ERC Hackathon 2025

General Overview

Welcome to the ROS2 Robotics Hackathon: Warehouse Automation Quest!

Problem Statement:

The rise of e-commerce has led to an unprecedented demand for efficient warehouse management systems. Traditional manual sorting and organisation processes are no longer sufficient to handle the massive volume of packages flowing through modern distribution centres. It is now up to us to revolutionise warehouse automation!

Your mission, should you choose to accept it, is to design an advanced autonomous warehouse management robot capable of navigating through complex warehouse environments, detecting misplaced packages, and organising inventory according to company-specific shelves.

This hackathon aims to introduce you to the broad domains that form a robotic system, namely Design (Mechanical and Electronics) and Automation (Perception, Planning, and Control), and guide you in building your robotic system from scratch. Each aspect of the task is crucial to achieving the final goal, yet they can be tackled independently of one another. Your solution should encompass mechanical design, electronics interfacing, and an implementation of autonomous behaviour.

Don't Panic!

At first glance, the problem statement may seem daunting. But fear not! It is intentionally complex and challenging, designed to stretch your abilities and creativity. Remember, the key is the thought and effort that goes into your attempt, so be sure to document everything meticulously!

We hope you will gain valuable insights and skills from this hackathon. The learning resources section of this document is a great starting point; however, we encourage you to explore and search online as extensively as possible.

Need Help?

If you encounter difficulties during the hackathon or have any doubts, don't hesitate to reach out for help. However, the best method to learn is through practice, so we encourage you to debug errors yourself as much as possible.

Ready to accept the challenge and automate the future of warehousing?

Fill out this form to register for the hackathon and embark on an automated adventure!

For BITS Goa students - [Link to Registration Form]

For other colleges' students - [Link to Registration Form]

Logistics

Submission Guidelines

The hackathon is meant to be a group activity, with groups consisting of a maximum of 6 members (ideally 2 for each subsystem). We will allot teams to those who want to participate but haven't formed a team. You are also allowed to tackle a particular subsystem independently. One of the team members should create a GitHub Repository consisting of all the required documents, files and code that the team wants to submit. It should be properly organized with the README.md file consisting of thorough documentation of your attempt. Also include the required CAD files, screenshots of your robot design, link(s) to electronics simulations, a screen recording of your robot performing the navigation as well as a rosbag in the repository.

You will be assigned mentors for each vertical who will help you in case of any queries and have general discussions about your approach. More details about this will be given in the groups.

Fun and creative team names will be rewarded :P

Submission Deadline: 16th August 11:59 pm

Index

Task Details

Your team needs to do all three tasks to be considered a complete submission. Different team members can work on different tasks, or on multiple tasks. The goal of the hackathon is to introduce everyone to the basics of all three domains and to give a taste of what system design feels like. Focus on learning and implementation of what you have learnt rather than the competition, although the winning team will be given a treat on campus :)

1. Mechanical

Objective:

Design a robust, reliable, and efficient mechanical structure for an autonomous warehouse management robot capable of navigating through warehouse environments, lifting and transporting boxes of various weights, and organising inventory on shelves.

Specifications:

Robot Parameters (Given):

- **Weight of boxes:** 800 g.
- **Dimensions of boxes:** 30cm x 30cm x 20cm
- **Height of shelves:** 0.4m, 0.75m, 1.1m, 1.4m (approximately)
- **Required operating speed:** 0.5 m/s traversal, 0.2 m/s lifting/lowering
- **Workspace dimensions:** 15m x 9m

Chassis Design:

- **Dimensions:** As per your requirements (should be able to justify your choice)
- **Mobility:** Differential drive system with 4 DC motors for navigation
- **Ground Clearance:** Ensure sufficient clearance for navigating warehouse floors
- **Compartments:** Design compartments for storing batteries, electronic components, and emergency equipment
- **Camera Mounts:** Include a front-facing camera for navigation and barcode scanning

CAD and Analysis:

- Use **Fusion 360, OnShape** to design the robot
- Sketch the forklift mechanism to determine appropriate dimensions and gear ratios
- Conduct static structural analysis using **Ansys** for both the chassis and the forklift mechanism
- **Torque Calculations:** Perform detailed torque calculations for:
    - Traversal system
    - Manipulator system

General Instructions:

- Calculate appropriate gear ratios for the lifting mechanism and movement
- Specify motors and gearboxes based on torque calculations
- Design a modular system for easy maintenance and wire routing
- Ensure the design can navigate through warehouse aisles and around obstacles

Bonus:

- **Detailed Engineering:** Show comprehensive torque calculations and safety factors
- **Safety Integration:** Advanced safety features and emergency systems
- New creative mechanisms, apart from ground vehicles, our welcome

2. Electronics

Objective:

Design the complete electronics system for an autonomous warehouse robot using TinkerCAD simulation. The system should be efficient, reliable, and capable of controlling the robot's movement and functions.

Requirements:

TinkerCAD Simulation:

- Design the entire electronics system using **TinkerCAD**
- Assume the components in TinkerCAD have sufficient torque, power, and other features
- Use **2 Arduino UNOs**:
    - **Master Arduino:** Communication and decision making
    - **Slave Arduino:** Motor control and sensor interfacing

System Functions:

The circuit you design should perform these three main functions:

1. **Differential Drive Movement:** Control 4 DC motors for robot navigation
2. **Forklift Control:** Control 2 DC motors with encoders for vertical movement
3. **Conveyor Belt Control:** Control 2 DC motors with encoders for the prong conveyor system

Sensing Systems:

- **Ultrasonic Sensor:** Position detection for objects on a conveyor belt
- **Limit Switches:** Stop forklift at upper and lower limits (use pushbuttons in TinkerCAD)
- **Encoders:** Position feedback for precise motor control

Power System:

- **Power Calculations:** Calculate total power requirements for the robot
- **Component List:** Provide a realistic part list based on power and torque calculations

PCB Design:

- Design **Arduino UNO shield (slave)** using **KiCad**
- Include all required footprints and components (motor drivers, encoders, etc.)
- Use **screw terminals** for external wire connections
- Calculate and justify the track thickness for power distribution
- Minimise wiring complexity through proper shield design

Code Requirements:

- **Master-Slave Communication:** Implement serial communication I2C between Arduinos
- **Comprehensive Comments:** Include thorough code documentation
- **Control Logic:** Implement proper control algorithms for all subsystems
- **Safety Features:** Include emergency stop and limit protection

General Instructions:

- Document all assumptions made during the design
- Ensure PCB design matches actual component specifications
- Note the differences between TinkerCAD and real-world components

Bonus Points:

- You can be creative with the choice of microcontrollers; feel free to use others apart from Arduinos.

3. Automation

Objective:

Design and implement the automation system for an autonomous warehouse management robot using ROS2 and Gazebo simulation. The robot needs to navigate through a warehouse environment, detect and organise misplaced packages, and avoid obstacles as well.

Task Instructions:

Setup:

- Use **ROS2 Humble** and **Gazebo Fortress Ignition** for simulation, with the provided world file and the robot files.
- The robot starts at a designated home position in the warehouse
- The warehouse contains multiple colour-specific shelves
- Obstacles include pallets, trolleys, people(static) and trash bins.

Primary Tasks:

1. **Inventory Management:**
    - Detect boxes on shelves and verify correct placement
    - Identify company affiliation through colour (RGBY) detection, and locate the shelf and rack number from the ARUCO marker ID 5x5 (range 0-19).
    - Reorganise misplaced boxes to correct the shelves [Refer to Appendix 1 for correct placement]
    - Create an optimal path for reorganisation tasks
2. **Floor Priority System:**
    - Detect boxes randomly placed on the warehouse floor
    - Be creative in your solution to this [For example, you may choose to complete an ongoing task or end it to finish this floor box]
    - Resume the previous task after floor clearance

Technical Requirements:

Path Planning:

- Implement a **sampling-based algorithm** (RRT, RRT*) or a **graph-based algorithm** (A*)
- Navigate through the maze-like warehouse with multiple shelves
- **Dynamic Obstacle Avoidance:** Account for moving people
- **Multi-objective Planning:** Optimise for both task completion and safety

Control System:

- **Collision Avoidance:** Prevent collisions with shelves, boxes, and people
- **Emergency Stop:** Immediate halt capability for safety

Computer Vision:

- **OpenCV Implementation:** Use OpenCV for all vision tasks
- **Real-time Processing:** Process camera feed for navigation and detection
- **Colour Recognition:** Distinguish between different colored boxes (RGBY)
- **AruCo Reading:** Read the aruco marker to determine the shelf and rack position from the ARUCO ID.

General Instructions:

- Document all assumptions and design decisions
- Implement robust error handling and recovery mechanisms
- Provide comprehensive code comments and documentation
- Test the system thoroughly in the simulation environment

Bonus Points:

- **Custom Messages:** Define specialised ROS2 messages for warehouse operations
- **Advanced Planning:** Implement multi-robot coordination capabilities
- **Machine Learning:** Use ML for improved box detection and classification

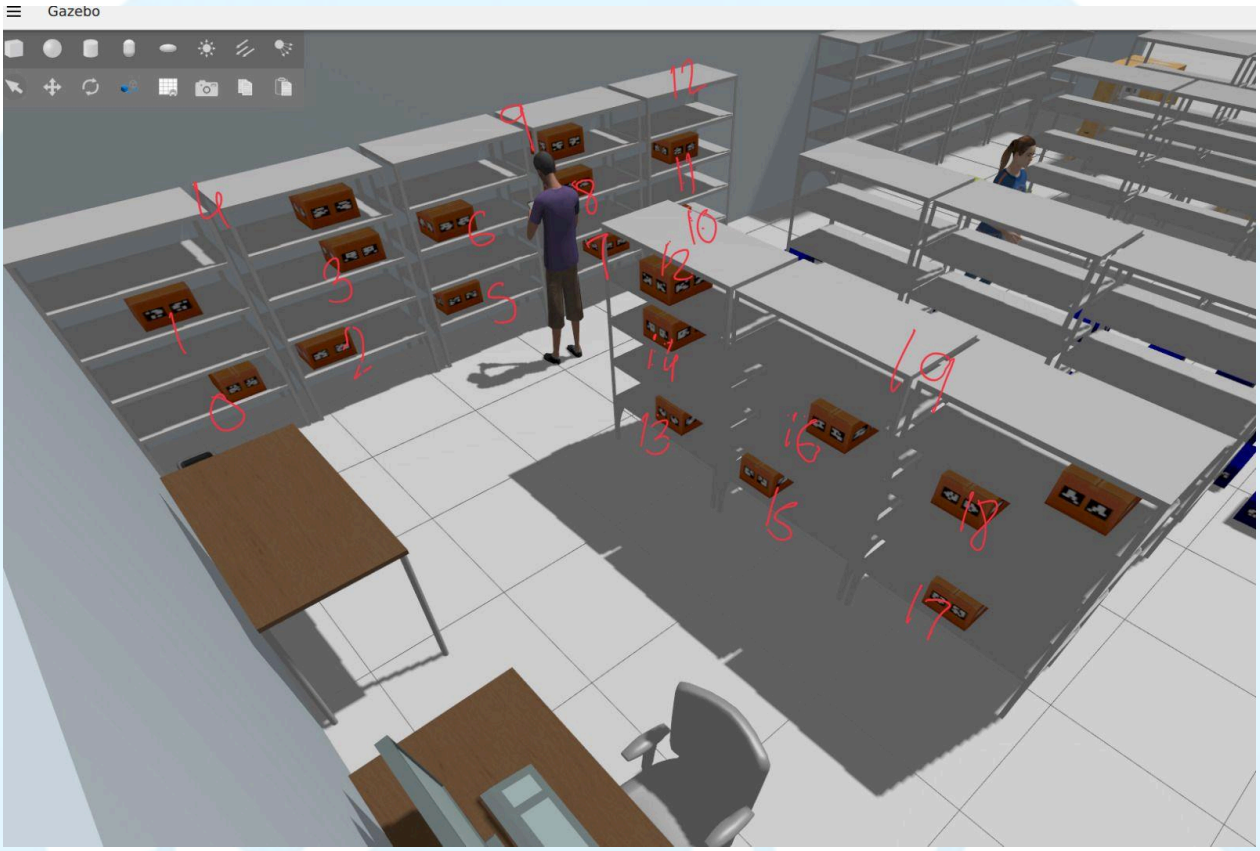
Setup Instructions: World File and Gazebo Environment

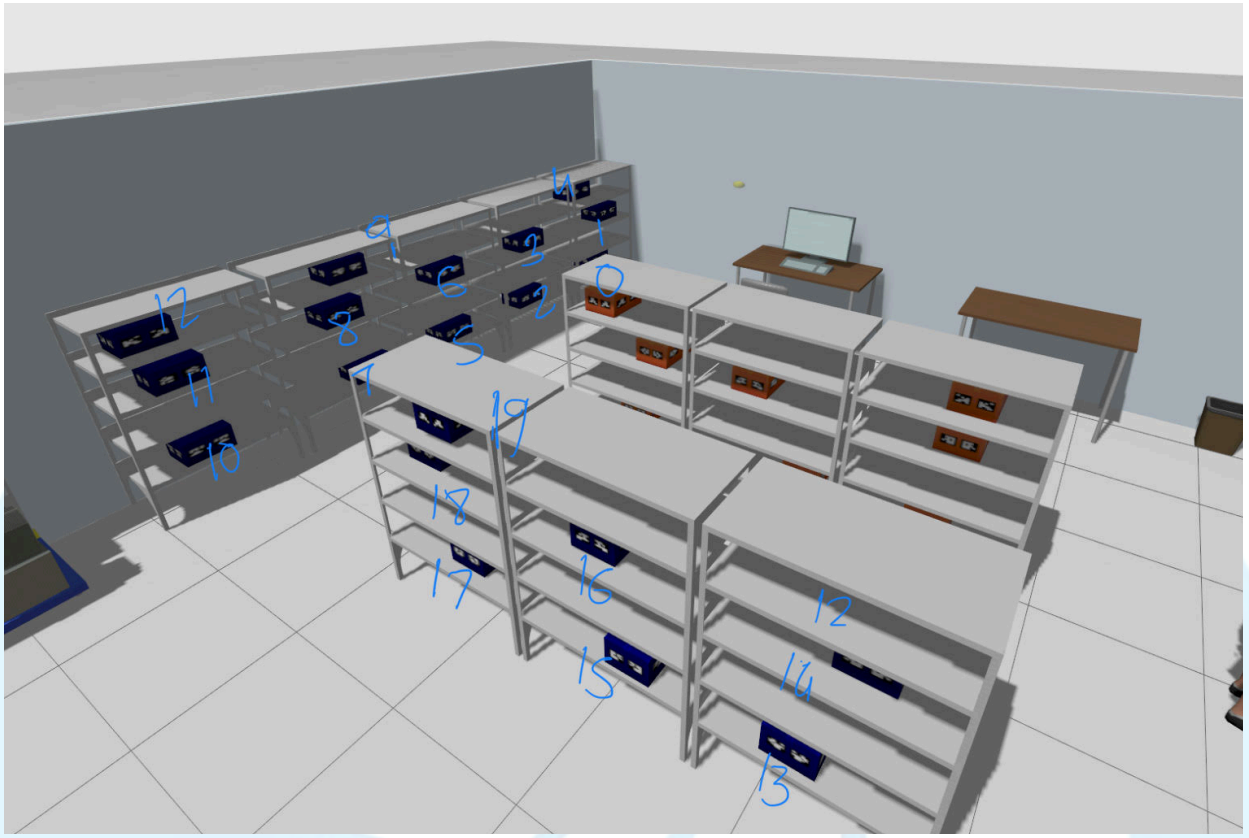
Note: We'll be adding the worldfile and robot's files to this doc shortly, till then, we recommend going through the installation process of gazebo and getting yourself familiar with it.
IMPORTANT: Install Gazebo Fortress for maximum compatibility. Instructions can be found here.

Appendix 1:

Correct Numbering of boxes on the shelves

Each type of box eg.(red-19) gets 2 rows of that shelf and the images below show the correct locations for the boxes. A box of the right kind can be placed any where on those 2 rows. For example, red 0 will go on the bottom 2 rows of the corner shelf as shown in the image while red 19 will go on the top rows of the 2 centre shelves




12
11
10
9
8
7
6
5
4
3
2
1
19
18
17
16
15
14
13

Learning Resources

The ERC Handbook is an extensive compilation of resources related to robotics so do check it out. The resources below can be used in addition to the handbook:

Python

Check out these video series (1, 2) by Cody Schafer. They cover the basics of Python from variables to functions and even object-oriented programming. This guide by Google is also a good supplementary resource.

Linux Terminal

You should do your best to familiarize yourself with the Linux terminal as it's essential for many of the development tools that you will use. Get started here and here.

Git

An integral part of most software projects, Git is a tool for version control and managing changes to code. Check out this course and the git cheat sheet, and this fun game Oh My Git!.

ROS2

The Robotics Operating System is the backbone of every complex robotics stack. You will need to first set it up on your machine. For this hackathon, we require you to install **ROS2 Humble** on your systems. Since ROS 2 is supported for Linux only, we recommend that you dual-boot with Ubuntu 22.04. If this isn't possible, you can try either a virtual machine or Windows Subsystem for Linux for Windows users. To use a GUI application like Gazebo with WSL, you'll need to set up XServer. For Mac users, follow this video to set up Ubuntu (remember to install Ubuntu 22.04 only). After getting to a Linux command prompt, follow the ROS2 Humble installation through this doc for instructions. Detailed Instructions are here.

The best learning resource for ROS is the official ROS2 Humble tutorials. Go through the tutorials thoroughly up till 'creating custom msg and srv files'. You are also free to refer to any other resources or video series for the same.

Path Planning

The following doc has several resources you could use to learn different path-planning algorithms. If you want a more mathematically rigorous resource, you can go through this book. Create a separate file demonstrating your path planning algorithm (GIF, video would be nice). Feel free to use the same algorithm in your ROS2 simulation as well.

Controls

Robotics deals with many continuously operating, dynamic systems, which are dealt with using Control Theory tools. For the basic concepts, check out the course on controls of a mobile robot by GATech and this video series by Steve Brunton. The PID controller is the most commonly used controller. Check out this introductory series by MATLAB. The PID controller section of our handbook provides more mathematical insight as well. Note that going above and beyond to implement different control algorithms will fetch extra credit ;)

Computer Vision

You can follow this playlist by Sentdex for learning OpenCV (till video 13). Refer to the OpenCV website or this page for more tutorials.

Mechanical

A robust and optimized mechanical design is the backbone of every deployed robotic system. Get started with an introduction from ERC's handbook. Refer to this playlist to get started with CAD in Fusion 360. ANSYS is a simulation software that you can use to perform structural analysis. Use this to learn more.

Electronics

Have a look at the ERC handbook's page on Arduino here. Jeremy Blum's playlist is a great place to get started. Arduino's official documentation is also well-made and useful. Use KICAD for designing your PCBs. Refer to these videos to learn about PCB design using KICAD.

Contact Information

Automation:

- Kevin B Mathew - 7776063339
- Dev Thacker - 9819824645
- Aryan Goyal - 8872251132
- Indrajit Mandal - 8016070647

Electronics:

- Saransh Agrawal - 7389076379
- Dev Thacker - 9819824645
- Aryan Goyal - 8872251132
- Parth Jaju - 8310685729

Mechanical:

- Kevin B Mathew - 7776063339
- Parth Jaju - 8310685729
- Ranadip Chakraborty - 9315053413
- Nilesh Bhatia - 9967039969