Team Interplanetar

Software Sub-team Recruitment, 2025

ASSIGNMENT INSTRUCTIONS FOR THE RECRUITMENT OF THE SOFTWARE SUB-TEAM

Rev 2.0

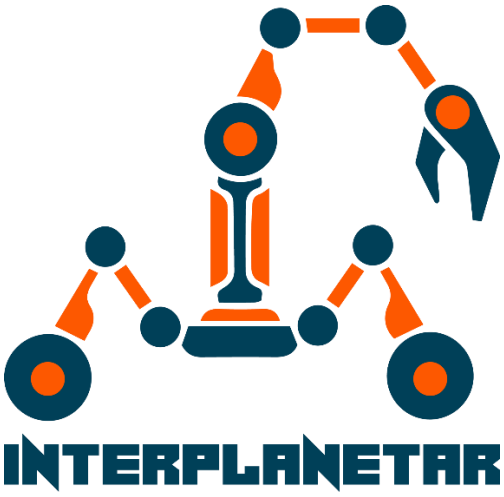

Table of Contents

1. Introduction

Team Interplanetar is a robotics team that focuses on designing and developing rovers, capable of competing in competitions with simulated environments. The team participates in rover challenges like the European Rover Challenge (ERC), the Anatolian Rover Challenge (ARC), and the University Rover Challenge (URC), and develops a rover capable of handling the challenges.

The software sub-team develops the control system for the rover, establishes communication mechanisms, develops control interfaces, and implements autonomous capabilities for specific tasks (e.g., navigating to a target location).

The tasks for the software sub-team can be divided into the following categories:

- **Implementation of actuator controls** (e.g., wheel motors, actuators of the rover arm, science module actuators, etc.). Requires microcontroller programming knowledge.
- **Development of control interfaces** (e.g., Graphical User Interface for rover control & monitoring, joystick control interface for various actuators, rover arm control interface, etc.). Requires a basic understanding of ROS communication protocols, and knowledge of various libraries for developing user interface and input hardware handling.
- **Wireless communication** (e.g., network interface for long-distance wireless communication, camera feed handling, latency management, FPV modules, etc.). Requires knowledge about long-distance communication hardware, a basic understanding of frequency bands & rules regarding the bands, WebSocket handling for camera feed transmission over the network, antenna adjustment for better connectivity, etc.
- **Control architecture** (e.g., autonomous navigation to a target location, obstacle avoidance, rover arm actuation, etc.). Requires the knowledge of basic autonomous algorithms, existing ROS packages for automation, etc.

The sub-team provides helpful resources to learn more about the topics mentioned. We encourage the participants to explore any of the mentioned topics and approach the assignment problems accordingly.

Note: If the topics seem overwhelming to you - don't worry, you're not alone. We didn't know any better during our recruitment either :3

2. General Instructions

The assignment is divided into two sections: **Control Architecture** and **User Interface**. The Control Architecture section contains 1 question, and the Manual Operations section contains 3 questions.

You need to attempt to ***solve the question from the Control Architecture section, and any one of the three questions from the User Interface section***.

Follow these instructions to solve and submit your solution:

1. Go through the details of each question described below. Each question focuses on a specific field of interest for rover development.
2. There is one question in the Control Architecture section. Attempt[1] that question and come up with a solution.
3. There are 3 questions in the User Interface section. Attempt **any one of the three questions** and come up with a solution.
4. Set up your environment as per requirements.
5. Once you are done with a solution, record a short video demonstrating the functionality of your solution.
    a. The video length should not exceed **7 minutes**.
    b. The video resolution should be high enough so that the code is readable. **1920x1080 is recommended**.
    c. You should **focus on the results**, rather than going through the code intensively.
    d. Provide a voiceover, briefly explaining your solution.
    e. Your explanation should contain the following aspects:
        i. Features that you wanted your solution to have (e.g. if designing a UI, you may want it to have buttons that are easily visible or intuitively positioned; or for the Control Architecture task, you'd need to mention any custom voice command you want to implement for a specific operation.)
        ii. The ROS architecture of your solution (how the nodes and topics are connected)
        iii. The general algorithm of your solution
        iv. How you implemented the features you mentioned in (i)
        v. Demonstration of the features

[1] Attempting refers to coming up with a feasible plan to provide a solution for the problem. Incomplete solutions are also welcomed, as long as the plan for solving the problem is described clearly enough. Explain your approach in the demonstration video.

        vi. If some of the features are not implemented due to time shortage, mention how you wanted to implement those.
        vii. Any additional feature/layout design you had in mind but couldn't implement due to lack of time and/or experience.
    f. If your solution is incomplete due to lack of time, describe your plan of approach for the problem.
6. If you can't record the explanation video due to time shortage, write down a brief explanation of your solution in the answer script, and provide a video with a demonstration only (no explanation needed). If you can't record the video, attach the necessary screenshots (that describe every feature of your solution) in the answer script alongside the explanation.
    a. Not providing a video demonstration, and/or too lengthy of an explanation (voiceover or written) will result in a reduction in marks.
7. Upload the code for your solution to a GitHub repository. Provide necessary instructions for building/installing the program.
    a. If your solution requires external libraries, provide a brief installation guide in a README.md file, and mention the necessary libraries in a `requirements.txt` file.
8. Upload the recorded video to a cloud storage of your preference so that it is accessible via a URL.
9. Submit a .txt file in the Google Classroom under the assignment with the following links:
    a. The link to the GitHub repository.
    b. The link to the recorded video.
    c. Other necessary links (if needed)

You may submit one .txt file containing the necessary links for all your solutions.

3. Section 1: Control Architecture

a. Problem i

In this problem, you will be developing a voice-assisted robot in a simulated world (gazebo). Create a proper architecture in ROS/ROS2 (ROS2 Recommended) to simulate a turtlebot3 in Gazebo.

Tasks:

- Your robot will be controlled by voice.
- Your robot has specific energy (say 100 initially). Each time the robot moves, the energy is spent. You need to keep track of energy. Also saying words like "Heal"/"Kaboom" will increase your robot's energy.
- When saying "Left", the robot will rotate to the left direction, saying "right" the robot will rotate to the right direction, saying "forward" the robot will move forward for 2s. But if the user says "Forward 5s", the robot will move forward for 5s. Also, create more custom commands as per your wish.

Sample project architecture:

1. Speaker Node: This node will handle voice-to-text conversion. The text then will be sent to certain topics for other nodes to subscribe to.
2. Control Node: This node will control the TurtleBot. This will take the message from the speaker node and process it. After processing it will be sent to the /cmd_vel topic for the TurtleBot to move.
3. The task is simple, right? But it's not done yet. After creating your project, you have to push your repo to GitHub. Also, you need to Dockerize your whole project.

Requirements:

To do the assignment you need to learn:

1. Git and GitHub
2. Docker
3. ROS/ROS2 (ROS2 Recommended) with Gazebo

Submit the docker container and the GitHub link (in a text file) in the assignment section of the respective classroom.

4. Section 2: User Interface

In this section, you will attempt to design graphical user interfaces (GUI) for the ease of the operators at the base station. Design the layouts according to your preference.

a. Problem i

The objective of this problem is to develop a graphical user interface (GUI) using Tkinter that interacts with a ROS-based robotic system. The GUI will enable the user to control a rover using buttons and will also display real-time rover statistics.

Task:

You are required to design and implement a GUI that provides both control and feedback functionalities for a rover. The GUI should:

- *Control Features:*

1. Implement a Tkinter-based GUI.
2. Include five buttons: Front, Left, Back, Right, and Stop.
3. Each button should publish a geometry_msgs/Twist message to the /cmd_vel topic.
    a. Front: Move the rover forward.
    b. Left: Turn the rover left.
    c. Back: Move the rover backward.
    d. Right: Turn the rover right.
    e. Stop: Stop the rover.
4. For testing your GUI, you can use the Turtlebot simulator or Turtlesim simulator to see if the control commands are accurate. (Not necessary for submission, it's only for your convenience.)

- *Feedback Features:*

5. The GUI should subscribe to the /rover_stats topic.
6. The /rover_stats topic provides a std_msgs/Float32MultiArray message containing three values:
    a. **Battery charge (%):** Represents the current battery level of the rover.
    b. **Rover speed (m/s):** This represents the average speed of the rover.
    c. **Latency (ms):** Represents the communication delay between the rover and the control system.
7. Display the received statistics on the GUI and update them in real time.

Note: Assume the /rover_stats topic already exists, and that data is continuously published on the topic. Assume the datatype to be Float32MultiArray. It's a list of float numbers with 3 positional data: battery level, rover speed, and latency. For testing, you can write a simple publisher node that publishes sample data into the topic with the same name. You don't have to add that node in your final submission.

Required Packages:

1. Python 3
2. Tkinter (GUI development)
3. ROS1 (Noetic) or ROS2 (Humble); ROS2 preferable
4. geometry_msgs (for Twist messages)
5. std_msgs (for Float32MultiArray messages)

b. Problem ii

The objective of this problem is to develop a graphical user interface (GUI) using Tkinter that visualizes a 4-DoF (degrees of freedom; *fancy way of saying 4 controllable joints*) robot arm and its joint angles. The GUI should provide an intuitive and illustrative way to represent the joint angles of the robot arm.

Tasks:

You are required to design and implement a GUI that visualizes the robot arm based on real-time joint angle data received from ROS. The GUI should:

- *Visualization Features:*

1. Implement a Tkinter-based GUI.
2. Provide a clear and easy-to-understand illustration of the robot arm's joint angles. The visualization can be:
    a. A 3D interactive model of the robot arm.
        i. If you wish to use a URDF file for 3D visualization, we encourage you to use the URDF file of any existing robot arm of your choice. However, the degrees of freedom of the arm may not correspond to the arm described in the question. In that case, you may proceed with the configuration of your existing URDF and its number of joints. This will save you some time and the hassle of reconfiguring the URDF (most of the available URDF files usually have 6 degrees of freedom).
    b. Orthographic views (top and front) showing all links and joints. An example concept is illustrated here:

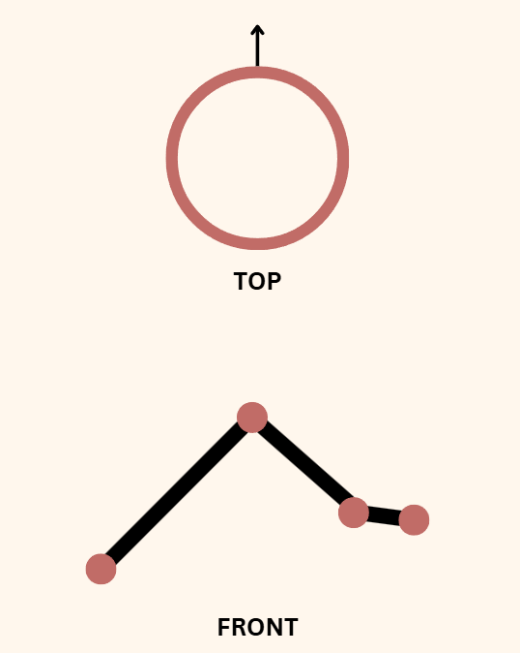

Figure 1 Concept illustration for visualizing arm joint angles.

3. The visualization should be updated in real-time as new joint angle data is received.
4. The reference angle (zero angular position) for each joint can be of your choice.

- *Data Subscription:*

5. The GUI should subscribe to the /joint_angles topic.
6. The /joint_angles topic provides a std_msgs/Float32MultiArray message containing four joint angles in sequence.
7. The joint angles should be processed and reflected in the GUI visualization.
8. The first angle in the array is the base angle, that can be visualized from the TOP view. The rest of the three angles are for the three links, as seen from the FRONT view.

- *Slider Control Panel:*

9. The GUI should also include a slider control panel with four sliders, each corresponding to one of the robot arm's joints.
10. Each slider should allow adjustment within a 180-degree rotation range.
11. When a slider is moved, the new joint angles should be published to the /joint_angles topic in real-time using a std_msgs/Float32MultiArray message. This functionality allows users to control the robot arm's joint angles manually while visualizing them.

Note: Be creative with the layout. The illustration should be easy to understand and should help visualize the current position of the end effector of the arm. ***Any other visual method that is easy to understand & the aesthetics of the layout will grant you bonus marks. If***

you have any method in mind that you couldn't implement due to time shortage, you can illustrate it in the demonstration video. In that case, a concept design should be provided. You will receive marks for each concept.

Required Packages:

- **Python**
- **Tkinter** (GUI development)
- **ROS1 (Noetic) or ROS2 (Humble)**
- **std_msgs** (for Float32MultiArray messages)
- **Matplotlib** or **PIL** (for rendering 2D/3D illustrations if needed)

c. Problem iii

The objective of this problem is to develop a WebSocket-based server capable of handling low-latency text streaming. A front-end interface should be created to display the received text and allow the user to send text to the server.

Tasks:

You are required to design and implement a WebSocket-based system that efficiently streams text between any client device within a network and the server with minimal latency. The system consists of:

- *Backend (WebSocket Server)*

1. Develop a WebSocket server capable of handling text stream from any number of clients within the network. Ensure low latency streaming over a wireless network.
2. Assign a client ID to each client.
3. The server should allow clients to start and stop individual text streams dynamically.
4. Choose a programming framework such as Python (FastAPI, Flask-SocketIO), Node.js (Socket.io), or literally any other suitable technology that you're comfortable with.

- *Frontend (Streaming Interface)*

5. Develop a web-based or desktop front-end to receive and send text messages from all the clients.
6. Display the client ID and the text from that client to all the clients connected to the server in that network.]
7. The interface can be built using:
    a. JavaScript frameworks (React.js, Vue.js, or vanilla JavaScript with WebRTC/WebSockets)

b. Python GUI frameworks (Tkinter, PyQt, or Kivy)
c. Any other framework suitable for text streaming. Explain the framework in the demonstration video.

Software Requirements

Backend Requirements:

1. Python 3 / Node.js / Other Framework
2. WebSocket library (FastAPI, Flask-SocketIO, Socket.io, etc.)

Frontend Requirements:

1. HTML, CSS, JavaScript (for web-based frontend)
2. WebSockets API / WebRTC (for real-time video streaming)
3. React.js / Vue.js / Tkinter / PyQt / anything else (for interactive UI)

Note: The UI should be clean and minimalistic, showing only the client IDs, the received text, and a text input area. ***A well-organized layout will grant you bonus marks.***

Marks Distribution

- Section 1
  - Problem I ---------- 100 marks (max)
- Section 2 (solve any one)
  - Problem I ---------- 75 marks (max)
  - Problem II ---------- 100 marks (max)
  - Problem III ---------- 100 marks (max)
- Maximum score: 200
- The ranking will be done based on achieved marks. Top scorers will be reached out for the next phase.

Note: We're eagerly waiting for you to join the team and work with us. Your time and effort will be the key to our next-generation rover development, and a great learning opportunity for you as well. Don't be disheartened if you cannot complete your assignment; submit whatever you've completed and explain your approach clearly. ***Don't try to make it perfect; try to make it work!***

6. Technical Experience & Skills Assessment

1. Do you have any prior experience in Robotics, Mechatronics, or related fields? *(Yes/No)*
2. Do you have any software skills related to these fields? *(Yes/No)*
    a. If you answered *Yes* to either of the above, please provide:
        i. A brief description of your experience.
        ii. The projects you have worked on (if any).
        iii. How did you acquire these skills (self-learning, courses, competitions, internships, etc.)?

(If you have no prior experience, you can skip the description.)

Add the answers to the .txt file along with the submission links and submit them under the assignment section.