**Project Altair Recruitment Week 2**

**Name: Rumman Adib**

**Student ID: 210041131**

**Department: CSE**

**Github Repository:** [**https://github.com/Rumman023/Project-Altair-Recruitment**](https://github.com/Rumman023/Project-Altair-Recruitment)

**PYTHON PROGRAMMING AND OPENCV**

Part 1: Python Programming

Task 1

Code: <https://github.com/Rumman023/Project-Altair-Recruitment/blob/main/210041131_Rumman%20Adib_Week%202_Task%201.py>

Solution Approach: The solution approach first begins by taking input from the users. The first line is the instruction which is stored in a string named ‘instruction’ and the second line takes two integer values which are the grid sizes in rows and columns which is later stored in the ‘grid\_size’ tuple. The these two stored variables are passed in the ‘process\_instructions()’ function which ultimately calculates the final result.

In this function, the solution is initialized by taking rover’s actual position 0 and direction also as 0 (indicating North direction). With each turn to right, the direction is incremented by 1 and with each turn to left it decrements by 1. Then the final direction is calculated by modulo arithmetic, where,

if direction modulo 4 is equals to 0, the direction is North

if direction modulo 4 is equals to 1 or -3, the direction is East

if direction modulo 4 is equals to 2 or -2, the direction is South

if direction modulo 4 is equals to 3 or -1, the direction is West

Again, the function initializes temporary variables along with x and y axis of the grid. The temporary coordinates are used to check if the rover will move within the boundaries of the grid, if it does so, the rover’s actual x and y coordinates are updated according to the temporary x, y variables. Finally the function returns the coordinates and directions in a tuple which is later stored in the ‘ans’ variable for better formatted output style.

Part 2: OpenCV

Task 2

Code: <https://github.com/Rumman023/Project-Altair-Recruitment/blob/main/210041131_Rumman%20Adib_Week%202_Task%202.py>

Video Link: <https://www.youtube.com/watch?v=nXIySDzb2xY&t=4s>

Reference:

1. <https://learnopencv.com/image-resizing-with-opencv/>
2. <https://www.tutorialspoint.com/color-identification-in-images-using-python-and-opencv>
3. <https://answers.opencv.org/question/134248/how-to-define-the-lower-and-upper-range-of-a-color/>
4. <https://stackoverflow.com/questions/10948589/choosing-the-correct-upper-and-lower-hsv-boundaries-for-color-detection-withcv>
5. <https://cvexplained.wordpress.com/2020/04/28/color-detection-hsv/#:~:text=The%20HSV%20values%20for%20true,10%20and%20160%20to%20180>. [red]
6. <https://dontrepeatyourself.org/post/color-based-object-detection-with-opencv-and-python/#:~:text=We%20can%20do%20this%20by,179%2C%2062%2C%20255>) [white]
7. <https://stackoverflow.com/questions/67964158/how-to-understand-the-max-function-in-opencv-python>
8. <https://stackoverflow.com/questions/53935359/how-to-find-out-average-pixel-value-of-an-image-scanning-it-from-top-and-bottom>
9. <https://sparkbyexamples.com/numpy/numpy-count-nonzero-values-in-python/#:~:text=NumPy%20count_nonzero()%20function%20in,arr%20%2C%20axis%20%2C%20and%20keepdims%20>.

**ROBOT OPERATING SYSTEM ( ROS )**

Task 1

1.

Publisher Node Code: <https://github.com/Rumman023/Project-Altair-Recruitment/blob/main/210041131_Rumman%20Adib_Week%202_Task_1%20(Publisher).py>

Subscriber Node Code: <https://github.com/Rumman023/Project-Altair-Recruitment/blob/main/210041131_Rumman%20Adib_Week%202_Task_1%20(Subscriber).py>

Resources:

1. <http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29>

2.

ROS Publisher & Subscriber Node in Python Video Link: <https://youtu.be/r0_buRPcjfQ>

3.

ROS Turtle Bot Simulation Video Link: <https://youtu.be/UhyJrwqsyuc>

**THEORETICAL PART**

Task 1

If I’m in a scenario where a rover is deployed in a remote and vast area, the LoRa module would be a better choice than a generic Radio frequency module. This choice has been taken considering several factors such as range, power consumption, data transmission rate, and the connectivity performance in challenging environments.

Justification: LoRa (Long-Range) is an RF modulation technology specifically engineered for long distance communication. It provides up to 15 KM line of sight distance coverage. Thus this technology makes more useful in the scenario where a rover is deployed in a remote and vast area. Using the LoRa Wan gateway, the 4G internet can be connected to the LoRa, providing a greater area coverage for communication. The generic RF module also has good range of coverage, but not as useful for the scenario. Also the distance varies with extenders like antenna, or the level of frequency or power. In terms of power consumption, a generic RF model receiver supplies 3.5 mA current, with an operating 5V voltage of receiver’s end. But a LoRa RF module draws only 4.2 milliamperes (mA) with an RF output power of +22 dBm, when transmitting or receiving data. So it is very Low Power consuming technology. The LoRa module has a data transmission rate of 50Kbps, which is relatively low than a generic RF module’s data transmission rate, mainly because it is typically used for short-range communication between the connected devices. Also the battery life in a LoRa supported module lasts longer than that of a generic RF module, which is a crucial advantage for a rover deployed in a challenging terrain. LoRa offers a perfect balance between sensitivity and data rate while operating in a fixed-bandwidth channel. RF module equipment requires a special attention to ensure a proper Line-of-Sight (LOS) clearance between the transmitter and receiver. Thus if it encounters any obstacle in-between these 2 nodes, the connectivity can be lost. So it is a a bit fragile architecture in terms of connectivity in challenging terrains. LoRa module, on the other hand, is designed to provide reliable connectivity because of its ability to handle Non-Line-of-Sight communication. So it offers much better performance in obstacle-prone areas, thus making it a better choice for a rover operating in such challenging conditions.

Therefore it can be reasonably concluded that, considering the specific requirements and constraints of the rover’s mission such as range, power consumption, data transmission rate, and the need for connectivity in challenging environments, LoRa module is the better choice for the communication architecture.

References:

1. <https://jfrog.com/connect/post/nrf24-vs-lora-for-wireless-communication-between-iot-devices/>
2. <https://www.engineersgarage.com/increasing-the-range-of-rf-module-by-using-antenna-and-increasing-transmission-power-part-5-23/#:~:text=Learn%20about%20the%20RF%20transmission,the%20RF%20transmitter%20and%20receiver>.
3. <https://robu.in/what-is-rf-transmitter-and-receiver/#:~:text=Features%20of%20RF%20Module%3A&text=Receiver%20supply%20current%203.5%20mA,transmitter%20frequency%20range%20433.92MHz>
4. <https://www.digikey.com/en/articles/how-to-implement-lora-fota-minimal-powerconsumption#:~:text=When%20transmitting%20or%20receiving%20data,output%20power%20of%20%2B22%20dBm>.
5. <https://www.mokolora.com/lora-and-wireless-technologies/>
6. <https://tankscan.com/downloads/documents/Line-of-sight-WP_REV.A.pdf>

Task 2

To develop an autonomous vehicle that can navigate and map an unknown indoor environment, the vehicle needs to be equipped with LIDAR and odometry sensors to perceive its surroundings and estimate its position and orientation. Because, the Simultaneous Localization and Mapping (SLAM) is a technique where any autonomous vehicle constructs a map of its current environment using sensor data while simultaneously estimating its own position and orientation in that given environment. Pose Graph Optimization (PGO) is a popular framework for solving the Simultaneous Localization and Mapping (SLAM) problem in autonomous navigation, that can operate in any complex environments only based on their on-board sensors and without relying on external GPS systems. A pose graph contains nodes connected by edges where each node estimate is connected to the graph by constraints that define the relative pose between nodes and the uncertainty on that measurement. Here pose refers to both position and orientation of the autonomous vehicle. PGO is the fundamental technique that is used to optimize and reduces error of the constructed map/graph by iterating over the previously visited positions of the vehicle on the map.

The key concepts involved in the Pose Graph Optimization process:

* Initial Estimation
* Pose Graph Generation
* Constraints
* Optimization
* Loop Closure

Explanation:

Initial Estimation: The initial key of building a map using the Pose Graph Optimization Process is the Initial Estimation. The vehicle first reads its surroundings with the sensor data and estimates it position and orientation on the given environment relative to its real position and orientation. Initially, the estimated pose lies right above the real pose of the vehicle. Then the PGO process refines the accuracy of its initial estimation.

Pose Graph Generation: A map or graph is constructed which represents the vehicle’s environment of its trajectory. This graph consists of node and an edge between two adjacent nodes. The node is basically the poses of the vehicle at different points in time and the edges are the constraints between these poses. The constraints are generated from sensor data collected from the mounted sensors like, the LIDAR and odometry sensors. LIDAR sensors measure distance by emitting laser pulses and measuring the total time it takes to ricochet after hitting any obstacle in its path. And the Odometry sensor estimates the position of the vehicle by measuring the changes in motor rotations. Thus combining these two sensor data with sensor fusion, the vehicle gets the necessary information about its surroundings in any environment, be it indoor or outdoor.

Constraints: This is one of the key concepts in PGO of any SLAM. It provides important information in the pose graph. The odometry constraints between two consecutive poses provide relative distance between these two poses. Basically, the edge constraints estimates a set of poses (positions and orientations) from relative pose measurements, and each constraints provide a loop relative to the prior vehicle trajectory.

Optimization: Optimization in the constructed graph refines the adjustment of the poses (positions and orientations) of an autonomous vehicle by minimizing the error factor between the real sensor measurements and the estimated poses. This process iteratively adjusts the poses to minimize the error, which is typically computed as the difference between the expected sensor measurements based on the estimated poses and the actual sensor measurements. With the help of different key constraints, this optimization yields the most consistent set of poses that best represents the vehicle's trajectory in the environment. There are various commonly used optimization algorithms, and the Gauss-Newton’s technique is the most popular ones. By refining the initial estimates based while incorporating various key constraints from odometry, loop closures, and other sensor measurements, this optimization process helps the autonomous vehicle to accurately navigate through the environment with the help of the reliable constructed map.

Loop Closure: Loop closure is detected when the vehicle visits its already visited location, and thus incorporating these constraints into the Pose Graph enabling the scope to correct the surmounting errors over time. With loop closure, the most optimized and consistent map is achieved.

By following these aforementioned key concepts, Pose Graph Optimization helps the autonomous vehicle to achieve simultaneous localization and mapping in an unknown indoor environment.

References:

1. <https://www.researchgate.net/figure/A-pose-graph-representation-of-a-SLAM-process-Every-node-in-the-graph-corresponds-to-a_fig2_292377670>
2. <https://ieeexplore.ieee.org/document/9596721/>
3. <https://www.mathworks.com/help/nav/ref/posegraph.html>
4. <http://rvsn.csail.mit.edu/graphoptim/#:~:text=In%20short%2C%20constraints%20in%20the,is%20distributed%20around%20the%20loop>.
5. <https://april.eecs.umich.edu/media/pdfs/wang2014icra.pdf>

Task 3

A systematic plan has to be taken to build a smart autonomous rover using Pixhawk along with other important components.

For a smart autonomous rover, a conventional steering system has to be sought out. Among some types of conventional steering systems, a crawler frame has to be implemented because of its robust nature and the advantage of drivability on almost any terrain. Then the drivetrain of the rover has to be planned out. On the drivetrain, 4WD chassis, a single brushed motor, brushed ESC, steering servo, signal system, drive integration, Pixhawk integration must be included. Some additional sensors like LIDAR, IMU, odometry sensor, GPS, etc are to be used. Also most importantly, a suitable Pixhawk flight controller according to the size and conditions of the rover has to be chosen. So, overall, all of these components and their integration has to be overviewed, and the final plan of implementation has to considered with great attention.

Before building a smart autonomous rover, the overall goal and objectives of the rover has to be planned out. This is the initial and one of the most important step, as it will later work as the basis for choosing suitable hardware, firmware, software components along with several autonomous algorithm later while building the rover. The most suitable Pixhawk flight controller has to be selected and implemented with the flashed out firmware. Otherwise, correct navigation output won’t be received through the mounted accelerometers, gyroscopes, GPS, etc. sensors. Along with this Pixhawk flight controller, a NVIDIA Jetson board has to used as a supporting computer for heavy calculation regarding the autonomous algorithms, sensor fusion and several conditional tasks. Then to understand rover its real time location and environment, then selecting its trajectory on its own, SLAM algorithm with the help of Pose Graph Optimization process has to be implemented. To have a remote monitoring system of the rover, a User Interface will have to implemented to get its real time telemetry data. Also to build a reliable communication system between the Pixhawk and the supporting computer, a suitable data communication protocol also has to be implemented. After following this systematic procedure of building an autonomous robot, it has also go through some test runs to identify the bug and errors it might have. The rover has to be tested in different condition under different environments to truly test its endurance as well as compatibility. Then from these test runs, important data can be scoured through which can be later implemented on the rover to have better performing rover. So, the main goal is to have a optimizing test operation, where after each test runs, the rover will be optimized to improve after every optimization process. Thus the safety and emergency mechanism will be sharper than ever, to survive critical conditions also.

By following the aforementioned plan, I can use Pixhawk along with other components to build a smart autonomous rover which can achieve various tasks in various environments according to the necessities.

References:

1. <https://ardupilot.org/rover/docs/common-pixhawk-overview.html>
2. <https://discuss.px4.io/t/some-newbie-questions-for-an-autonomous-rover/25849>

THE END