ERC Hackathon 2023

1 General Overview

The hackathon is focused on the design of a **landmine detection and extraction robot**. Your team has to provide a solution consisting of the mechanical design, electronics interfacing and an implementation of autonomous behavior. All three tasks are linked to the same final goal and yet can be completed independently.

This hackathon aims to introduce you to the broad domains that form a robotic system i.e. Design (Mechanical and Electronics) and Automation (Perception, Planning and Control) and guide you to build your own robotic system from scratch. Please read all the instructions carefully before starting. All the files related to the assignment questions and necessary for solving them are present in this GitHub repository.

At first glance, the problem statement may be intimidating. ***Don't panic!*** It is meant to be complex and challenging and will take some time and effort to complete. In the end, what matters is the thought that went into the attempt - so do document everything!

We hope that you will learn something valuable from this hackathon. The learning resources section of this document is a great starting point, but we encourage you to search online as much as possible.

If you face an difficulties during the hackathon or have any doubts then don't hesitate to reach out for help, but the best method to learn is by practice, and hence try your best to debug errors yourself.

Fill this form to register for the hackathon!

2 Logistics

2.1 Submission Guidelines

The hackathon is meant to be a group activity, with groups consisting of a maximum 6 members (ideally 2 for each subsystem). We will allot teams to those who want to participate but haven't formed a team. 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 on the groups.

Fun and creative team names will be rewarded :P

Submission Deadline:

2.2 Contact Information

- Arjun Puthli
- Aditya Kurande
- Yash Bhisikar
- Priyanshu Kumar
- Shantanu Deshmukh
- Ishan Gokhale
- Murali Nair
- Pulkit Singla
- Adarsh Gopalakrishnan
- Akshat Tubki
- Aasim Sayyed
- Vishnuvarhdan Reddy Peddireddy
- Smrithi Lokesh

3 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:

3.1 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.

3.2 Linux Terminal

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

3.3 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.

3.4 ROS

The Robotics Operating System is the backbone of every complex robotics stack. You will need to first set it up on your machine. Since ROS 1 is supported for Linux only, we recommend that you dual boot with Ubuntu 20.04. 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 up xserver. For mac users, follow this video to setup Ubuntu (remember to install Ubuntu 20.04 and not 22.04). After getting to a Linux command prompt, follow the ROS installation instructions in the wiki. This book by Morgan Quiqley is highly recommended (the first 4 chapters + the examples in the second half of the book should be sufficient), and will walk you through the ROS workflow to make you comfortable with it.

3.5 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. You don't need to use shapely for obstacle avoidance, we have provided you with the functions to do so.

3.6 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.

3.7 Computer Vision

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

3.8 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 which you can use to perform structural analysis. Use this to learn more.

3.9 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 EAGLE for designing your PCBs. Refer to these videos to learn about PCB design using EAGLE.

4 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 how 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 :)

4.1 Mechanical

Assume the diameter of the landmine to be 20cm. The detector has the following tasks to achieve:

1. A fixed sensor can be used to detect landmines. For the metal detector the electromagnetic coil array of the sensor could be arranged in a circular or rectangular array and coupled to a signal processing device. The intended detection range and sensitivity would determine the sensor's size.
2. A tracked wheel design would be appropriate for navigating low friction surfaces such as mud. The system could be made up of two parallel tracks. Traction patterns on the tracks should be used to improve surface grip. A sprocket should be designed for each track to engage with the track links and create propulsion.
3. When designing the chassis, consider the dimensions needed for the mechanisms and electronics components, and ensure they fit without interfering with each other. Use compartments to separate components, plan cable routing to avoid entanglement. Utilize CAD software for virtual modeling and perform simulations to identify and address any design flaws before manufacturing the chassis. Using ANSYS, perform static structural analysis of this chassis and ensure there are no regions which are below FOS (Factor of Safety) of 2. Make necessary changes to the model if required.
4. To dig up a landmine, a mechanism is required to loosen the soil without triggering the mine. One approach is to use a digging arm or shovel attached to the detector. The arm can be equipped with a digging tool such as a scoop or a rotating cutting edge. The mechanism should allow controlled and precise digging while avoiding contact with the landmine. Additionally, a separate mechanism can be designed to remove the landmine safely from the ground and place it on the surface for later collection.

Bonus: show the torque calculations for the required motors

4.2 Electronics

1. Assuming that the robot to be designed drives around using a differential drive mechanism and has 4 wheels (2 front wheels and 2 rear wheels), design a circuit which will run the 4 DC motors associated with the wheels based on a command it receives in the form of linear velocity (in the direction that the robot is facing), and the angular velocity of the robot about the z-axis.
2. The robot comes equipped with a metal detector for the detection of landmines. The detector has 2 coil windings, a sender coil and a detector coil. The sender coil has an external 9V battery connected to it. Implement a circuit wherein this constant power supply is converted into a sinusoidal or near-sinusoidal waveform which then powers the sender coil.
3. When a landmine is detected by the metal detector, it rings a piezoelectric buzzer and changes the color of an RGB LED depending on how close the landmine is to the detector. For the sake of simplicity assume that all the landmines are at the exact same depth below ground. Design and interface a circuit which makes the buzzer ring and the LED change color depending on the distance between the detector coil and the landmine underground as follows:

Case	Detector coil	Buzzer	RGB LED
1	No landmines detected	Rings for 1s, then doesn't for 2s, repeat	Green
2	Landmine detected, $>$ 3m away	Rings for 600ms, doesn't ring for 1s, repeat	Yellow
3	Landmine detected, $<$ 3m away	Rings for 500ms, then doesn't ring for 500ms, repeat	Orange
4	Landmine directly below sensor	Rings constantly non-stop	Red

4. Implement a switching mechanism to turn it off or stop detecting while the landmine is being dug up. In this state, the RGB LED turns blue, while the buzzer stops ringing completely. Once the mine has been dug up, the mechanism will switch the detector back on so that it would start looking for landmines again.

5. In order to dig up the landmine, there needs to be a mechanism to loosen up the soil before picking it up (refer to Mechanical Task). Design and interface a circuit using motors for loosening the soil and for removing the landmine. Try to make the circuit as consistent with the Mechanical Design of this mechanism as possible.

6. Now suppose that we need to count the number of landmines that have been detected. Display the number of landmines detected on a set of 7 segment displays. Assume that the count increases when the detector is turned off to dig up the landmine. Note that regardless of whether the detector is on or off, the count should be displayed.

You are required to do design the drive mechanism in Tinkercad.
Design a PCB and a create a parts list for the metal detector circuitry, and the previous two mechanism as well. This can either be a single PCB or multiple, depending on your robot's design.
Make sure to include a brief explanation for every design decision you take

Brownie points for minimizing Arduino use in the metal detector. We just think it'd be really cool if you did that.

4.3 Automation (Perception, Planning, Control)

Assume that due to some manufacturing error and technical malfunction the robot is not safe enough to use for landmine extraction and can't switch between detection and extraction mode (message Arjun for the lore drop), and it is therefore being used for detection of metals. It now requires two passes of the environment to function properly. On the first pass the bot works in detection mode and marks the coordinates of the metal deposits and on the second pass the bot traverses the shortest path between these marked locations to extract the metals. Yours is the second task of traversing already labelled and marked metal deposit locations.

Getting Started

Use **Turtlebot 3** and **Gazebo** for simulation. Instructions for setup of Turtlebot 3 for simulation in Gazebo can be found in section **4.3.1**.

Task Instructions

Coloured cones are placed at the metal extraction sites. Your robot should detect the colour of the cone using the camera mounted on top. Publish the following strings to the ROS topic 'task_status' based on the following conditions:

- Blue Cone: "Iron extraction ongoing"
- Red Cone: "Zinc extraction ongoing"

- After message for the final cone is published: ”Task completed”

The coordinates of the goals in the matplotlib frame are as follows.

- (5.18, -2.19)
- (1.58, -2.26)
- (-2.28, 1.86)
- (0.57, 0.33)
- (-3.61, -2.20)

Points to Note:

- If (X, Y) is a node in your planned path to the goal, while feeding these values to the controller, change it to (X + 1.79, Y + 0.66)
- Begin planning from (-5.06, -3.12)
- Do not account for the size of the robot while planning. The obstacles in matplotlib have already been adjusted taking that into account.
- Ensure that the robot doesn't collide with the cones. Add this control logic to your controller node.
- The following is provided in **obstacle_detection.py**:

  A function called **`isValidPoint(parentX, parentY, childX, childY)`**.

  While plotting a point in sampling based algorithms, you can call this function to check if that point is valid or not, by passing the arguments: x,y of parent node (the existing node in tree that the child node is going to connect to) and, x,y of child node.

  The function will return true, if it is a valid point and false if it is not. You do not need to implement your own obstacle detector as we have provided the required functionality in obstacle_detection.py (including the coordinates of all wall obstacles), however you are free to do so if you wish to.

- Add the following in the ROS package `robotics_hackathon_automation`:

  1. **path_planner.py** - (Node name - “`path_planner`”)
     A path planner using a sampling-based algorithm (RRT, RRT*). [You will have to create your own map for graph based algorithms such as A*, you are free to do that]
  2. **controller.py** - (Node name - “`tb3_controller`”)
     A PID controller to have the robot follow the path planned by your path planner.
  3. **colour_detection.py** - (Node name “`colour_detector`”)
     To detect the colors from the camera feed of the object using OpenCV

  Eventually, your planner should plan a path from the robot's starting position to where it needs to move next and publish the path to a topic (say it's named “`planned_path`”). The controller should subscribe to this topic and receive the path planned by the planner.
  That is, in detail:

Name of **Node** (not file)	Subscribing to	Publishing to
`path_planner`		`/planned_path`
`tb3_controller`	`/planned_path, /odom`	`/cmd_vel`
`color_detector`	`/camera/rgb/image_raw`	`/task_status`

4.3.1 Environment Setup

To set up the simulation environment, the following packages are required:

- Turtlebot 3
  Follow setup instructions here
- Clone the hackathon repo in your catkin workspace source folder (remember to install shapely to be able to run this package) and then run catkin make/build.

The launch file needed to launch the world as below is provided in the ROS package named `robotics_hackathon_automation`. Run the following block of code to launch the world with the turtlebot along with the three nodes.

```
roslaunch robotics_hackathon_automation automation_task.launch
```

The lines that launch the three nodes are commented out in the launch file so remember to uncomment them when you want to test them together.

5 Final Points

All this might seem a little intimidating if you haven't worked on a big project before. The key thing, however, is to break it up into manageable chunks. We recommend you start off by building a small publisher which publishes to `/cmd\_vel` to move the robot. After you are comfortable with this and can see the robot move properly in the simulator, you can work on a path planner. Once the planner is ready, you can separate everything into separate ROS Nodes.

Another essential skill one should have is "The art of googling" (not kidding). Because of the availability of platforms such as stack overflow and ROS Wiki the task of solving errors and debugging has become a lot easier. Therefore, we highly encourage you all to google all your errors and queries first before asking any of the seniors.

The steps listed below are just to give you a hint of how to proceed, you can, of course, follow any method you want. (Make changes in the launch file accordingly)

Although all three tasks are related, they are independent of each other. The electronics interfacing or the path planner does not depend on the mechanical design in this hackathon. (although if you were to build a real world system, all three will be linked)

Finally, this hackathon is meant for you to learn and have fun :) We hope that you have a great time solving the hackathon!