ISRO Robotics Challenge

IRoC-U

Design And Test Report

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

Team Name
YUVAAN IITG

Team ID
10320

Team Lead
Aditya Paul

Institute Name
Indian Institute of Technology, Guwahati

Contact Info:
Aditya Paul
+91 91679 56235

Table of Contents

No.	Topic	Page
1	Introduction	01
2	Rover Mobility	04
3	Navigation	11
4	Manipulator	19
5	Sensor System	36
6	Emergency Stop	40
7	COTS Components	42
8	Test Plan	45

00

Introduction

0.1 Introduction

This design report documents the comprehensive process and methodologies employed by our team in the design and development of "Anveshak", a rover specifically made according to the problem statement given for IRoC-U. The word "Anveshak" id derived from the word "Anveshakah", which means explorer in Sanskrit.

The primary objective of our team was to engineer a rover capable of navigating and investigating unexplored terrains.

0.2 Interdependencies Schematic

The diagram below shows a high-level overall System Architecture of our rover.

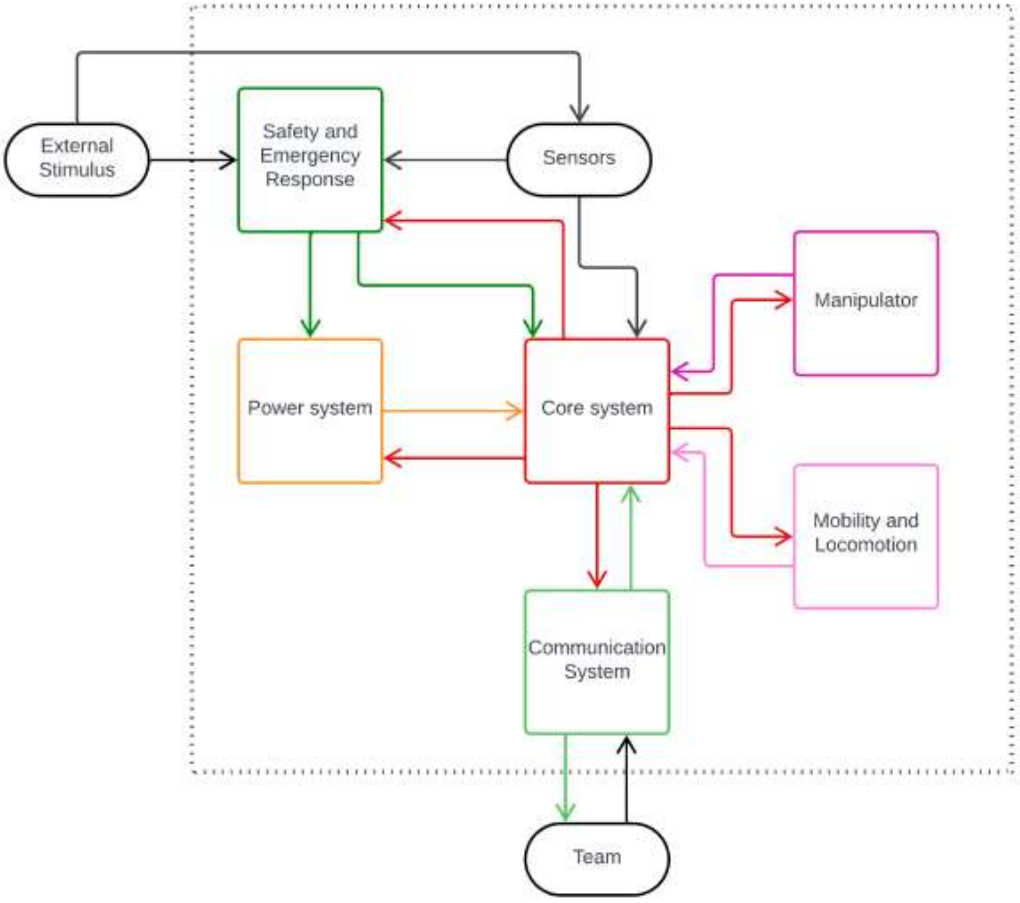


Figure 1.1

0.3 Hardware Description Schematic

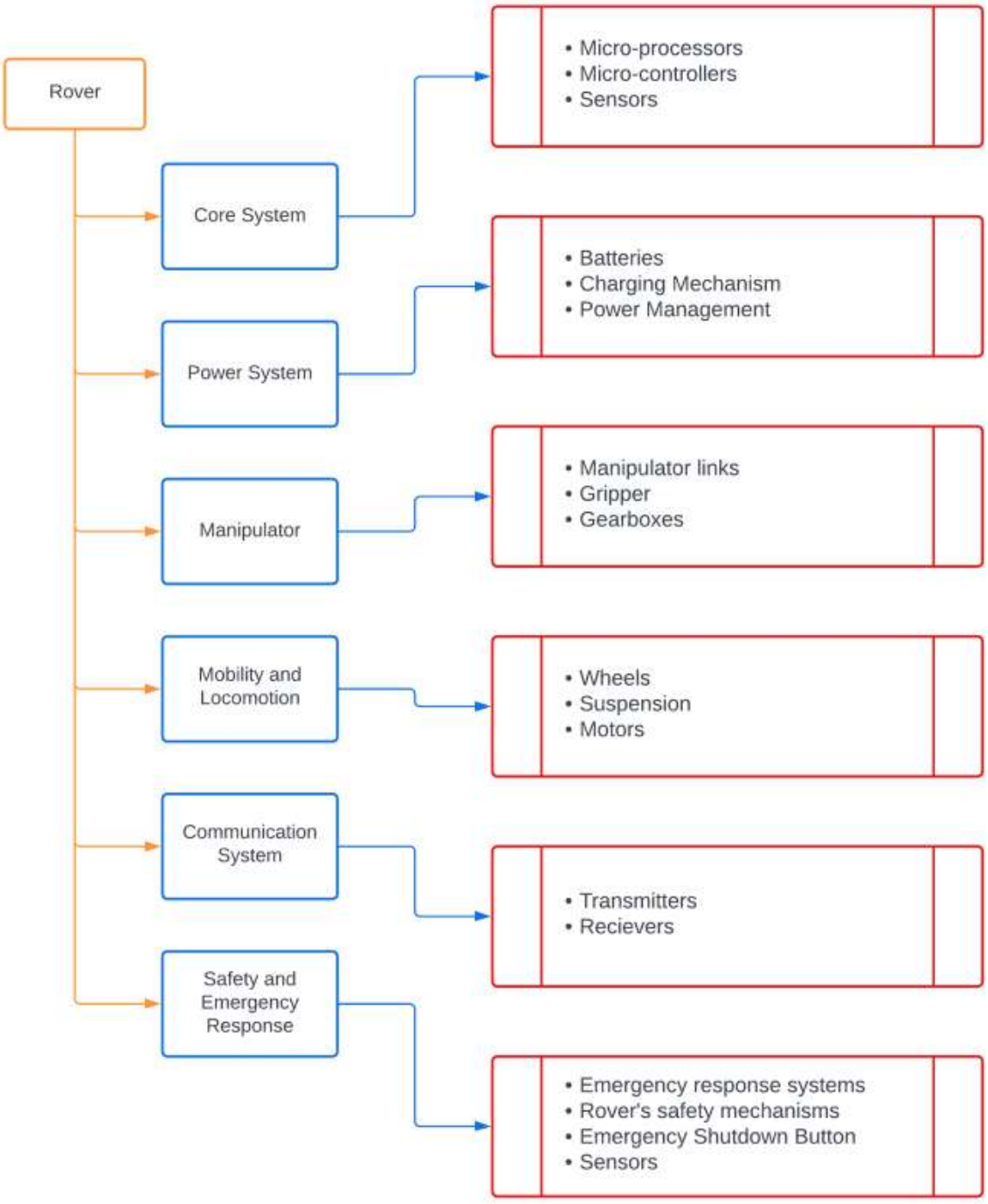


Figure 1.2

The above figure describes various components which constitute a subsystem. The whole team will work on the systems in the given order, from top to bottom.

0.4 Team Division

For smooth development of the rover, the team was divided into the following sub-teams:

Sub-Team	Responsibilities
1. Mechanical Design Team	
• Ground Vehicle Team	Development of physical structure and mobility systems of the rover.
• Manipulator Team	Development of manipulator arm for Object Pick and Place Mission.
2. Electronics and Control System Team	Development of sensory and control mechanisms of the rover.
3. Software Team	Development of algorithms and interfaces for autonomous operations.

01 Rover Mobility

1.1 Sub-System Division

This section will cover the following subsystems:

1. Mechanism
2. Drive Electronics Logic
3. Design Calculations

Software, Sensor System and COTS Component details are provided in a dedicated sections later on.

1.2 Mechanism

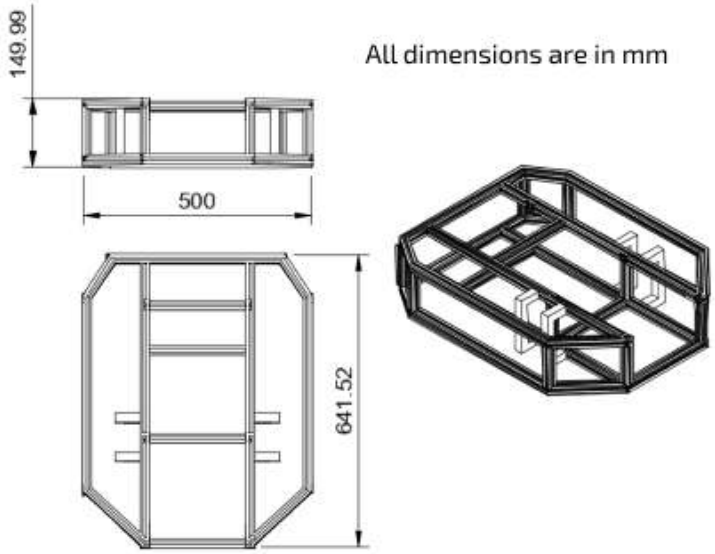
The Mechanisms of our rover are divided into six sub-systems namely,

1. **Chassis**
2. **Suspension**
3. **Wheels**
4. **Differential**

1.2.1 Chassis

Chassis is one of the most fundamental parts of the rover. It serves as the backbone, providing the necessary structure and stability. It supports all the other components of the rover.

The chassis is made up of Aluminium T-Slots, which are welded into form, and encompasses the robotic arm, suspension system and houses the electronic components of the rover. Its main aim is to provide structural integrity to the rover and act as a protective housing for all other subsystems.

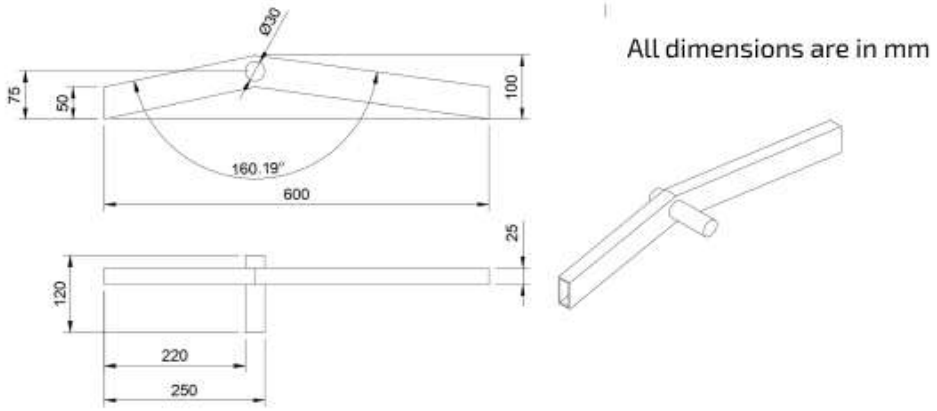


All dimensions are in mm

Dimensions of Chassis
Figure 2.1

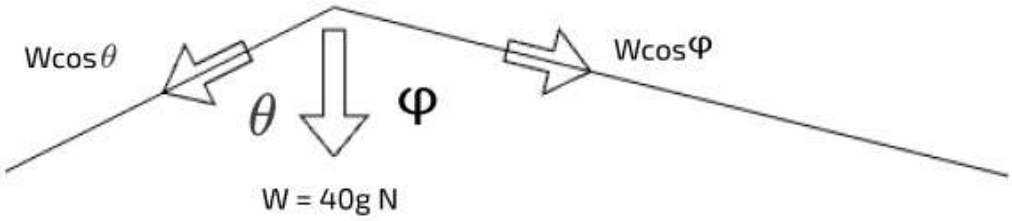
1.2.2 Suspension

The rover's Rocker Suspension system is crucial for moving smoothly on bumpy ground. The Rocker Suspension is made for flexibility and adaptability. These traits let the rover go on lots of types of ground. The rover can move and adjust by itself, which is vital to keep the wheels on the ground and have a good grip on many surfaces.



Dimensions of Suspension
Figure 2.3

The design philosophy behind the suspension was influenced by two important factors, weight distribution and ground clearance. The following diagram explains the logic.

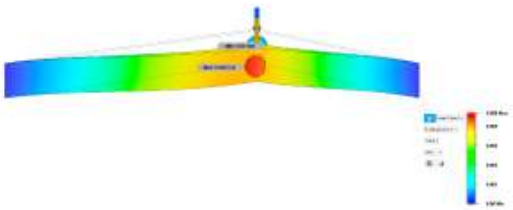


Free Body Diagram of a Suspension Link
Figure 2.4

We have assumed that a single suspension link, in the worst case, would have to bear a loading of 25kgs, assuming the weight of the rover to be 50kgs. Including a factor of safety, we have applied a total weight of 40kgs to one link, making the net total load become 40*g Newtons, where g is acceleration due to gravitational force.

From the laws of Trigonometry, It is clearly noticed that increasing the value of both angles phi and theta would decrease the amount of load that will be transferred down to the wheel subsystem. This is beneficial, as increase in amount of normal load on wheels would increase the friction generated between the wheel grips and the ground surface, which could lead to the rover digging a hole on the sandy surface of the terrain instead of traversing over it.

Thus, the value of theta and phi was selected such that we could maximize the angle between the two legs of the suspension while maintaining our desired ground clearance and axle to axle distance. Aluminium was our material of choice due to it's superior Strength-to- Weight ratio and resistance to abrasion and corrosion compared to other metal.



Displacement Analysis
Figure 2.5.a

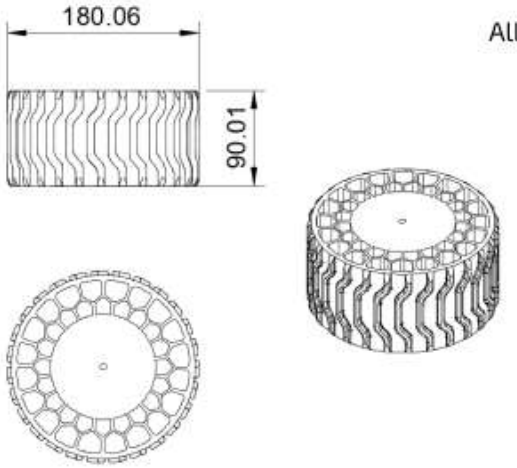
Strain Analysis
Figure 2.5.b

Static Analysis of our design further solidified our belief in our design, as we achieved a Factor of Safety of 15 for our design.

1.2.3 Wheels

The rover's mobility ultimately depends on the wheels of the rover. We have designed the wheels considering the given problem statement and mission in mind.

1. Terrain Material: M-Sand
2. Slope: 15 degrees, stretching 2 metres in length
3. Obstacle to be traversed over: Wooden Cube, side length 150mm
4. Crater to be traversed through: Hemisphere, diameter 200mm

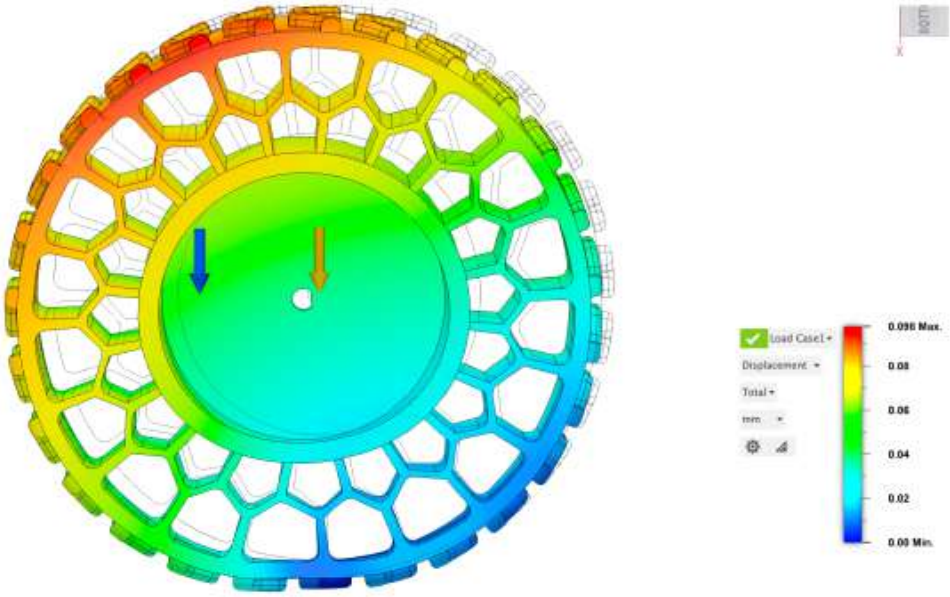


All dimensions are in mm

Dimensions of Suspension
Figure 2.6

Our design acknowledges the dynamic nature of the arena, accommodating variations in terrain gradient, obstacle dimensions, and crater formations. By addressing these challenges, our wheel aims to facilitate smooth rover navigation and operational efficiency. Some of the salient features of our wheel design is as follows:

1. **Honeycomb Structure**: Our wheel features a hexagonal cell honeycomb structure, strategically engineered to distribute weight effectively and enhance flexibility. This structure by design allows controlled deformation of wheel at the point of contact, which increases the area of contact between the wheel and the ground. This results in better gripping while clearing obstacles and better weight distribution.
2. **Aluminium Rim**: Employing aluminium for the rim ensures robustness and structural integrity, vital for traversing rugged terrains. This also acts as a housing unit to the motors and encoders, protecting them from being exposed to dust in the region of exploration
3. **Material**: TPU 95A: Utilising Thermoplastic Polyurethane (TPU 95A) for its flexible and shock-resistant properties. TPU is specifically chosen because of its ability to provide elastic deformation over a large range of temperature and stresses. TPU is also highly abrasion resistant which reduces the wear and tear of the grips in harsh terrain.

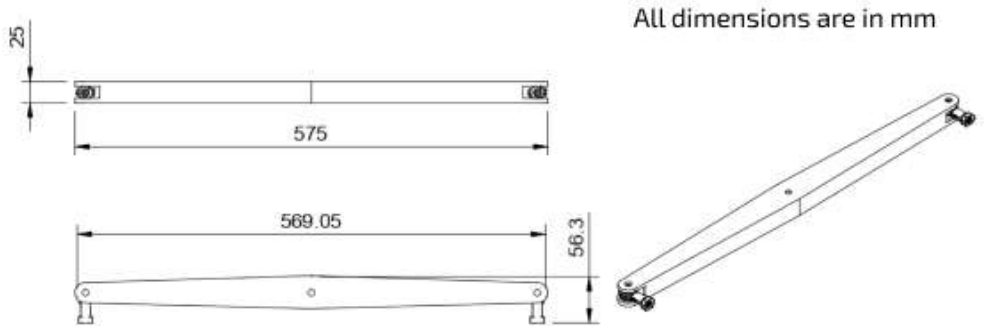


Displacement Analysis of Wheel
Figure 2.7

Through Static Analysis, We have found the minimum factor of safety of 4.89 for our design, with load on the wheel being 150N.

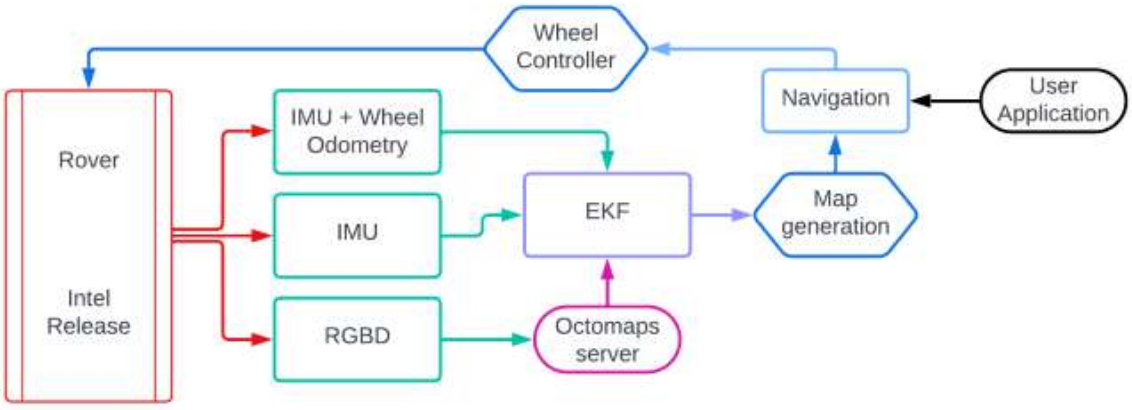
1.2.4 Differential

The differential bar is a pivotal component of the rover, ensuring that the vehicle maintains balance on uneven terrain and the base as a result remains almost horizontal. It operates on a principle of balance and counterbalance, crucial for the rover's ability to navigate challenging landscapes. Its function is to act as a balancing scale, responding to the shifts in the rover's weight as it climbs over rocks or dips into craters, while maintaining an equal radius of curvature of both the suspension links during turning operations. The differential is made out of Aluminium.



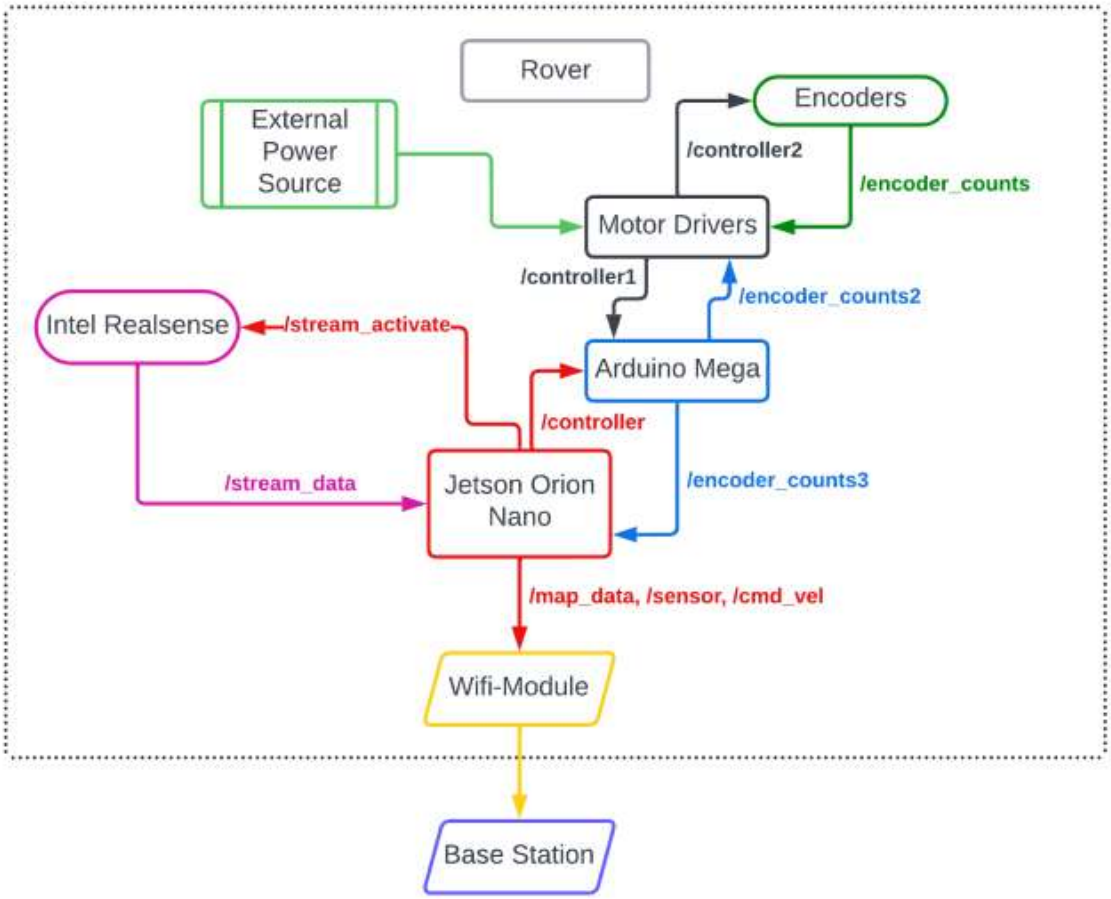
Dimensions of Differential
Figure 2.8

1.3 Drive Electronics Logic



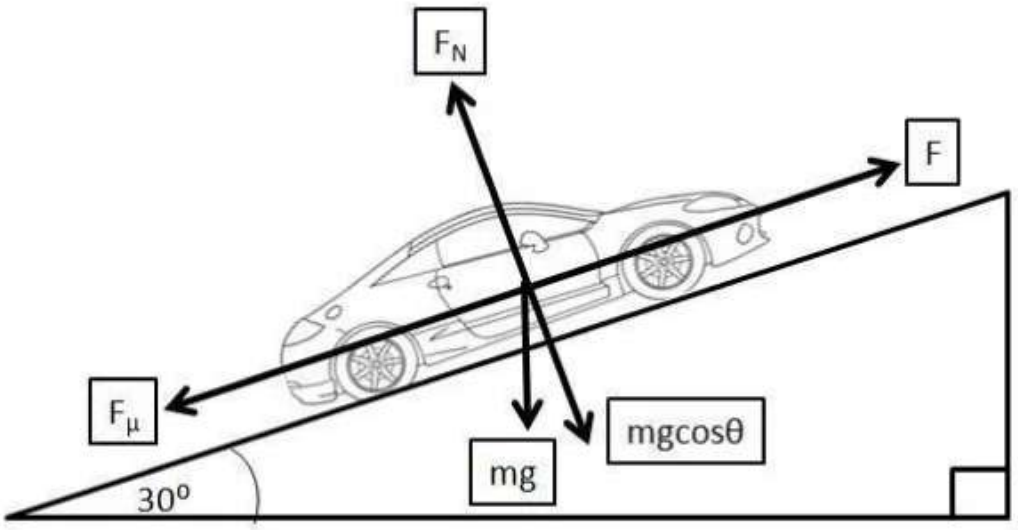
Roving Software Architecture
Figure 2.9

The above image shows the overall software logic of the roving mechanism. The nuances of the mechanism are explained in detail in the navigation section ahead.



Roving Electronics Architecture
Figure 2.10

The above figure displays the interactions between different electronic components of the rover. The interactions and their working is explained in detail in the navigation section ahead.



According to the given problem statement, the rover needs to be capable of travelling over a blanked surface with a slope of minimum 15 degrees. We have assumed the maximum consistent slope of 30 degrees. The mass of the rover is assumed to be 75kgs, 25kgs in excess for factor of safety

Using torque equations of wheels, we get the expression:

Required torque = ((m*r)/4)*(g*sin(s) + a - mu*g*cos(s))
where,

m = Mass of rover = 75kgs
r = Radius of wheel = 0.18m
s = slope of bank = 30 degrees
g = acceleration due to gravity = 9.81 m/s2
mu = Coefficient of static friction = 0.2
a = Desired acceleration on slope = 0.5 m/s2

On plugging the values, we get the required torque to be = 12.5 N-m = 125 kg-cm

From this calculation, we have selected a motor with a torque rating of 140 kg-cm, details regarding which will be provided in the COTS section.

Navigation

4.0 Overview

An overview of the basic computational processes in navigation:
Jetson Nano is going to be our Computational device which hosts ROS (Robot operating system) for the respective nodes and topics to be initiated.
Arduino UNO is going to be used to establish motor encoder power control to maintain the RPM at a desired rate using a Cytron Motor driver.
We are going to use both LIDAR and camera data as an input to the navigation stack and use SLAM tool from ROS for GMapping to produce a desired velocity and direction to reach a target location from our autonomous navigation algorithm.

4.1 Components/Software Used:

- LIDAR
- Camera
- Jetson Nano
- Arduino Uno
- Cytron Motor Driver
- Motor with Encoder

4.2 Software configuration required:

4.2.1 ROS (Robot Operating Systems)

ROS (Robot Operating System) provides libraries and tools to help create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more.

4.2.2 Gazebo Simulator

Gazebo provides a realistic 3D simulation environment where we can simulate our rover in complex scenarios with obstacles, and other objects. It handles interactions like gravity, friction, and inertia in the virtual environment, allowing us to evaluate and test your rover's behaviour without any harm to the physical model of our rover.

4.2.3 SLAM tool for GMapping

SLAM (Simultaneous Localization and Mapping) is a technique use to create maps of unknown environments while simultaneously estimating the position of our rover within the maps.

Gmapping is a ROS package that provides laser-based SLAM. It allows to create a 2D map using laser scans and pose data collected during the movement of our rover.Gmapping follows the Particle Filter approach, which combines data from odometry and scan matching to estimate the rover's incremental movement and localization .

4.2.4 Navigation stack package

The ROS Navigation package serves the purpose of moving a robot from its starting position to a goal position while avoiding collisions with the environment. It provides an implementation of several navigation-related algorithms, making it easier to achieve autonomous navigation for mobile robots.

- Path Planning Algorithm:

Algorithms that compute a collision-free path from the robot's current position to the goal. Here we will be using A*algorithm.

- Localization:

Algorithms that estimate the position within the environment for which we will use Monte Carlo Localization (MCL) and particle filters.

4.2.5 Arduino Library

Arduino libraries play a crucial role in controlling motors after taking inputs from ROS topics by providing pre-built functions and abstractions that simplify the interaction with various hardware components.

4.3 Initialization

Let us go through the step-by-step process of how this happens:

Gazebo 3D CAD tool is used to create and simulate the bot and path planning over local and global maps in virtual environment by using a URDF file which contain all essential information about the model.
Then we set up the ROS packages in Jetson Nano and configure the Navigation stack, which include algorithms for Gmapping (SLAM), path planning, and obstacle avoidance.

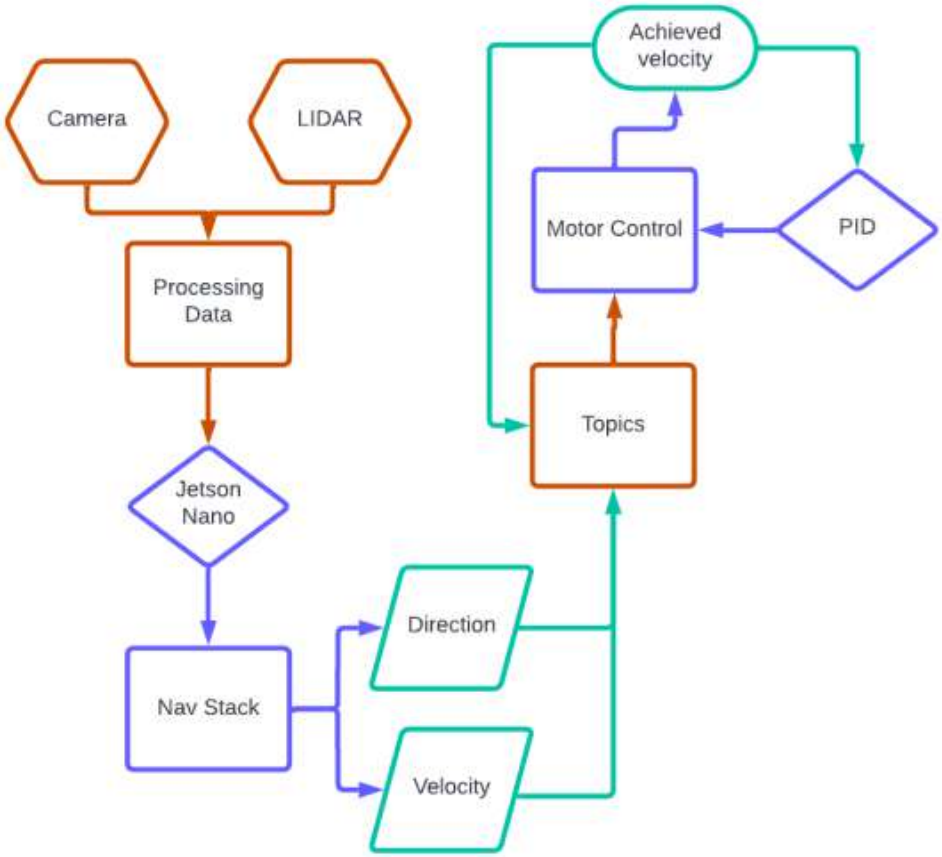
Jetson Nano is going to initiate all the Nodes and Topics required for this process in the beginning and link them with the Sensors and Output devices before the process begins.

The Code for motor control is also uploaded in the Arduino UNO for us to be able to achieve a desired speed without any jerk even when we invert directions.

4.4 Running

After subscribing the data from camera and LIDAR, the mapping node creates map of the environment using SLAM tool (i.e. Gmapping).

- This map is used to localize the robot within the environment and AMCL (localisation algorithm) is used in to estimate rover's position.
- Navigation stack now handles the path planning using global and local maps, virtually adjusting the rover's trajectory and update the planned path to avoid collision with obstacles detected by sensors. The desired velocity and direction are now obtained from the updated path.
- After we obtain the desired velocity and direction, we publish it to /cmd_vel topic to Jetson Nano and the Motor driver with Encoder control node which is uploaded to Arduino is subscribed to /cmd_vel topic to receive the target velocity and direction.
- Arduino node after receiving the target velocity and direction, takes the encoder data and finds the current target velocity and direction and publishes to /new_vel in Jetson Nano.
- Now the PID controller node is subscribed to both /cmd_vel and /new_vel and computes a factor which is linear combination of proportional, differential and integral error to change the velocity gradually and publishes it to /pose topic in Jetson Nano.
- Arduino node is now subscribed /pose to receive the factor from Jetson Nano. Capping the factor and adding it to the power value which is going to be input and configuring the direction into Cytron Motor Driver we achieve the desired speed gradually by repeating the process recursively.



Navigation Control Schematic
Figure 2.1

1.5.2 Obstacle Identification

The problem statement states we have to identify 2 types of obstacles and a tube: 2 cubes of dimensions 150mm*150mm*150mm & 300mm*300mm*300mm. 2 craters of dimensions 200mm and 400mm diameter. A tube of OD 80 mm, L 125 mm
We approached object detection focusing on the utilisation of YOLOv8 for initial detection, calibration with the Intel® RealSense™ D400 Series Dynamic Calibration Tool, subsequent classification of obstacles using perspective projection.

1.5.2.1 Methodology

a Training Dataset

1. **Dataset Curation**: A dataset comprising 1000 images for cube and crater identification was curated and utilised for model training.
2. **Dataset Diversity:** The diversity of the dataset ensured the robustness and adaptability of the subsequent model. It was made sure that the dataset is diversified to prevent any bias in the model.
3. **Bias Prevention**: Early in development, it was observed that YOLOv8 had a tendency to classify the object based on colour rather than shape. To prevent such bias, the following steps were taken:
   a. The dataset comprises a mix of grayscale images, along with RGB images of cylinders painted with multiple colours to avoid colour bias.
   b. The dataset includes multiple other bodies of different shapes surrounding the object of interest to emphasise learning based on shape.

b Object Detection

YOLOv8 was employed for real-time detection of cubes and craters in the Moon terrain. This served as the foundation for subsequent analysis.

c Calibration

Depth data obtained from the RealSense camera facilitated perspective projection for size-based classification of objects. Cubes were differentiated based on their dimensions, while craters were classified using the breadth of their bounding boxes.

d Classification

Depth data obtained from the RealSense camera facilitated perspective projection for size-based classification of objects. Cubes were differentiated based on their dimensions, while craters were classified using the breadth of their bounding boxes.

e Pose Estimation

Pose estimation is performed using OVE6D to calculate all position vectors of the tube. This enables the rover to accurately determine the orientation and position of detected cylinders relative to its surroundings.

1.5.2.2 Softwares and Packages Used

a Python

Programming language used for implementation, training, and analysis. OpenCV Library used for computer vision tasks such as image processing and manipulation.

b PyTorch

Deep learning frameworks used for model training and inference. YOLOv8 Employed for real-time object detection of cubes, cylinders, and craters.

c Intel® RealSense™ D400 Series Dynamic Calibration Tool

Utilised for calibration to determine intrinsic parameters for accurate spatial understanding.

d CUDA

NVIDIA's parallel computing platform and programming model utilised for GPU acceleration to speed up model training and inference processes.

1.5.2.3 Justificatioons

a YOLO (You Only Look Once)

1. **Speed:** YOLO offers real-time detection capabilities, which is a significant advantage for applications that require immediate feedback.
2. **Unified Model Structure:** YOLO uses a single convolutional neural network to replace the disjoint parts in other models like DPM. This unified architecture leads to a faster, more accurate model.
3. **Direct Output:** YOLO directly outputs the position and category of the bounding box, simplifying the detection process.
4. **No Post-Processing Requirement:** Unlike Overfeat, YOLO does not require significant post-processing to produce coherent detections.

a.i COMPARISON WITH OTHER MODELS

Property	YOLO	R-CNN	DPM	Overfeat	MultiGrasp
Speed	Fast (real-time detection)	Slow (more than 40 seconds per image at test time)	Fast (due to unified architecture)	Efficient (but requires significant post-processing)	Fast (only needs to predict a single region)
Model Structure	Straightforward (directly outputs position and category of bounding box)	Complex (each stage must be precisely tuned independently)	Unified (replaces disjoint parts with a single convolutional neural network)	Disjoint (optimises for localisation, not detection performance)	Simple (only needs to predict a single graspable region)
Focus	Shape and colour of object	Region proposals	Static features	Localisation	Region suitable for grasping
Post Processing Requirement	No	Yes	No	Yes	No

b Perspective Projection

1. Accuracy: Perspective projection takes into account the spatial relationship between objects and the camera, leading to high accuracy in depth perception.
2. Realism: It produces realistic images with depth cues such as foreshortening and perspective distortion, mimicking the way human vision perceives depth.
3. Low Computational Intensity: Perspective projection involves direct mapping of a 3D scene onto a 2D image plane, making it less computationally intensive compared to techniques like Photogrammetry.
4. Practicality for Real-Time Applications: Due to its simplicity and efficiency, perspective projection is highly practical for real-time applications.

b.i COMPARISON WITH OTHER ALGORITHMS

Property	Perspective Projection	Direct Measurement Algorithm	Photogrammetry Technique
Accuracy	High (takes into account spatial relationship between objects and camera)	Low (struggles with accurate measurements due to perspective distortion)	High (triangulates object positions based on multiple camera views)
Realism	High (produces realistic images with depth cues such as foreshortening and perspective distortion)	Low (does not consider perspective)	High (estimates sizes and positions in three-dimensional space)
Computation Itensity	Low (direct mapping of 3D scene onto 2D image plane)	Low (direct estimation of object lengths or sizes from image data)	High (requires precise camera calibration and multiple camera views)
Practicality for Real-Time Applications	High (simple and efficient technique)	High (simple direct measurement from image data)	Low (computationally intensive due to requirement of multiple camera views)

03 Manipulator

3.0 Sub-System Division

This section will cover the following subsystems:

1. **Mechanism**
2. **Drive Electronics Logic**
3. **Design Calculations**
4. **Sensor System (discussed in topic no. XX)**
5. **Software**
6. **COTS Equipment Details (discussed in topic no. XX)**

Each point separately covered for:

1. Robotic Arm
2. Gripper

3.1.1 Mechanism: ROBOTIC ARM

The Robotic Arm is the prime component for the pick and place activity, the main mission task. The arm is responsible for taking the gripper to the desired location, for the object to be picked or placed. The arm must be stable but light and have the required workspace.
The design has the following features

1. Degrees of Freedom = 5
2. Parts are named:
    a. Base
    b. Link 1
    c. Link 2
    d. Wrist
3. The first three parts are rigid and actuated by the motor and gearbox combination
4. The wrist is not a fully deployed rigid part, but is just a serial combination of motors and couplers (The motors for wrist are discussed later), which we are calling overall as 'wrist assembly'
5. The joints are as follows
    a. Base joint
        i. It joins the chassis and the main base of the manipulator.
        ii. It gives the whole manipulator a rotary motion about an axis perpendicular to the chassis base.
    b. Shoulder joint
        i. It joins the base of the manipulator and Link 1.
        ii. It gives the shoulder-like motion to the manipulator arm. The whole manipulator gets an overall up – down motion due to this joint.

Elbow joint

i. It joins Link 1 and Link 2.
ii. It gives a motion like the shoulder joint but away from the base of the manipulator arm. It imitates the elbow joint of a human arm joining the hind arm and the forearm.

d. Pitch
   i. There is a pitch joint (axis of revolution parallel to Link 2) at the end of Link 2
e. Roll
   i. There is a roll joint (axis of revolution skew to axis of revolution for pitch) connected to the coupler at the pitch motor

6. Finally, the end effector connects to the coupler attached to the motor actuating the roll joint.
7. Construction
   a. The main links of the robotic arm are made by cutting aluminium sheets into the correct shapes and joining them to the mild steel shafts directly using screws (as can be seen from the CAD screenshots)
   b. The couplers are made using aluminium

3.1.2 Mechanism: GRIPPER

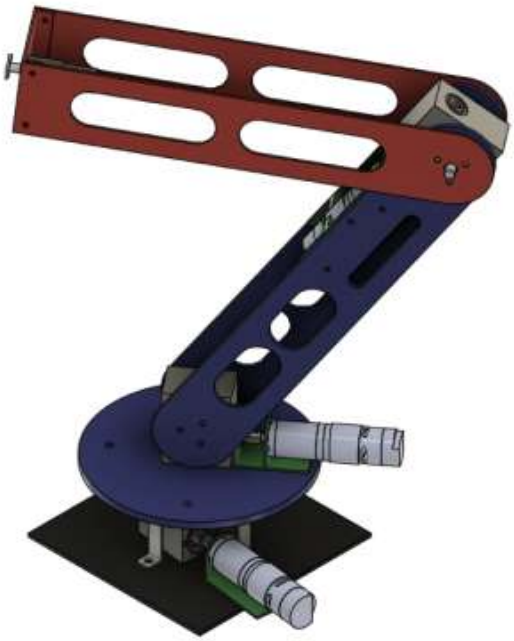
The end effector required for the task at hand is a 'gripper'. We have designed an underactuated adaptive gripper which is actuated by a servo and the extra degree of freedom is kept at check by an elastic band. The gripper is made of 3D printed PLA material.

The design has the following features

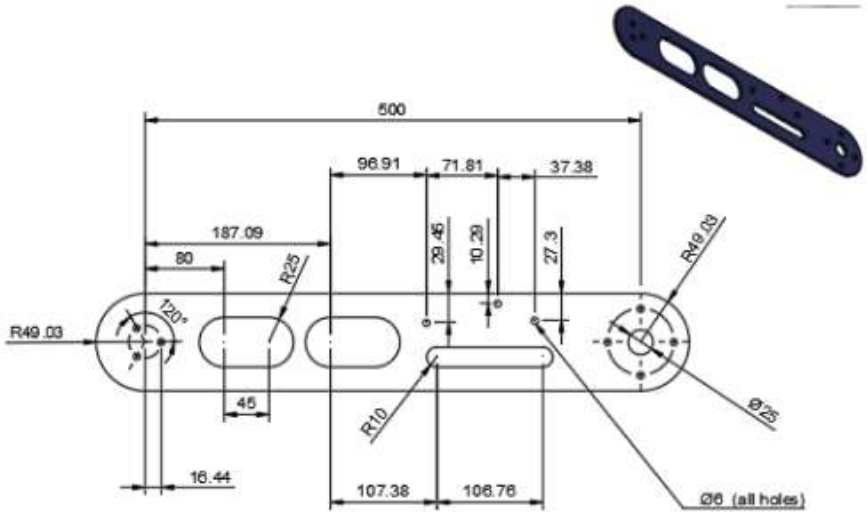
1. Base Module: This houses all the other parts of the gripper viz. Spur gears, Gripper Links, Servo (Mg995), Intel RealSense Camera.
2. Gear Mechanism: The servo rotates a pinion which in turn rotates a gear wheel embedded in one of the gripper links. This wheels also drives another similar gear embedded link.
3. This results in motion of the links in the opposite direction, hence finally driving the 2 jaws.
4. Each half (both constrained to each other) of the gripping mechanism is a 5 bar mechanism with DOF = 2 each, and the net DOF = 3.
5. The servo is utilised to actuate through one of its degrees of freedom. The other 2 are kept in check with the use of 2 elastic bands.
6. The elastic bands provide a false constraint, and hence enables adaptive behaviour
   a. pinching action when load is towards the far end of the jaw.
   b. Curling in due to the elastic bands when load is towards the near end of the jaw.

3.1.3 CAD AND DRAWINGS

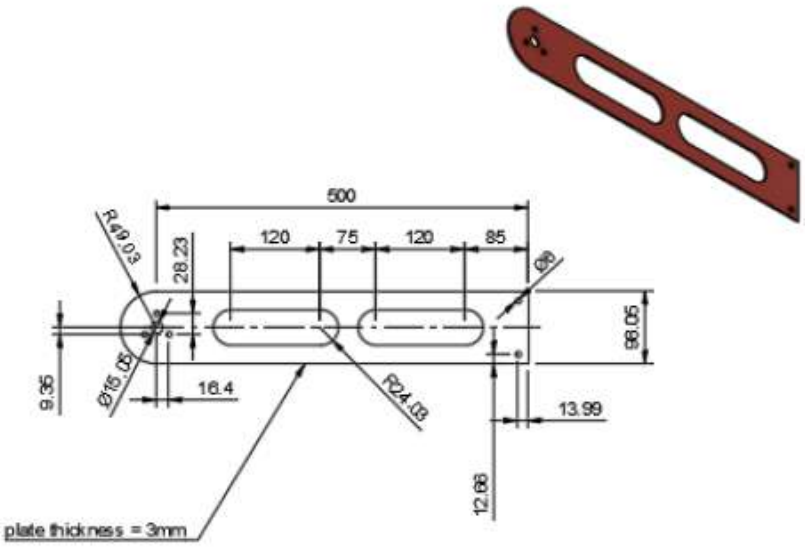
A ROBOTIC ARM



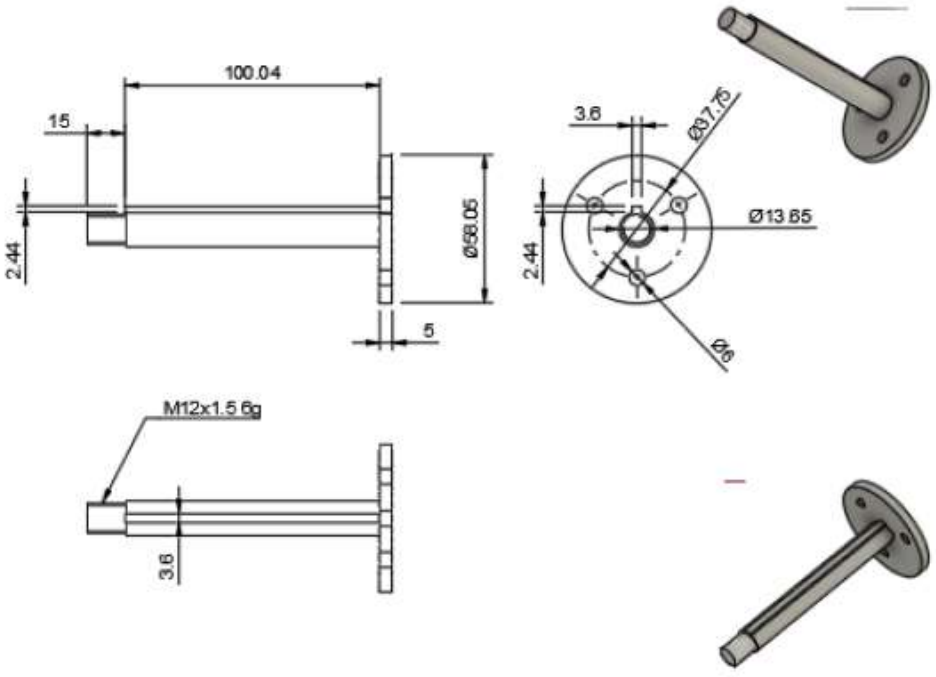
Robotic Arm
Figure 3.1



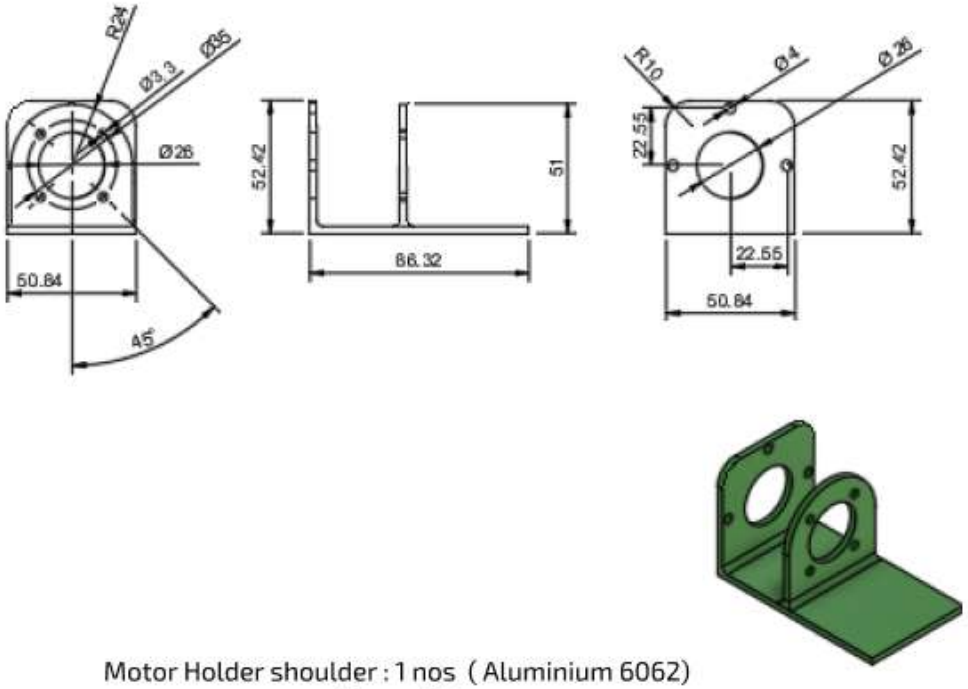
Link 1:2 nos (Aluminium 6062)
Figure 3.2



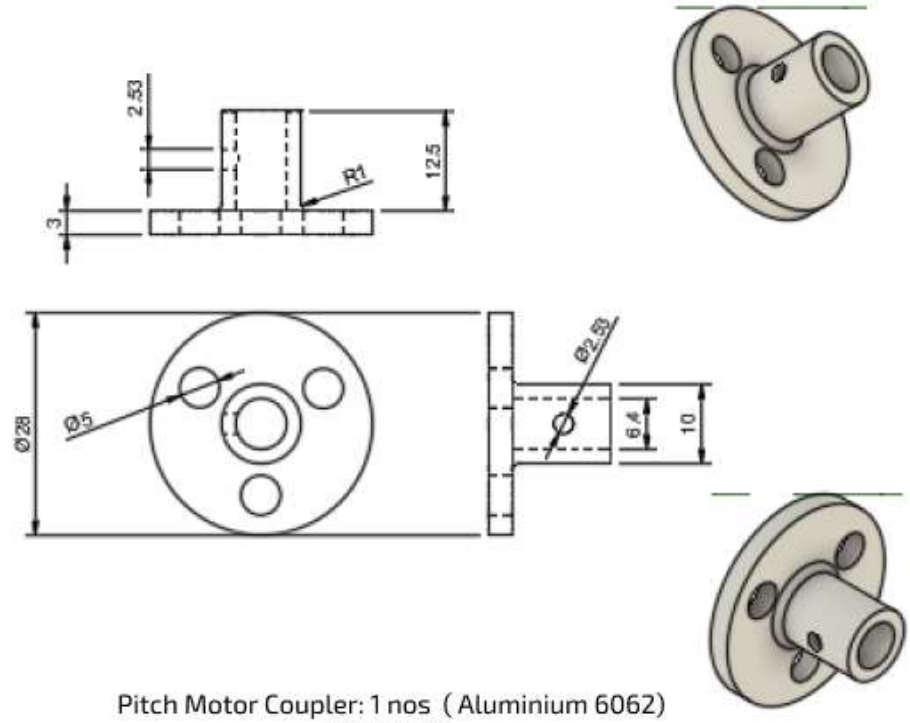
Link 2:2 nos (Aluminium 6062)
Figure 3.3



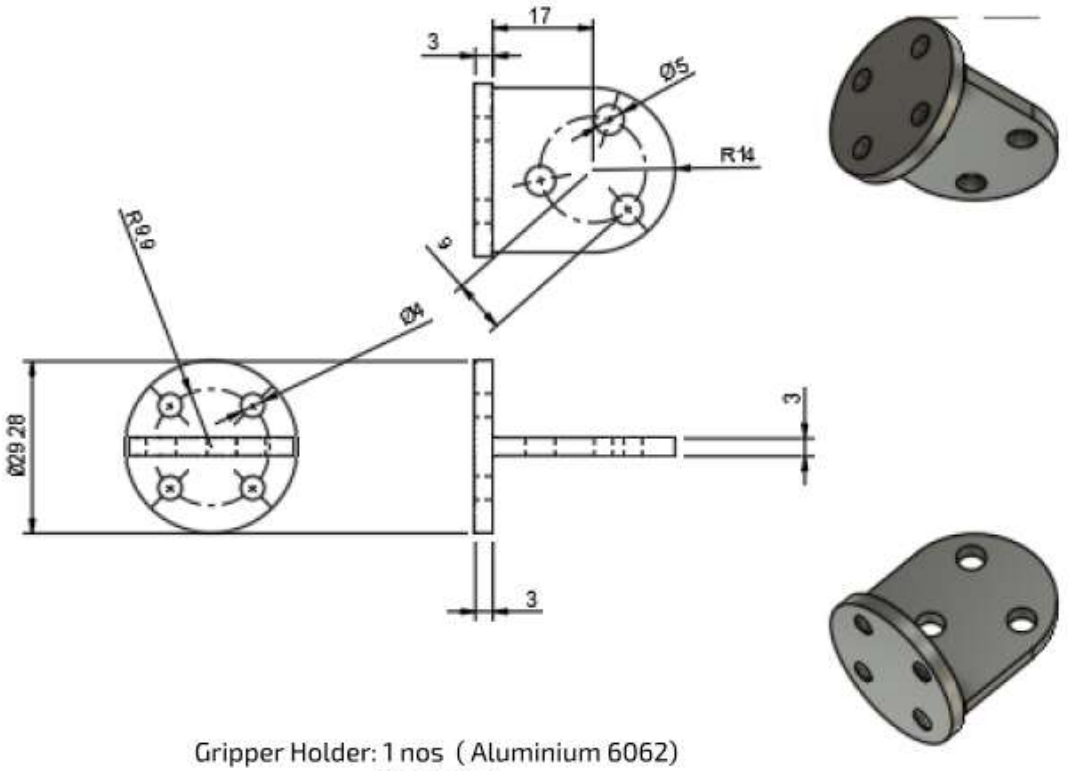
Elbow Shaft : 1 nos (MS)
Figure 3.4



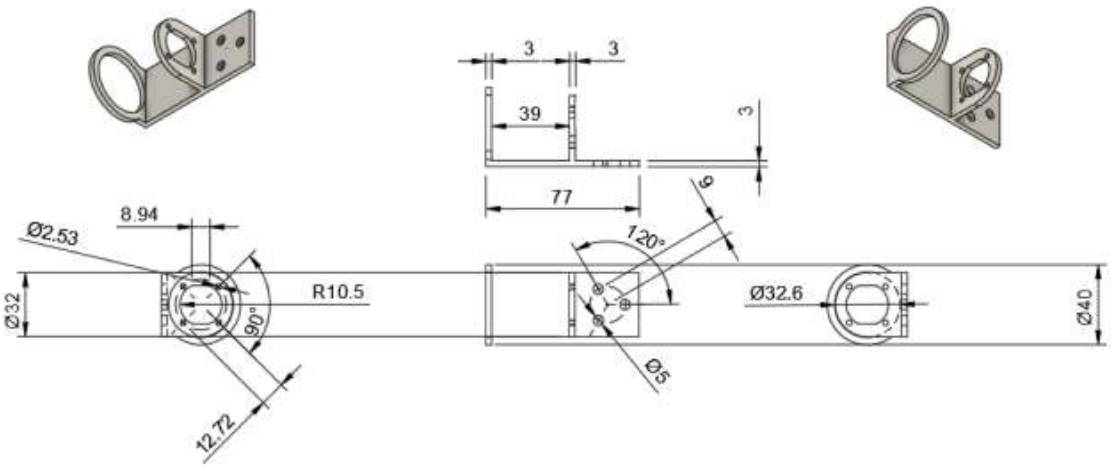
Motor Holder shoulder : 1 nos (Aluminium 6062)
Figure 3.5



Pitch Motor Coupler: 1 nos (Aluminium 6062)
Figure 3.6

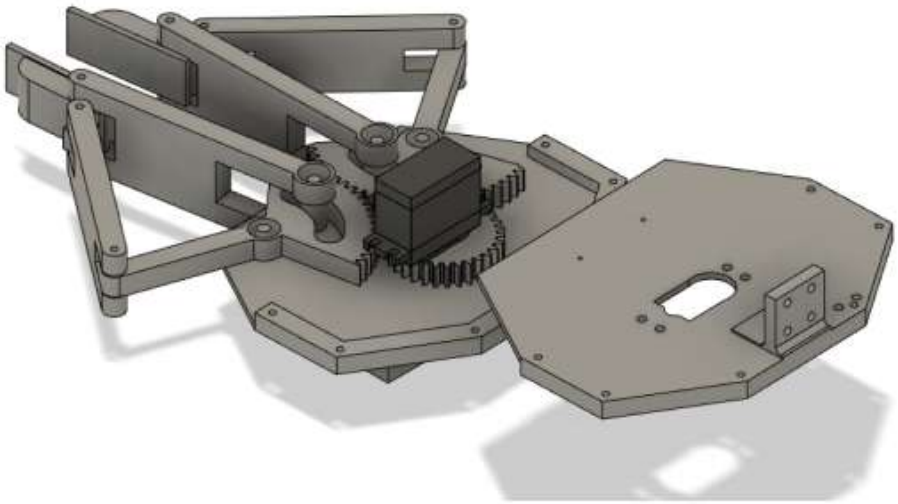


Gripper Holder: 1 nos (Aluminium 6062)
Figure 3.7

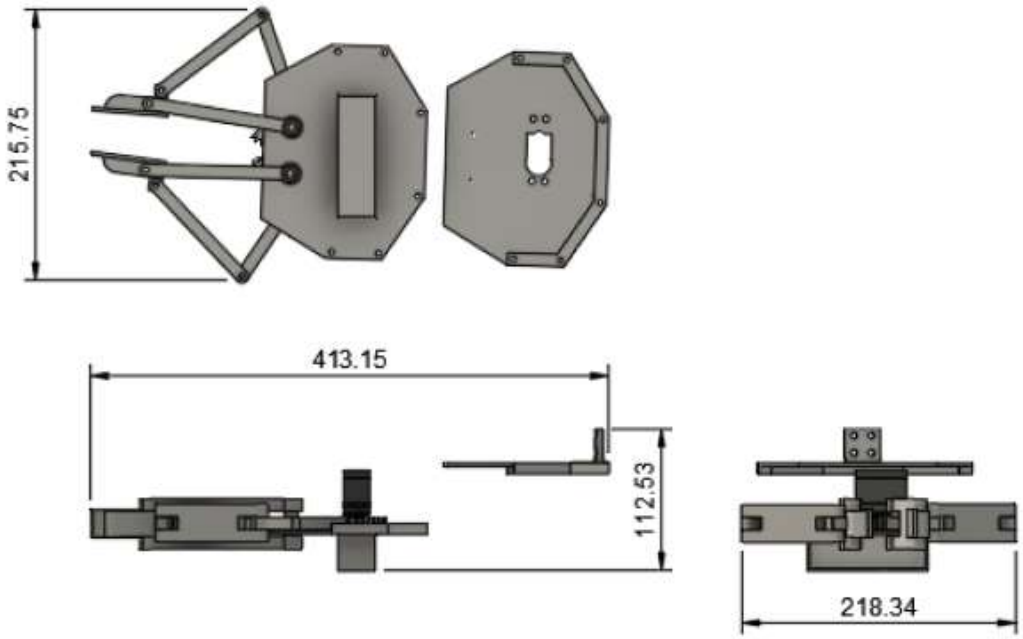


Pitch Motor Holder: 1 nos (Aluminium 6062)
Figure 3.8

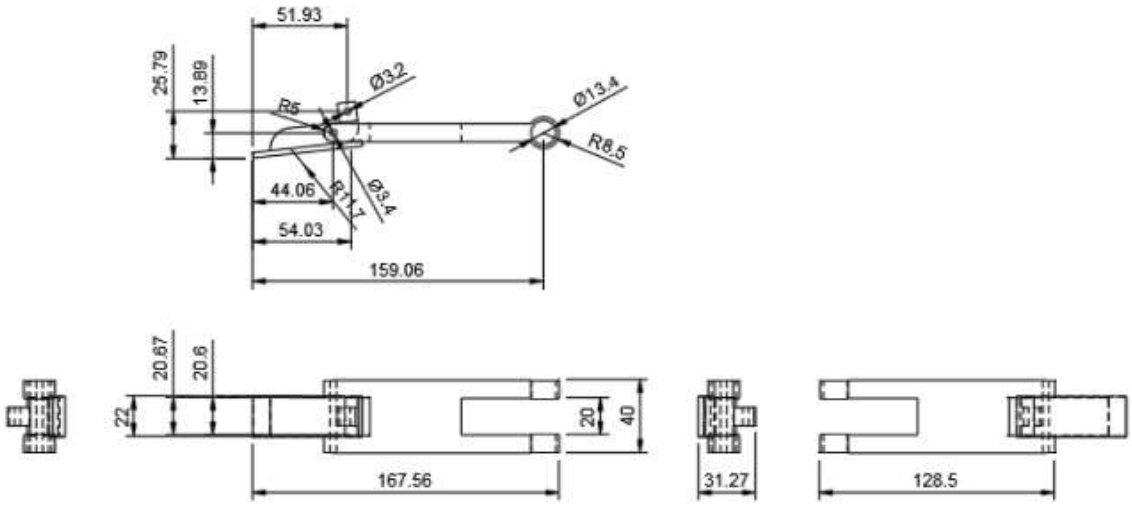
B GRIPPER



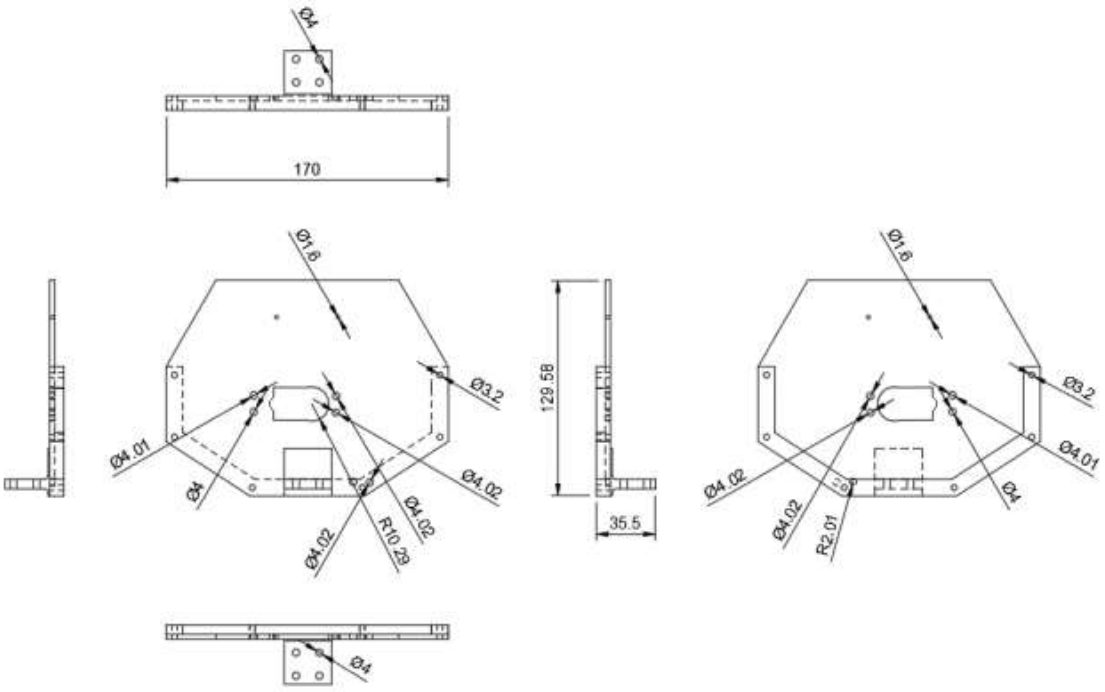
Gripper
Figure 3.8



Gripper Views
Figure 3.9



Technical Drawing 1
Figure 3.9



Technical Drawing 2
Figure 3.10

3.2.0 Drive Electronics

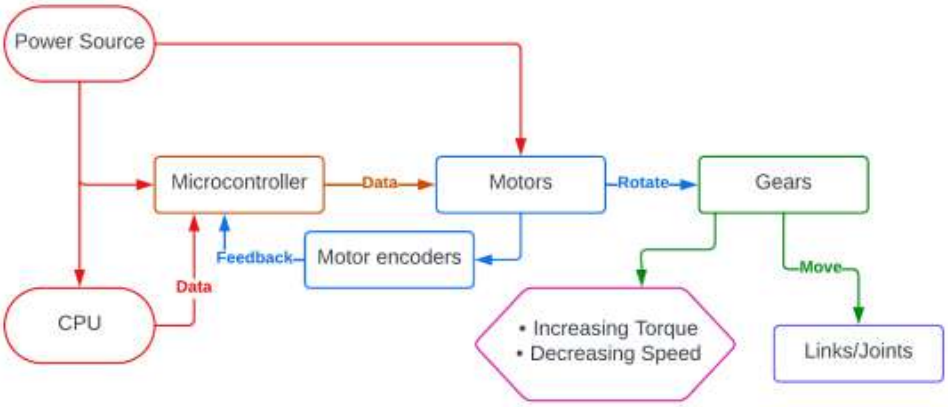


Figure 3.11

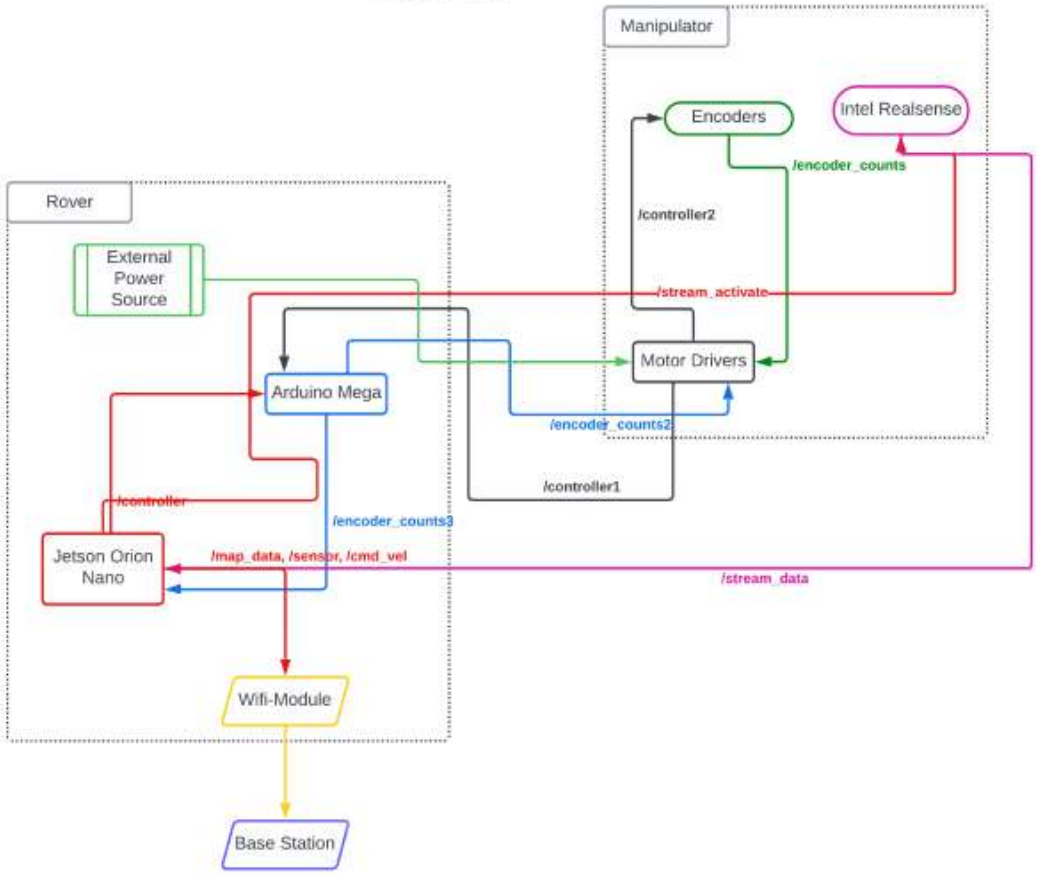


Figure 3.12

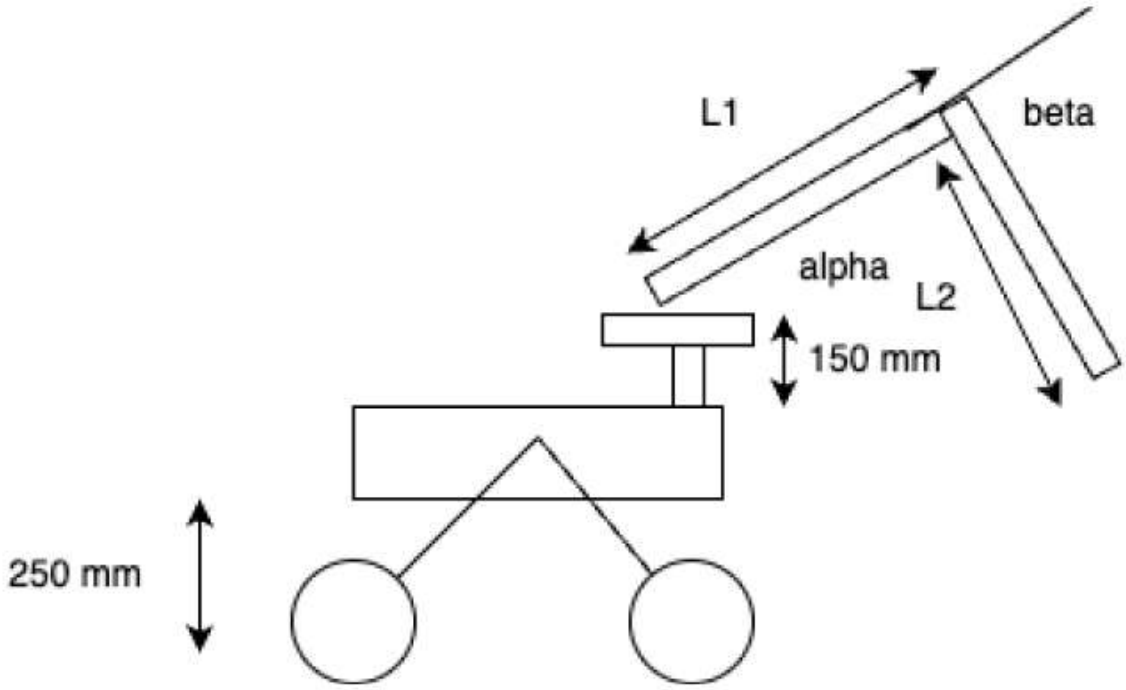
The above figure displays the interactions between different electronic components of the manipulator. The interactions and their working is explained in detail in the manipulator software ahead.

3.3.0 Design Calculations

We have arrived at the following results after calculations:

1. L1 = 45cm
2. L2 = 60cm
3. L2/L1 = 1.33, which is close to the desired value of 1.36

Calculation details are shown below:



Calculation
Figure 3.13

We're given:

- L_1 = length of link1
- L_2 = length of link2
- m_1 = mass of link1
- m_2 = mass of link2
- $m_1 + m_2 = 2$ kg (approximately)
- $m_{\text{shoulder}} = 1.5$ kg (approximately) = mass of motor + mass of gearbox
- $m_{\text{payload}} = 1.5$ kg (approximately) = mass of gripper + mass of payload

We've derived:

$m_1 = \frac{L_1}{L_1+L_2} \times 2$

$m_2 = \frac{L_2}{L_1+L_2} \times 2$

For the final resting position with $\alpha = 120^\circ$ and $\beta = 165^\circ$, we want the center of mass (COM) to be at the base's center. This leads us to set up an equation based on moments about the base:

$$m_1 \left(\frac{L_1 \cdot \cos(60^\circ)}{2}\right) + m_{\text{shoulder}}(L_1 \cdot \cos(60^\circ)) - m_2 \left(-\frac{L_1 \cdot \cos(60^\circ)}{2} + \frac{L_2 \cdot \cos(45^\circ)}{2}\right) - m_{\text{payload}} \left(-\frac{L_1 \cdot \cos(60^\circ)}{2} + \frac{L_2 \cdot \cos(45^\circ)}{2}\right) = 0$$

By substituting values and simplifying, we reach an equation involving L_1 and L_2. Eventually, through algebraic manipulations, we arrive at:

$$5 \cdot L_2^2 + \left(3 - 4 \cdot \sqrt{2} \cdot L_1 \cdot L_2 - 4 \cdot \sqrt{2} \cdot L_1^2\right) = 0$$

We introduce a new variable $x = \frac{L_2}{L_1}$, allowing us to rewrite the equation as:

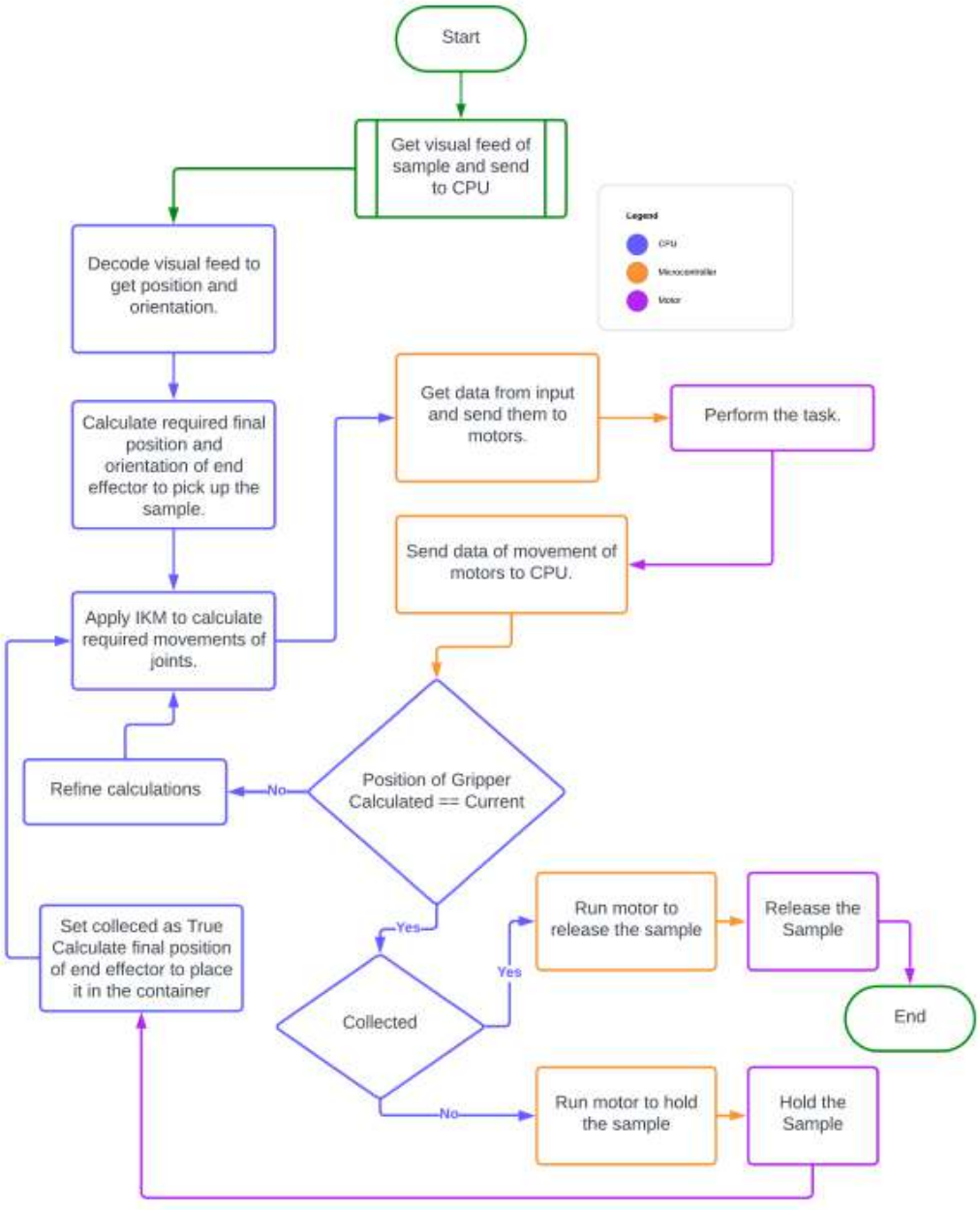
$$5 \cdot x^2 + (3 - 4 \cdot \sqrt{2} \cdot x - 4 \cdot \sqrt{2}) = 0$$

Upon solving this equation, we find x, representing $\frac{L_2}{L_1}$, indicating the ratio of lengths of link 2 to link 1. We obtain a positive value for x, as L_2 must be greater than L_1.

Hence, we conclude that $\frac{L_2}{L_1} = 1.3623$.

1.5.0 SOFTWARE FOR MANIPULATOR MODULE

1.5.1 Software Architecture



1.5.2 OBJECT IDENTIFICATION AND POSE ESTIMATION

The problem statement states we have to identify 2 types of obstacles and a tube: 2 cubes of dimensions 150mm*150mm*150mm & 300mm*300mm*300mm. 2 craters of dimensions 200mm and 400mm diameter. A tube of OD 80 mm, L 125 mm
We approached object detection and pose estimation focusing on the utilisation of YOLOv8 for initial detection, calibration with the Intel® RealSense™ D400 Series Dynamic Calibration Tool, subsequent classification of obstacles using perspective projection and using OVE6D for pose estimation.

1.5.2.1 Methodology

a Training Dataset

1. **Dataset Curation**: A dataset comprising 500 images for cylinder detection and 1000 images for crater and cube identification was curated and utilised for model training.
2. **Dataset Diversity:** The diversity of the dataset ensured the robustness and adaptability of the subsequent model. It was made sure that the dataset is diversified to prevent any bias in the model.
3. **Bias Prevention**: Early in development, it was observed that YOLOv8 had a tendency to classify the object based on colour rather than shape. To prevent such bias, the following steps were taken:
   a. The dataset comprises a mix of grayscale images, along with RGB images of cylinders painted with multiple colours to avoid colour bias.
   b. The dataset includes multiple other bodies of different shapes surrounding the object of interest to emphasise learning based on shape.

b Object Detection

YOLOv8 was employed for real-time detection of cubes, cylinders, and craters in the Moon terrain. This served as the foundation for subsequent analysis.

c Calibration

Depth data obtained from the RealSense camera facilitated perspective projection for size-based classification of objects. Cubes were differentiated based on their dimensions, while craters were classified using the breadth of their bounding boxes.

d Detection

RGB + Depth data obtained from the RealSense camera facilitated perspective projection for size-based classification of objects. Cubes were differentiated based on their dimensions, while craters were classified using the breadth of their bounding boxes.

e Pose Estimation

Pose estimation is performed using OVE6D to calculate all position vectors of the tube. This enables the rover to accurately determine the orientation and position of detected cylinders relative to its surroundings.

1.5.2.2 Softwares and Packages Used

a Python

Programming language used for implementation, training, and analysis. OpenCV Library used for computer vision tasks such as image processing and manipulation.

b PyTorch

Deep learning frameworks used for model training and inference. YOLOv8 Employed for real-time object detection of cubes, cylinders, and craters.

c Intel® RealSense™ D400 Series Dynamic Calibration Tool

Utilised for calibration to determine intrinsic parameters for accurate spatial understanding.

d OVE6D

Used for pose estimation to calculate all position vectors of the cylinders.

e CUDA

NVIDIA's parallel computing platform and programming model utilised for GPU acceleration to speed up model training and inference processes.

1.5.2.3 Justificatioons

a YOLO (You Only Look Once)

1. **Speed:** YOLO offers real-time detection capabilities, which is a significant advantage for applications that require immediate feedback.
2. **Unified Model Structure:** YOLO uses a single convolutional neural network to replace the disjoint parts in other models like DPM. This unified architecture leads to a faster, more accurate model.
3. **Direct Output:** YOLO directly outputs the position and category of the bounding box, simplifying the detection process.
4. **No Post-Processing Requirement:** Unlike Overfeat, YOLO does not require significant post-processing to produce coherent detections.

a.i COMPARISON WITH OTHER MODELS

Property	YOLO	R-CNN	DPM	Overfeat	MultiGrasp
Speed	Fast (real-time detection)	Slow (more than 40 seconds per image at test time)	Fast (due to unified architecture)	Efficient (but requires significant post-processing)	Fast (only needs to predict a single region)
Model Structure	Straightforward (directly outputs position and category of bounding box)	Complex (each stage must be precisely tuned independently)	Unified (replaces disjoint parts with a single convolutional neural network)	Disjoint (optimises for localisation, not detection performance)	Simple (only needs to predict a single graspable region)
Focus	Shape and colour of object	Region proposals	Static features	Localisation	Region suitable for grasping
Post Processing Requirement	No	Yes	No	Yes	No

b Perspective Projection

1. Accuracy: Perspective projection takes into account the spatial relationship between objects and the camera, leading to high accuracy in depth perception.
2. Realism: It produces realistic images with depth cues such as foreshortening and perspective distortion, mimicking the way human vision perceives depth.
3. Low Computational Intensity: Perspective projection involves direct mapping of a 3D scene onto a 2D image plane, making it less computationally intensive compared to techniques like Photogrammetry.
4. Practicality for Real-Time Applications: Due to its simplicity and efficiency, perspective projection is highly practical for real-time applications.

To perform inverse kinematics in 3D, the first thing that we need to calculate is θ, which is the angle around which the entire robotic arm revolves. This is the extra degree of freedom added to the hip joint

b.i COMPARISON WITH OTHER ALGORITHMS

Property	Perspective Projection	Direct Measurement Algorithm	Photogrammetry Technique
Accuracy	High (takes into account spatial relationship between objects and camera)	Low (struggles with accurate measurements due to perspective distortion)	High (triangulates object positions based on multiple camera views)
Realism	High (produces realistic images with depth cues such as foreshortening and perspective distortion)	Low (does not consider perspective)	High (estimates sizes and positions in three-dimensional space)
Computation Itensity	Low (direct mapping of 3D scene onto 2D image plane)	Low (direct estimation of object lengths or sizes from image data)	High (requires precise camera calibration and multiple camera views)
Practicality for Real-Time Applications	High (simple and efficient technique)	High (simple direct measurement from image data)	Low (computationally intensive due to requirement of multiple camera views)

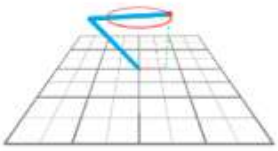
c OVE6D

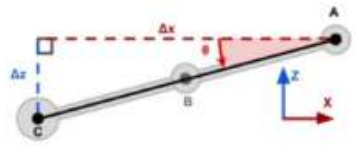
1. Depth-Based: OVE6D is a depth-based object viewpoint encoder. This allows it to capture more detailed and accurate information about the object's position and orientation in space compared to methods that only use RGB data.
2. Feature Vector: OVE6D captures the object viewpoint into a feature vector. This compact representation of the object's viewpoint can be easily processed and used for further analysis.
3. Invariance and Sensitivity: The encoded representations are trained to be invariant to the in-plane rotation around the camera optical axis, but to be sensitive to the camera viewpoint. This allows OVE6D to accurately estimate the object's pose even when the camera's viewpoint changes.
4. Efficient Inference: At the inference time, OVE6D first utilises the viewpoint encodings to determine the camera viewpoint, and subsequently estimates the remaining pose components. This two-step process makes the inference more efficient and accurate.

c.i COMPARISON WITH OTHER ALGORITHMS

Property	OVE6D	Pose Estimation from RGB data	Pose Estimation from Depth data	Pose Estimation from RGB-D data
Data Type	Depth-based	RGB-based	Depth-only	RGB and Depth
Method	Captures the object viewpoint into a feature vector	Establishes sparse or dense 2D-3D correspondences	Performs 6D pose regression from the point cloud segments	Initial pose estimation based on RGB images and then further refine with depth images
Invariance/ Sensitivity	Invariant to the in-plane rotation around the camera optical axis, but sensitive to the camera viewpoint	N/A	N/A	N/A

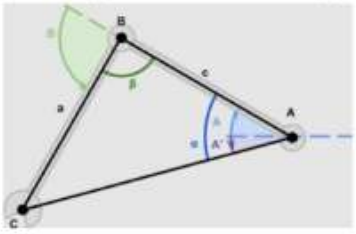
1.5.2.4 Inverse Kinematics Model

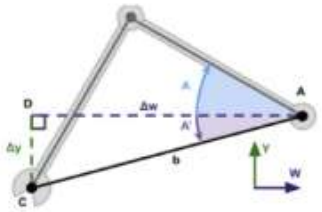




$$\theta = \tan^{-1}\left(\frac{\Delta z}{\Delta x}\right) = \tan^{-1}\left(\frac{C_Z - C_Z}{C_X - A_X}\right)$$

A is the angle which controls the first joint and B is the angle which controls the second joint





$$\theta = \tan^{-1}\left(\frac{C_Z - C_Z}{C_X - A_X}\right)$$

$$A = \cos^{-1}\left(\frac{b^2 + c^2 - a^2}{2bc}\right) + \tan^{-1}\left(\frac{C_Y - D_Y}{\sqrt{(C_X - A_X)^2 + (C_Z - A_Z)^2}}\right)$$

$$B = \pi - \cos^{-1}\left(\frac{a^2 + c^2 - b^2}{2ac}\right)$$

Sensor System

3.0 Sub-System Division

This section will cover the following subsystems:

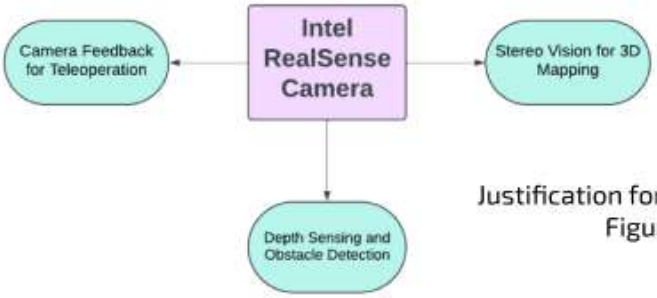
1. **Intel RealSense camera D435i**
2. **Sensor Data Acquisition**
3. **Data Fusion and State Estimation**
4. **Data Processing and Distribution**
5. **3D Mapping and Localization**

COTS components are discussed at a later section

3.1 Intel RealSense camera D435i

The Intel RealSense camera on our rover is like its eyes, helping it navigate through tricky terrain and obstacles with ease. It does this by using following:

1. **Depth Sensing and Obstacle Detection**
   a. Structured Light: It projects a pattern onto the ground and measures how it gets all wiggly when something blocks it. That way, it can create a map of how far away everything is.
   b. Time-of-Flight (ToF): It sends out a quick flash of light and measures how long it takes to bounce back, which helps it figure out how far away stuff is.
   c. This comes in handy for
      i. Avoiding obstacles (like rocks, craters, or uneven ground)
      ii. Changing directions quickly if it senses something in its way.
      iii. Navigating safely even in tricky spots.
2. **Stereo Vision for 3D Mapping:** The camera uses RGBD sensors to recognise the colours and measure the depth of the objects it sees. This helps with:
   a. Understanding the lay of the land (like hills and valleys).
   b. Planning the best route to avoid obstacles and steep slopes.
   c. Knowing where it is in relation to other things.
3. **Camera Feedback for Teleoperation:** When someone's controlling the rover remotely, they can see what the camera sees in real-time. This helps them:
   a. Drive it more precisely
   b. Avoid collisions with obstacles.
   c. Make smart decisions based on what they see.



Justification for IntelRealSense
Figure 4.1

3.2 Sensor Data Acquisition - IMU and Motor Encoders

3.2.1 INERTIAL MEASUREMENT UNIT (IMU)

IMU is like a rover's inner ear and gyroscope all in one. This sensor constantly measures the robot's acceleration along its axes (forward/backward, side-to-side, up/down) and its angular velocity (roll, pitch, yaw). This raw data helps determine the robot's tilt, rotation speed, and potential changes in direction. It has gyroscopes that help it keep track of how fast it's spinning around in a circle, like when it's doing a pirouette. This raw data lets the robot know if it's tipped over, how fast it's spinning, and whether it's about to turn left or right.

3.2.2 Wheel and Manipulator Joints Odometry - ENCODERS

Sensors on each wheel in case of mobility system and each joint in case of manipulator track how far they have rotated (in encoder counts).

1. By measuring the distance each wheel travels and considering the robot's geometry (wheelbase - distance between wheels), we can estimate the robot's displacement (movement) since the last measurement.
2. However, wheel slippage on some surfaces can cause these calculations to go a bit off.

1. By measuring the angle rotated in each joint and considering the manipulator geometry, we can estimate the change in pose of the manipulator at each time interval. With the knowledge of the initial pose, we can get the current pose of the manipulator
2. However, gear backlash error may cause wrong estimation; and the errors may get accumulated over time.

Hence, frequent calibration is important

3.3 Data Fusion and State Estimation

Extended Kalman Filter (EKF)

This algorithm is like the brain of the rover, taking in all the messy data from its sensors and deriving results out of it. It has a model of how the rover moves and how the different sensors work, so it can figure out what is happening. It is like a traffic controller and GPS navigator all in one. It uses math to predict where the rover thinks it should be based on where it's been and what it's doing, then compares that to what the sensors are telling it. When there's a difference, it makes adjustments and keeps going. The EKF is good at cutting through the noise and making sure the rover knows where it is and where it's headed. It's especially helpful in situations where the sensors are giving conflicting info or the terrain is changing a lot.

3.4 Data Processing and Distribution

Intel Release

1. Data Publishing: The processed and improved robot state data from the EKF is published on a communication channel (e.g., message queue) within the robot's internal system. This allows other modules, like the motor control unit, to access the robot's most recent location and orientation for navigation purposes.
2. Data Transmission: Also, "Intel Release" signifies transmitting the robot's state data to an external server for further processing, visualisation, or logging. This allows for centralised monitoring and control of multiple robots or facilitates tasks requiring more computational power than available onboard the robot.

3.5 3D Mapping and Localisation

OCTOMAP

The Octomap plays a critical role in the robot navigation system by constru cting a 3D representation of the environment around the robot. This map isn't a simple blueprint; it's a probabilistic map, meaning it encodes the likelihood of space being occupied by an object within a 3D grid structure. This is how Octomap works:

1. Building the Grid:

Voxels are the foundation of the Octomap. A voxel (volumetric pixel) represents a small volume of space in the environment. Smaller voxels provide higher resolution but require more memory.

2. Assigning Probabilities:

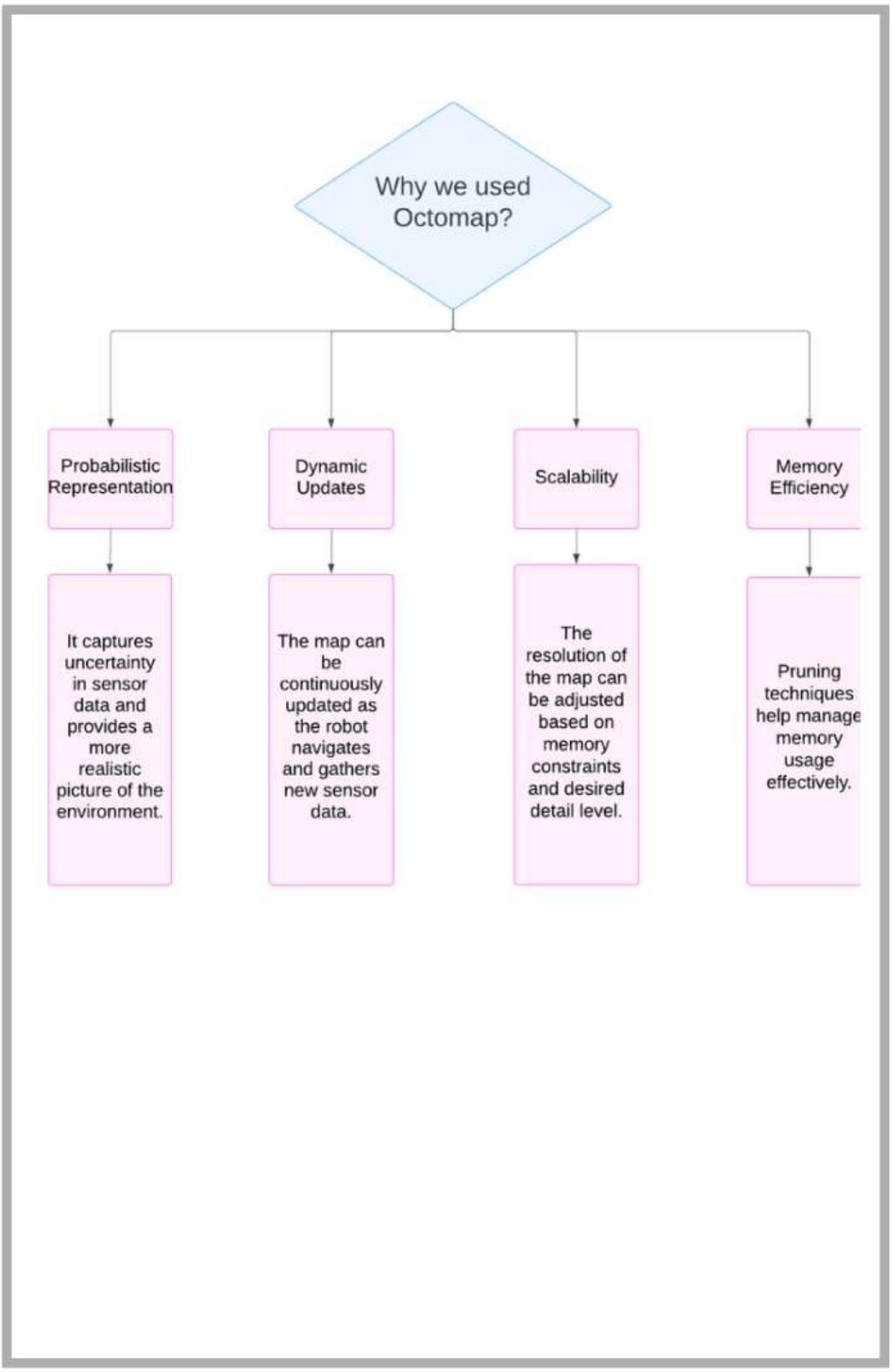
The core of Octomap lies in the probability value assigned to each voxel. This value represents the likelihood that the space occupied by that voxel is filled with an object. The value ranges from 0 (completely free) to 1 (completely occupied), with values in between indicating varying degrees of uncertainty.

3. Raycasting and Refining:

A technique called ray casting is used to refine the map probabilities as the robot moves and gathers new sensor data. It shoots rays out from the robot's position in different directions. When a ray intersects an object (based on the sensor data), the corresponding voxels along the ray path have their occupancy probabilities increased, reflecting the higher likelihood of those spaces being occupied. Conversely, if the ray doesn't hit anything, the probabilities of the free space voxels along the ray path are slightly increased, while those further away might be decreased to account for potential uncertainties.

4. Memory management:

As the robot explores its environment, the Octomap grows. To maintain efficiency and avoid memory overload, techniques like pruning are employed. Pruning removes voxels with very low occupancy probabilities (indicating they are likely free space) from the map, reducing the computational burden.


Why we used Octomap?
Probabilistic Representation
Dynamic Updates
Scalability
Memory Efficiency
It captures uncertainty in sensor data and provides a more realistic picture of the environment.
The map can be continuously updated as the robot navigates and gathers new sensor data.
The resolution of the map can be adjusted based on memory constraints and desired detail level.
Pruning techniques help manage memory usage effectively.

Emergency Stop

The E-stop system provides a rapid response mechanism to halt potentially dangerous equipment while maintaining the functionality of the control and diagnostic components. This selective power distribution is a critical aspect of modern safety systems, allowing for a controlled and informed approach to managing emergencies and resuming operations.

Activation

An E-stop is initiated upon perceiving an emergency situation such as malfunctioning or posing a danger. This can be done manually by an operator or through an automated system response.

Selective Power Cut-off

Upon activation, the E-stop switch interrupts the power supply to the actuators only. This design is intentional to instantly halt all potentially dangerous mechanical movements during emergencies.

System Response

Despite the power-down of actuators, other parts of the system including ARDUINO units, CYTRON motor drivers, and JETSON NANO remain functional. This allows for real-time monitoring and diagnostics, which are crucial in understanding the cause of the emergency and ensuring safe restart of operations.

Safety Check

With the actuators stopped, operators can safely assess and address the emergency. This might involve:

- Repairing equipment
- Clearing obstructions
- Taking other necessary safety measures

Reset Procedure

Once the situation is under control and it's safe to proceed, the E-stop switch can be reset by following these steps:

- Locate the E-stop switch.
- Manually press the E-stop switch to reset it.

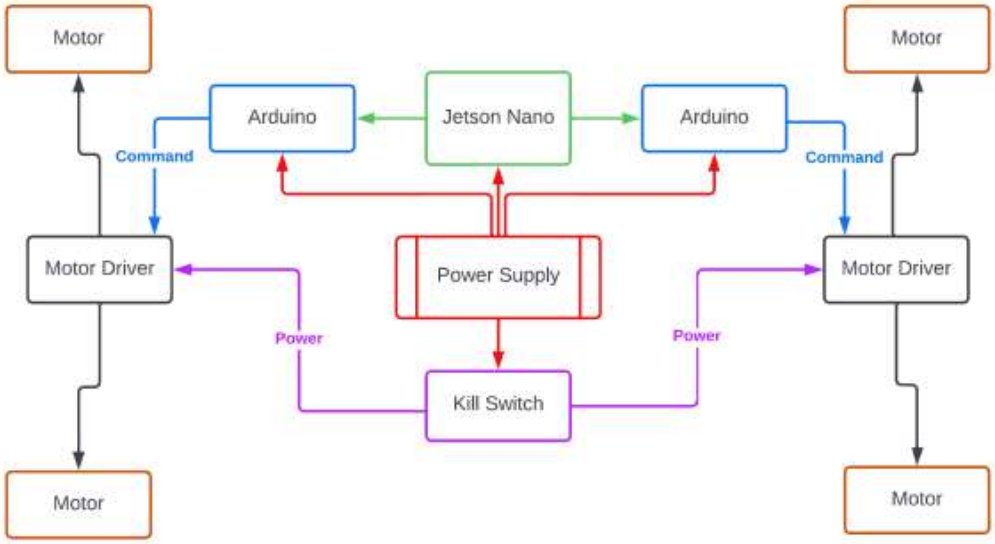
This step is essential to prevent the accidental reactivation of the actuators and to ensure that the system can only be restarted intentionally.

Resume Operation

After the E-stop switch is reset and the system is deemed safe, operation can be resumed by following these steps:

- Verify that all personnel are clear of the equipment.
- Restore power to the actuators.
- Monitor the system for any abnormalities or malfunctions.

The continuation of power to the control units during the emergency stop ensures that they are ready to resume their functions immediately.



COTS Components

Onboard Processor

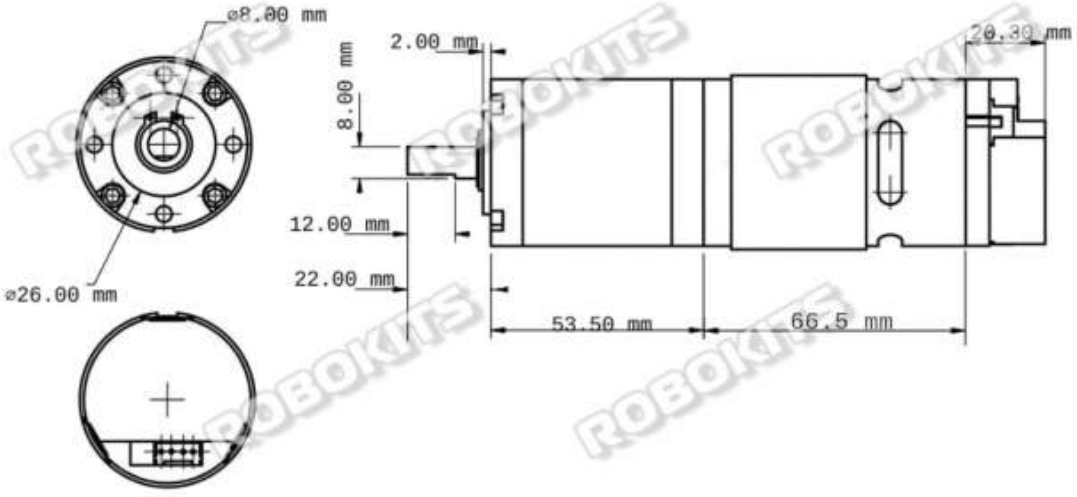
PARAMETER	SPECIFICATIONS
Name of Component	Nvidia Jetson Orin Nano.
GPU	Ampere, 1024 CUDA Cores, 48 Tensor Cores, 625 MHz
CPU SPEC int 2k6 SPEC int rage	6X A78, 1.5 GHz 25 106
Power	15 W
Memory	8GB, 68 GB/s

Onboard Microcontroller

PARAMETER	SPECIFICATIONS
Name of Component	Arduino Mega
Microcontroller	ATmega2560
Input Voltage	7-12V
Flash Memory	256 KB of which 8 KB used by bootloader
Clock Speed	16 MHz

Motors (Roving and Manipulators)

PARAMETER	SPECIFICATIONS
Name of Component	Mega Torque DC Planetary Geared Encoder Servo Motor
Rated Voltage	18V
Rated Speed	200 RPM
Rated Current	18.7 A
Encoder PPR	**13**
Stall Current	130 A
Base Motor CPR	52
Breaking Torque	200 kg-cm



Gearbox

PARAMETER	SPECIFICATIONS
Name of Component	FW-30-10-63B5
Ratio	1:10
Material	Aluminium
Weight	1.2 kgs

08 Test Plan

8.1 Test Plan for Manipulator

8.1.1 Range of Motion - DONE

TEST	OBSERVED RESULT	FAILURES
Set the arm position manually and check for the maximum and minimum range the arm can reach	The observed range of motion must match the calculated range of motion.	Range is less than calculated range. Faults in motors, joints, or links. Links &/or gears break near the end points due to excess torque.

CORRECTION	EMERGENCY
• Correct mechanical faults • Recalculate the range and test again. • Improve gears, and revise link lengths and torque calculations.	The IKM program is to be made such that the extreme angular values of the joints act as global valued parameters so that in case the problem happens onsite; one can solve it by changing the global parameter value and recompile the script.

8.1.2 Vibration - DONE

TEST	OBSERVED RESULT	FAILURES
Hold an object with the gripper and move the manipulator with the rover in jerky motions. Check for unnecessary vibrations	Vibrations are in range of acceptable limits. The held object does not slip during the motion.	Vibrations during the motion of arm causing the joints to loosen. Vibrations cause the held object to slip.

CORRECTION	EMERGENCY
• Revise the link lengths and slightly change the link weights to remove any resonant vibrations. • Establish a safe speed of movements for each joint.	Lower the max possible RPM of the motor by controlling the power using software. Replace any damaged part which is causing the vibration.

8.1.3 Load lift - DONE

TEST	OBSERVED RESULT	FAILURES
Start with a lightweight load and increase the weight of the load to check the max. weight that can be lifted by the gripper without slipping.	The maximum load must exceed the given mission statement payload mass of 200gm	Unable to pick up required load. Structure fails before carrying the maximum specified load

CORRECTION	EMERGENCY
• Revise gear ratio and/or motor. • Change gripper material for better friction.	Emergency Intervention not possible. Failure will lead to failure in mission.

8.1.4 Torque - DONE

TEST	OBSERVED RESULT	FAILURES
We perform the Load lift test with the 5 DOF motors active and will try to move the load from one place to another	The manipulator must pass higher external torques than the required limits.	Links unable to move due to weight. Structure fails during the process.

CORRECTION	EMERGENCY
• Run software simulations for required torque before manufacturing. • Revise gear ratio and/or motor.	Not applicable.

8.1.5 Catastrophic Failure Prevention - Done

TEST	OBSERVED RESULT	FAILURES
Most expensive test. We will test the strength and integrity of the manipulator links and joints.	Make sure all the weak points are rectified before deploying in the arena	Any part of the manipulator breaks during testing.

CORRECTION	EMERGENCY
• We will run software tests before manufacturing to check if the structure and materials can handle the necessary stresses.	Not applicable.

8.1.6 Force Sensing (Tactile Feedback)

TEST	EXPECTED OBSERVATION	FAILURES
Check the force transducer values when the gripper is in maximum closed position	We will make sure the observed values match the calibrated values after repeated usage of the gripper	Due to sensor malfunction, calibration error and wear and tear over time; gripper may experience inaccuracies in force testing leading to unreliable tactical feedback during operation.

CORRECTION	EMERGENCY
• Regularly calibrate and maintain force sensing components to ensure accuracy.	Check for loose connections, damaged sensor, or calibration discrepancies. Have spare sensors available for quick replacement. If issue persists, consider gripper force manually based on visual feedback or relying on alternating gripping strategies.

8.1.7 Force Sensing (Tactile Feedback)

TEST	EXPECTED OBSERVATION	FAILURES
Check the force transducer values when the gripper is in maximum closed position	We will make sure the observed values match the calibrated values after repeated usage of the gripper	Due to sensor malfunction, calibration error and wear and tear over time; gripper may experience inaccuracies in force testing leading to unreliable tactical feedback during operation.

CORRECTION	EMERGENCY
• Regularly calibrate and maintain force sensing components to ensure accuracy.	Check for loose connections, damaged sensor, or calibration discrepancies. Have spare sensors available for quick replacement. If issue persists, consider gripper force manually based on visual feedback or relying on alternating gripping strategies.

8.2 Test Plan for Mobility and Locomotion

8.2.1 Obstacle Avoiding and Climbing - Done

TEST	EXPECTED OBSERVATION	FAILURES
• We will place different obstacles in front of the rover and check the rover's ability to detect the size of obstacles and distance from the rover. • We will also test the rover's ability to climb smaller obstacles and avoid larger ones.	• Path planned must be clear and far from obstacles. • Rover can clearly detect and avoid the obstacles.	• Rover is unable to cross the smaller obstacles. • Wheels stuck near the obstacles.

CORRECTION	EMERGENCY
• Rewrite the path planning algorithms to pass all the obstacles without intervention	• Switch to manual mode to readjust path in case rover is about to get stuck

8.2.4 Structural Strength - Done

TEST	EXPECTED OBSERVATION	FAILURES
• Testing the hardware integrity during the mission operation.	• Components must be intact even when rover crosses over obstacles.	• Loosening of joints during operation.

CORRECTION	EMERGENCY
• Before the mission, tighten all the joints. • Use of thread lock liquid during final assembly, especially in joints which may face high levels of vibrations	• Emergency repair kit and thread lock liquid

8.2.5 Water Resistance

TEST	EXPECTED OBSERVATION	FAILURES
• We will test the rover's chassis and the electronic compartments for leakages by placing it in an environment which imitates rainy environment.	• The internals of the rover must be waterproof.	• compartments exhibit leakages

CORRECTION	EMERGENCY
• Rectify the leaks, replace the coverings properly.	• Shutdown the rover to avoid electrical short circuit. • Cover the leaks and restart

8.3 Power System Test Plans

8.3.1 Battery Endurance - Done

TEST	EXPECTED OBSERVATION	FAILURES
• Full charge the batteries and operate the rover at its full capacity. Test the run time of the rover with different sub-systems on.	• Rover can complete the mission with its primary supply. • Must last at least 60min.	• Battery leaks. • Sparks • Rover discharges early.

CORRECTION	EMERGENCY
• Replace batteries with new ones. • Add extra batteries. • Increase the efficiency of the power system	• Switch the extra backup batteries. • Keep extra batteries with team with full charge.

8.3.2 Temperature - Done

TEST	EXPECTED OBSERVATION	FAILURES
• Testing over wide temperature ranges	• Components must not overheat during operation.	Cooling system fails Insulation of power line may melt due to high stall current of motors iin case the rover is stuck

CORRECTION	EMERGENCY
• Recheck cooling mechanism	• Turn off some non-essential sub-systems and cool the rover using cooling mechanism.

8.4 Communication Test Plans

8.4.1 Wireless Range - Done

TEST	EXPECTED OBSERVATION	FAILURES
• We test the rover's maximum range of stable communication. • We will place the rover and controller at different distances and measure the lag in performing tasks.	• This test ensures that the maximum range of 25m is achieved.	• Rover is unable to communicate at larger distances. • There is noise in communication signal

CORRECTION	EMERGENCY
• Check the signal strength of the antennas and controller. • Check and replace the antennas.	• Backup comms hardware.

8.4.2 Penetration of Signal through Obstacles - Done

TEST	EXPECTED OBSERVATION	FAILURES
• We will separate the rover and controller putting different obstacles in between and test the communication.	• The rover and base station can communicate even when they are not in line of sight of each other	• Unexpected connection breakage

CORRECTION	EMERGENCY
• Check the signal strength of the antennas and controller. • Check and replace the antennas.	• Autonomously readjust rover's position to a place where connection can be re-established

8.4.3 Signal Loss - Done

TEST	EXPECTED OBSERVATION	FAILURES
• We will test the response of the rover when the communication is lost with the controller.	• We want the rover to backtrack to the location where the signal has been lost and try to re-establish the signal. • We will implement backtracking algorithms for the above. • Even then, if the rover is unable to gain communication, the rover will go into standby mode and wait for the team to manually operate.	• The Rover shows unexpected and uncalculated behavior

CORRECTION	EMERGENCY
• Readjust the antenna's location and refine the communication systems.	• Mentioned in emergency response system (first row of the table)

8.4.4 Signal Noise Management - Done

TEST	EXPECTED OBSERVATION	FAILURES
• We will test the tolerance of the communication system to external noise. • We will provide external noise manually and checks for communication problems. • We will also check if the rover is responding to any other frequencies than the one set by us.	• This test is to ensure that the system is not interrupted by external noise. • This test also ensures that no other frequencies can be used to control the rover.	• High noise values to which the rovers software starts behaving abnormally.

CORRECTION	EMERGENCY
• We will implement noise filters in our code to eliminate most of the noises.	• We will have a backup channel to which we will switch when there is unexpected noise.

8.5 Mission Tasks Testings

8.5.1 Manual Control

TEST	EXPECTED OBSERVATION	FAILURES
The rover will be controlled manually and the whole mission will be completed in manual mode.	The rover must be able to perform all the tasks.	Rover might not traverse or work according to the given input.

CORRECTION	EMERGENCY
• The channel and program need to be reconfigured to get the desired controls.	NA

8.5.2 Terrain Mapping

TEST	EXPECTED OBSERVATION	FAILURES
We will place the rover in different terrains and test its mapping capabilities.	This test will ensure the rover can detect obstacles and their distances from the rover.	The rover might not map the terrain appropriately and may not be able to judge the obstacles.

CORRECTION	EMERGENCY
• The algorithm for Terrain Mapping needs to be reconfigured	NA

8.5.3 Detect and Pick Up

TEST	EXPECTED OBSERVATION	FAILURES
Once the rover is at the endpoint. We will test its ability to drop the package in the given location and move out of the endpoint	This will ensure that the rover finishes the mission successfully.	The rover may prematurely drop the package, or it may not drop it at the end point..

CORRECTION	EMERGENCY
The program needs to be rechecked and fixed. Faulty hardware for package drop needs to be fixed.	NA

8.5.4 End point Identification

TEST	EXPECTED OBSERVATION	FAILURES
The camera and algorithm with the rover will be tested with the identification of the end location in the arena.	To ensure that the rover can identify the endpoint during the mission.	The rover might not be able to judge the endpoint correctly.
CORRECTION		**EMERGENCY**
The algorithm needs to be reconfigured with correct values.		NA

8.5.5 Drop the package

TEST	EXPECTED OBSERVATION	FAILURES
Once the rover is at the endpoint. We will test its ability to drop the package in the given location and move out of the endpoint	This will ensure that the rover finishes the mission successfully.	The rover may prematurely drop the package, or it may not drop it at the end point.
CORRECTION		**EMERGENCY**
The program needs to be rechecked and fixed. Faulty hardware for package drop needs to be fixed.		NA

8.5.6 Path Planning

TEST	EXPECTED OBSERVATION	FAILURES
After terrain mapping, put the rover in different locations in a mock arena to test path planning capabilities	We will also ensure the rover can avoid obstacles and take efficient paths.	There might be an obstacle in the path planned by the rover.
CORRECTION		**EMERGENCY**
The path planning algorithm needs to be rechecked and reconfigured		NA