Electronics and Robotics Club

BITS Pilani, K.K. Birla Goa Campus

erc-bpgc.github.io

Robotics Hackathon 2022

1. General Note

The hackathon is focused on the design of an **autonomous floor cleaning 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 repo.

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!
Please remember that keeping your code well commented is necessary so that other people can understand it.

We hope that this hackathon gives you first-hand experience with the tools and technologies we use and how to find solutions to problems yourself. These skills will definitely be of help in any future projects you may wish to pursue.

In the Learning Resources section of the doc, we have tried to include a comprehensive set of resources, but they may end up falling short in a few places. We encourage you to search online as much as possible learning resources for any issues or doubts you may have. In some cases, you may find resources better suited to you. We hope you will learn some valuable skills through this hackathon, and searching for yourself will be a key part of that process.

Fill this form to register for the hackathon as a team, or as an individual and we will allot you a team

2. Logistics

2.1 Submission guidelines

The hackathon is meant to be a **group activity**, please form a group of a **maximum of 6** people (ideally 2 from each vertical and the same timetable to avoid schedule clashes). We will allot teams to members who wish to participate but have not formed a team. We expect all members to actively contribute to solving the task. 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.
The submission deadline is **11:59 P.M. on 31st August**.

2.2 Contact Information

Feel free to ask your doubts on the ERC WhatsApp groups. You can also contact the team of mentors directly-

- Aditya Parandekar
- Ajay Krishna Gurubaran
- Atharva Ghotavadekar
- Dhruv Potdar
- Divith Mahajan
- Ishan Tandon
- Laukik Nakhwa
- Manan Arora
- Sahil Shingote
- Sai Akshit
- Siddh Gosar
- Yash Chavan
- Yash Yelmame

3. Learning Resources

Check out the ERC handbook for an extensive compilation of information and resources for everything robotics, use the resources below to supplement them.

3.1 Python

This language is essential for robotics, being very easy to learn and prototype with. Check out this guide by Google to start. This book and the docs are good references. If you prefer video tutorials, refer to this playlist by sentdex.

3.2 Linux Terminal

This will be essential for working with the various development tools you require. Get started 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 for an intro. Use this cheat sheet for reference.

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. After getting to a Linux prompt, follow the ROS installation instructions in the wiki. Feel free to reach out to us if you run into any issues. To introduce the basic concepts, the tutorials section of the wiki will be your best friend. A good supplement that delves a little deeper is this book by Morgan Quigley.

3.5 Motion Planning

For video tutorials and articles on path planning algorithms, use resources at this link. Use these (1, 2) books for a brief introduction and reference. To navigate around obstacles, you need not use Shapely, we have provided you with commands which you can use to program your path planning algorithm.

3.6 Control Systems

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 Control of Mobile robots by GAtech and this video series by Steve Brunton (1). The most commonly used controller everywhere is the PID controller. Check out this video series by MATLAB (1) to understand it better .And also you can refer to the handbook here.

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 Design

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.

3.9 Electronics and Arduino

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.

To further supplement these resources, you can refer to the handouts of the mini projects.

4. Problem Statement - Design of an Autonomous Cleaning Robot

The description of all the tasks is given below:
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 :)

Robotics is broadly categorized into the following interlinked divisions: Design (Mechanical and Electronics), and Automation (Perception, Planning and Control)
The problem statement includes at least one aspect of each subdivision.

4.1 Design (Mechanical)

The cleaning robot has has three states:

- Not cleaning
- Suction mode
- Sweep mode

You are to design the following

- Chassis
    - The chassis should be a differential drive chassis with two wheels and should house the required electronics and accommodate the sensors.
    - Dimensions: 340-350 mm in diameter and 120-140 mm in height.

- Mechanism to shift between modes
    - Design a mechanism to iterate through the three modes upon receiving a signal from the controller (assume signal)
    - The sweep mode has a spinning cloth attachment.
    - The suction mode involves a vacuum and a dry brush to collect the waste.
    - The robot also needs to have a compartment to house the dry waste.

You are free to make the design as creative as you want while maintaining efficiency. Make sure you show the required torque calculations and also search for motors that meet the requirements.
Do make a datasheet listing all material choices.

You will be judged on the creativity of the design and innovation of the switching mechanism.

4.2 Design (Electronics)

The robot can be in one of three modes (suction, sweep or not cleaning) that the user sets through a **user interface**. You implement this either through pushbuttons or a keypad or any other way that you prefer.

In the sweep mode, a mop is attached to the robot which is connected to a DC motor. In suction mode, a vacuum pump is deployed. In 'not cleaning' mode, don't do anything.

Assume that for this task, the robot has 4 independent DC motors as described below:

Sr. No	Description	Function
1	Vacuum Pump Motor	Rotates only in vacuum mode to operate the pump
2	Sweeper Motor	Rotates only in sweep mode
3	Left Wheel Motor	Rotates left wheel of the robot
4	Right Wheel Motor	Rotates right wheel of the robot

Design and interface a circuit that will be able to control all 4 motors according to user commands from the interface designed above. The bot will need to display what mode it is currently in as well as prompt the user for the mode that the user wants to set. The user also gives motor commands for motion in terms of linear velocity in the direction the robot is facing, and angular velocity about the z axis. (Refer to the differential drive model of a robot here)

Now, (in both modes), suppose on reaching the dirty parts of the room and identifying the dirt, the dirt (or liquid, for sweep mode) is found to be in a linear trail.

Implement a **line following algorithm** using Arduino Uno and necessary sensors and any other components such that the robot is able to go through the dirty area while cleaning it. Since it is difficult to simulate a line-following robot, submit an explanation of how and why your algorithm works. Hardware implementation can be tried once we are back on campus.

Hint: For simplicity assume that the area to clean is black and the clean area is white.

You are required to implement the circuit for suction and sweep modes on the Tinkercad simulator using an Arduino Uno.

NOTE: The line following algorithm is independent of the following automation task.

4.3 Automation (Perception, Planning, Control)

The robot has to autonomously navigate through a maze and *'pick up the trash*' (for simulation, assume that the trash is picked up automatically as you pass over the region) without colliding into any obstacles.
Use Turtlebot 3 in Gazebo (an open source physics engine) for simulation. TurtleBot3 is a small and programmable ROS-based mobile robot.
Instructions for installation and setup of Turtlebot3 for simulation in Gazebo are given below.

The areas to be cleaned will be marked in one of two colors, blue or green. Your robot should detect the color using a camera mounted on its top. Publish the following strings on the ROS topic 'cleaning_mode' according to the color detected:
.

- Green - 'Suction mode'
- Blue - 'Sweep mode'
- Neither blue nor green - 'Not cleaning'

The coordinates of cleaning areas are given as follows:
(Use these points as they are for the planner)

- (-4, -2) - Blue
- (-3, 2) - Green
- (-1, -2) - Blue
- (0.3, 0.2) - Blue
- (1.36, -1.8) - Green

The robot needs to navigate from the starting position to the successive areas that need to be cleaned. Plan a path from the robot's starting point to all of the areas to be cleaned. The order and path along which the robot navigates is up to you. Try to make it as efficient as possible. After cleaning all areas, your robot should stop in its place.

Important points to note:

- **The origin point for planning should be (-5.06, -3.12)**

- Due to some differences between how the coordinate system works, you must change all the coordinates that are planned by the path planner in the following manner:
  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)**
  When you plan your path using the above origin and goal points, you need to make these adjustments. You just need to make these changes to all the points (including the goal point as well).

- 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.

- Do not account for the dimensions of the robot in the path planner, it has already been accounted for in the obstacles. That is, treat the robot as a point which has no dimensions.

- You need not worry about the collision of the robot with the areas marked with green and blue, as collision for those objects has been turned off.

- 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)	Subscribed to	Publishing to
path_planner	/odom	/planned_path
tb3_controller	/odom, /planned_path	/cmd_vel
colour_detector	/camera/rgb/image_raw	/cleaning_mode

4.3.1 Environment Setup

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

- Turtle bot 3
  Follow the setup instructions here
- Clone the hackathon repo in your catkin workspace source folder
  (To run this package, you also need to install shapely)
- Run catkin_make or catkin build in your ROS catkin workspace

The launch file needed to launch the world as below is provided in the robotics_hackathon_automation ROS package
The command `roslaunch robotics_hackathon_automation automation_task.launch` will launch the world with turtlebot and also the three nodes as mentioned in the launch file.

5. Guidelines on how to proceed

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)

5.2 Important instructions

1. 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)
2. The hackathon is structured in this way so that you can think about the various design and automation aspects that need to be considered while designing a robot.
3. A very important part of any project is planning what is to be achieved, breaking it down into small manageable tasks and finally integrating everything together. Using a pen and paper to design your system first can be very helpful.
4. Mentor mentee groups will be formed after the registrations are complete.

Finally, this hackathon is meant for you to learn and have fun :)
This is a great chance to explore a field that you have always been interested in, interact with other ERC members and have a functional project at the end of two months.

We hope that you have a great time solving the hackathon!