Assignment - Robotics Engineer

[This is a FlytBase, Inc. confidential document. This is not to be shared with anyone other than the candidate who is being evaluated for a role with FlytBase. The candidate is required to destroy his/her copy of this document after the intended purpose has been served.]

FlytBase is a technology product startup and is a global market leader in drone automation software products. Team members at FlytBase are young achievers who are not afraid to dream big and work hard with their teams to realize those dreams. Mutual trust, respect, the pursuit of excellence, efficient execution, and a sense of responsibility & ownership are the core-values that drive us as a team. We aspire to build one of the best workplaces in the world where our team of innovators can thrive and work towards realizing their potential, enabling FlytBase to continue to strengthen its position as a technology leader in this space.

As part of the Robotics team, we would expect the same level of passion, commitment, and pursuit of excellence from you towards your role.

Problem Description:

- The problem makes extensive use of the ROS TurtleSim package.
- The package provides ground robots that can move in a 2D environment, take linear and angular velocity commands.
- You should go through all the APIs (ROS services and topics) provided by the node to understand how you can interact with TurtleSim.
- You are provided with 5 subgoals, with an increasing level of complexity.
- Each goal has its own success criteria and usually builds on the previous goal.
- You may use any language of your choice (C++, Python) to solve the problem.
- For plotting, you may use PlotJuggler, Rqt Multiplot

Goals:

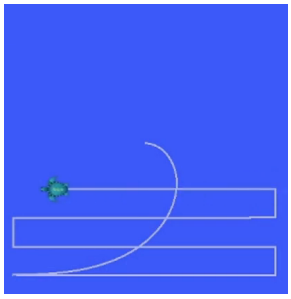
Goal 1: Control turtle

- Spawn a turtle at random locations.
- Make it reach a goal in 2D env by controlling linear and angular velocity.
- Implement a PID controller to ensure the turtlebot reaches the goal as fast as possible without overshooting.

- Show performance variation for various gain values. Figure out the variables/parameters to plot e.g. control inputs, current state etc. to explain why it works.

Goal 2: Make a grid

- You may have noticed that the Turtle stops moving if you stop publishing velocity. In real life there is a deceleration profile for every vehicle. Even applying the brakes does not usually result in vehicles stopping at the current stop. Since the turtlesim does not provide this deceleration profile, let us add it from outside.
- In your code, ensure that there is a limitation on maximum acceleration and deceleration. You can use separate parameters for each and vary them independent of each other.
- Check if you need to change the PID gains to accomplish goal 1 once again.
- Use this relatively realistic robot, and make a grid pattern as shown in the image below. You can ignore the curve formed by the turtle in the image.



- Capture the video for varying profiles of acceleration and deceleration.
- Do you need to change gains every time? If yes/no, why?
- Accompany your videos with plots that show how you are maintaining the limitations on acceleration and deceleration.

Goal 3: Rotate turtle in circle

- Write code that will keep moving a turtle from goal 2 in circles. Provide variables to control the speed and radius of the circle.
- Publish the pose of this turtle every 5 secs. Let us call this topic '/rt_real_pose'.
- Publish another topic with the pose of the turtle, added with random gaussian noise, every 5 sec. Let us call this topic '/rt_noisy_pose'.
- Set the noise standard deviation at 10 units.

Goal 4: Chase turtle fast

- Use the turtle from Goal 3 (which can move into circles), let us call it RT (Robber Turtle).
- Use the turtle with limits on acceleration and deceleration, let us call it PT (Police Turtle).

- Spawn RT from location A, which will then keep moving in circles. Set circle radius to more than 30 units.
- Spawn PT from a random location, 10 secs after RT is launched.
- The PT can access RT's real position every 5 secs through topic 'rt_real_pose'.
- The PT can move at a higher speed than RT.
- PT needs to chase down RT. Whenever the distance between both of them is less than or equal to a 3 unit distance, we can say that the chase is complete.

Goal 5: Chase turtle slow

- Same setup as Goal 4.
- Only difference being, the PT can move only half as fast as RT.
- PT needs to chase down RT.
- Do you need a planning element to accomplish this?

Goal 6: Chase turtle noisy

- Same setup as Goal 5:
- Only difference being, the PT can access RT's noisy pose every 5 secs through topic 'rt_noisy_pose'.
- PT needs to chase down RT.
- Do you need a better estimator to guess where the RT is going to be?

Submission Details:

- Code:
  - Submit the code through a private git repository (gitlab provides them for free of cost).
  - Create a repo on GitHub and upload your source code. (Please keep the repo private). Add assignments@flytbase.com as a contributor for the same.
  - Share the link of the repo by mailing it to careers@flytbase.com
  - In case of any issue, please write to careers@flytbase.com.
- Videos:
  - One or more videos (screen record) for each goal.
- Report:
  - Summarize the approach to the problem.
  - Explain any assumptions that were made.
  - Explain the reasoning behind the choice of algorithms/tools for solving the problem.
  - Show the results through graphs/videos.
  - Explain what could have been done better if submission time was not constrained.

- Send the report in PDF over email with any videos attached.
- Additional notes:
  - In case you do not have past experience with tools/algorithms, mention that in your report. The evaluation will be done accordingly. In such cases, take it up as a challenge and do your best.
  - The report carries equal weightage in your assessment, so make sure you are spending enough time on it.
  - If you have any questions or doubts regarding the assessment, send an email to havish@flytbase.com, snehal.mali@flytbase.com

Please note: Any candidate if found guilty of plagiarism, will result in disqualification from the assignment; and will not be considered for further rounds of evaluation.

- Ensure appropriate viewing rights when sharing the access to the repository.
- Provide appropriate instructions for environment setup, execution of the project.

Assessment metrics:

- Overall understanding of the problem statement
- The justification behind selection of approach, choice of tools/protocols
- Correctness of solution
- Coding skills (use of OOP, modularity, etc.)
- Coding style (PEP 8, etc.)
- Code documentation