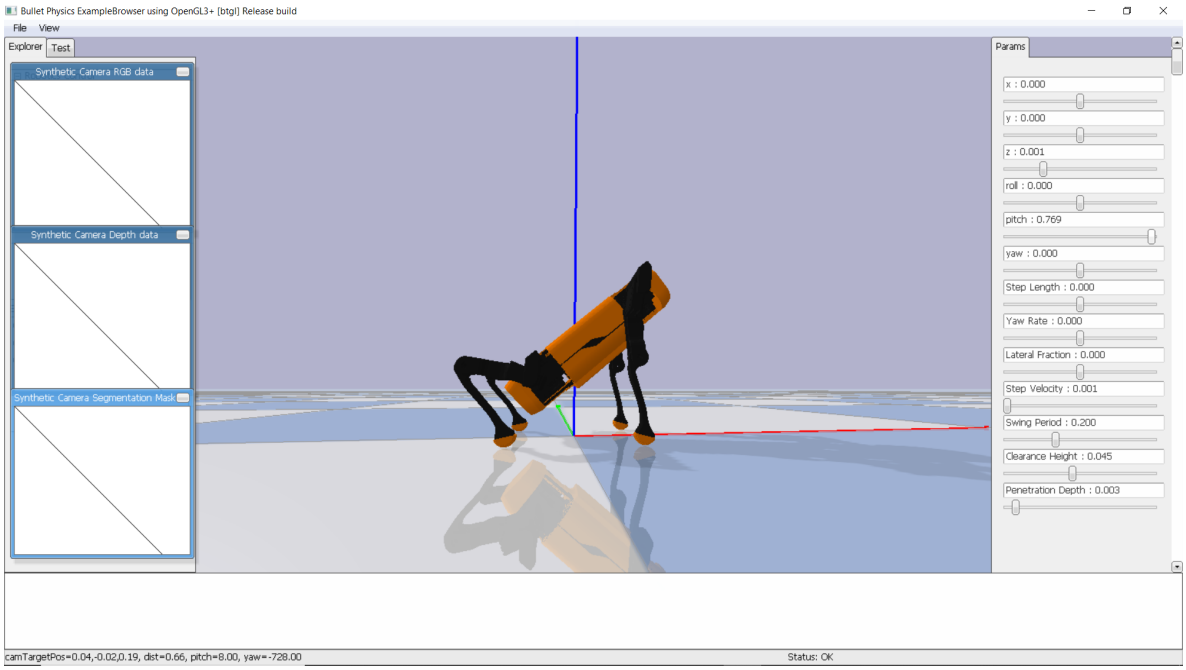

Bullet Physics ExampleBrowser using OpenGL3+ (btgl) Release build
File
View
Explorer
Test
Synthetic Camera RGB data
Synthetic Camera Depth data
Synthetic Camera Segmentation Mask
Params
x : 0.000
y : -0.100
z : 0.000
roll : 0.000
pitch : 0.000
yaw : 0.000
Step Length : 0.000
Yaw Rate : 0.000
Lateral Fraction : 0.000
Step Velocity : 0.001
Swing Period : 0.200
Clearance Height : 0.045
Penetration Depth : 0.003
camTargetPos=0.00,0.09,0.20, dist=0.66, pitch=0.00, yaw=-258.00
Status: OK

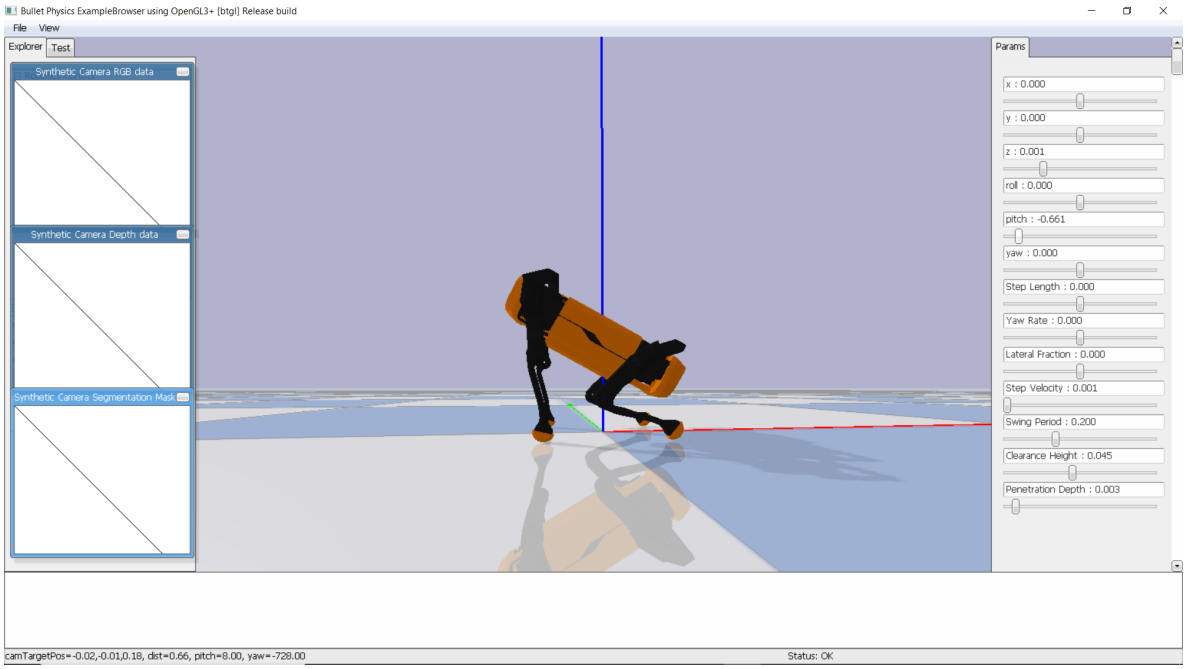

Bullet Physics ExampleBrowser using OpenGL3+ (btgl) Release build
File
View
Explorer
Test
Synthetic Camera RGB data
Synthetic Camera Depth data
Synthetic Camera Segmentation Mask
Params
x : 0.000
y : 0.000
z : 0.102
roll : 0.000
pitch : 0.000
yaw : 0.000
Step Length : 0.000
Yaw Rate : 0.000
Lateral Fraction : 0.000
Step Velocity : 0.001
Swing Period : 0.200
Clearance Height : 0.045
Penetration Depth : 0.003
camTargetPos=0.01,-0.02,0.09, dist=0.66, pitch=0.00, yaw=-359.00
Status: OK


Bullet Physics ExampleBrowser using OpenGL3+ (btgl) Release build
File
View
Explorer
Test
Synthetic Camera RGB data
Synthetic Camera Depth data
Synthetic Camera Segmentation Mask
Params
x : 0.000
y : 0.000
z : -0.038
roll : 0.000
pitch : 0.000
yaw : 0.000
Step Length : 0.000
Yaw Rate : 0.000
Lateral Fraction : 0.000
Step Velocity : 0.001
Swing Period : 0.200
Clearance Height : 0.045
Penetration Depth : 0.003
camTargetPos=0.02,-0.02,0.23, dist=0.66, pitch=0.00, yaw=-359.00
Status: OK


Bullet Physics ExampleBrowser using OpenGL3+ (btgl) Release build
File
View
Explorer
Test
Synthetic Camera RGB data
Synthetic Camera Depth data
Synthetic Camera Segmentation Mask
Params
x : 0.000
y : 0.000
z : 0.001
roll : 0.785
pitch : 0.000
yaw : 0.000
Step Length : 0.000
Yaw Rate : 0.000
Lateral Fraction : 0.000
Step Velocity : 0.001
Swing Period : 0.200
Clearance Height : 0.045
Penetration Depth : 0.003
camTargetPos=0.02,-0.04,0.20, dist=0.66, pitch=8.00, yaw=-609.00
Status: OK


Bullet Physics ExampleBrowser using OpenGL3+ (btgl) Release build
File
View
Explorer
Test
Synthetic Camera RGB data
Synthetic Camera Depth data
Synthetic Camera Segmentation Mask
Params
x : 0.000
y : 0.000
z : 0.001
roll : -0.719
pitch : 0.000
yaw : 0.000
Step Length : 0.000
Yaw Rate : 0.000
Lateral Fraction : 0.000
Step Velocity : 0.001
Swing Period : 0.200
Clearance Height : 0.045
Penetration Depth : 0.003
camTargetPos=0.02,0.00,0.20, dist=0.66, pitch=8.00, yaw=-609.00
Status: OK


Bullet Physics ExampleBrowser using OpenGL3+ (btgl) Release build
File
View
Explorer
Test
Synthetic Camera RGB data
Synthetic Camera Depth data
Synthetic Camera Segmentation Mask
Params
x : 0.000
y : 0.000
z : 0.001
roll : 0.000
pitch : 0.769
yaw : 0.000
Step Length : 0.000
Yaw Rate : 0.000
Lateral Fraction : 0.000
Step Velocity : 0.001
Swing Period : 0.200
Clearance Height : 0.045
Penetration Depth : 0.003
camTargetPos=0.04,-0.02,0.19, dist=0.66, pitch=8.00, yaw=-728.00
Status: OK


Bullet Physics ExampleBrowser using OpenGL3+ (btgl) Release build
File
View
Explorer
Test
Synthetic Camera RGB data
Synthetic Camera Depth data
Synthetic Camera Segmentation Mask
Params
x : 0.000
y : 0.000
z : 0.001
roll : 0.000
pitch : -0.661
yaw : 0.000
Step Length : 0.000
Yaw Rate : 0.000
Lateral Fraction : 0.000
Step Velocity : 0.001
Swing Period : 0.200
Clearance Height : 0.045
Penetration Depth : 0.003
camTargetPos=-0.02,-0.01,0.18, dist=0.66, pitch=8.00, yaw=-728.00
Status: OK

CONCLUSION AND FUTURE WORK

The LISA model is a reasonably priced fully simulated quadruped robot developed using PyBullet that can be controlled with remote bluetooth controllers like PS4 or XBOX can be used in both the rough and tough terrain environments. The Robot is used to monitor the terrain environments where humans are unable to pass through and handle various works accordingly. The Gait patterns implemented here depict the original beings to mimic them. The Improved stability provides the robot to wake up immediately after a dash or a fall.

The Robot can be extended to use in exacavation sites, monitoring areas, dangerous terrains, military bombings areas. The Interfaced Webcam provides the feed of the surroundings to the owner several feet away. The Further software improvements can provide features like facial recognition, object detection, intruder detections.

REFERENCES

https://en.wikipedia.org/wiki/Raspberry_Pi
https://www.raspberrypi.org/documentation/
https://www.raspberrypi.org/products/raspberry-pi-4-model-b/
https://learn.adafruit.com/
https://dev.bostondynamics.com/
https://www.autodesk.in/products/fusion-360/personal
https://www.sportsvet.com/muscle-actions-of-the-legs-during-locomotion-of-the-dog/
https://gitlab.com/custom_robots/spotmicroai
https://sites.google.com/view/drgmbc/home?authuser=0
https://dspace.mit.edu/handle/1721.1/98270