Team RoboManipal
RoboCup@Work
Team Description Paper 2024

Team members names

Manipal Institute of Technology
Email:
Website: www.robomanipal.com

Abstract: This paper presents Team RoboManipal's solution to the challenges of RoboCup@Work. The underlying software framework, developed capabilities, and the hardware required for operation in industrial environments are described in detail. **K.A.R.M.A.** was entirely developed in-house by Team RoboManipal, featuring custom-designed electronics and an arm constructed from repurposed scrap metal.

1. Introduction

Team RoboManipal is the official robotics team of Manipal Institute of Technology (MIT), established in 2010 to pursue innovation and excellence in robotics. Comprising over thirty undergraduate engineering students from various disciplines, the team operates under the guidance of Dr. Ishwar Bhiradi, Assistant Professor in the Department of Mechatronics Engineering at MIT. Over the years, Team RoboManipal has become a center for pioneering research and development in robotics.

The team represents Manipal Academy of Higher Education (MAHE) in prestigious robotics competitions such as ABU Robocon and Technoxian (World Robotics Championship), competing on both national and international stages where they've had considerable success. These events challenge RoboManipal to design and build cutting-edge robotic systems, pushing the boundaries of technology while fostering a spirit of collaboration and technical expertise. RoboManipal's projects emphasize practical, real-world applications of robotics, focusing on areas like autonomous navigation, object manipulation, and intelligent control systems. With a commitment to iterative prototyping and innovation. By combining creativity with technical rigor, RoboManipal continues to make significant strides in robotics, both as a competitive force and a contributor to the field's advancement.

Table 1: History of RoboManipal

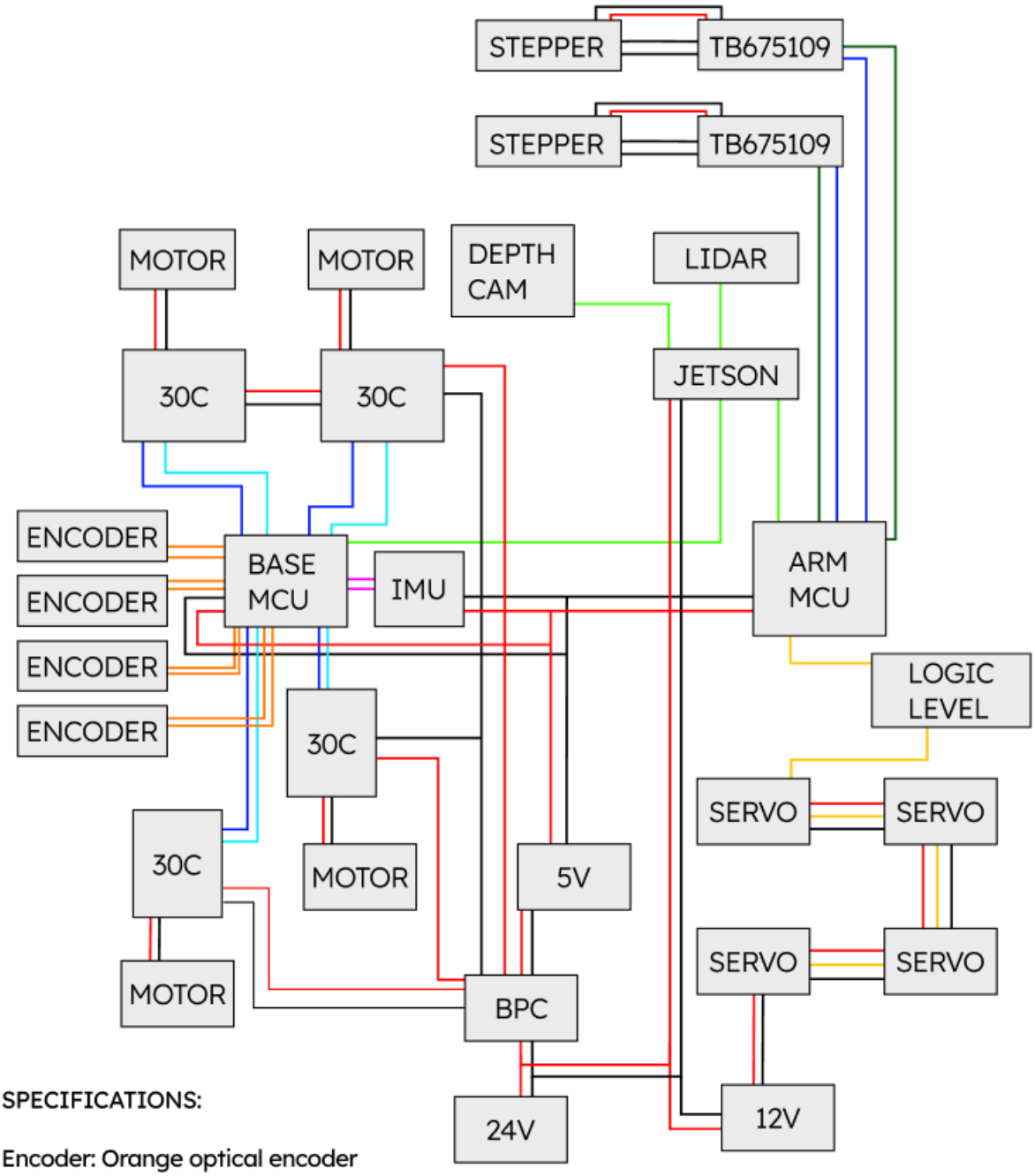
Year	**Location**	**Event/Competition**	**Result**
2016	Delhi, India	ABU Robocon	9th
2018	Ahmedabad, India	World Robotics Olympiad (WRO)	2nd
2021	Erode, India	Disenyo PCB Challenge	1st
2023	Delhi, India	RoboSoccer at Technoxian, World Robotics Championship	3rd
2023	Delhi, India	ABU Robocon	21st
2024	Delhi, India	ABU Robocon	22nd

(insert team list)

2. Hardware Description

2.1. Robot Control Flow

Hi



SPECIFICATIONS:

Encoder: Orange optical encoder
30C: Cytron MDD30C
Motor: Rhino DC motor
IMU: Bosch BNO055
Depth Cam: StereoLabs ZED 2i
LIDAR: Rplidar a2m8
24V: 24V LiPo Battery
Jetson: NVIDIA Jetson Orin
5V and 12V: DC-DC Buck Converter
Base and Arm MCU: STM32F411
Servo: WaveShare Serial Bus Servo
Stepper 1: NEMA 23
Stepper 2: NEMA 17

SDA/SCL
A/B
Power
Ground
Step
USB
DIR
Half Duplex UART
PWM

Fig.1 : This diagram shows the robotic system's electrical connections, featuring motors, sensors, MCUs, and power distribution, with key components like Cytron drivers and NVIDIA Jetson Orin. Color-coded lines indicate power and signal flow

The bot's power electronics and hardware are designed for efficiency and reliability, featuring a custom control board with an integrated power distribution system. This all-in-one board simplifies the management of power flow throughout the bot, reducing wiring complexity and enhancing performance. The system utilizes two 24V batteries, ensuring sufficient energy storage for prolonged operation. To meet the power requirements of various components, we have incorporated 200W buck converters, which efficiently step down the voltage to the required levels. This setup ensures that the bot can operate smoothly under varying loads, providing a stable and robust power solution for all its functions.

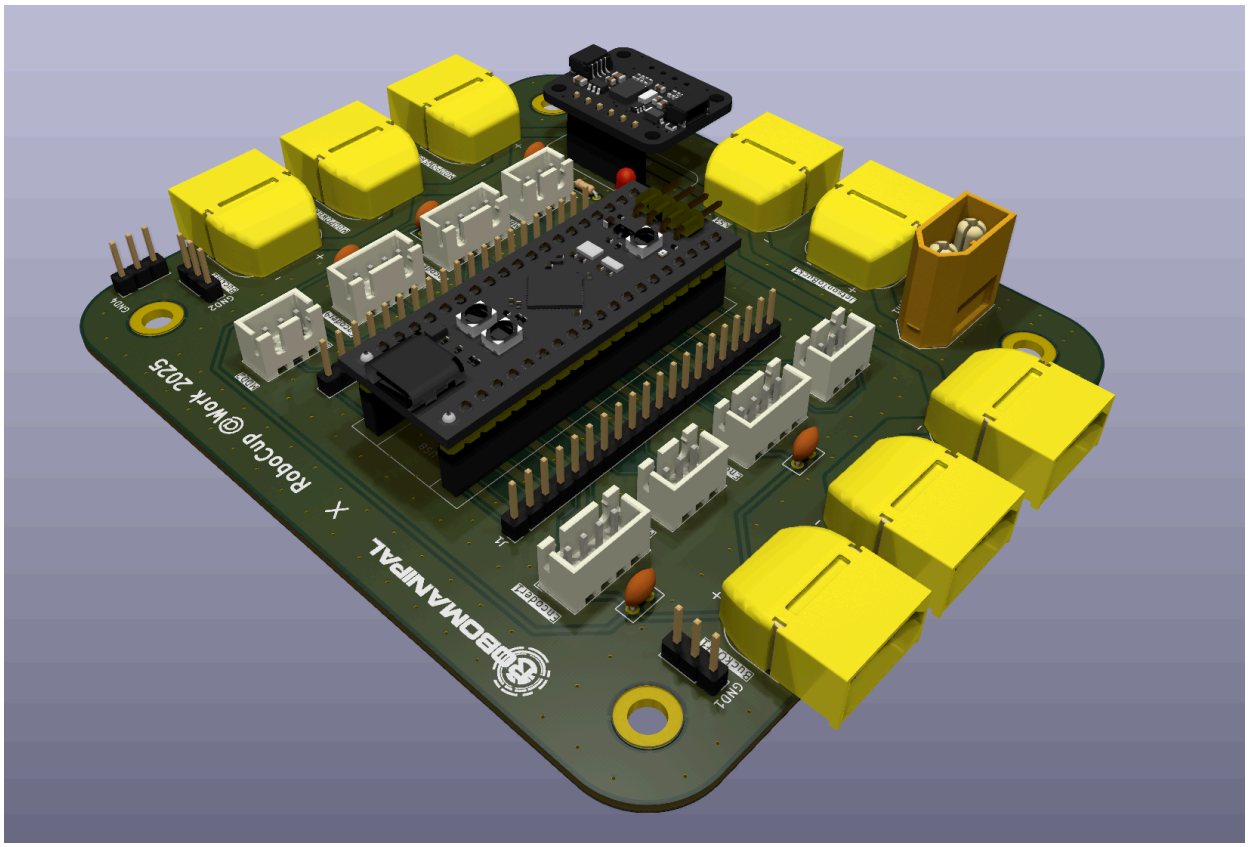


Fig. 2: a) Custom Control and Power Distribution Board for the Base

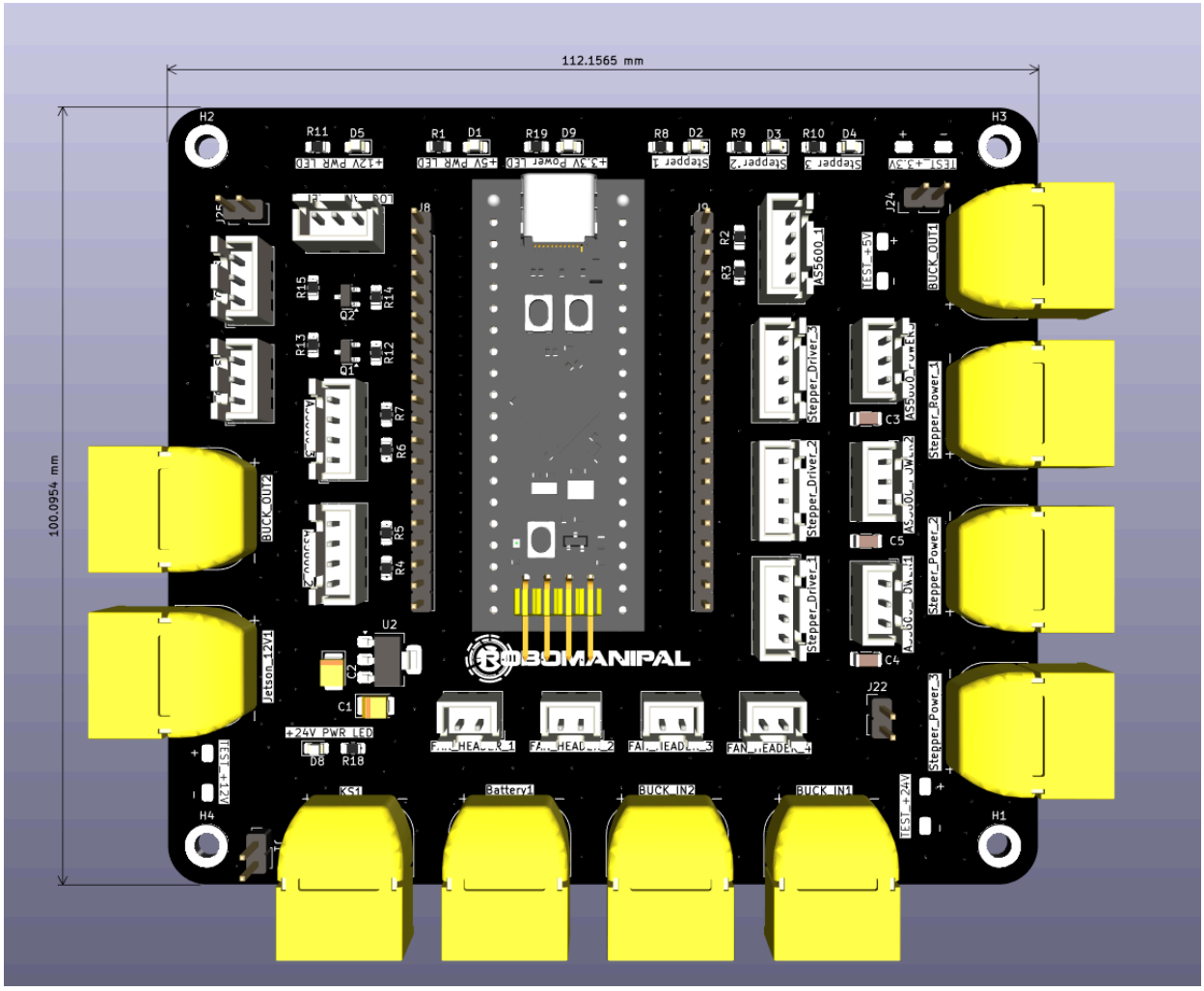


Fig. 2: b) Custom Control and Power Distribution Board for the Arm.

Table 3: Hardware Components and Pricing Breakdown:

Components Used	**Price in Indian Rupees**	**Price in USD***
Nvidia Jetson Orin	80000	915.97
STM32F411 MCU	592	6.79
RpiLIDAR	23,599	270.20
ZED 2i Depth Cam	43121.26	493.72
MDD30C Motor Driver	10800	123.65
Rhino IG52	19116	218.87

Waveshare ST3215/Dynamizel 18A	21548.45	246.72
Orange Optical Encoders	8800	100.75
AS5600 Contactless Potentiometer	298	3.41
TB67S109 Stepper Motor Driver	1212	13.87
Logic Level Shifter	12	0.14
LiPo Batteries	23999	274.79

*The prices from INR to USD were calculated based on 5th February 2025 conversion rate.

2.2. Safety Features

2.2.1. BPC (Battery Protection Circuit):

We developed a custom low-voltage cutoff circuit designed to protect Li-Po batteries from over-discharge. This circuit continuously monitors the battery voltage and automatically disconnects the power supply if the voltage falls below a predefined safe threshold. By preventing the battery from being discharged beyond its limit, this circuit helps to prolong the battery's lifespan and ensures safe operation of the connected devices.

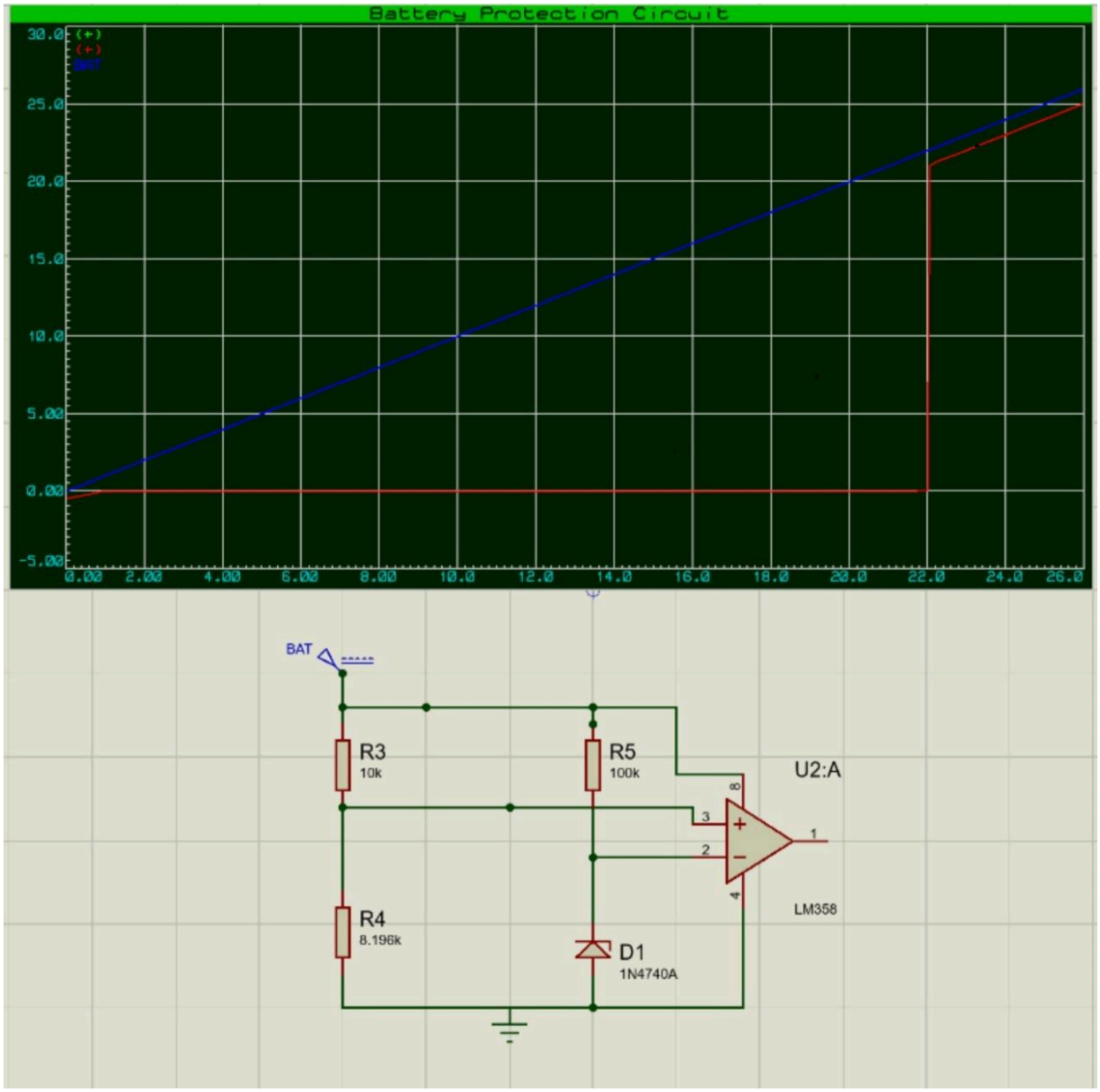


Fig. 3: Simulation of the Battery Protection Circuit for a 24V LiPo on Proteus

2.2.2. Serial bus servo:

Fig. 4: Dynamixel AX-18A

Serial bus servos, like the Dynamixel servos, are equipped with multiple safety features to ensure reliable and secure operation. Running on in-house custom firmware that operates UART in half-duplex mode, these servos offer enhanced control and communication efficiency. Overload protection prevents damage by halting operation when excessive load or torque is detected, while overheating protection actively monitors internal temperatures and stops functionality if safe limits are exceeded. They also provide motion range limitations, allowing users to define safe angle boundaries to prevent mechanical damage. Additionally, overcurrent protection limits abnormal current surges, and voltage protection safeguards against undervoltage or overvoltage conditions, ensuring stable operation. Position error detection further ensures accurate responses by identifying discrepancies between the desired and actual positions.

2.3 Structural Design

The design of the arm was created using Fusion 360. Fusion 360 provides optimized performance and methods for dynamic modeling to understand the movement of the arm. It is useful for creating ergonomic or organic shapes, such as smooth joints or enclosures for components, which is crucial for ROS implementations. Fusion 360 also allows you to simulate the motion of the robot arm, ensuring that it performs as expected and avoids collisions between parts.

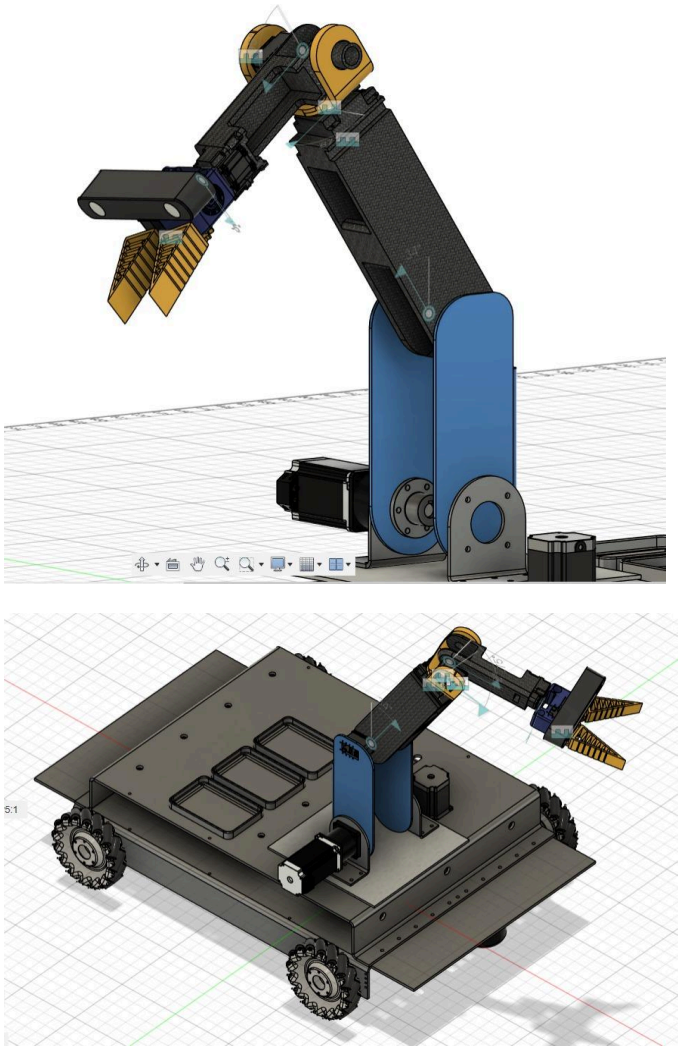


Fig 5: Computer Aided Design of the Bot

Our robot is built on a robust four-wheel Mecanum drive platform, enabling omnidirectional movement and precise maneuverability. The primary construction material for the robot is aluminum 6061(Alloy 61S), chosen for its lightweight properties, strength, and corrosion resistance. No gearboxes have been used with the motors driving the wheel, since enough torque was achieved by the selected motors.

For the first degree of freedom (DOF) of the arm, the load of the entire arm was initially exerted directly on the motor shaft, leading to potential reliability and performance issues. To address this, we integrated a timing pulley system supported by bearings. This setup effectively transferred the load away from the motor shaft, ensuring enhanced durability and smoother

operation. Additionally, a jockey was employed to enhance tension on the belt, which helped prevent slippage during high torque requirements.

Initially, the arm links were designed with 3mm aluminum, but it was later changed to polylactic acid (also known as PLA) due to its lightweight nature and exceptional dimensional stability. This material also reduced current draw from the servo motors, helping to prevent them from overheating quickly during operation.

2.3.1. Gripper Design

The design of the gripper is intentionally simple and straightforward. It is primarily constructed from polylactic acid (PLA), while the fingers of the gripper are made from thermoplastic polyurethane (TPU). TPU is chosen for its excellent flexibility and grip, making it highly suitable for soft robotic applications. This material allows the gripper to effectively handle a wide variety of objects during pick and place operations. Additionally, the design features finger mounts that resemble a spur gear, enabling simultaneous actuation of the fingers using a single servo motor. This mechanism enhances efficiency and functionality, making the gripper versatile and effective for various applications.

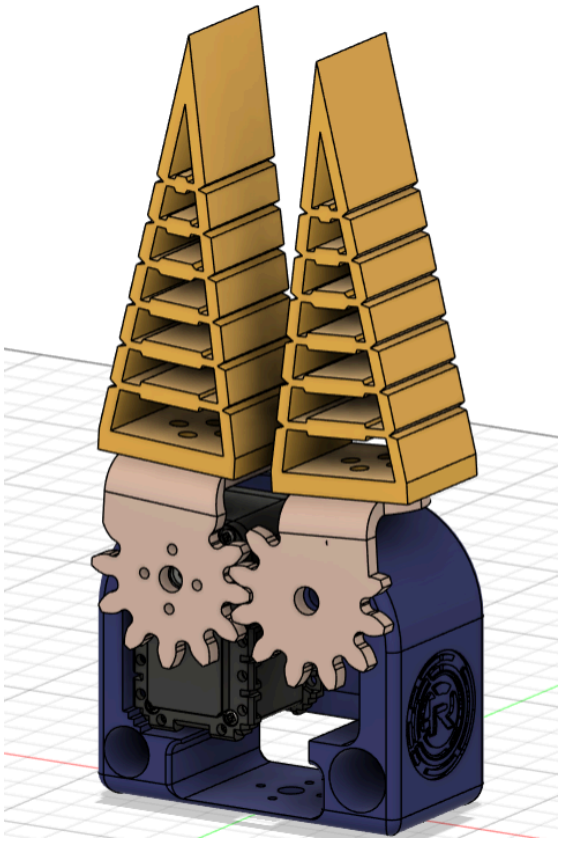


Fig 5: Optimised Gipper Design

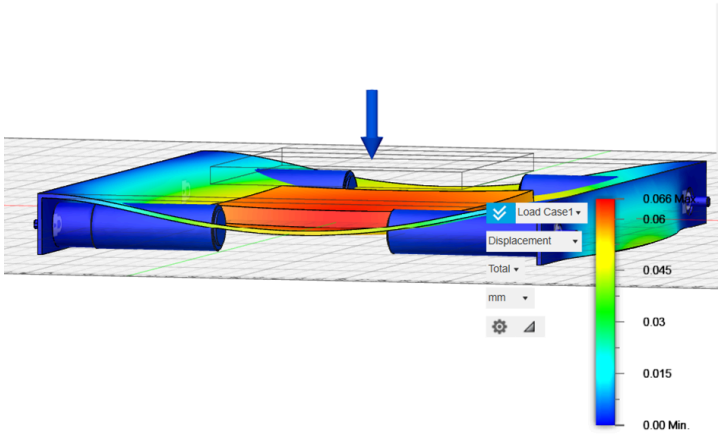


Fig 6: Load Bearing Simulation

3. Software Description

3.1 Control Systems

In-House Closed-Loop Stepper Motor Control

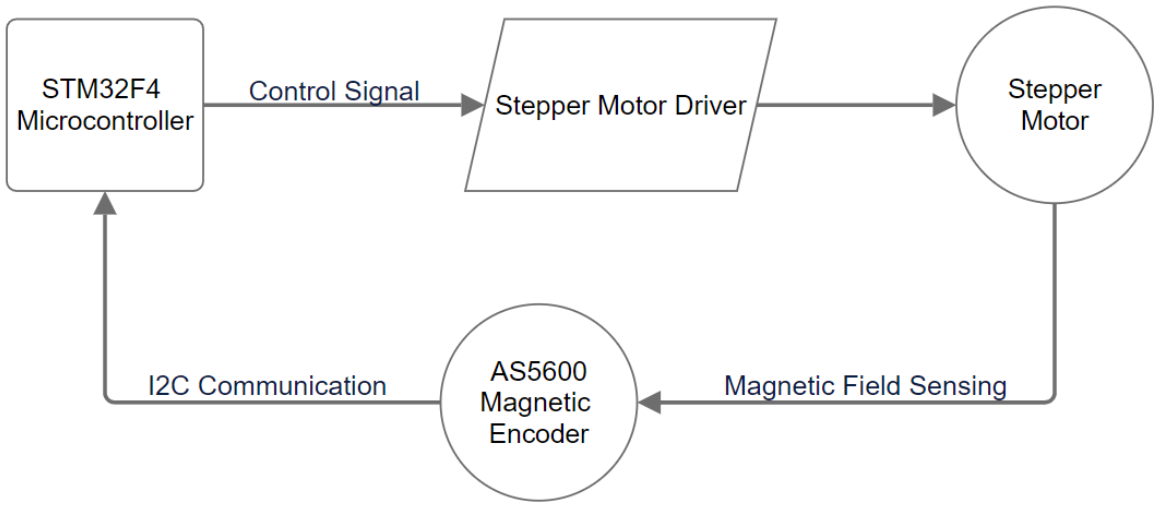


Fig. 5: Closed loop feedback utilizing magnetic encoder (wireless potentiometer).

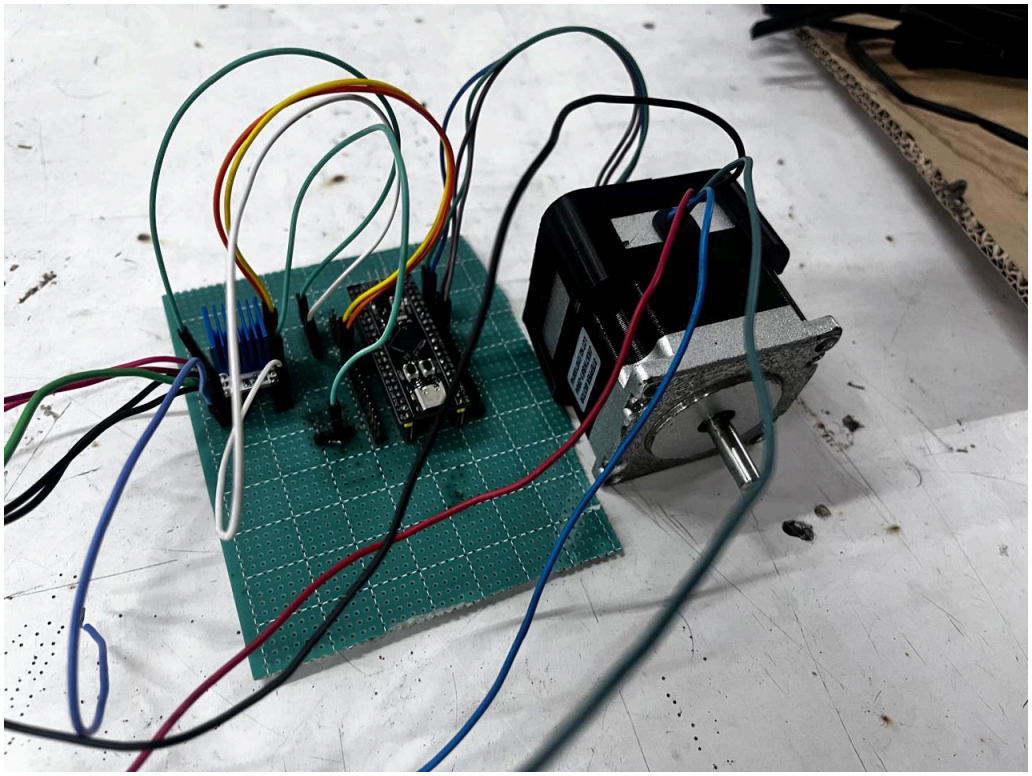


Fig . 6: Testing setup for working stepper feedback.

We developed an in-house closed-loop stepper motor control system with PID regulation and magnetic feedback for precise motion and dynamic torque control. Running on an STM32F4, it ensures real-time position correction, reducing step losses and enhancing efficiency for high-precision robotics applications. It controls the first two Degrees of Freedom.

Serial bus servo motor control

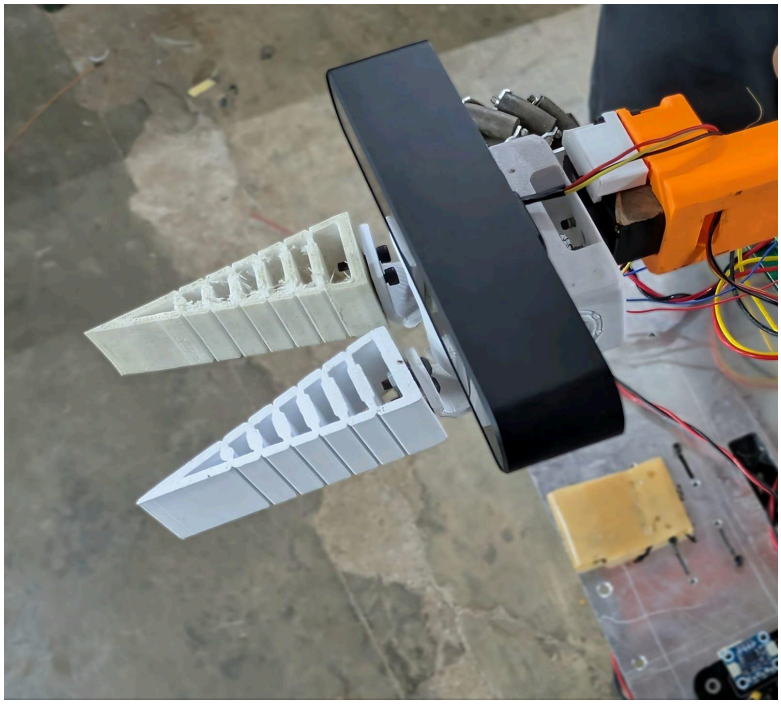


Fig. 7: Testing of our gripper

We used Dynamixel serial bus servo to control 3 DOF of the 6 DOF robotic arm. The servo motors are powered and managed by the STM32F4 microcontroller, which is specifically configured for this application. To facilitate communication between the microcontroller and the servo motors, we developed an in-house custom firmware that operates UART in half-duplex mode. This library is equipped with error detection features, ensuring reliable and accurate data transfer over a single communication line. This setup provides precise and dependable control over the robotic arm's movements.

IMU and Odometry Integration

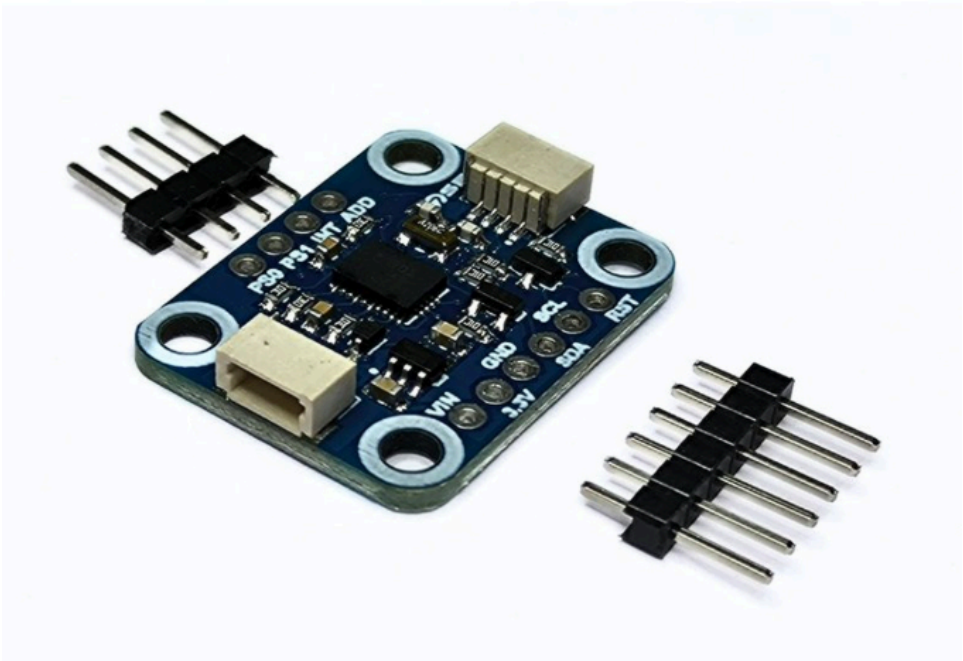


Fig. 8: Bosch BNO055 for accurate quaternions.

Initially, we used the MPU6050 IMU with a custom library for its DMP functionality on STM32, but its accuracy and drift issues led us to switch to the Bosch BNO055. The BNO055's onboard sensor fusion significantly improved orientation stability and reduced drift.
Integrated with four optical encoders, the IMU provided precise odometry data, processed in real-time on the STM32. This data was sent via USB to the Nvidia Jetson for SLAM and navigation, enabling accurate positioning and path planning in dynamic environments.

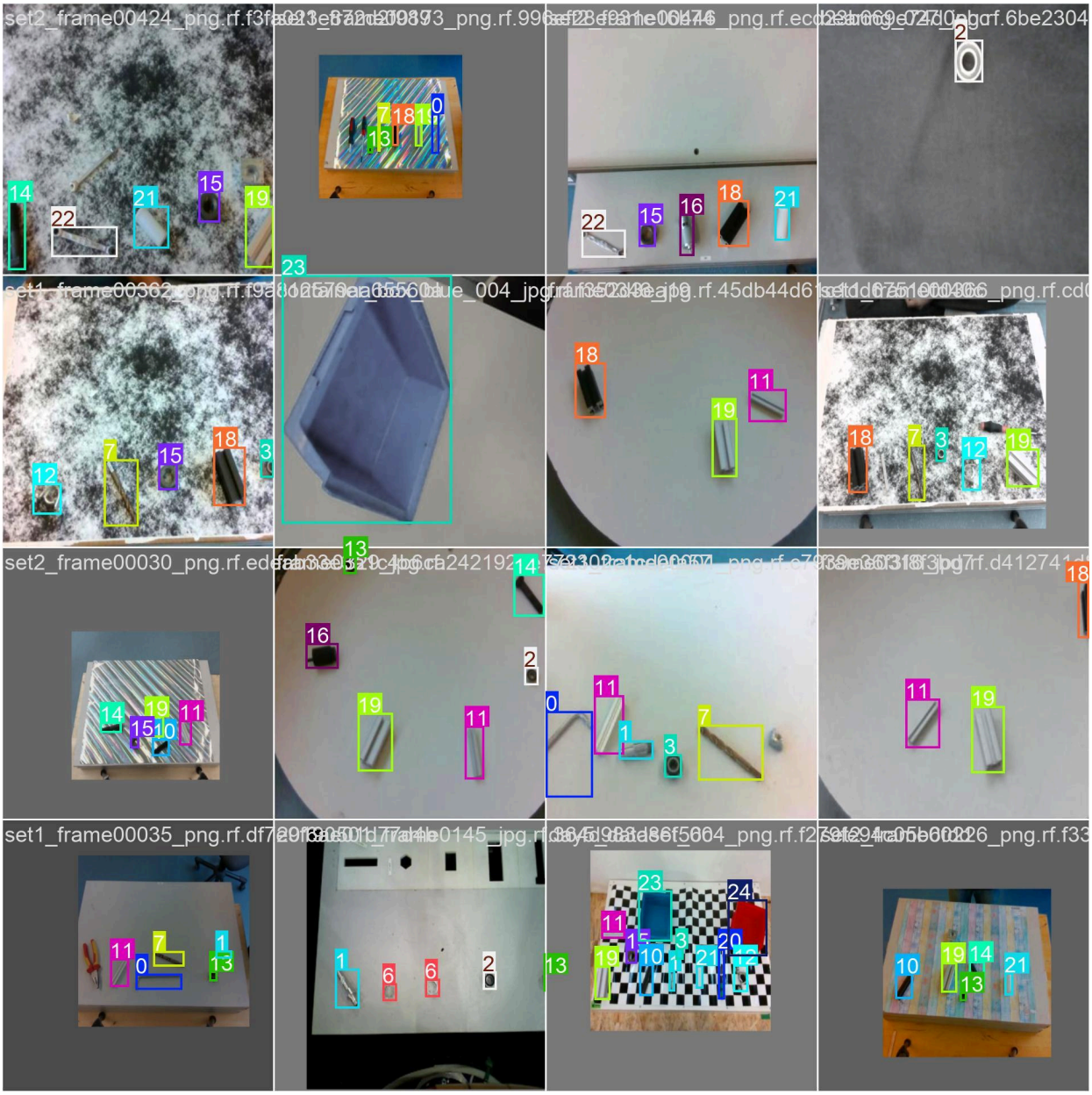
3.2 Pose

YOLOV8

The robot employs a ZED RGB-D camera, strategically mounted near the end-effector of its robotic arm, for object detection and recognition tasks. The camera's captured images are processed using YOLOv8, a state-of-the-art neural network optimised for real-time object detection. This model has been trained on the RoboCup Objects 2024 dataset, enabling the identification of objects across 22 distinct classes and providing precise coordinates of detected objects in real-time, as depicted in Figure 6.

To enhance inference speed and simplify integration with C++ frameworks, the trained YOLOv8 model is converted into the ONNX (Open Neural Network Exchange) format. This conversion ensures compatibility with highly efficient inference engines, significantly boosting the real-time performance of the robot. Additionally, the streamlined training process, which takes approximately two hours on a GPU,

allows for rapid retraining to adapt to new objects at competitions, ensuring the robot's detection capabilities remain up-to-date and effective.



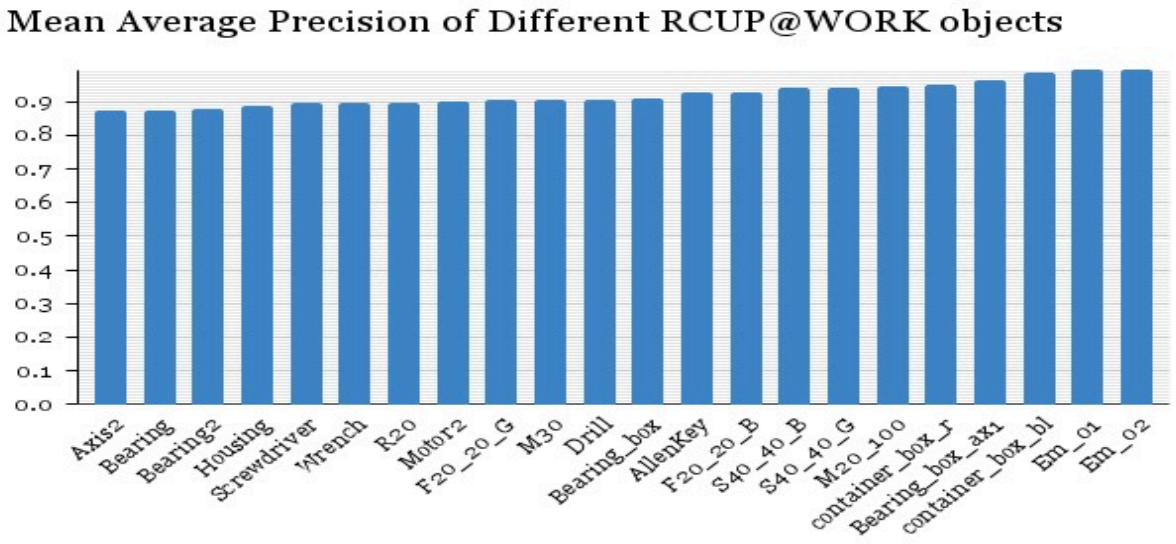


Fig. 9: a) Multiple object recognition using YOLOv8. b) Mean average precision of different RCUP @Work objects.

BarrierTape Detection

The vision system detects and highlights specific objects in video frames using OpenCV. It processes each frame by converting it to the HSV colour space and creating binary masks for regions of interest based on predefined thresholds. The masks are refined with morphological operations, including opening to remove noise and closing to fill gaps. Contours are then extracted and combined if they are close to each other, using bounding boxes to merge nearby regions. Final bounding boxes are filtered based on their aspect ratio and size to eliminate non-relevant shapes. Detected regions are labelled and displayed on the video feed in real-time, showcasing the system's ability to analyze and classify objects dynamically.

Empty Space Detection

To place an object, the robot identifies empty areas in the workstation by processing 3D point cloud data from the RGB-D camera. Edge detection and Hough Transform techniques are used to detect horizontal planes. The data is converted to grayscale, edges are detected, and lines marking boundaries of potential free spaces are identified. Contour approximation refines these spaces into rectangles to evaluate their suitability.

The system then evaluates neighbouring areas around candidate points, ensuring sufficient free space. This process is repeated until at least two viable placement locations are identified.

3.3 Movement

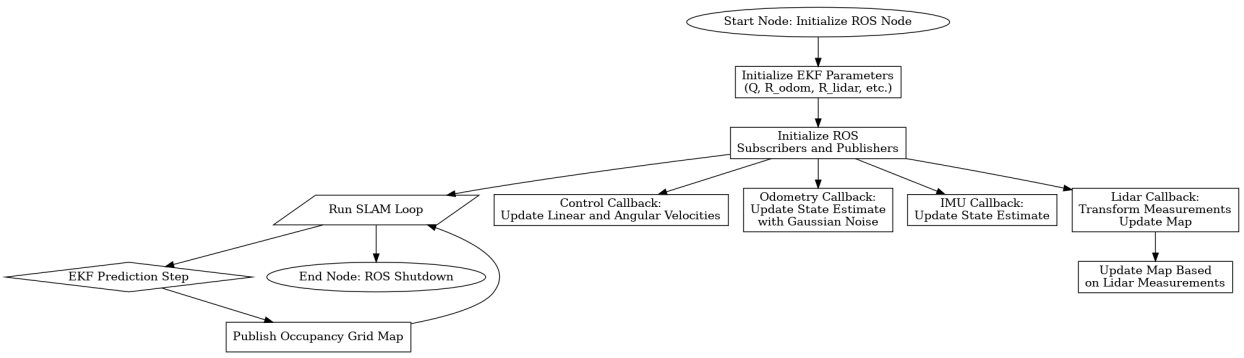
SLAM

The **RPLiDAR A2M8** is utilized to gather approximately **180-degree** laser scan data of the robot's surroundings. This data facilitates **obstacle detection** and **environment mapping**. The scan data is processed by the **slam_toolbox**, which generates a real-time map of the environment, a critical component for autonomous navigation.

Simultaneously, the **Nav2 framework** is employed for **path planning** and **motion control**, leveraging the map generated by SLAM. To ensure precise localization within the map, the **Adaptive Monte Carlo Localization (AMCL)** algorithm is integrated into the system. Additionally, an **Extended Kalman Filter (EKF)** fuses data from multiple sensors, including wheel odometry and IMU measurements. This fusion reduces the effects of noise and drift in individual sensors, significantly enhancing localization accuracy.

For visualization and monitoring, **RViz** provides a graphical interface displaying the robot's position, map updates, and navigation progress. To ensure seamless integration of sensor data and navigation logic, static and dynamic transforms maintain proper alignment and communication between various frames, such as the base link, laser, and odometry.

This comprehensive setup enables the robot to **autonomously navigate** unknown environments, effectively **avoiding obstacles** and following planned paths with **high precision**.



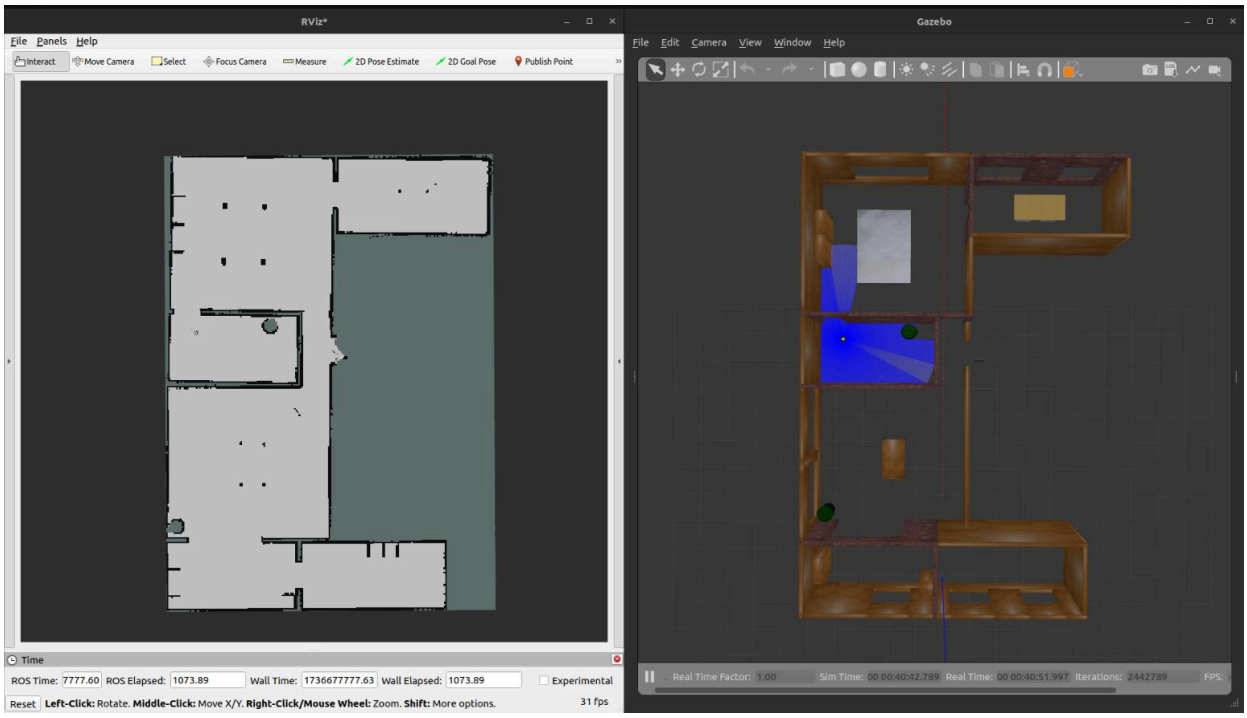


Fig. 10: a) Flowchart representation of EKF used to understand SLAM. b) the worlds were completely mapped using the slam_toolbox package.

Navigation with AMCL

The **Nav2 stack** facilitates autonomous navigation by allowing the robot to navigate to a specified Nav goal pose within the environment. The process begins with the user defining a goal position and orientation on the map, typically through a graphical interface like **RViz**. The Nav2 stack leverages the map generated by the SLAM Toolbox which is preloaded as a static map for global planning.

The **Global Planner** computes an optimal path to the goal pose, considering static obstacles, while the **Local Planner** dynamically adjusts the path to account for moving obstacles and environmental changes. It also ensures smooth and collision-free motion by generating velocity commands for the robot. This robust navigation framework enables the robot to reach the designated 2D goal autonomously pose efficiently and reliably, even in dynamic and complex environments.

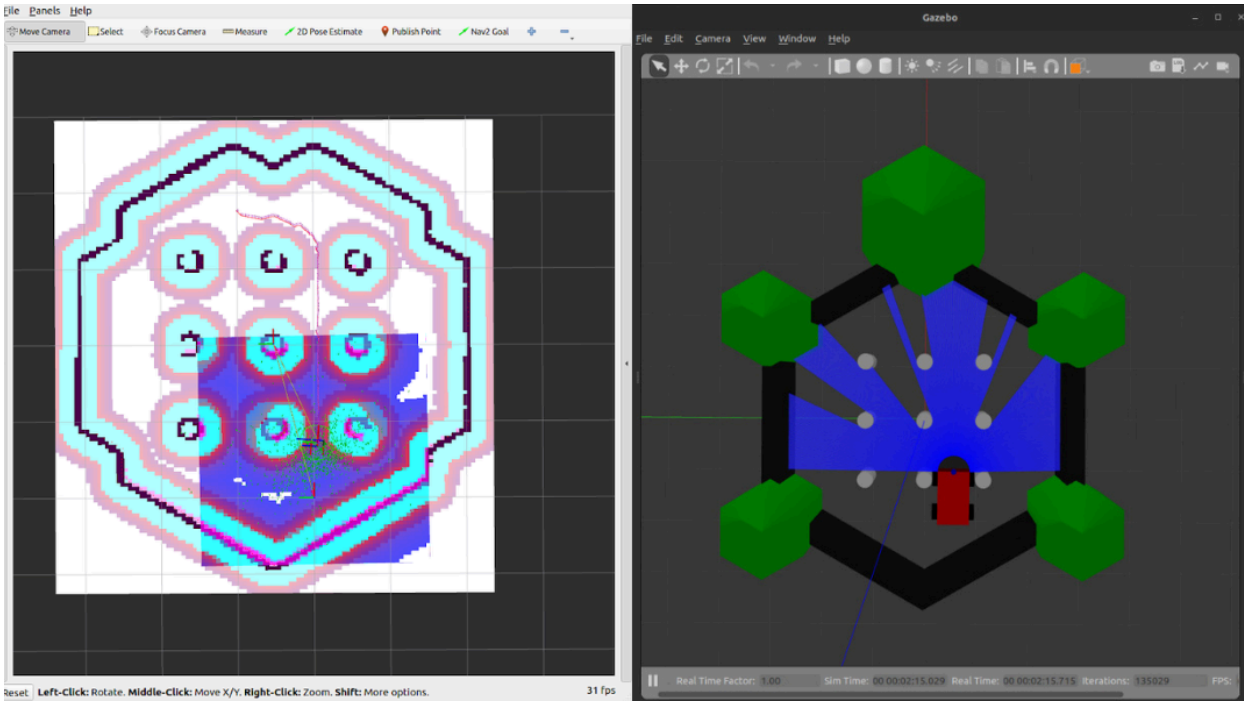


Fig. 11: movement of robot based on the 2D goal pose provided by Nav2 stack.

The rqt_graph provides a high-level visualization of the ROS2-based navigation stack, showcasing the interaction between nodes, topics, and action servers. At its core, the system integrates sensor data, localisation, mapping, path planning, and motion control to enable autonomous navigation. Nodes such as **/slam_toolbox, /amcl,** and **/ekf_filter_node** play crucial roles in processing laser scans, odometry, and IMU data to create accurate maps and state estimates. These outputs are essential for real-time localization and obstacle detection, forming the foundation for efficient navigation.

The system employs a dual-layer costmap strategy, with **/global_costmap** handling long-term path planning and **/local_costmap** focusing on real-time obstacle avoidance. The **/controller_server** node coordinates motion control, while nodes like **/follow_path** ensure precise execution of planned paths. Behavior tree-based nodes, such as **/bt_navigator_navigate_to_pose_rclcpp_node**, oversee the navigation process and handle recovery actions like spinning or backing up when obstacles disrupt the robot's path. This modular approach ensures flexibility and reliability in dynamic environments.

The architecture is designed for seamless integration, with components like **/map_server** and **/lifecycle_manager_localization** ensuring proper initialization and data flow. Visualization tools like RViz allow real-time monitoring of the robot's position, map updates, and navigation status, enhancing system usability and debugging capabilities. The use of sensor fusion through the **/ekf_filter_node** minimizes noise and drift, ensuring accurate localization and smooth navigation. Overall, this architecture balances efficiency, scalability, and robustness, enabling autonomous robots to operate effectively in complex and dynamic environments.

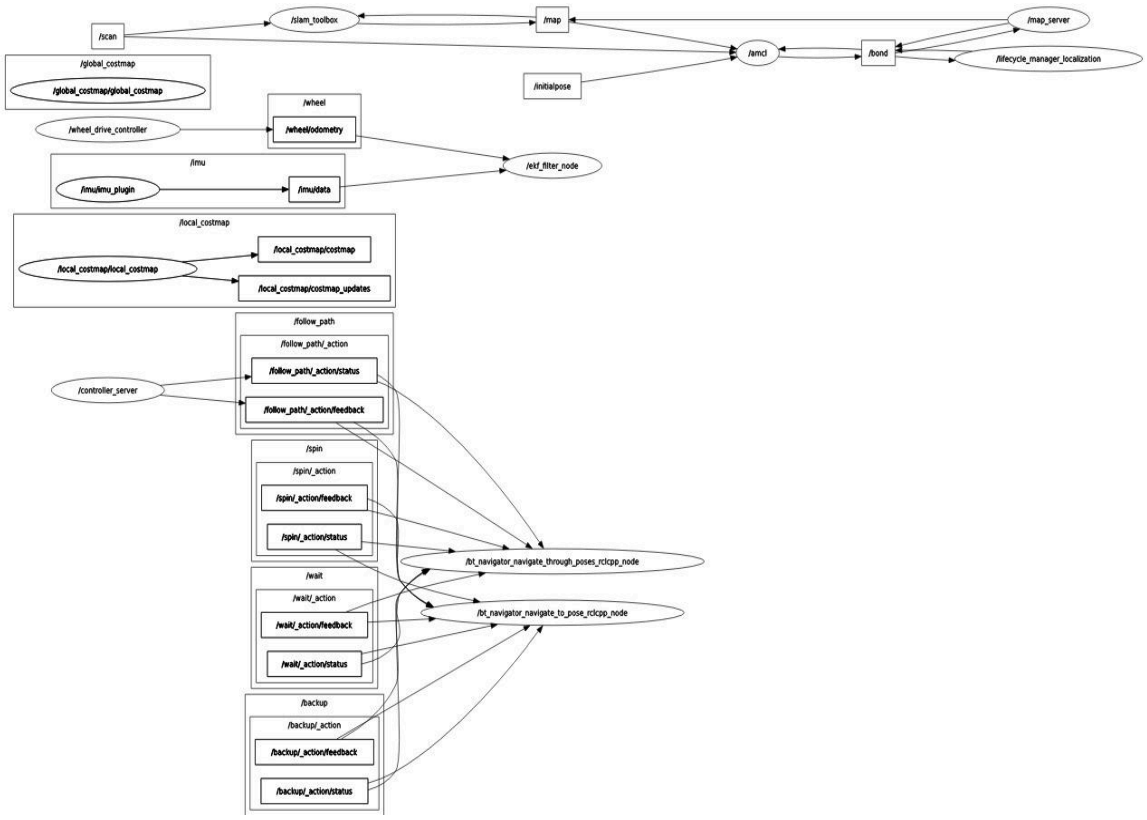
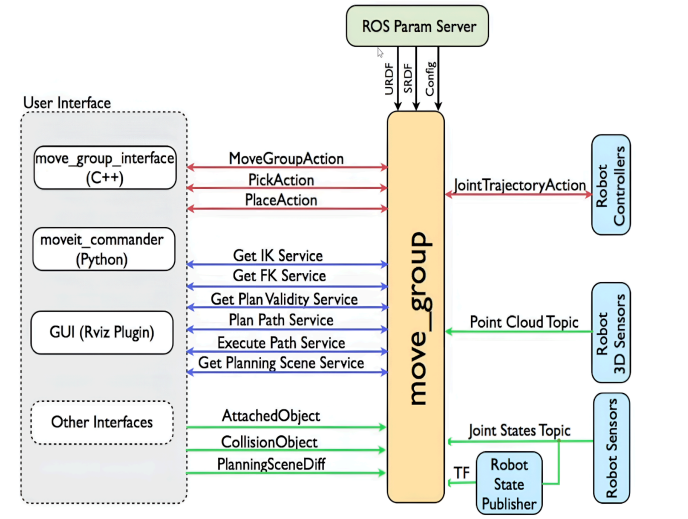


Fig.12: rqt_graph of the overall robot.

3.4 Arm Control

MoveIt2

An advanced open-source motion planning framework initially developed by Willow Garage, has been chosen for implementing arm kinematics. This robust framework integrates key functionalities such as **motion planning**, **kinematics**, **collision detection**, and **dynamic 3D environment representation**, enabling precise and efficient robotic arm control.
Tailored for **ROS2**, MoveIt2 ensures seamless compatibility with existing ROS2 packages and tools, including **RViz**, which enhances visualization and interaction capabilities. Its modular design and extensive features make it a powerful solution for complex robotic applications, providing reliability and ease of use in dynamic environments.


ROS Param Server
URDF
SRDF
Config
User Interface
move_group_interface (C++)
moveit_commander (Python)
GUI (Rviz Plugin)
Other Interfaces
MoveGroupAction
PickAction
PlaceAction
Get IK Service
Get FK Service
Get Plan Validity Service
Plan Path Service
Execute Path Service
Get Planning Scene Service
AttachedObject
CollisionObject
PlanningSceneDiff
move_group
JointTrajectoryAction
Robot Controllers
Point Cloud Topic
Robot 3D Sensors
Joint States Topic
TF
Robot State Publisher
Robot Sensors


/home/soham/ws_moveit2/install/moveit2_tutorials/share/moveit2_tutorials/launch/panda_moveit_config_demo_empty.rviz* - RViz
Camera
Measure
2D Pose Estimate
2D Goal Pose
Publish Point
Views
Stored Scenes
Save
Right-Click/Mouse Wheel: Zoom. Shift: More options.

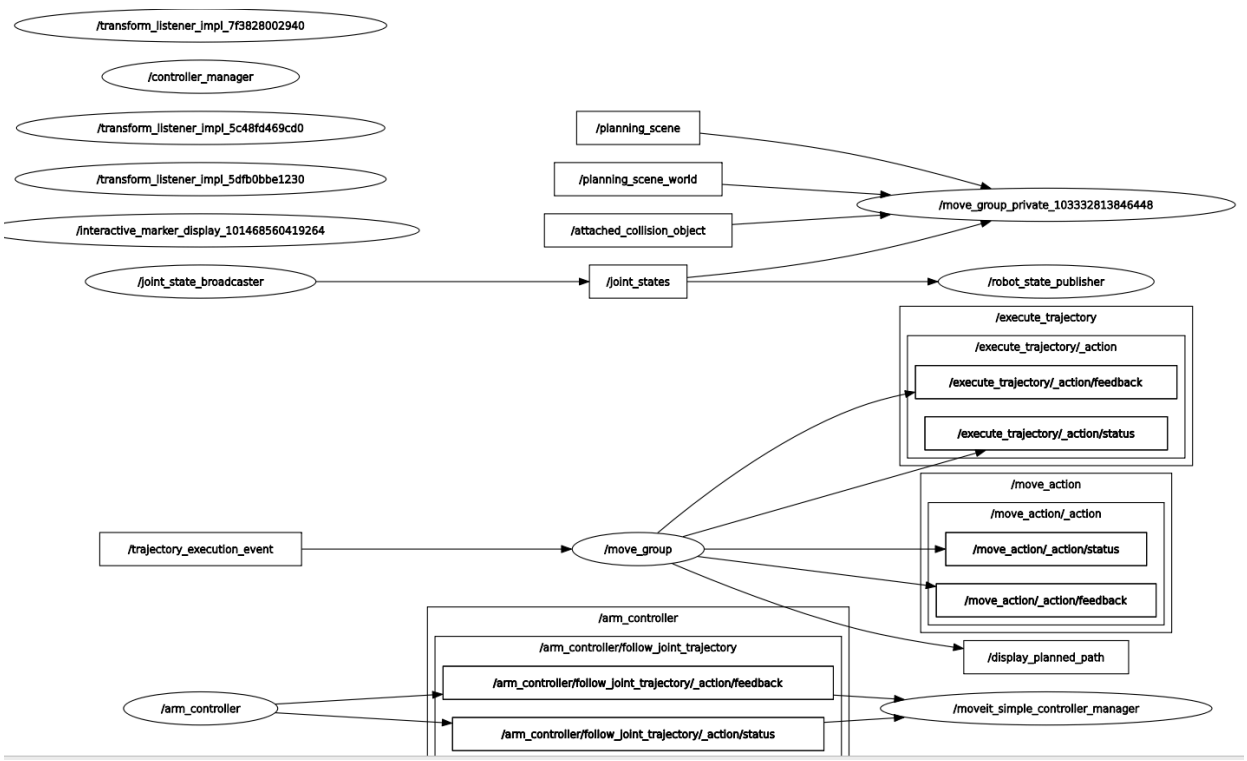


Fig. 13: a) workflow of moveit2 [1] b) rviz visualization of robotic arm using moveit2 c) RQT Graph for MoveIt2 simulation

The **StereoLabs Zed-2i depth camera** serves as the vision system, delivering **point cloud data**, **6-DOF pose information**, and **plane segmentation outputs**. This rich dataset is seamlessly integrated into the **inverse kinematics module** within **MoveIt2**, enabling the robotic arm to perform precise **pick-and-place operations**. The integration allows the system to identify and interact with objects in real-world environments with high accuracy. Furthermore, MoveIt2's compatibility with **RViz** is leveraged to simulate and visualize the arm's movements in real-time, ensuring accurate planning and reliable execution of tasks.

https://drive.google.com/file/d/1bzEw0gcjW7TGYpTW_1DHmGaHJLuxbOSd/view?usp=drive_link
Simulation of *arm_xyz* in Rviz using MoveIt2

4. Task Planner

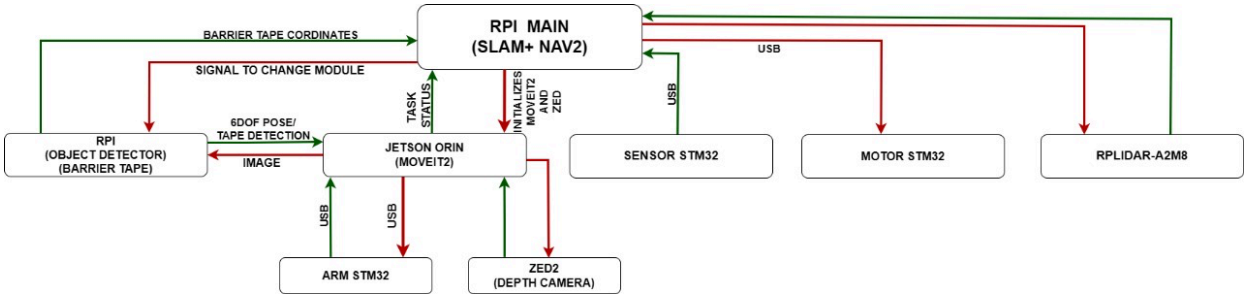


Fig. 14: Task Planner for proper completion of given tests.

Our RoboCup robot features a distributed computing architecture utilizing two Raspberry Pi 5s, a Jetson, and multiple STM32 microcontrollers to efficiently handle SLAM-based navigation and object manipulation.

Primary Raspberry Pi (Main Controller)

- Runs **SLAM and the Nav2 stack** to handle navigation.
- Directly connected to the **LiDAR**, which provides real-time mapping and localization data.
- Interfaces with two STM32 microcontrollers:
  - **Motor Control STM**: Processes commands from the RPi to drive the motors.
  - **Sensor Fusion STM**: Collects data from the **IMU and wheel encoders** and sends it back for SLAM processing.

Secondary Raspberry Pi (Vision Processing Unit)

- Runs an **ONNX-based model** for object detection and tape detection (not simultaneously).
- Receives depth camera data from the Jetson to enhance object recognition.
- Simultaneously executes the tape detection model to assist in navigation.

Jetson (Motion Planning & Coordination)

- Runs **MoveIt2** for robotic arm control.
- Transfers depth camera data to the RPi for vision processing.
- Waits for a signal from the main RPi to trigger MoveIt operations upon reaching the target location.

Task Execution Flow

- The main RPi navigates the robot using **SLAM and Nav2**.
- Once the goal is reached, it signals:
  - The Jetson to **start MoveIt2 operations** for object manipulation.
  - The secondary RPi is to **switch from tape detection to object detection** for precise object handling.

- This ensures the **separation of navigation and object-handling tasks**, preventing interference.

Arm Control

- A dedicated **STM32 microcontroller** drives the robotic arm's motors, executing movement commands based on values computed by MoveIt2 on the Jetson.

This architecture ensures smooth coordination between navigation and object handling while optimizing computational efficiency across different hardware components.

5. Future Scope

We plan to extend the capabilities of our robotic system by leveraging the **MoveIt Task Constructor** to implement advanced manipulation functionalities such as pulling, pushing, and twisting. This will enhance the system's versatility in handling diverse tasks within the RCUP@WORK environment.

Looking ahead, we aim to develop a **custom tech stack** tailored to our specific requirements, integrating a fine-tuned **GraspNet implementation** optimized for RCUP@WORK objects. This will enable more precise and efficient grasp planning, improving object handling in real-world scenarios.

For enhanced perception and navigation, we plan to integrate **3D LiDAR and a depth camera** with **RTAB-Map** for **3D SLAM**, creating a dense, feature-rich representation of the environment while ensuring **loop closure detection** to minimize long-term drift. To enable real-time obstacle avoidance and smooth navigation, a **local planner** is essential. While a **global planner** like **SmacPlanner** can generate high-level paths using the 3D map, a **local planner** such as **TEB (Timed Elastic Band)** or **MPC (Model Predictive Control)** can refine these paths dynamically. The depth camera will enhance obstacle detection beyond LiDAR's range, improving safety in cluttered spaces. Additionally, for **3D path planning**, frameworks like **OctoMap, Voxblox, or FIESTA (Fast Incremental Euclidean Distance Fields for Online Motion Planning)** will allow the robot to navigate complex terrains, detect overhanging obstacles, and traverse multi-level environments efficiently. These upgrades will be particularly beneficial for **warehouse automation, search-and-rescue missions, and autonomous navigation in unstructured terrains**, where precise 3D perception and adaptive path planning are crucial.

For simulation and refinement, we intend to incorporate **Isaac Sim**, allowing for realistic physics-based testing and validation of our robotic models. Additionally, we plan to integrate **DenseFusion**, which combines RGB and depth data to enhance object pose estimation, ensuring more accurate and reliable object interactions.

Advancements in electronics are bringing exciting improvements to boost performance and functionality. One key area is better sensor fusion, which will help systems process data from multiple sensors more accurately and reliably. Adding features like Electrostatic Discharge (ESD) protection will also make devices more durable and resistant to environmental damage. The shift to BLDC motors for actuation will allow for smoother motion, better control, and greater efficiency, extending the lifespan of mechanical components. Additionally, moving to a swerve drive system will make movement more precise and agile, especially for applications that need complex manoeuvres. Altogether, these upgrades will take electronic systems to the next level.

References:

1. https://learn-robotics-with-ros-s-school.teachable.com/p/mastering-robotics-software-engineering-with-ros-develop-and-deploy-advanced-manufacturing-ssolution
2. https://universe.roboflow.com/dir-wg01w/robocup_objects_2024/dataset/1
3. https://github.com/ivyknob/bno055_stm32
4. https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/
5. https://emanual.robotis.com/docs/en/dxl/ax/ax-18a/
6.