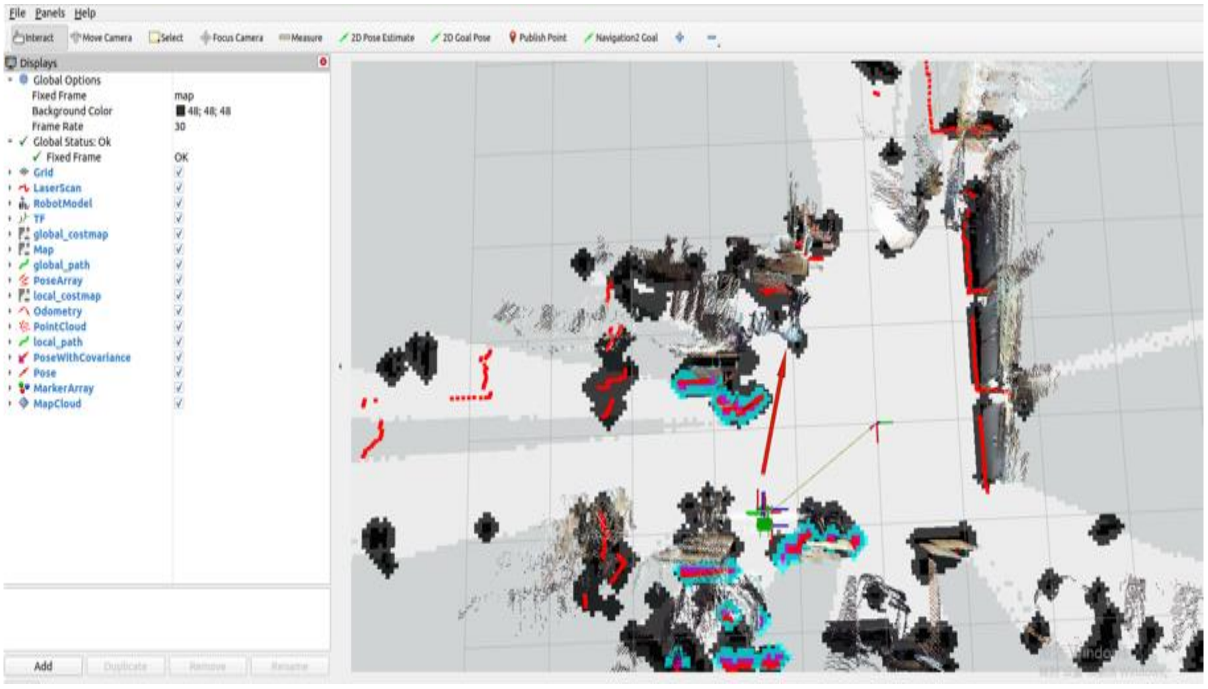
1. Complete and Exhaustive Description of All Logical Steps in the Working of the Robot

A. Description of Robot:

Overview: The rover is designed for navigating outdoor fields and challenging terrains, incorporating a rocker-bogie suspension system that allows for superior ground clearance and obstacle traversal. The frame is built from lightweight, durable materials like aluminium or composite, optimized for strength and flexibility.

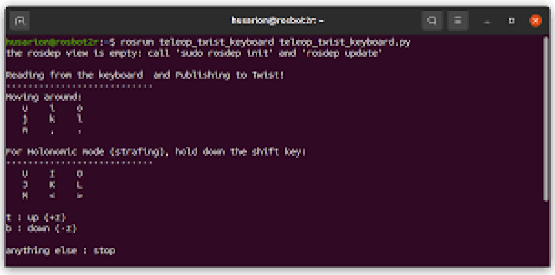
Navigation Capabilities:

- Autonomous Navigation: The rover navigates from a starting point to a designated endpoint, set by either a specified distance or GPS coordinates.
- Obstacle Detection: Utilises Lidar technology to identify and avoid obstacles, including short fences, tree branches, plastic bottles
- Obstacle Avoidance: When an obstacle is detected within range. The rover adjusts its trajectory to navigate around the obstacle. It resumes forward movement after avoiding the obstacle.
- Integration with SLAM: Combines Lidar output with SLAM (Simultaneous Localization and Mapping) to enhance navigation capabilities.
- Framework Utilisation: Integrates ROS FOXY and the Nav2 framework for precise autonomous navigation in the desired direction.
- Mapping: The rover will use the radar scanning data and the depth data of the depth camera to build a map using the RTAB-MAP algorithm during the movement and save the map after the construction is completed.



Global Options	
Fixed Frame	map
Background Color	■ 48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
LaserScan	✓
RobotModel	✓
TF	✓
global_costmap	✓
Map	✓
global_path	✓
PoseArray	✓
local_costmap	✓
Odometry	✓
PointCloud	✓
local_path	✓
PoseWithCovariance	✓
Pose	✓
MarkerArray	✓
MapCloud	✓

- Semi-Autonomous Control: The rover can also be controlled manually using a manual controller or by keyboard remote controller, providing flexibility in operation.



```
husarion@rosbot2r:~$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py
the rosdep view is empty: call 'sudo rosdep init' and 'rosdep update'

Reading from the keyboard  and Publishing to Twist!
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

t : up (+z)
b : down (-z)

anything else : stop
```

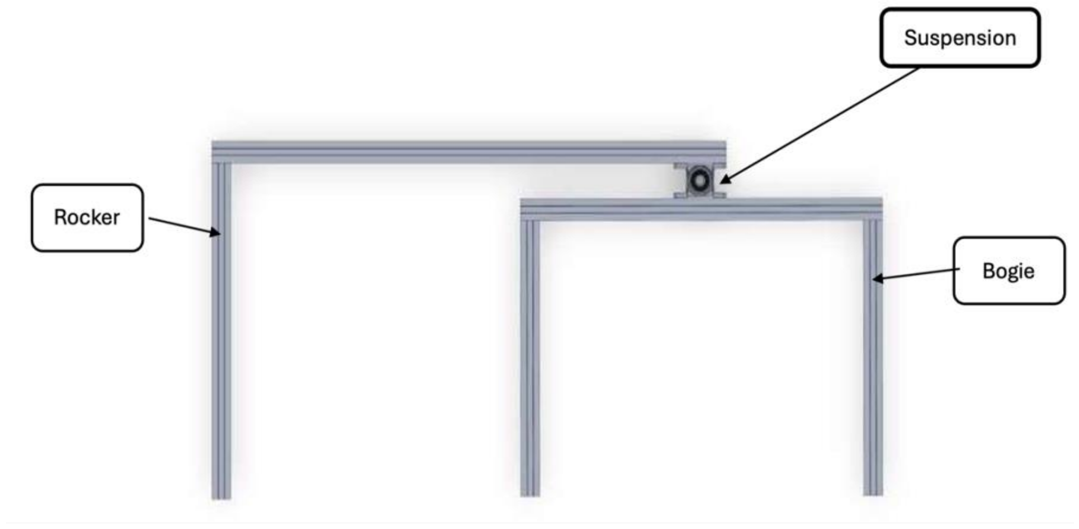
GPS Visualization:

- The rover is equipped with an integrated high-performance GNSS system that facilitates the storage of its trajectory as GPS coordinates.

Rover Structure and Mechanics:

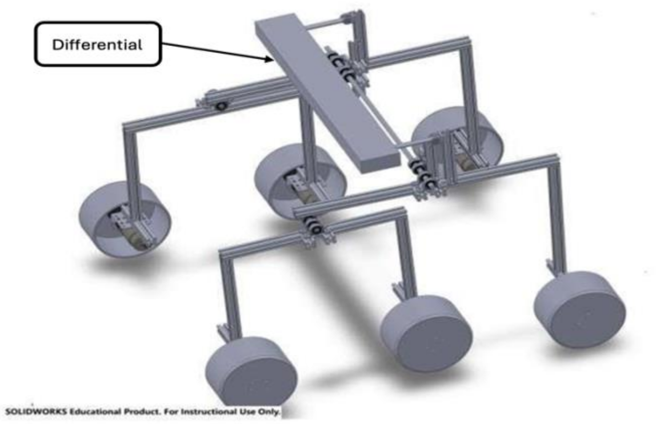
- We constructed the rover's framework using aluminium extrusion, selected for its cost-effectiveness, high strength-to-weight ratio, and excellent corrosion resistance. The rover's body, measuring 40 cm x 36 cm x 19 cm, is also composed of aluminium extrusion, providing ample space for electronic components and batteries.
- To enhance stability, we employed mechanical differentials within the design. The two rockers are interconnected to the body via the differential mechanism, which ensures that the rover's body remains level. As one rocker rises, the other descends, maintaining the body at an angle that is the average of the two rockers' positions relative to the ground.
- The differential consists of a central bar pivoted to the body, with its ends linked to the rockers through short connecting rods. When the rover body is held steady and one rocker is tilted upward, the central bar shifts: one end moves backward while the other moves forward, causing the opposite rocker to tilt downward.
- For propulsion, we incorporated 12V DC motors equipped with an Orange Planetary Gear system, delivering a rated torque of 100 N·m and a rated speed of 262 RPM, ensuring effective movement and control of the rover

- **Suspension system:** The rocker-bogie design has no springs or stub axles for each wheel, allowing the rover to climb over obstacles, such as rocks, that are up to twice the wheel's diameter in size while keeping all six wheels on the ground. As with any suspension system, the tilt stability is limited by the height of the centre of gravity. Systems using springs tend to tip more easily as the loaded side yields.



- The mechanism is designed such as due to the independent motion of right and left rockers, the pitching of the chassis or the rover body remains an average of the two rockers.
- The design incorporates independent motors for each wheel. There are no springs or axles, making the design simpler and more reliable.
- The design reduces the main body motion by half, compared to any other suspension. The jerk experienced by any of the wheel is transferred to the body as a rotation via the differential connecting the two rockers, not as translation like conventional suspensions.
- The rocker-bogie suspension is a mechanism that, along with a differential, enables a six-wheeled vehicle to passively keep all six wheels in contact with a surface even when driving on severely uneven terrain.

- **Differential:** Rover mobility systems play a critical role in enabling exploration missions by facilitating movement across diverse terrains. The rod differential is a fundamental component within rover drivetrains, contributing to enhanced manoeuvrability, traction, and control. This report elucidates the functionality, design, and advantages of rod differentials in rover mobility systems.



- **Overview of Rod Differential:** A rod differential, also known as a torque-biasing differential or limited-slip differential, is a mechanism that allows for the distribution of torque between the wheels of a rover. Unlike traditional open differentials, which distribute torque equally to all wheels, a rod differential can vary torque distribution based on wheel slippage and traction conditions.

- **Functionality:** The rod differential operates by utilizing a set of interconnected rods or mechanisms that transfer torque between the wheels. When one wheel experiences slippage, such as when traversing uneven terrain or encountering obstacles, the rod differential transfers torque to the wheel with greater traction, thereby improving overall traction and stability. By enabling torque distribution based on wheel conditions, the rod differential enhances rover manoeuvrability and performance across various terrains.

- **Design Considerations:** The design of a rod differential involves careful consideration of factors such as torque bias ratio, material selection, and durability. The torque bias ratio determines the extent to which torque is transferred from one wheel to another based on slippage conditions. Materials with high strength-to-weight ratios and resistance to wear are chosen to ensure the reliability and longevity of the rod differential in demanding operational environments.

- **Advantages of Rod Differential:** Improved Traction: The rod differential enhances traction by redirecting torque to wheels with better grip, enabling the rover to traverse challenging terrains with reduced slippage. Enhanced Manoeuvrability: By dynamically adjusting torque distribution, the rod differential improves the rover's ability to navigate obstacles, negotiate inclines, and maintain stability during turns. Minimal Impact on Power Consumption: Unlike locking differentials that may increase power consumption, the rod differential operates efficiently by transferring torque only when necessary, optimizing energy utilization during rover operations.

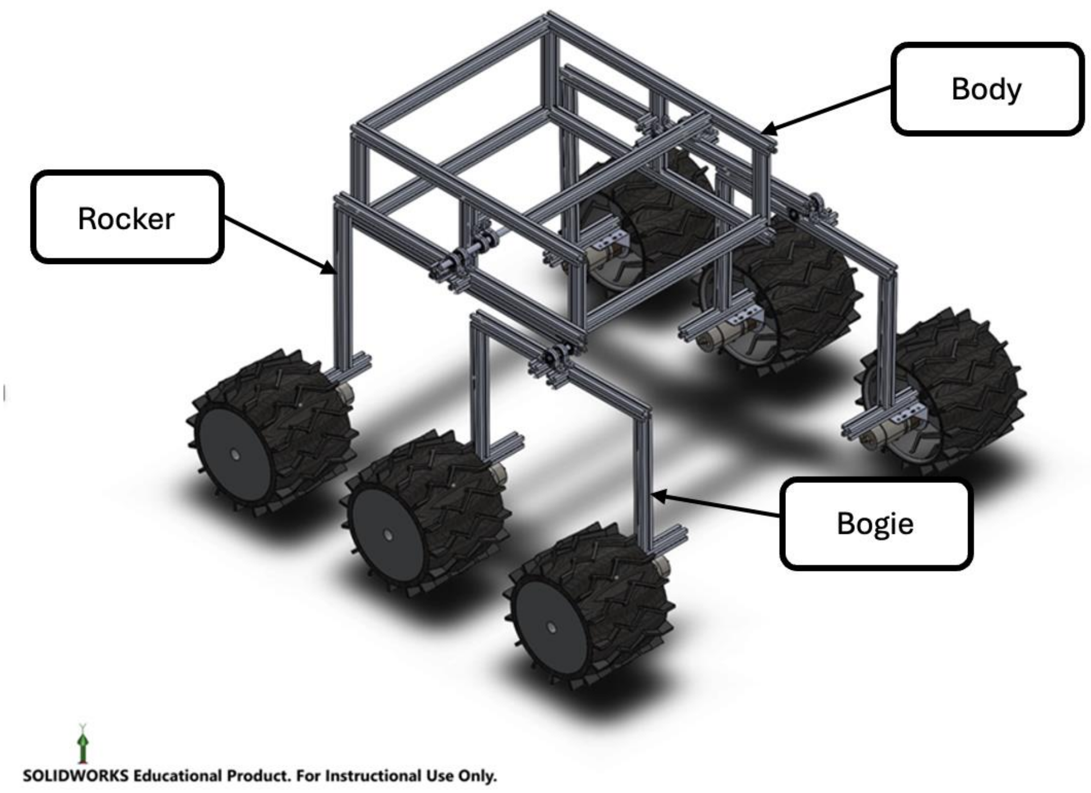
- **Applications and Implementation:** Rod differentials find applications in various rover designs, including planetary exploration rovers and terrestrial vehicles used in rugged environments. Implementation involves integrating the rod differential into the rover's drivetrain system, ensuring compatibility with other components such as motors, wheels, and control systems. Testing and validation procedures are conducted to assess the performance and reliability of the rod differential under simulated and real-world operating conditions.

    1. The Rocker Mobility Test is a specific evaluation used in physical therapy and rehabilitation settings to assess an individual's dynamic balance and mobility. It focuses on the ability to shift weight and maintain stability during walking or other movements. Here's an outline of what the test typically involves:
       Initial Assessment: The individual's baseline mobility and balance are assessed through observation, subjective questioning, and possibly other standardized tests.
       Test Setup: The individual is asked to stand with feet together or in a comfortable stance on a firm surface.

Board: The individual may be asked to stand on a rocker board or similar device designed to challenge balance. A rocker board typically consists of a platform that pivots in the middle, requiring the individual to maintain balance by shifting weight and adjusting their position.

2. Task Execution: Depending on the specific protocol, the individual may be instructed to perform various tasks on the rocker board, such as:
   Shifting weight from side to side while maintaining balance.
   Rotating the board in different directions. Performing reaching or bending movements while standing on the board.
   Assessment Criteria: The therapist or evaluator observes the individual's performance, looking for signs of instability, loss of balance, compensatory movements, or difficulty completing the tasks.
   Scoring: The individual's performance may be scored based on predefined criteria, such as the ability to maintain balance, the range of motion during movements, and the quality of movement execution.
   Interpretation: The results of the Rocker Mobility Test are interpreted in the context of the individual's overall mobility and functional abilities. They may inform treatment planning, goal setting, and interventions aimed at improving balance and mobility.

Overall, the Rocker Mobility Test provides valuable information about an individual's dynamic balance control and can help guide rehabilitation strategies for improving mobility and reducing the risk of falls or injuries.

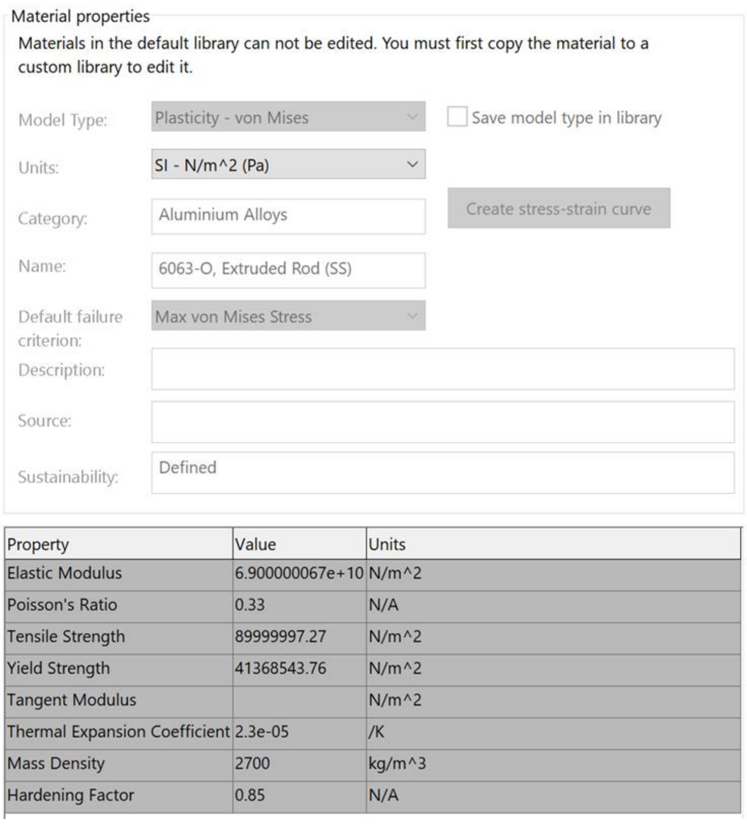


Selection of material is an important step in designing of any component The main advantages of material selection are:

- It increases the reliability of product
- It reduces the cost of product
- It can also optimize the weight of product

We have selected aluminium alloy that is extrusion which is light weight, and it have higher strength among the materials in that weight category.

Following image shows the data of the material test of the extrusion.



Material properties

Materials in the default library can not be edited. You must first copy the material to a custom library to edit it.

Model Type:	Plasticity - von Mises
Units:	SI - N/m^2 (Pa)
Category:	Aluminium Alloys
Name:	6063-O, Extruded Rod (SS)
Default failure criterion:	Max von Mises Stress
Description:	
Source:	
Sustainability:	Defined

Save model type in library

Create stress-strain curve

Property	Value	Units
Elastic Modulus	6.900000067e+10	N/m^2
Poisson's Ratio	0.33	N/A
Tensile Strength	89999997.27	N/m^2
Yield Strength	41368543.76	N/m^2
Tangent Modulus		N/m^2
Thermal Expansion Coefficient	2.3e-05	/K
Mass Density	2700	kg/m^3
Hardening Factor	0.85	N/A

- **Wheels:** Wheels are the fundamental to the mobility of the rover, dictating its ability to traverse various terrains.
  The important things to take in the consideration before designing the wheels are as follow:

  - **Terrain compatibility:** The ability of rover wheels to navigate diverse terrains encountered during exploration missions is crucial for mission success. Rover wheels must navigate over rugged and uneven rocky surfaces commonly found on planetary bodies like Mars or the Moon. Increased wheel diameter and width can help traverse larger obstacles and maintain stability over rocky terrain. The wheels can encounter the various degree slopes.

- **Material composition:** Rover wheels are typically constructed from lightweight yet durable materials to withstand the rigors of extraterrestrial exploration. Keeping this aspect in mind we have choose a lightweight and extra durable material which is light weight too that material is polyvinyl chloride.

- **Wheels Flexibility:** Flexibility in wheel design is crucial for traversing uneven terrain and absorbing shocks to minimize impact on the rover's chassis and internal components.

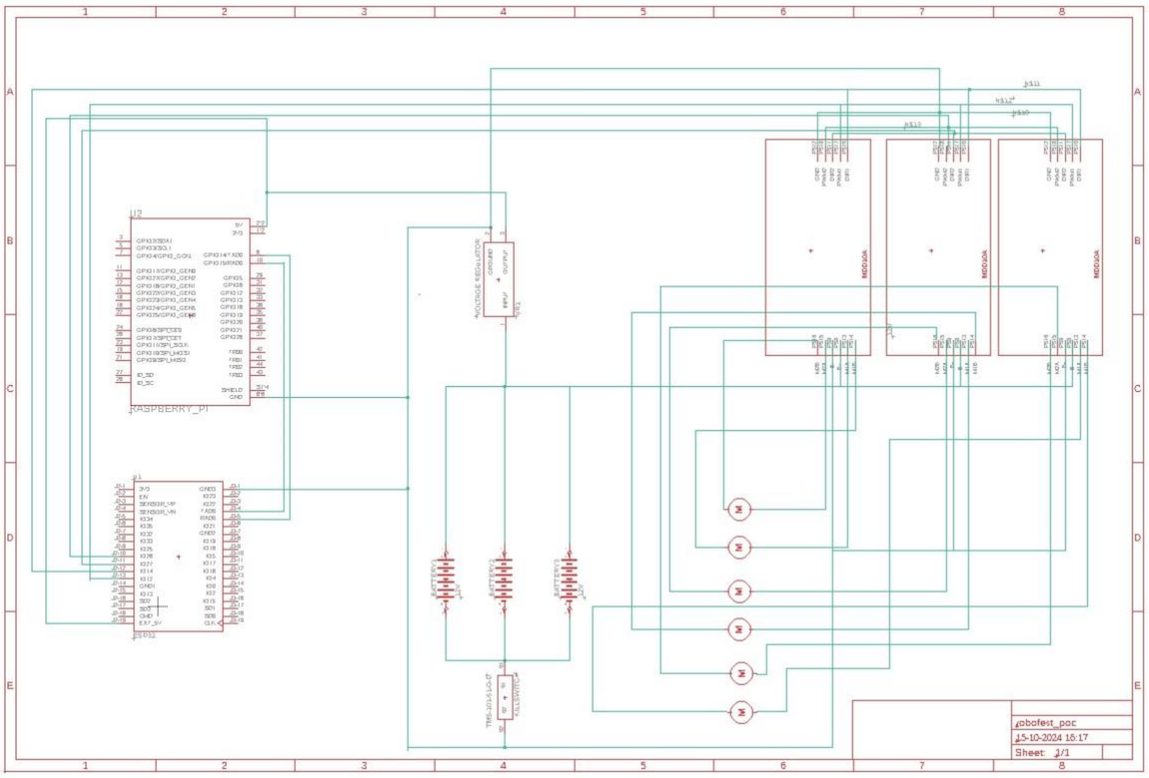
- **Dimensions:** The diameter of the is 18cm and the wheels are hollow.

- **Cost Efficient:** As the wheels are made up of the polyvinyl chloride polymer which is cheap as compared to the other materials having the same specifications and the wheels are designed such that the internally, they are hollow all the load is evenly distributed over the surface is the key aspect of the success of this type of the wheel.

B. Electronics Description:

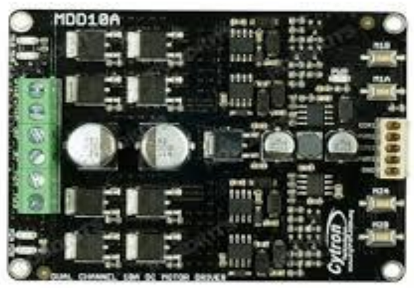
Power Supply and Power Distribution PCB:

- Role: Our rover is powered by a rechargeable 12V battery with around 13700mAh battery life. A custom-designed PCB handles power distribution to all components, ensuring efficient and stable power delivery. The PCB includes voltage regulators and protection circuits to manage power to the Raspberry Pi, motor drivers, and sensors.
- Function: A custom-made power distribution PCB is made to distribute power to all electronics which includes:
- Voltage regulators to step down power for components like the Raspberry Pi (5V).
- Power protection circuits to prevent overvoltage or undervoltage conditions.
- Connections for motor drivers, sensors, and other peripherals.



MDD10A Motor Driver

- Role: The MDD10A dual-channel motor driver controls the speed and direction of the planetary motors. It receives PWM signals from the Raspberry Pi for precise control over the motors.
- Connectivity: PWM pins from Raspberry Pi GPIO for speed control. Connected to the 12V battery to power the motors.

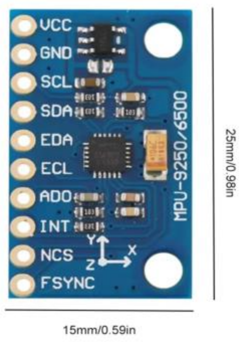


Planetary Motors

- Role: High-torque planetary motors are used to move the rover across rough terrain. The motors are driven by the MDD10A motor drivers, with speed and direction controlled by the Raspberry Pi via PWM.
- Specifications:
  Operating voltage: 12V
  High torque output for steady movement and navigation.
- Connectivity:
  PWM pins from Raspberry Pi GPIO for speed control.
  Connected to the 12V battery to power the motors.

MPU9250:

- Role: The MPU9250 measures the robot's orientation in 3D space, specifically tracking pitch, roll, and yaw, which is essential for accurate navigation and stability.
- Function: The MPU9250 integrates its data with other sensors, such as LiDAR and GPS, using tools like the robot localization package. It publishes orientation data (in quaternion format), angular velocities, and linear accelerations on the /imu/data topic, while the magnetometer data is published on the /imu/mag topic. This fusion of data enhances the robot's real-time localization capabilities.



RUSH FPV GNSS PRO:

- Role: It serves as a high-precision GPS receiver that plays a critical role in collecting accurate geospatial data for the robot .
- Function: It continuously gathers GPS data, including latitude, longitude, and altitude, which is published on ROS2 topic (/gps/fix). This data facilitates sensor fusion with other data sources, enhancing the robot's localization. This GNSS data can be visualized in Mapviz, providing a graphical representation of the robot's location on a map.

TF-LUNA (1D LiDAR) :

- Role: The TF-LUNA 1D LiDAR detects small or low-height obstacles (e.g., rocks, 100 mm cuboids) in front of the rover that might be missed by the 2D LiDAR.
- Function: It continuously publishes real-time distance data to a ROS 2 topic (e.g., /tf_luna_scan), which is used for mapping and obstacle avoidance, ensuring accurate detection of small obstacles close to the ground, enhancing the robot's navigation and SLAM capabilities.

YDLIDAR X2 360 Degree :

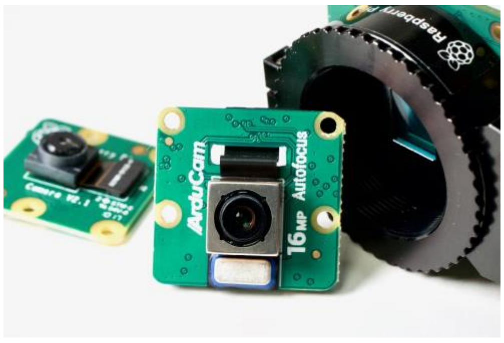
- Role: To laser scan the environment for obstacle detection and mapping.
- Function: The 2D LiDAR scans the surrounding area in a 360-degree or partial plane, continuously measuring distances to objects. This data is published to ROS 2 topics (/scan) and is used to generate a 2D occupancy grid map. The map supports SLAM by helping to construct the environment's layout while tracking the robot's position for navigation and path planning using ROS 2's Nav2 stack.

Intel RealSense Depth Camera D435i :

- Role: To depth sense the environment, navigating landscape and easily create accurate scales of objects.
- Function: It captures and publishes RGB and depth data on ROS 2 topics (/camera/color/image_raw, /camera/depth/image_raw),this RGB-D data is used in the RTAB-Map package for simultaneous localization and mapping (SLAM), for creation of accurate 3D maps while tracking the robot's position.

Arducam 16MP IMX519 Camera Module:

- Role: To provide high-resolution visual data for the robot. Allowing the operator to monitor the robot's environment in real-time.
- Function: The camera’s data will be published on ROS 2 topics, which can then be accessed by the object detection node running YOLOv5 for further processing, enhancing autonomous decision-making.



Power Flow:
We are using four 12V batteries to power the electronic components in our rover. Each motor driver is powered by a dedicated 12V battery, utilising three batteries in total. The fourth 12V battery is used to distribute power to all the electronic components via a voltage regulator, which provides 5V to the other components and to the rover's brain, the Raspberry Pi.

PCB Design and Power Distribution:
A custom PCB is designed to integrate sensor data, manage power distribution, and ensure smooth communication between the Raspberry Pi, motors, and sensors. This board includes connectors for voltage regulation, ensuring battery power is distributed efficiently to each component.

C. Features

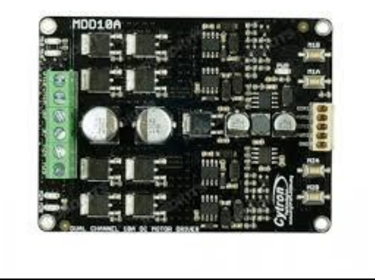
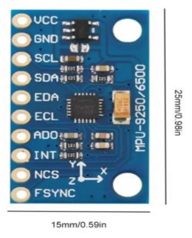
Autonomous Navigation: Uses advanced sensor fusion to map the environment and navigate without human intervention, avoiding obstacles smoothly.
Real-Time Mapping: Through SLAM and GPS, the rover generates real-time maps of the environment, helping in tracking its path and ensuring efficient route optimization.

Mechanically Efficient Design: The rocker-bogie suspension ensures smooth traversal of rocky and uneven surfaces without losing balance.

2. Complete and Exhaustive Listing of All Hardware / Component / Equipment

A. Electronics Components

- **Electrical Modules:**

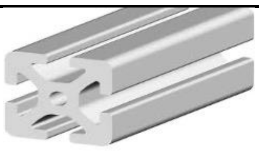
Sr.no.	***Module***	***Diagram***
1)	*Raspberry-Pi 5*	
2)	*MDD10A*	
3)	*Intel RealSense Depth Camera*	
4)	*Rush FPV GNSS Pro*	
5)	**MPU 9250 IMU Sensor**	 VCC GND SCL SDA EDA ECL ADO INT NCS FSYNC MPU-9250/6500 25mm/0.98in 15mm/0.59in

- **Electronics Components Used for Fabrication**
  - Custom PCB: Designed to integrate the power supply, motors, and sensors. Includes connectors for the lidar units, camera, and power regulation.
  - Power Components: Batteries (lithium-ion), voltage regulators, and power distribution circuits that ensure each module receives the correct voltage.

- **Tools and Equipment Used**
  Soldering kit, multimeter, oscilloscope (for debugging circuits), breadboards, jumper wires.

B. Mechanical Components

Mechanical Components

Sr.no.	Module	Diagram
1)	*Aluminium Extrusion*	 *Aluminium extrusion is used because it is a material which requires minimum machining.*
2)	*Pillow Bearing*	*It is used in the suspension system and to join the rocker and the bogie.*
4)	*Smooth Rod*	*Smooth rod is inserted inside the pillow bearings and the via this the suspension system is completed*
5)	*Anti Slip Motor Coupling*	*This is used to connect the wheels and the shaft and motor and we have used this module because it is anti slip*
6)	*Clamp*	*Clamp is used to make the motor stable and attach it to the extrusion.*

7)	*Wheels*	*Wheel are of polyvinyl chloride and the outer covering i.e. grip is of rubber And we are working on the modified roboust wheels*

Fabrication Component

Sr.no.	**Components**	**Description**
1)	*Series Pivot joint*	*Used in the differential to connect the rods withs extrusion*
2)	*Corner bracket*	*Used to connect the joints in the extrusion from outer side*
3)	*inner brackets*	*Used to connect the joints of the extrusion from the inner side.*
4)	*Allen bolts*	

		Allen bolts are used in corner brackets and it avoids the drilling operation
5)	*T nuts*	*T nuts are used in corner brackets and it avoids the drilling operation*
6)	*Nut and Bolts*	*Used to connect the wheels and the hub assembly.*
7)	*Allen grub screw*	*Grubs are used in the inner bracket and in the pillow bearing to fix the smooth rod*

Tools, Equipment's Used

Sr.no.	Tools or Equipments
1)	Angle grinder
2)	Screw driver set
3)	Allen keys
4)	Metal filler
5)	Nose pyler
6)	Drilling machine
7)	CNC Router
8)	3D Printer

Electromechanical Components

- Planetary Motors: Integration of high-torque motors with the rocker-bogie system for smooth operation, even on inclines.
- Sensors on Moving Parts: Lidars mounted on rotational platforms to allow for dynamic scanning.
- Kill Switch acts as a simple safety mechanism enabling to turn off the flow of power

3. Complete and Exhaustive Listing of All Software Used in Robo-Making

A. Software Used:

The software used in the rover is completely based on ROS2 framework. In our rover we used the Foxy version because of its strong stability and rich information.

CPU:

- The Raspberry Pi 5 features a quad-core ARM Cortex-A76 CPU, clocked at 2.0 GHz.The ARM Cortex-A76 supports a wide range of instruction sets, including SIMD (Single Instruction, Multiple Data) operations, which accelerate multimedia processing and computations essential for our rover and image processing.

- It is available with 8GB of LPDDR4-3200 SDRAM.With the increased memory capacity, users can run multiple ROS nodes simultaneously, manage high-resolution video streams from cameras, and perform complex data processing .
- The integrated VideoCore VI GPU processes graphics and allows for the execution of OpenGL ES 3.0 applications. This capability is particularly useful for running visualisation tools like RViz and handling 3D rendering tasks. It also assists in parallel processing tasks.

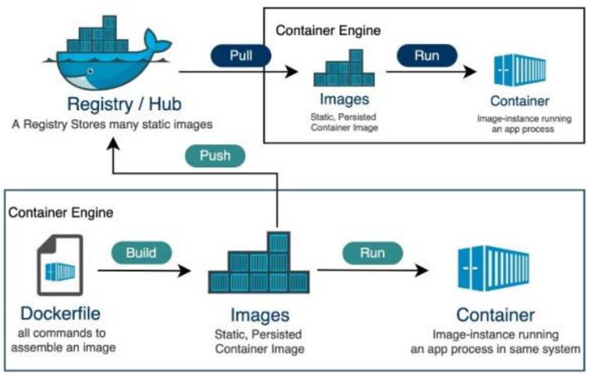
Operating System :

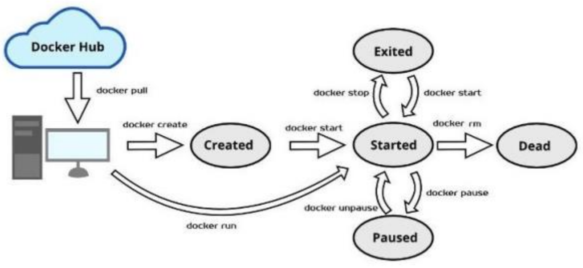
- The 64-bit operating system of Raspberry Pi is used due to its larger memory addressing space and higher performance, 64-bit operating systems can also better support data-intensive applications, such as large-scale data processing, machine learning.
- Tailored Kernel and Drivers: Pi OS (64-bit) is specifically designed and optimized for Raspberry Pi hardware. It includes kernel patches and drivers tailored to fully utilise the features of the Raspberry Pi 5. It has better support for HATs(Hardware Attached on Top) and GPIOs pins.
- Broadcom BCM2712 SoC: The RPi 5’s quad-core Cortex-A76 CPU (2.4GHz) benefits greatly from Pi OS (64-bit), which has been specifically compiled with ARMv8-A architecture optimizations. It helps in proper handling multiple sensors like Lidar , Depth Camera .

- OpenGL ES 3.1 & Vulkan Support: With better support for graphics APIs like OpenGL and Vulkan, Pi OS (64-bit) allows seamless rendering of complex visualizations in RViz for ROS 2.

Development Environment:
Docker :

Docker is an open-source platform which helps to concentrate on creating software rather than handling difficult infrastructure. It is based on the idea of software containers.
Docker Containers :
These containers are independent of underlying OS . They are lightweight and portable as they share the host's OS kernel, unlike VM that mimic full hardware systems. It is a runtime instance of Docker Image.

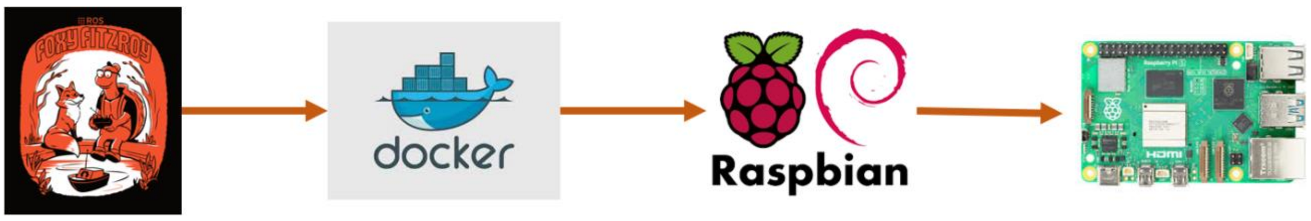

Registry / Hub
A Registry Stores many static images
Pull
Container Engine
Images
Static, Persisted Container Image
Run
Container
Image-instance running an app process
Push
Container Engine
Dockerfile
all commands to assemble an image
Build
Images
Static, Persisted Container Image
Run
Container
Image-instance running an app process in same system



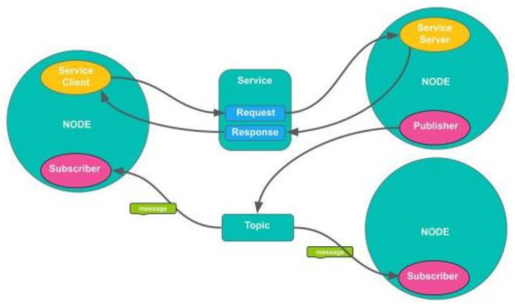
- Docker Hub : It is a cloud-based repository service that allows users to store, share, and manage Docker container images.
- Docker Images: They are self-contained templates that are used to build containers.Changes made affect to the running program happen inside a container, not to the image itself.
- Dockerfile: It is a set of instructions for creating a Docker image.

Since ROS 2 only supports Ubuntu 22.04, which is not yet available for the Raspberry Pi 5, we are unable to run ROS 2 directly on it. However, we can use Docker to run ROS 2 (Foxy) by pulling the official image `ros:foxy-ros-base` and executing it in an appropriate environment.

ROS 2 (Foxy):
ROS 2 is the second generation of the Robot Operating System, an upgrade from ROS 1, designed for more robust, real-time, and distributed robotic applications. It supports multiple programming languages, including C++, Python, and Java, enabling developers to work with a wide range of tools and environments.



- Nodes
  A node is a process that performs a specific task in a robot. For instance, a node can be responsible for reading data from a sensor or controlling motors.
  Example: One node reads data from a camera, while another node processes the camera images to detect obstacles.

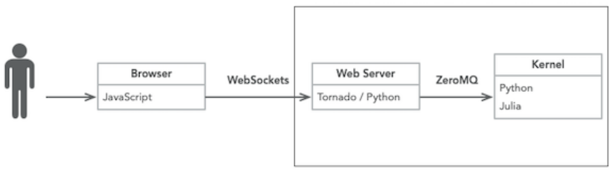


- Topics
  Topics are used for communication between nodes. A node can publish data to a topic, and other nodes can subscribe to that topic to receive the data.
  Example: A LiDAR sensor node publishes distance data to the /lidar_data topic, and a navigation node subscribes to this topic to make decisions.
- Messages
  Messages are the format in which data is sent between nodes over topics. Each message type defines the data structure used (e.g., sensor readings).
  Example: A GPS node sends a message (/gps/fix) with latitude and longitude data through a topic, and another node processes the message to locate the robot.
- Services
  Services allow a node to make a request to another node and receive a response. Unlike topics (which are asynchronous), services are synchronous.
  Example: A node can request a map from a mapping node by calling a service. The mapping node responds with the current map.
- Actions
  Actions are used for tasks that take time to complete and provide feedback during the process. They are useful for long-duration tasks, like robot movement.
  Example: A node sends an action to move the robot to a specific location. The robot navigation node provides feedback on its progress and reports when the task is done.
- Parameters
  Parameters are values that can be set to configure a node’s behaviour. They can be changed without stopping the node.
  Example: A parameter might define the maximum speed of the robot. You can adjust this value without restarting the node controlling the motors.
- Launch Files
  Launch files start multiple nodes together with their configurations. They simplify the process of starting a robot system by handling dependencies and settings.

Example: A launch file might start a camera node, LiDAR node, and navigation node all at once with the proper configurations.

- Transforms (tf2)
  Transforms (managed by the tf2 package) allow nodes to keep track of coordinate frames and convert data between different frames of reference.
  Example: A robot's camera provides data in the camera frame, but the navigation system needs it in the robot's base frame. tf2 handles this conversion.
- Middleware (DDS)
  ROS 2 uses DDS (Data Distribution Service) as its middleware for node communication. DDS automatically discovers nodes and handles data exchange.
  Example: When a new sensor node starts, it can immediately communicate with other nodes because DDS takes care of discovery and message delivery.
- Quality of Service (QoS)
  QoS settings control how messages are sent and received over topics. We can set different QoS policies to handle message reliability or latency.
  Example: A sensor that sends critical safety data might have a higher QoS setting to ensure that no data is lost during transmission.
- Lifecycle Nodes
  Lifecycle nodes have managed states (e.g., inactive, active) that allow them to be controlled more predictably. This is useful in complex systems where some nodes need to be brought up in stages.
  Example: A node responsible for controlling a robot arm might be started in an inactive state and only transitioned to active once the system is fully initialised and ready.

- **Jupyter**
  The architecture of Jupyter notebooks includes several key components:
  Browser Interface: Users interact with Jupyter notebooks through a web browser.

  HTTP Server: Tornado serves as the web server handling client requests.
  Kernel Communication: Communication with the kernel is managed via ZeroMQ, allowing different channels for message types (e.g., execute_request, status).
  Data Model: Notebooks are stored in JSON format (.ipynb), which includes code and outputs.
  Multiple Clients: Multiple clients can connect to a single kernel.
  Hosting Options: Jupyter can run locally or on remote servers.
  By integrating YOLOv5 in JupyterLab, we can conduct object detection

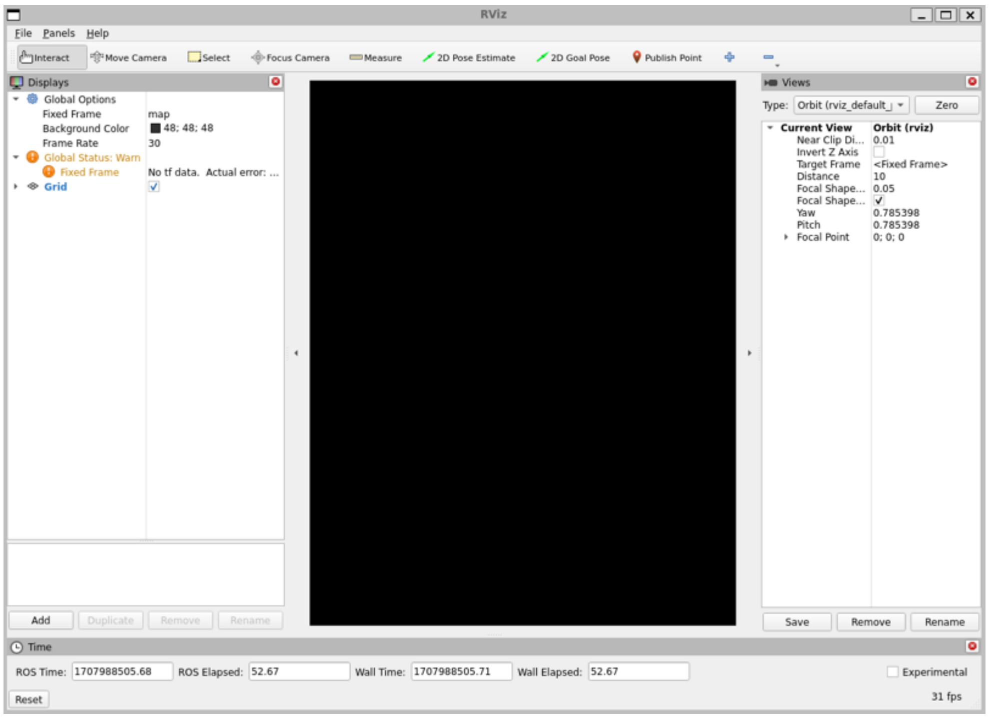
interactively while leveraging the flexibility of Python and its data processing libraries.

VS code :

It is a lightweight, open-source code editor developed by Microsoft, designed for building and debugging applications.
The Docker extension for VS Code simplifies Docker usage by providing:
Integrated Docker Management: Allows users to build, run, stop, and manage containers and images directly within the editor.
Effortless Dockerfile and Compose Management: Users can easily create and edit Dockerfiles and docker-compose.yml files with syntax highlighting and validation.
Container Debugging: Enables debugging of applications inside containers with breakpoints and interactive debugging.
Seamless Workflow: Provides commands for automating containerized development, making Docker usage more intuitive and accessible in a streamlined development environment.

Visualization Tools :
Rviz: It is a 3D visualizer for the Robot Operating System (ROS) framework.
The big black window in the middle is the 3D view. The left is the Displays list, right now it just contains the global options and a Grid.
A display is something that draws something in the 3D world.

Each display gets its own status to help let you know if everything is OK or not.



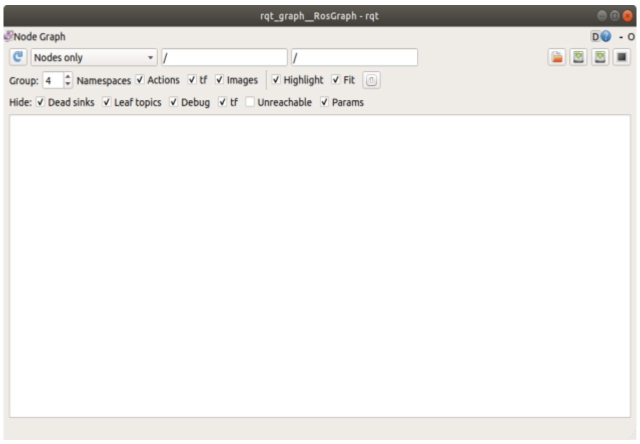
```
Global Options
  Fixed Frame          map
  Background Color     ■ 48; 48; 48
  Frame Rate           30
Global Status: Warn
  Fixed Frame          No tf data.  Actual error: ...
Grid                   ✓
```

```
Type: Orbit (rviz_default_j
Current View           Orbit (rviz)
  Near Clip Di...      0.01
  Invert Z Axis        ☐
  Target Frame         <Fixed Frame>
  Distance             10
  Focal Shape...       0.05
  Focal Shape...       ✓
  Yaw                  0.785398
  Pitch                0.785398
  Focal Point          0; 0; 0
```

```
ROS Time: 1707988505.68  ROS Elapsed: 52.67  Wall Time: 1707988505.71  Wall Elapsed: 52.67   ☐ Experimental
                                                                                              31 fps
```

Name	Description	Messages Used
Camera	Creates a new rendering window from the perspective of a camera, and overlays the image on top of it.	sensor_msgs/msg/Image, sensor_msgs/msg/CameraInfo
Laser Scan	Shows data from a laser scan, with different options for rendering modes, accumulation, etc.	sensor_msgs/msg/LaserScan
Map	Displays a map on the ground plane.	nav_msgs/msg/OccupancyGrid
TF	Displays the tf2 transform hierarchy.	
RobotModel	Shows a visual representation of a robot in the correct pose (as defined by the current TF transforms).	
Range	Displays cones representing range measurements from sonar or IR range sensors. Version: Electric+	sensor_msgs/msg/Range
Odometry	Accumulates odometry poses from over time.	nav_msgs/msg/Odometry
Path	Shows a path from the navigation stack.	nav_msgs/msg/Path

RQT graph:
It is a GUI tool used to introspect log messages in ROS 2. Typically, log messages show up in your terminal.
RQT is a GUI tool in ROS for visualizing the ROS computation graph.
It shows how nodes and topics interact, helping to debug and understand complex systems.
rqt_graph is specifically useful for displaying node-to-topic connections in real-time.
It helps identify communication issues or misconfigurations in a ROS network.


rqt_graph__RosGraph - rqt

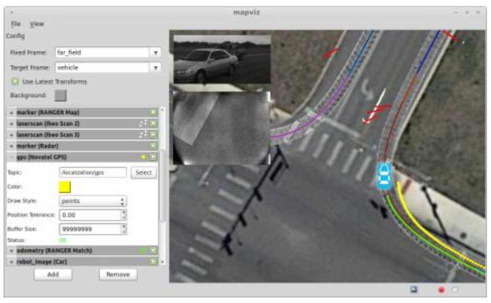
Node Graph

Nodes only | / | /

Group: 4 Namespaces ✓ Actions ✓ tf ✓ Images ✓ Highlight ✓ Fit

Hide: ✓ Dead sinks ✓ Leaf topics ✓ Debug ✓ tf ☐ Unreachable ✓ Params

Mapviz :



Fixed Frame: far_field
Target Frame: vehicle
Use Latest Transforms
Background:
marker (RANGER Map)
laserscan (Ibeo Scan 2)
laserscan (Ibeo Scan 3)
marker (Radar)
gps (Novatel GPS)
Topic: /localization/gps
Color:
Draw Style: points
Position Tolerance: 0.00
Buffer Size: 99999999
Status: OK
odometry (RANGER Match)
robot_image (Car)

It is a ROS-based 2D visualization tool focused on displaying large-scale data in a simple, customizable way. It allows you to overlay different types of data on a map, such as GPS information, laser scans, and robot positions.
We use Mapviz with GNSS sensor by leveraging the GPS plugin, which allows the visualization of GPS coordinates on a map, giving us a real-time view of our rover's position overlaid on a real-world map.

Object Detection and Recognition:
It is essential for our rover which provide capability to "see" and understand the environment .

Obstacle Detection :
The environment may includes obstacles like boulders, short fences, tree branches, and plastic bottles. Object Detcection helps rover to identify these obstacles in real-time, allowing it to have a safe path.
Terrain Awareness: Detecting different terrain types (artificial grass, natural surfaces, or crater-like areas) enables the rover to adapt its speed and traction, ensuring smooth and stable movement across various surfaces.
In the case of manual control, object detection and recognition assist the operator by providing real-time visual feedback on the location and type of obstacles.

YOLOv5:
It is a popular version of the YOLO family of models that focuses on detecting objects in images or video streams. It can identify multiple objects within a frame, classify them, and provide bounding boxes around the objects.

ultralytics
YOLOv5

Working:

- Single-stage Object Detection: Unlike two-stage detectors like Faster R-CNN, YOLOv5 uses a single neural network to predict bounding boxes and class probabilities directly from full images. This makes it faster and more suitable for real-time applications.
- Image Input and Processing: The input image is divided into a grid, and for each grid cell, YOLO predicts multiple bounding boxes along with confidence scores and class probabilities for the object detected.
- Anchor Boxes: YOLOv5 uses predefined anchor boxes to predict bounding boxes more efficiently. These anchor boxes are optimised based on common object sizes in the dataset.

- Loss Function: The model calculates loss based on the difference between the predicted bounding box and the actual bounding box (ground truth). YOLOv5 employs a Binary Cross-Entropy (BCE) loss for classification and an Intersection Over Union (IoU) loss for localization accuracy.
- Post-processing (Non-Maximum Suppression): After predicting multiple bounding boxes for the same object, YOLOv5 applies Non-Maximum Suppression (NMS) to remove redundant boxes, retaining only the best prediction.
- Algorithms Used In YOLOv5 :
- Convolutional Neural Networks (CNNs): YOLOv5 is built using a series of convolutional layers, which extract spatial hierarchies and features from input images.
- Leaky ReLU Activation: Used as an activation function to introduce non-linearity into the model, helping it learn complex patterns from the data.
- Batch Normalisation: Applied to normalise the outputs of convolutional layers, which helps stabilise the training process.
- Sigmoid Activation: This function is applied to the class probabilities and object confidence scores, ensuring that the predictions are between 0 and 1.
- Training and Deployment:
- Transfer Learning: YOLOv5 can be fine-tuned on different datasets using pre-trained weights, allowing for faster training on new tasks.
- Real-Time Inference: Due to its efficient architecture, YOLOv5 is suitable for real-time object detection applications, such as surveillance systems, autonomous vehicles, and robotics.

YOLOv5 Integration With ROS2 :
Image Subscription: A ROS 2 node subscribes to a camera feed (/camera/image_raw), receiving images in ROS message format.
Image Conversion: The received image is converted from ROS format (using CvBridge) to a format compatible with YOLOv5 or OpenCV.
YOLOv5 Processing: The converted image is passed to the YOLOv5 model, which detects objects and generated results are formatted and published as ROS messages on a new topic (yolov5/detections), allowing other nodes to access the object detection data.

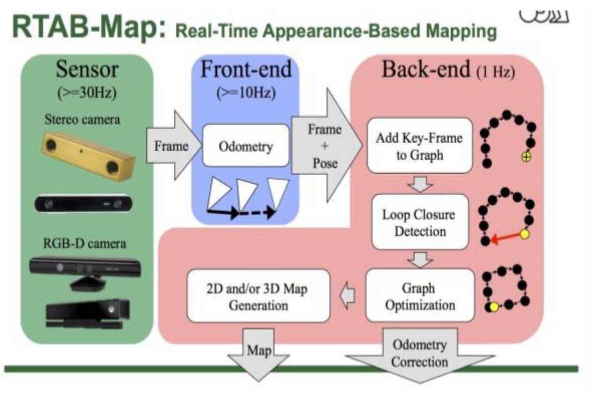
Localization and Mapping :
- The rover's primary task is to navigate a 200m track either by specifying a direction and distance or through GPS coordinates. The integration of high-precision GNSS sensors plays a critical role in determining the start and end points, while the MPU9250 IMU helps calculate the rover's position by monitoring its orientation, accelerations, and angular velocities.
- In localization, unexpected obstacles are inevitable, which is where mapping becomes essential. The environment is mapped using Lidar and depth

cameras to generate a real-time model of the surroundings. This mapping is done using RTAB-Map for simultaneous localization and mapping (SLAM), allowing the rover to create a detailed map while tracking its position in real time. To visualize this process, Rviz provides a graphical interface, and the rover’s position on the map is represented using a URDF model, which acts as the rover's 3D visualization.

Navigation and Control

- Once the environment is mapped, the next step is to guide the rover in the desired direction using the current sensor data. Nav2 is the key framework for this, managing the complex tasks of path planning, sensor fusion, and navigation control. It processes data from various sensors (Lidar, IMU, and GNSS) and calculates the optimal path while avoiding obstacles.
- The localization is supported by sensor fusion, where data from the GNSS, odometry, and IMU is fused to maintain an accurate position estimate. The robot_localization package in ROS 2, often through an Extended Kalman Filter (EKF), is employed for this sensor fusion, stabilizing the position estimate and providing odometry corrections based on sensor readings. This ensures smooth localization even when individual sensors might have errors, as demonstrated by other projects using ROS 2 and Nav2 for autonomous vehicles
  To move the rover, Nav2 sends velocity commands based on the desired path, using real-time sensor data to avoid obstacles. Lidar and camera data continuously update the map, while Nav2 recalculates the path as needed using planners like smac_planner. For motor control, ROS 2 topics are used to send signals, guiding the rover’s wheels or tracks toward the desired trajectory.
- Data from Mapviz and Rviz help monitor and visualize the rover's movements, with the GNSS coordinates offering global visualization. The depth cameras and Lidar are crucial for obstacle detection, enabling real-time path adjustments. Filters such as complementary filters or Kalman filters are used to smooth sensor data and handle noise, ensuring stable movement on rough terrain or slopes.

Rtab and SLAM



- Sensor Measurement
  The process begins with data acquisition from sensors such as RGB-D or stereo cameras. In this stage, the visual information (color and depth) is captured along with any auxiliary

sensor data, like from an IMU or LiDAR. These sensors help in obtaining 3D models of the environment.

- Frontend (Odometry and Feature Extraction)
  In the frontend, the system extracts feature from the sensor data to estimate the robot’s motion (odometry). Visual odometry determines the robot's movement by comparing consecutive frames to detect common features like edges or corners. This is done through various feature detection algorithms (e.g., SURF or ORB). This stage also calculates the relative transformations (pose) between frames, which helps track the robot's position and orientation.

- Loop Closure Detection
  One of the critical aspects of RTAB-Map is its loop closure detection. It uses a bag-of-words technique to compare the current frame with previous ones, checking if the robot has revisited a known location. When a loop is detected (i.e., the robot revisits a location), it corrects the map by adding a constraint to minimize errors accumulated during navigation. This step is essential for reducing drift in the estimated trajectory.

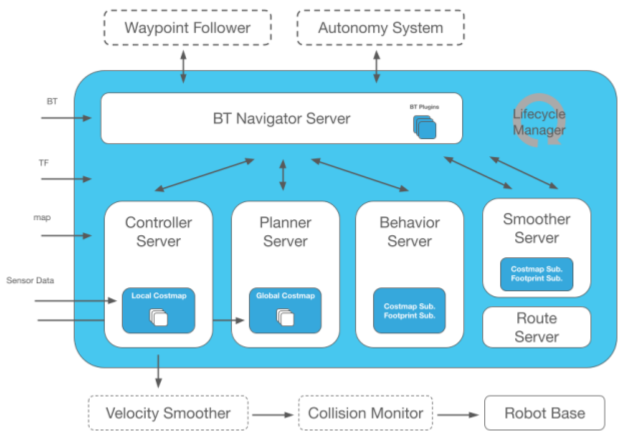
- Memory Management
  It uses a memory management mechanism to divide the map into working memory (WM), short-term memory (STM) and long-term memory (LTM) to limit the number of anchor points for closed-loop detection and graph optimization to ensure real-time.

- Backend (Global Map Optimization)
  The backend is where the system corrects accumulated errors in the map through global optimization. Once loop closures are detected, the system uses graph-based optimization techniques, such as g2o, to minimize inconsistencies between different parts of the map. This ensures that the final 3D map is accurate and globally consistent, even in large and complex environments.

- Map Generation
  After optimization, the system generates the final 3D map. This map consists of a dense or semi-dense reconstruction of the environment based on the 3D points obtained from the sensor. The map can be used for further navigation tasks, such as path planning or obstacle avoidance.

- Visualization: rtabmap_viz is the visual interface of RTAB-Map, which is the package of RTAB-Map GUI graphics library, similar to rviz but has options for RTAB-Map. It can subscribe to different topics, such as odom, rgb/image, depth/image, scan to display the process and results of SLAM, and load the 3D map into rviz at the same time.

NAV2 :



- Nav2 uses behavior trees to create customized and intelligent navigation behavior via orchestrating many independent modular servers. A task server can be used to compute a path, control effort, behavior, or any other navigation related task. These separate servers communicate with the behavior tree (BT) over a ROS interface such as an action server or service. A robot may utilize potentially many different behavior trees to allow a robot to perform many types of unique and complex tasks.
- The expected inputs to Nav2 are TF transformations conforming to REP-105, a map source the Static Costmap Layer, a BT XML file, and any relevant sensor data sources. It will then provide valid velocity commands for the motors
- It has tools to :
  - Load, serve, and store maps
  - Control the robot to follows the path and dynamically adjust to avoid collision
  - Convert sensor data into an environmental model of the world
  - A smoother on output velocities to guarantee dynamic feasibility of commands

B. Software Developed

- Raspberry Pi 5 Setup: I successfully set up the Raspberry Pi 5 running Raspberry Pi OS (64-bit), optimizing it for handling high-performance tasks such as autonomous navigation and sensor data processing.
- Docker Installation & ROS Foxy: Docker was installed on the Raspberry Pi to ensure containerized environments for the robotic software stack. I pulled and configured the ROS 2 Foxy base image, which was pivotal for running the various ROS nodes.
- Sensor Integration on RViz: Basic sensor integration was achieved using RViz, where different sensor plugins, including LIDAR, IMU, and encoders, were visualized and tested to ensure accurate environmental perception and robot state feedback.
- ESP32 Micro-ROS Integration: Micro-ROS was employed to interface the ESP32 with the Raspberry Pi 5. I established a serial communication pipeline to publish the MPU6050 IMU data from the ESP32 to ROS 2, which is used for orientation and motion feedback.
- Manual Control of Rover Using PS5 Controller: A PS5 controller was interfaced through the ESP32 to manually control the rover. This provided the ability to operate the motors and adjust navigation parameters manually, allowing precise control in non-autonomous scenarios.
- Autonomous Navigation using Nav2 and SLAM Toolbox: For autonomous navigation, I utilized the Nav2 stack in conjunction with SLAM Toolbox to map the environment in both simulated and real-world scenarios. The robot, equipped with 2D LiDAR, IMU, encoders, and a depth camera, was simulated in a custom-built Gazebo environment. The generated maps were saved and used for path planning and navigation tasks.
- Simulated Environment and Visualization: A customized simulation environment was created in Gazebo, including models of the robot and various obstacles. This simulation was visualized using RViz, where real-time mapping and localization outputs were monitored, ensuring a complete understanding of the robot's surroundings and navigation performance.

4. Additional Features in Robo-Making

- We are integrating BDIC(Battery diagnosis integrated circuit) which will help us calculate our batteries SOC and SOH.
- Modified mild steel wheels enhance rover robustness through optimized design, reinforcement, and surface treatments for improved strength and durability in diverse terrains.
- In case of electronic malfunctioning or over heating a automated circuit breaker is mounted.

5. Concurrence / Deviation from Level-1 Document with Reasoning

Stage-1 Report Comparison:

- In our initial stage-1 proposal, we aimed to perform environment mapping using LiDAR data. However, during research and simulations, it became clear that relying solely on LiDAR is not sufficient for a comprehensive environmental map, particularly in complex terrains. Based on our findings in the Gazebo simulation platform, we decided to incorporate a depth camera and utilize RTAB-Map SLAM for 3D mapping. This approach provides a complete overview of the crater and helps visualize obstacles in a 3D perspective.
- In the initial proposal, we assumed that obstacles would always be taller than the rover itself. However, after receiving updated guidelines, we realized that obstacle heights could vary significantly, from a few centimeters to several feet. Consequently, we adjusted our setup by using multiple LiDAR sensors positioned strategically, including both 1D and 2D LiDARs, to better detect varying obstacle heights.
- In the initial Level-1 document, standard wheels were proposed. However, after testing on rough terrain, specialised treads were added for enhanced traction. And also we are working on the wheels made up of mild steel.

Control Methods:

We proposed three methods for controlling the rover:

- Autonomous Control: The rover utilizes sensors like LiDARs, depth cameras, and encoders for environmental mapping and localization. Additional sensors like the MPU9250, GNSS, and encoders provide real-time feedback on the rover's status. This allows the system to autonomously navigate and avoid obstacles while maintaining real-time situational awareness.
- Semi-Autonomous Control: In this mode, the Raspberry Pi 5 serves as the central processing unit, receiving data from various sensors to generate motor control signals for localization and movement. These signals are relayed to an ESP32, which is connected to a PS5 controller. This setup allows for customized manual intervention while still maintaining an overview of the environment through real-time mapping.
- Manual Control: Manual operation is possible by directly connecting the PS5 controller to the ESP32, bypassing the Raspberry Pi 5. In this mode, the ESP32 generates motor control signals to move the rover in any desired direction without the need for the full system to be powered on.

Object Detection and Localization:
After conducting various tests with the rover, we identified the need for live telemetry and obstacle detection when operating from remote locations (>100m). Object detection not only assists in navigation but also enhances the localization capabilities of the rover by providing additional context for avoiding obstacles.

GPS and Heading Control:
By integrating a GNSS sensor along with a magnetometer, the rover can maintain a precise heading and avoid unnecessary turns. This system ensures accurate directional movement and aids in localization by providing real-time positional data.

Deviations:
The use of the Intel RealSense Depth Camera was a later addition, replacing earlier plans for a basic webcam, due to the need for better depth perception and obstacle recognition.

6. Additional Information

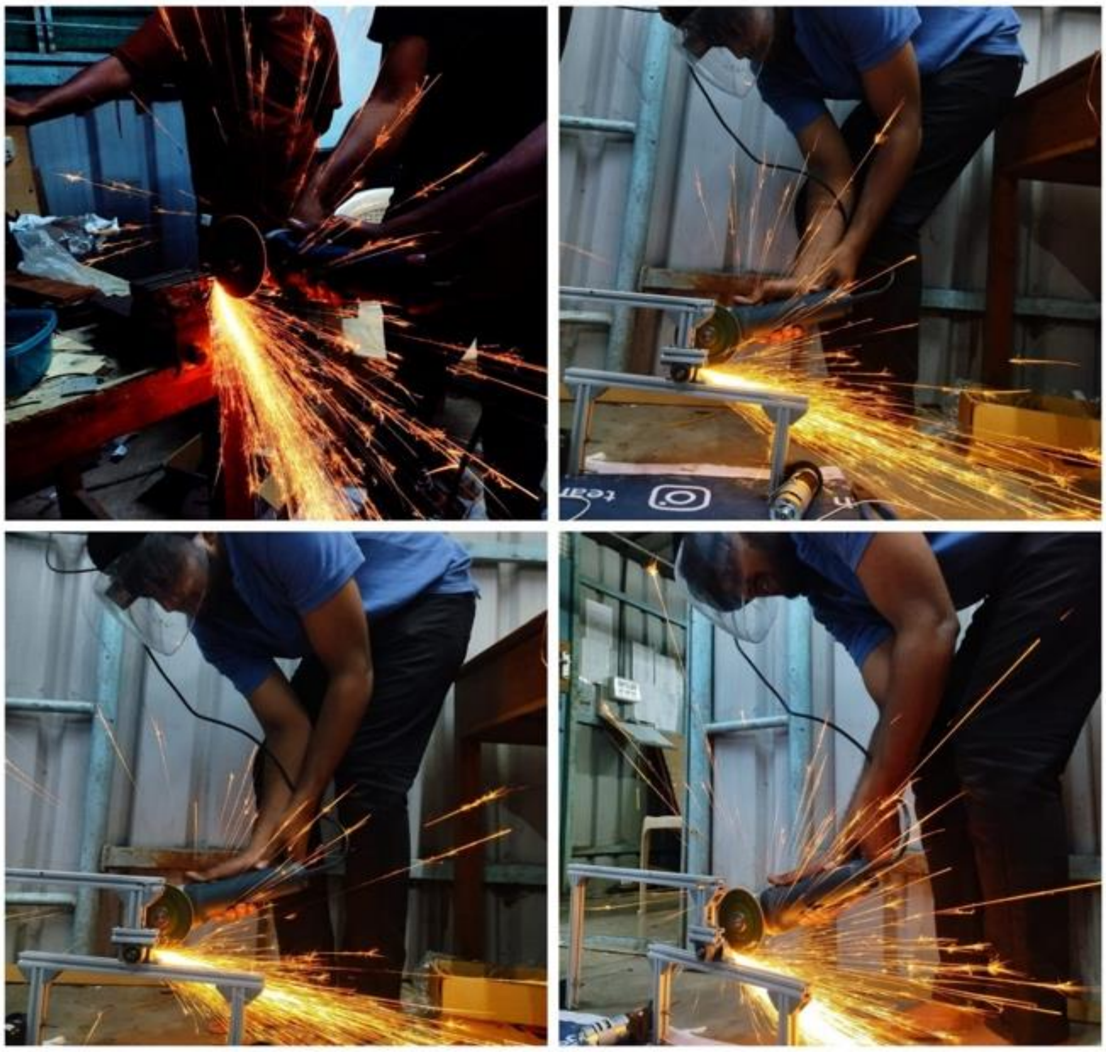
Power Consumption Analysis:

- The robot uses a 12V lithium-ion battery pack, with power consumption optimized for long-distance travel. Real-time monitoring of voltage and current draw ensures efficiency and longer battery life.
- Semi-Autonomous Control: If needed, the rover can be controlled manually using a joystick or laptop via wireless communication, allowing for operator intervention during difficult obstacles.

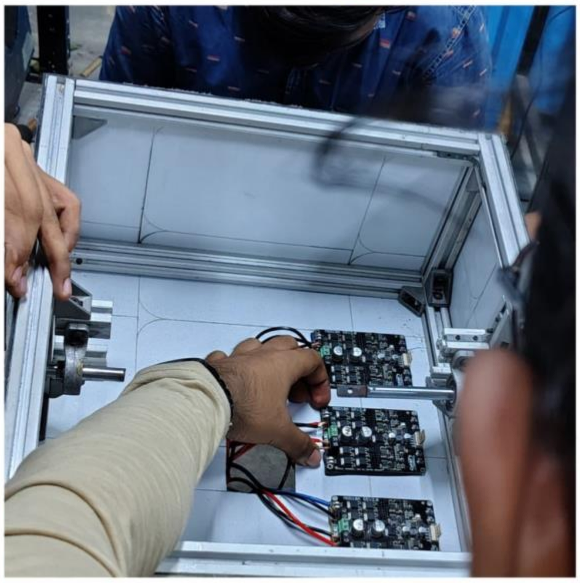
Procedure for making Rover:

- Prepare the rough sketch of the rover considering its dimensions and task to perform.
- Now design the rover in the CAD software e.g. in our case we have designed the bot in the solidworks.
- While designing we have used the material i.e. Aluminium extrusion so that we can get the required properties in the design
- In design take the extrusion and edit it via the command edit the sketch you can change the length of that extrusion, we have taken this length firstly as 15 cm height , 32 breadth cm and 40 length cm and designed the cuboid like structure which is our main body in which all the electronics will be mounted.
- After that for joining the parts use the command called mates and join all the parts and create the assembly.
- Likewise do for right side rocker bogie and left side rocker bogie .
- For analysis of this assembly open it in the simulations of solidworks and do the analysis and you will understand that your design is proper or not and how much load it can sustain
- After the simulations now add this file in gazebo and test the working of rover virtually after successfully working we can start the manufacturing .
- For starting the manufacturing the material selection is very important step , after selecting proper material i.e. aluminium extrusion start fabrication .
  - Take the extrusion rods and cut them according to the dimensions with the cutting machine .
  - Join the extrusion with help of corner beackets and inner brackets and use T nuts and Allen bolts to avoid the drilling operation.
  - Likewise the cuboid of the dimension 40 x 32 x 15 cm is ready.
  - After this make the suspension system i.e. rocker and boggie for each side .
  - The main part in the rocker boggie is the joint which is made up of pillow
  - bearing , Make the assembly of joint via the boggie and the rocker will be joined .
  - Now attach the main chasis and the suspension system via smooth rod .
  - Mount the PCB on the chaisis and attach the sensors and components on their specific location on the PCB board

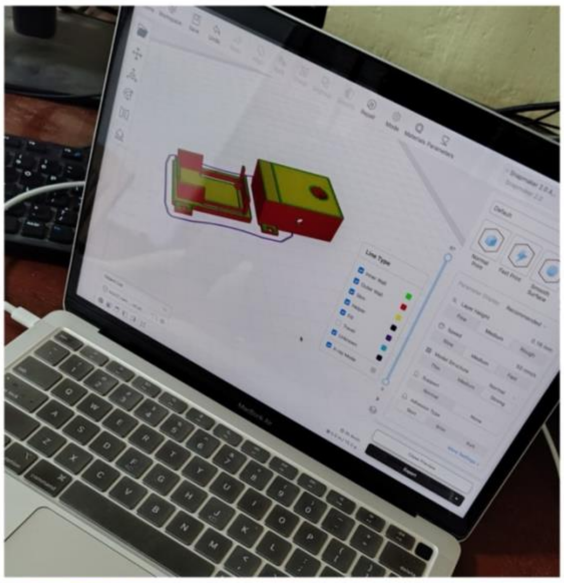
7. Photos of Robo-Making

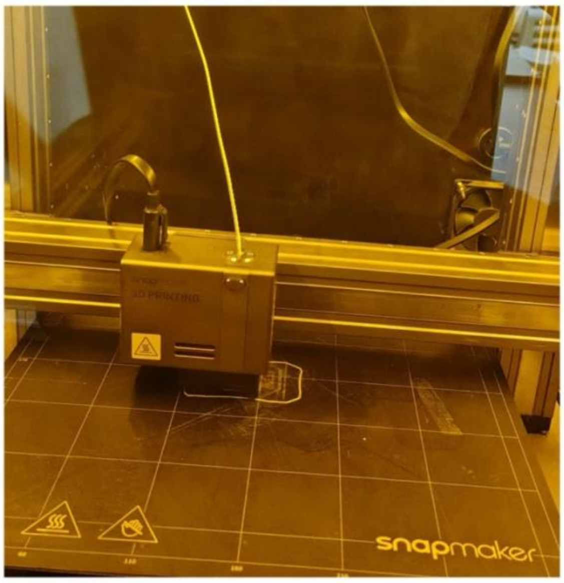


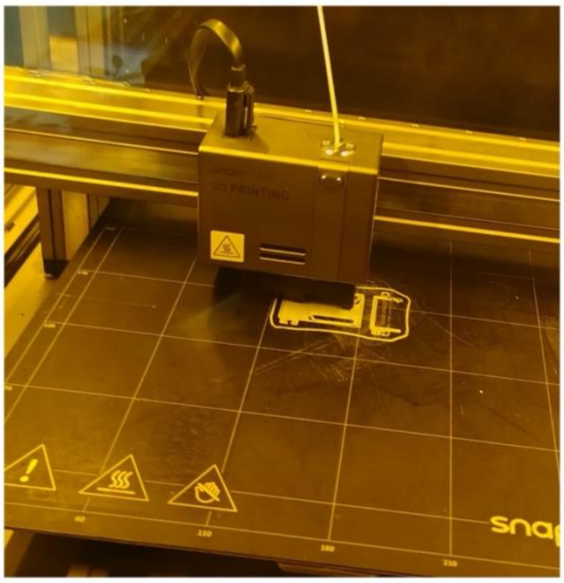
Construction of Rover Mechanics



Sachin
Education Consultancy
(A Perfect Career Guidance Junction)
Best Service Result







Videos :

[Drive Link for Videos]