free PRO

Internship Report

DECEMBER 7, 2023 – MAY 31, 2024

KENAN BLASIUS

Table of Contents:

I. Section 1 (Company and Project Overview for a New Employee)

A. Introduction

1. Company overview

Welcome to Free Pro! As a new member of the team, it's crucial to understand the evolution and values of the company. Free Pro was created in 2001. Initially established under the name Jaguar Network, the company quickly made its name in the industry of telecommunications, known for its commitment to the innovation and its customer-centric solution.

In 2019, Jaguar Network was acquired by Iliad, the parent company of various subsidiaries including Free and Free Mobile. This acquisition marked a new chapter in the company's journey, providing access to greater resources and expertise within the Iliad group. In 2021, Jaguar Network was officially rebranded Jaguar Network by Free Pro, while still maintaining the name of its original brand. By the end of 2022, Jaguar Network disappeared to make way for Free Pro.

Free Pro is a company in the telecommunication industry that has the objective to bring the best of Free to other company, since Free is made for personal networks. At Free Pro we have a research and development center where we focus on enhancing telecommunications, particularly through the development of advanced 5G technologies, artificial intelligence, and edge-computing.

2. Project Overview

For my internship, I have been received in the research and development center where I had the opportunity to contribute to two distinct projects.

The first project involved the deployment of a 5G core network to test the documentation and performance of the 5G core network. This documentation served as a tutorial guide for the deployment process.

The second project focused on the development of an Automated Guided Vehicle (AGV) **to evaluate network coverage and performance within office environments. The AGV autonomously navigated through office spaces, mapping the environment, and assessing network coverage quality to optimize network performance.**

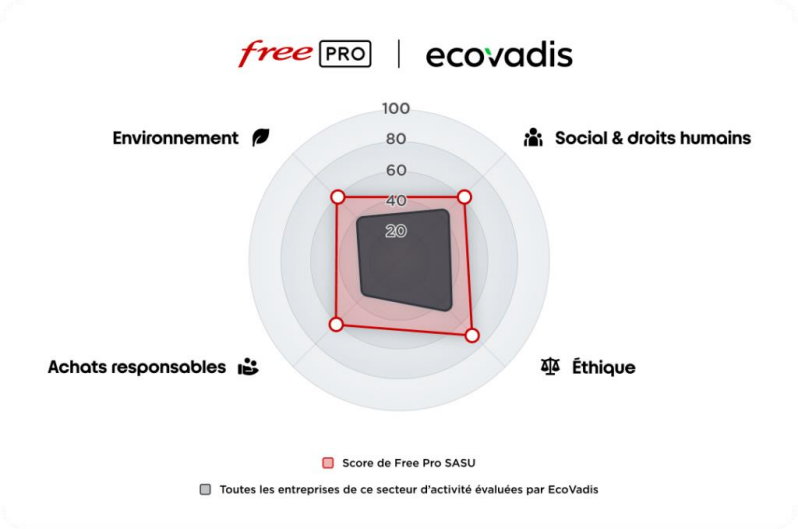
B. Organizational Structure and Company Culture

Free Pro is organized into several key departments, each with specific focuses and responsibilities. The Human Resources department manages recruitment, employee relations, benefits, and organizational development, while the Information Systems department oversees the management and maintenance of the company's IT infrastructure. The LAB team is dedicated to research and development, focusing on cutting-edge technologies such as 5G, artificial intelligence, and edge computing. In parallel, the Technical and Innovation department also engages in R&D activities, particularly in the cloud, voice, and security domains. The Communication department handles both internal and external communications, including marketing and public relations. The Sales department is divided into major accounts (custom offers) **and professional accounts** (standard offers)**, while the Customer Relations department manages customer support and service. Product Strategy develops and implements the company's product strategies, and Operations includes deployment and logistics, integration and services, support, and exploitation.**

Responsibilities and hierarchical reporting within Free Pro are well-defined, with each department headed by a supervisor who oversees team members. These department heads report to the General Director, and major decisions are escalated based on their impact to ensure thorough review and alignment with the company's strategic goals. Collaboration among departments is essential to achieving Free Pro's objectives. While the LAB team is currently somewhat isolated, communication with the sales team is vital, especially for upcoming project launches. This collaboration ensures that pricing, target audience, and other strategic aspects are well-defined. Decision-making at Free Pro follows a hierarchical approach, where employees discuss decisions with their immediate supervisors, who in turn consult their superiors if the decisions are significant.

Free Pro's core values are Simplicity, Expertise, and Proximity. These values are reflected in the company's daily operations and interactions, guiding employees in their work and decision-making processes. The work environment at Free Pro is friendly and collegial, fostering a positive atmosphere where employees feel comfortable and supported. Innovation and creativity are highly encouraged, with the company assigning projects that align with employees' interests and passions, driving the development of innovative solutions. Free Pro is also committed to corporate social responsibility and sustainability. The company has been rated by EcoVadis, an independent rating agency assessing corporate social responsibility (CSR) and sustainable procurement. Free Pro achieved a

silver medal with a score of 61/100 in 2023, placing it in the top 25% worldwide. For 2024, Free Pro aims to achieve a gold medal and rank in the top 5% worldwide.



EcoVadis ratings of Free pro in red.
EcoVadis ratings of other companies in the same sector of activity as Free Pro in grey/black.

To facilitate communication and collaboration within the company, Free Pro utilizes several tools, including Teams for conversations and instant messaging, email for official communications and announcements, and Confluence for documentation and knowledge sharing. Within the LAB, meetings are organized as needed, with weekly check-ins to monitor progress, ensuring that team members stay aligned and any issues are promptly addressed. Employees at Free Pro work exclusively on virtual machines and do not work directly on their personal computers. They are encouraged to reach out to their supervisors or relevant personnel if they have any questions or need assistance.

C. First Project (Deployment of a 5G Core Network)

1. Introduction

The objective of this project was multiple.

Firstly, I had to deploy a 5G core network (5GC) **to test the internal documentation of Free Pro made to help the deployment process. This documentation, designed as an internal tutorial, provided step-by-step instructions for the set-up, configuration and deployment of the 5GC.**

Secondly, the project aimed to evaluate the performance of the deployed 5GC by evaluating its network's efficiency and reliability.

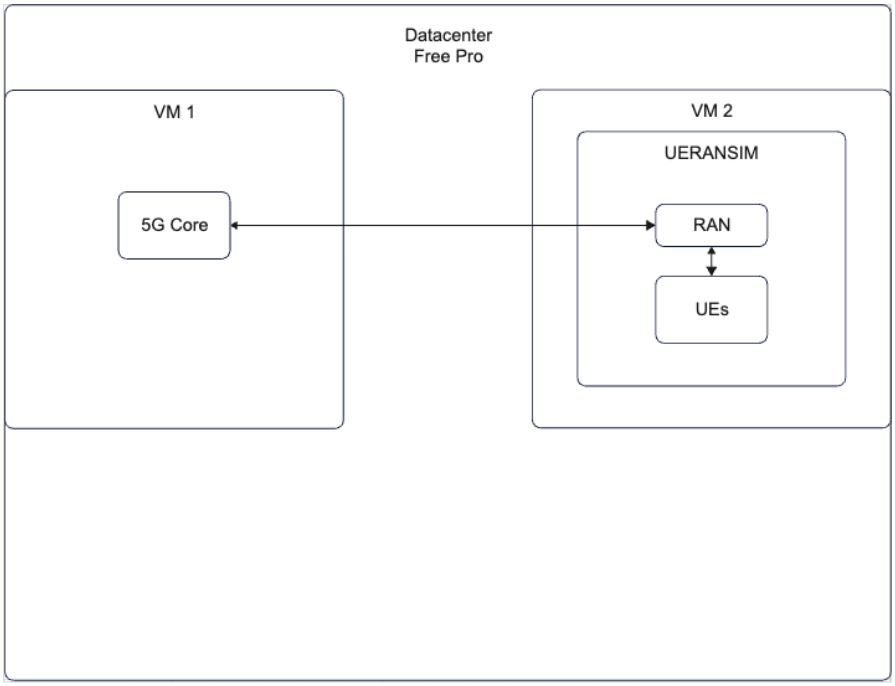
Finally, I had to determine my personal interest in this type of project. This approach was part of my supervisor's initiative to guide me towards projects that align with my preferences and skills. In an environment focused on research and development, where innovation is at the core of activities, it is crucial to work on projects that interest us.

2. Project Structure and Difficulty Faced

First, I requested a Virtual Machine (VM) **to deploy the 5GC. Once the VM was available, I followed the documentation step-by-step and successfully deployed the 5GC. However, there were some points in the documentation that I found unclear, so I reported them for updates. With the deployment complete, it was time to proceed with testing.**

Testing the 5GC required connecting some user equipment (UE) (for example: a phone, a computer equipped with a dongle, or any devices that can be connected to a 5G network)**. However, for a UE to connect to the 5GC, a radio access network** (RAN) **is needed to establish the connection between the UE and the 5GC. Fortunately, to test our 5GC, we use a software named UERANSIM that emulates both a RAN and multiple UEs.**

To execute the test, I requested a second VM and installed UERANSIM and configured it to connect to the deployed 5GC. With everything set up, I proceeded to test the performance of the 5GC by gradually increasing the number of UEs connected to it. During the test, every UE were simultaneously downloading some random data to simulate real-world usage scenarios.



Final architecture diagram of the deployed 5GC in VM1 and UERANSIM in VM2.

3. Conclusion

In conclusion, this project of deployment of a 5G core network was an instructive experience, although I was not as passionate about this type of configuration task as other aspect of development. My effort to overcome the difficulty encountered during this project allowed me to develop my adaptability and collaboration within the team. However, to better fit with my interests and skills, I was assigned to a new project which offers me the opportunity to be more involved in programming and innovation. This change was beneficial for me as it allowed me to better exploit my talents and enthusiasm for the development of innovative technological solutions.

D. Second Project (AGV: Learning and Familiarization with ROS and the Robot)

1. Introduction

The objective of this second project during my internship at Free Pro was to familiarize myself with the ROS (Robot Operating System) (ROS is a set of open-source software libraries and tools that helps building robot applications, facilitating communication between robot system components and providing tools for building) **development environment and to gain experience with manipulating and controlling the AGV** (Automated Guided Vehicle) **robot. This initial phase of the project aimed to assess my interest and skills in robotics and prepare myself for the following step of this project.**

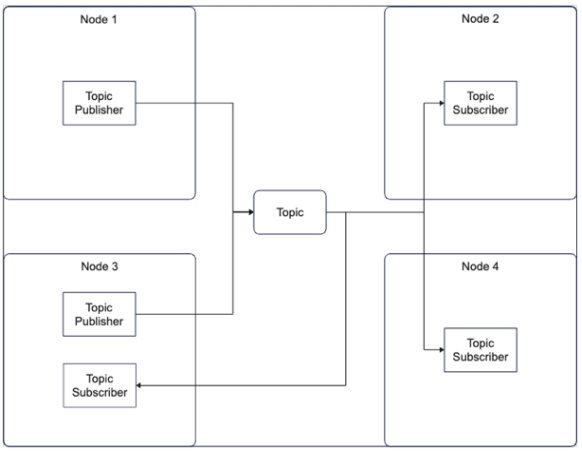
ROS Logo

2. Project Structure and Difficulty Faced

To start, I immersed myself in the ROS documentation, which offers comprehensive tutorials to understand the fundamental concepts of ROS. Using one of the virtual machines previously employed in the preceding project, I carefully worked through these tutorials, mastering the basics of ROS without encountering any significant challenges.

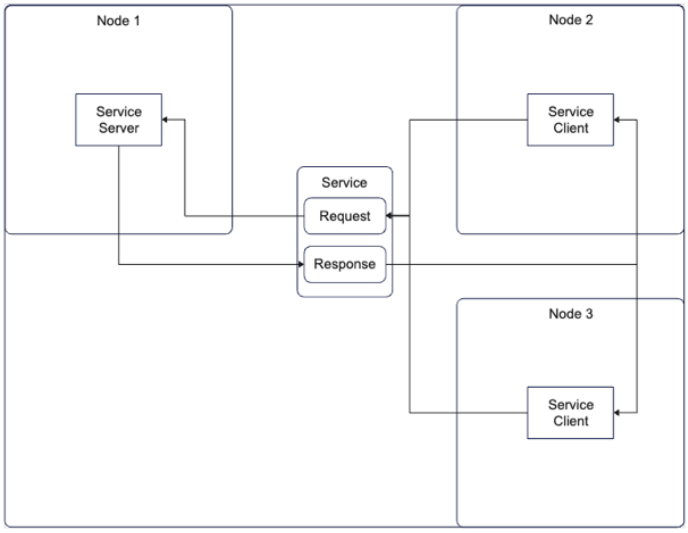
One of the key learnings during this phase was understanding the three primary communication types in ROS: topics, services, and actions.

Topics function similarly to a forum, where messages are published by one entity and received by all subscribers.



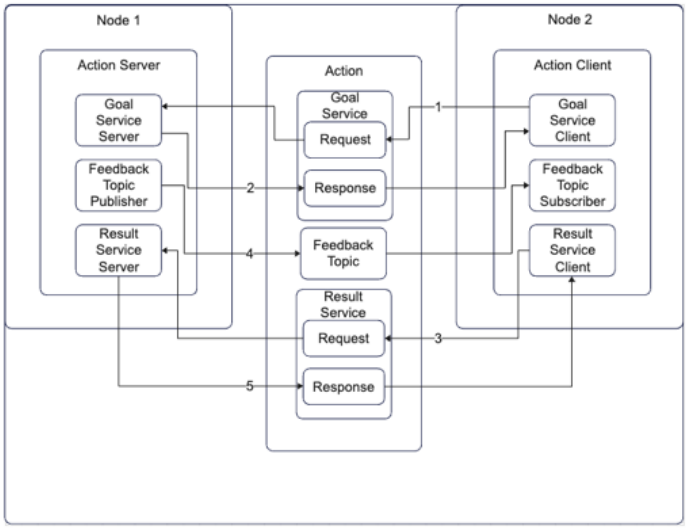
Topic Diagram where nodes 1 and 3 send data to the topic and nodes 2, 3, and 4 listen data from the topic.

Services enable clients to send requests for specific tasks, receiving responses upon task completion.



Service Diagram where node 1 contains the service server (we can have only one service server per service) and nodes 2 and 3 contain service client to send a request to the service server.

Actions, on the other hand, provide additional functionality by allowing communication during task execution, like tracking progress with a loading bar.



Action Diagram where node 1 contains the action server (as for the service we can only have one action server per action) and node 2 contains an action client to request the action server.
1) The action client sends the initial request to the action server via the goal service.
2) The action server accepts or refuses the received request.
3) If the request was accepted, the action client would call the result service to wait it's response. If the request was refused, the action would stop.
4) The action server sends the progress of the action to the action client using a topic.
5) The action server sends its response to the action client via the result service.

During this phase, I also learned how to create my own ROS node. A ROS node is an executable that can be made in either Python or C++ which will communicate with ROS to receive and send data via the previously explained topics, services, and actions from the other running nodes.

With a solid understanding of ROS fundamentals, I transitioned to interfacing with the robot. I created my own node in Python and leveraging the pre-configured sensors, which transmitted data to their respective topics, I made the robot move forward in the absence of obstacles and rotate left or right based on detected obstructions, facilitated by the 360° LiDAR 2D sensor. Implementing these functionalities required creating a publisher to relay movement data to the motor control topic.

3. Conclusion

In conclusion, this project of familiarizing myself with ROS and the AGV robot was engaging. Discovering an interest in robotics and successfully implementing functionalities like obstacle avoidance fueled my enthusiasm. This experience solidified my commitment to continue exploring and innovating in this field. As a result, I was tasked to continue working on this robot to make it map autonomously its environment.

E. Third Project (AGV: Autonomous mapping)

1. Introduction

Now that I learned how ROS works and confirmed my interest in this project, it's time to enable the robot to move autonomously, rather than simply going forward or turning when encountering obstacles.

The objectives of this third project were to make the robot move autonomously to explore its environment, mapping the environment that was explored, implement a system for the robot to locate himself in the generated map and finally save the generated map in a file to use for the next project.

Photo of the AGV Robot

2. Project Structure and Difficulty Faced

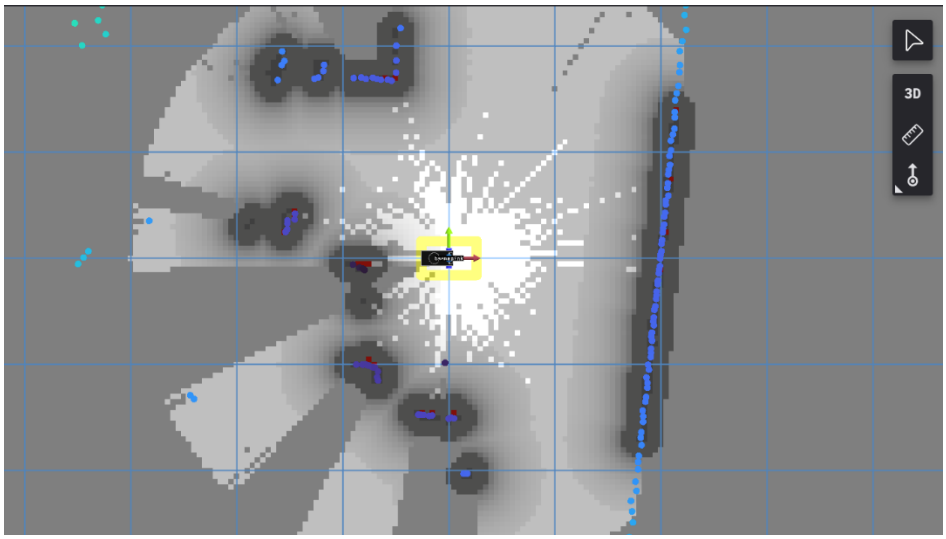
Initially, I attempted to create the map manually using data from the robot's movement, computed from the motor's encoders, and LiDAR sensor. I tried to calculate the absolute position of each point received from the LiDAR (which provides 360 points, one per degree, along with the distance from the LiDAR) **relative to the current robot's position. The idea was to generate the map without the need for self-localization, as I could determine the robot's position based on the robot's movement data. However, this approach proved to be messy and ineffective.**

Later, I received advice from a colleague who had set up the robot's functionalities, suggesting that I search about "SLAM" (Simultaneous Localization And Mapping) **online. Given that ROS benefits from community contributions, he informed me that dedicated ROS nodes had already been developed for SLAM. Additionally, he mentioned the existence of nodes designed to navigate from one point to another using only a map, the robot's movement, and LiDAR data, eliminating the need for manual path computation, and following.**

Following the advice received, I searched for a node for SLAM (SLAM node) **and a node in charge of the navigation** (Nav node) **and downloaded them. Once downloaded, I proceeded to configure them according to my requirements. While I initially used the default configuration files for both nodes, certain adjustments were necessary. For the SLAM node, I had to specify the LiDAR and robot movement data sources. Similarly, for the Nav node, I had to define the data sources for the map, robot movement, and LiDAR. Additionally, I had to set some other parameters such as the robot's dimensions, maximum and minimum linear and angular speeds within the Nav node's configuration.**

After configuring the nodes, testing commenced. Although the initial mapping encountered some issues, investigation of the configuration settings revealed no errors. Upon further exploration, I discovered an issue in the calculation of the robot's rotational movement. This issue resulted from an inaccurate registration of the robot's dimensions, leading to incorrect rotational computations. Once fixed, the mapping process proceeded smoothly, allowing for accurate generation of maps. Furthermore, I successfully tested the robot's navigation capabilities by providing destination points, which it navigated to without encountering any issues.

Once the SLAM and Nav nodes were operational, I began integrating them with my own node. Before proceeding, it was necessary to understand how the map was generated. The SLAM node generated a map composed of three elements: free space, obstacles, and unknown space. Free space represented areas where the robot could move freely, obstacles were areas where movement was obstructed, and unknown space indicated areas that had not yet been explored or mapped.

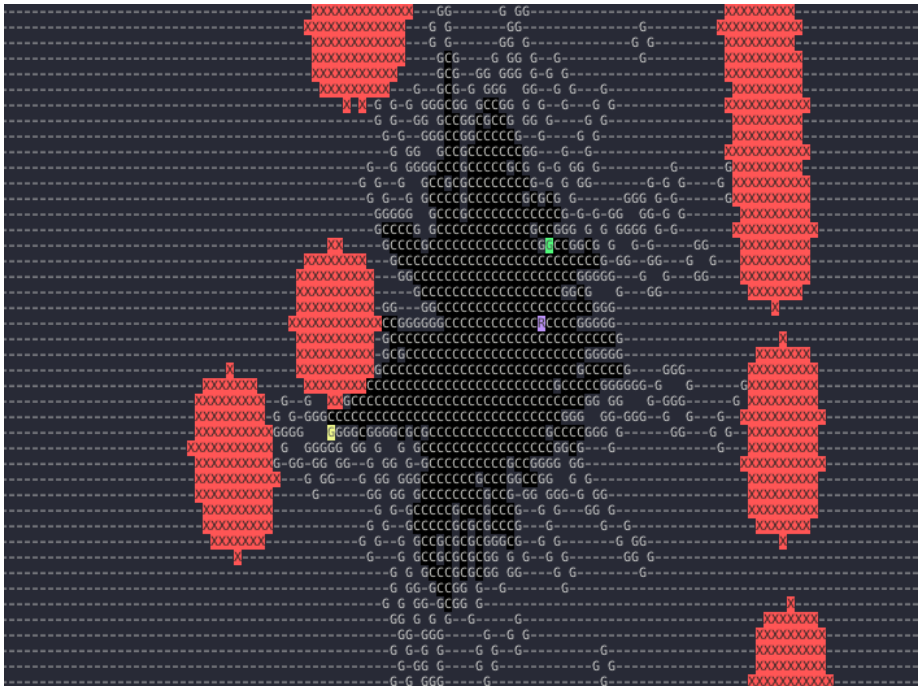


The map in black (Obstacles), grey (Unknown), and white (Free).
The LiDAR data in rainbow.

Initially, I attempted to retrieve the map from the SLAM node and locate myself on it using the provided position data. However, the position data was in meters relative to the starting point, while the map was represented as a grid of characters. As a result, I needed

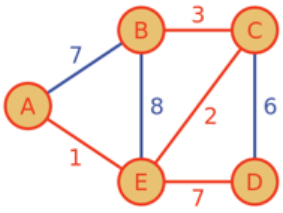
to convert the position data into grid coordinates based on the grid resolution (how many meters a single character represents) **and origin.**

Having determined the robot's location, the next step was to plan my route. Since the objective was to explore the environment, I identified every unknown space next to a free space as a potential destination. To select the destination, I employed the Dijkstra algorithm to find the furthest reachable point while considering obstacles. Opting for the furthest point allowed for further map exploration during movement.



The map seen by the robot with Obstacles as red X, Unknown space as -, Free space as space, Robot as blue R, Potential destination as G, Closest potential destination as Green G, Furthest potential destination as Yellow G, and Dijkstra movement from robot in black C.

Dijkstra's algorithm finds the shortest path to all point in a graph with positive weights (distance)**. It starts from the starting point, assigns infinite distance to all others except the start point which is set to zero. It selects the point with the minimum provisional distance, updates the distances of its neighbors if a shorter path is found, and repeats until all points are processed. This process ensures that the distance found for each point is the shortest possible.**

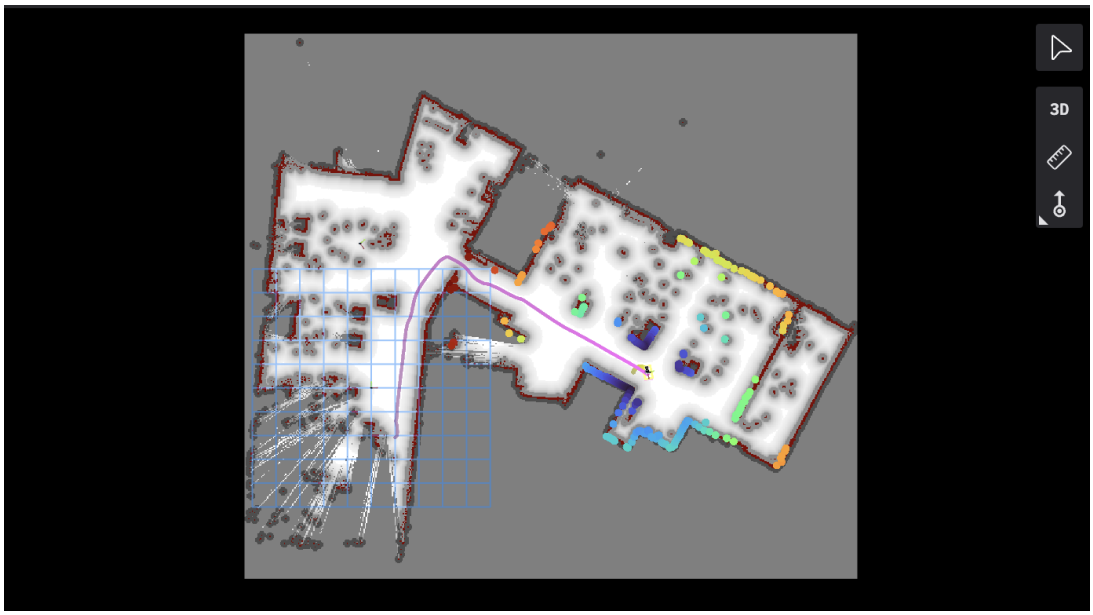


	Vis	Dist	Prev
A	1	0	-
B	1	6	C
C	1	3	E
D	1	8	E
E	1	1	A

Example of the Dijkstra algorithm in a graph with the result.
Dist: Distance of the path from the starting point (A)
Prev: From where we arrive to access the node

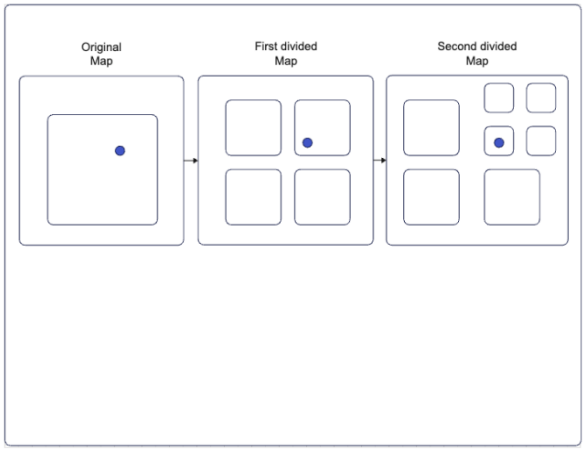
Now that I had chosen the destination, I converted the coordinates into metric coordinates and transmitted them to the Nav node via appropriate topics. However, a challenge emerged regarding feedback on the node's progress. Without feedback, the robot would continuously recompute paths, resulting in unnecessary movement. Fortunately, the Nav node developers had implemented an action feature that provided feedback on movement progress and a final success or failure when movement ended. This allowed me to determine if the robot was moving and avoid redundant computations.

All was working fine but there were two issues. The first problem emerged because the robot always aimed for the furthest reachable point, leading to inefficiencies in long or large spaces. It traversed the entire office, even if it hadn't explored the entire area, often returning later, ending up making numerous round trips, resulting in significant time loss. The second issue occurred when the map size increased, causing the Dijkstra algorithm to slow down considerably. Consequently, the robot remained stationary for approximately three to four seconds while computing its new destination.



Example of an unnecessary round trips thus making a long path
The current robot's path in pink/purple.

To address these issues, I implemented a solution involving partitioning the map into multiple submaps using a grid-based approach. If the map was too large or complex, determined by user-defined parameters and thresholds, I divided it by four and recursively split only the submap containing the robot's position. This recursive partitioning process continued until the resulting submaps were sufficiently small and simple. This partitioning allowed for the Dijkstra algorithm to be computed on much smaller submaps, reducing computational time, and preventing the robot from making unnecessary round trips.



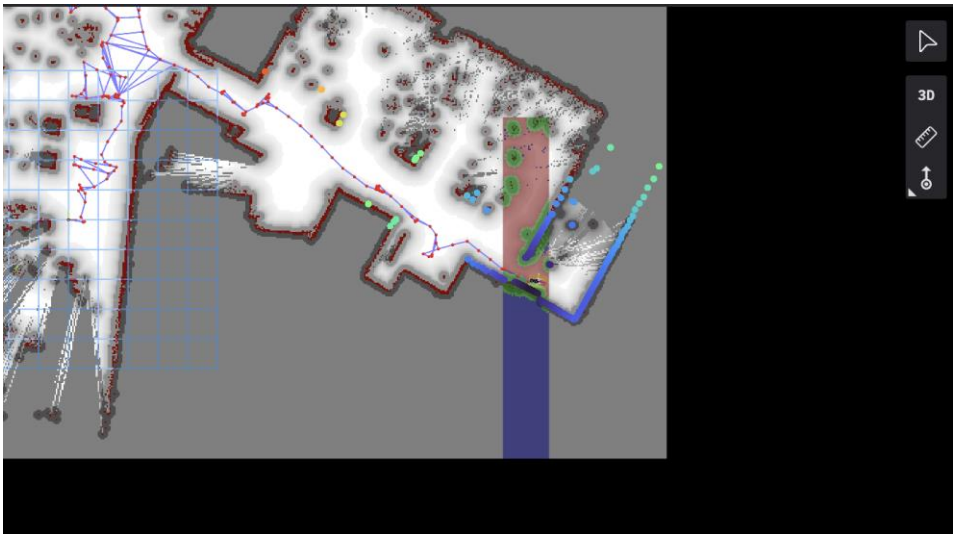
Map subdivision process.
The blue dot is the robot
Only the submap containing the robot is divided by four.

However, new problems have arisen, I had to keep my partitioned map updated with the map received from the SLAM node. Firstly, I had to take care of change in the new map's origin by shifting all the submaps to the left or top and deleting all elements that fell in negative position after the shift, or by creating some new submaps to the top or left of the partitioned map depending on the new map's origin change. Secondly, I had to take care of change in the new map's size by shrinking the submaps on the right or bottom, or by creating some new submaps on the right or bottom of the partitioned map depending on the new map's size change. Once I adjusted the partitioned map accordingly to align with the new map, updating all elements from the new map to the partitioned map became relatively easy.

The map and the current submap of the robot in Green (Obstacles), Blue (Unknown), and Red (Free)

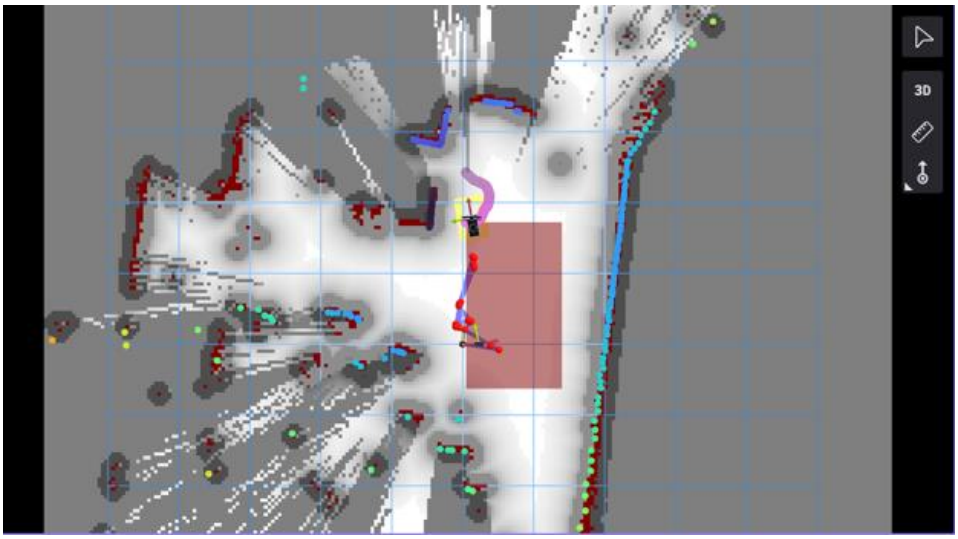
Everything is now functioning correctly. However, considering that the new map may only increase in width or height by one or two elements, the resulting submaps could become excessively long, such as being 500 elements in height but only one or two in width (although this is an extreme case, it can occur occasionally)**. To fix this issue, I introduced a user setting to determine the maximum aspect ratio of a submap. If a submap has an**

aspect ratio smaller than the defined one, it will be split either horizontally or vertically, depending on its longer side. Additionally, to prevent the partitioned map from being composed of too many thin submaps, if the number of submaps with a ratio below the user-defined ratio exceeds a certain user-defined value, the map is recreated to maintain a cleaner partitioned map.

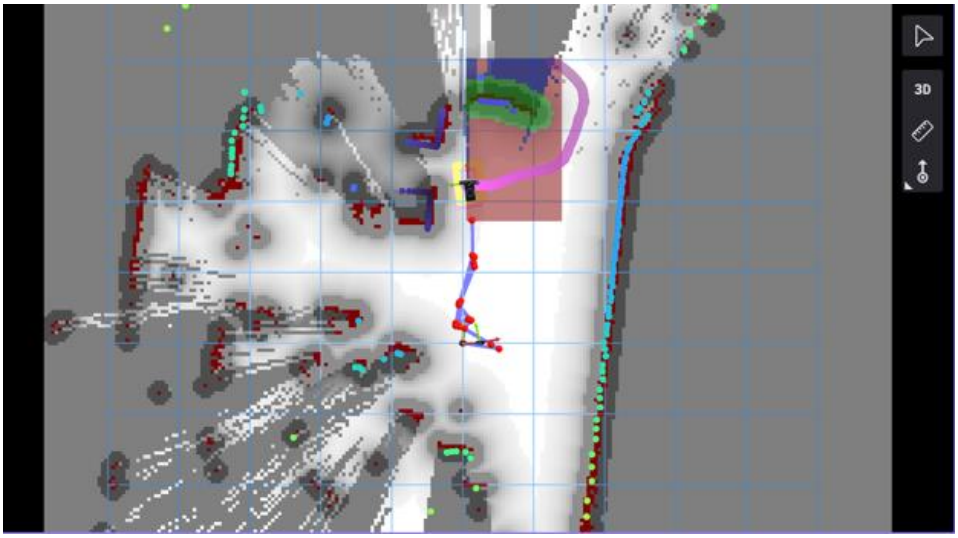


Example of long submaps that may appear.

Now that I have a functional partitioned map and know on which submap I am, I can easily determine the coverage percentage. If it's below a user-defined threshold, I continue exploring this submap using the same logic as before, aiming for the furthest unexplored point. If it's above the threshold, or if the rest of the submap is not explorable (for example, if there is a wall splitting the submap in half)**, I navigate to the closest unexplored point on the entire map. This allows the robot to change its current submap and begin covering the new submap.**



Going to the closest accessible unexplored point in the entire map because the current submap is fully covered.



Going to the furthest accessible unexplored point in the current submap to cover it.

The robot is now capable of autonomously exploring its environment and generating a map of it. However, even with the partitioned map, computation remained slow due to the high computing demands of the SLAM and Nav nodes. To fix this, I used a specialized computer, named GPU, available at the research and development center, which is designed for AI task and thus more than well-suited for the robot's needs.

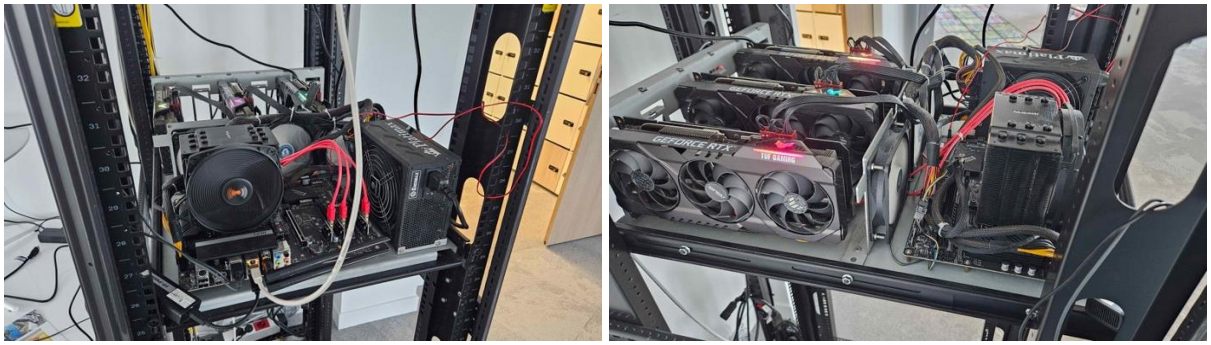
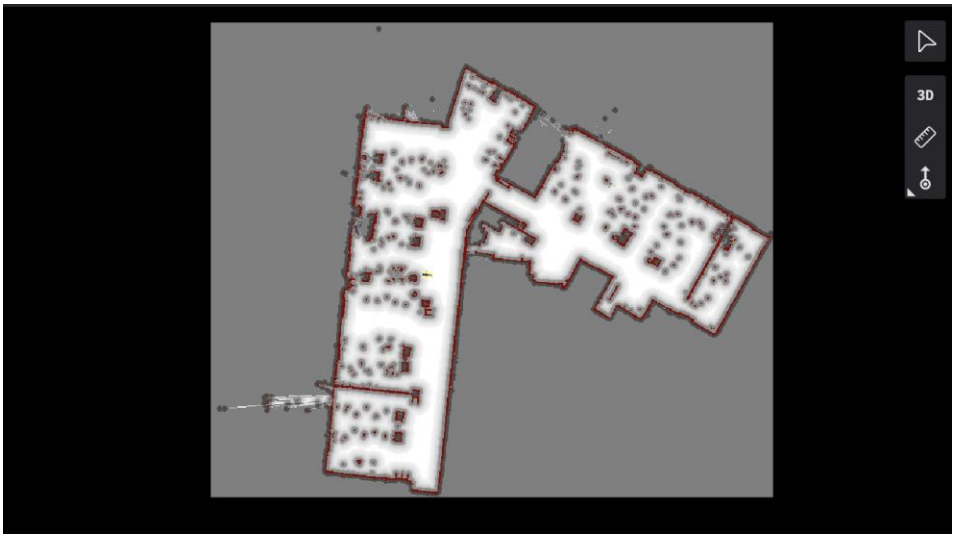


Photo of the GPU

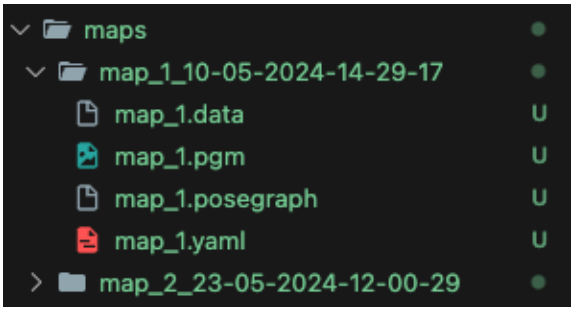
The robot and the GPU are both connected in 5G, the GPU via a dongle and the robot via a 5G chip, in the same subnetwork which enable them to communicate together. Since ROS can be distributed across multiple PCs, I chose to use the GPU for its ample computing resources. Initially, I attempted launching ROS without configuration, as it is supposed to automatically discover all devices on the network. However, this approach failed. Moving forward, I discovered the possibility of creating a server (by default, ROS manage by itself where the topics, services, and actions are registered and communicate depending on which machine they're used on, but with a server every topic, service, and action are registered on the server and must pass through it) **using ROS's default middleware** (software used by ROS to communicate)**. I attempted configuring it accordingly, which partially worked, except for the monitoring functionality, which required superuser access. Even after granting superuser access, the monitoring feature still did not function properly. Searching for an**

alternative solution, I discovered that it was possible to change the middleware used by ROS. Eventually, I found another middleware that only required a configuration file containing the IPs of the machines on which ROS would run. After specifying the use of this new middleware and providing the path to the configuration file, I launched ROS, and it worked perfectly. As a result, the robot can now autonomously explore and map its environment without any slowdowns, thanks to the edge computing setup I just implemented.



Result of the generated map

Now that everything is functioning properly without any slowdowns, it's time to save the generated map. This task is relatively straightforward, as both the Nav node and the SLAM node have services for saving the map. The Nav node provides a service to save the map in a human-readable format, such as an image, while the SLAM node offers a service to save it in a computer-readable format, allowing it to be used in future executions to skip the mapping process or continue an unfinished mapping. To save the map, I simply had to call the appropriate service, specify the desired saving location for the map, and wait for the response to determine whether the operation succeeded or failed.



The maps are saved in their dedicated folder in a human-readable format (yaml/pgm) and in a computer-readable format (data/posegraph)

3. Conclusion

In this project, I successfully enabled the robot to move autonomously, explore its environment, generate a map, locate itself within the map, and save the generated map for future use. Implementing SLAM and Nav nodes significantly enhanced the robot's capabilities, allowing it to map its environment accurately and navigate without issue. To optimize performance and address computational constraints, I partitioned the map into smaller submaps, improving path planning efficiency.

Using edge computing resources, specifically the GPU, proved to be useful in overcoming computational limitations, enabling seamless mapping and navigation tasks without any slowdowns. Overall, this project represents a significant milestone in developing an autonomous mapping system using ROS.

This project will be useful for the following project, focusing on the network coverage to determine then connection speed at different points on the map.

F. Fourth Project (AGV: Autonomous network coverage)

1. Introduction

Now that the robot can detect its environment, move autonomously, and locate itself in its environment, this next project involves instructing it to travel to desired points, perform some connection tests, and send the result for analysis. Then, these data will be transmitted to a server to store them. Finally, the project aims to develop a graphical user interface to visualize the network coverage, providing a comprehensive overview of the network performance within the robot's environment. This graphical user interface will facilitate the identification of areas of both strong and weak connectivity. The goal of this project is to identify zones with varying levels of connection strength to optimize the number and placement of cells.

2. Project Structure and Difficulty Faced

To begin, I needed to develop a new node dedicated to performing connection tests, ensuring that it would operate on the robot itself, since my other node now runs on the GPU. It was essential that the tests originate from the robot's position rather than the GPU, which isn't moving. So, I created the necessary node and service to execute multiple connection test when receiving a request from the GPU node. Then, the results are sent to the GPU for further processing.

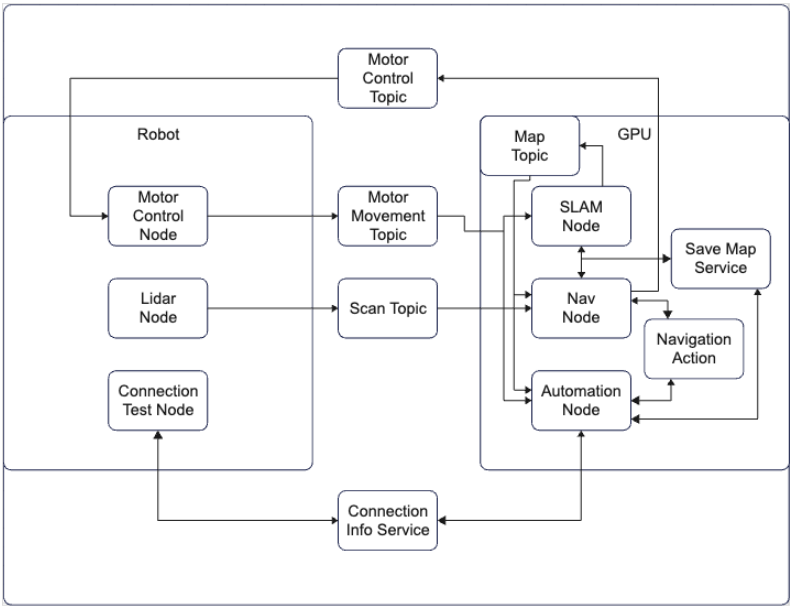
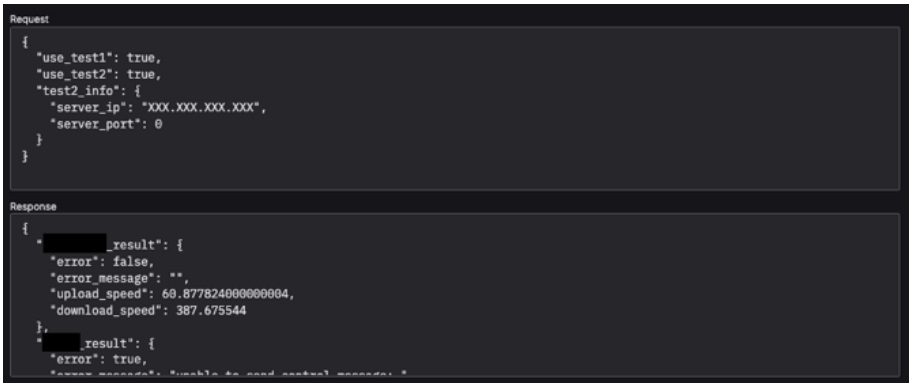


Diagram showing where each node is and the topic/service/action connections

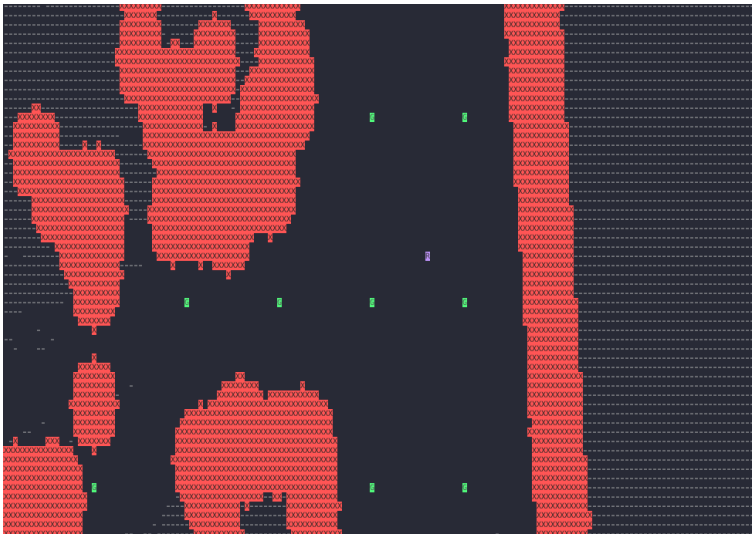
However, to create the service I had to create my own ROS message type, which is only possible in a C++ package. Given that I initially developed the nodes in Python, I had to create a new C++ package dedicated to creating my custom ROS message type. Once my custom ROS message was created, I could easily make the GPU node request for a connection test whenever is needed.



Call of the created service to test the connection speed.

Now that I have a map and can perform connection tests from any location, I can instruct the robot to navigate to specific points and perform the tests. However, to achieve this, I needed to determine whether the robot was in state of idle, mapping its environment, or performing the network coverage. Previously, the robot only needed to know if it was in control or not, performing mapping when it was. To address this, I implemented a global state and substate system for the robot. This system allows the robot to identify whether it is in idle, mapping, or network coverage mode thanks to the global state, while the substate provides further details such as whether the robot is in motion, computing its next move, or anything else. Additionally, I created a new custom ROS message type to relay this state information, enabling users to monitor the robot's current global and substate.

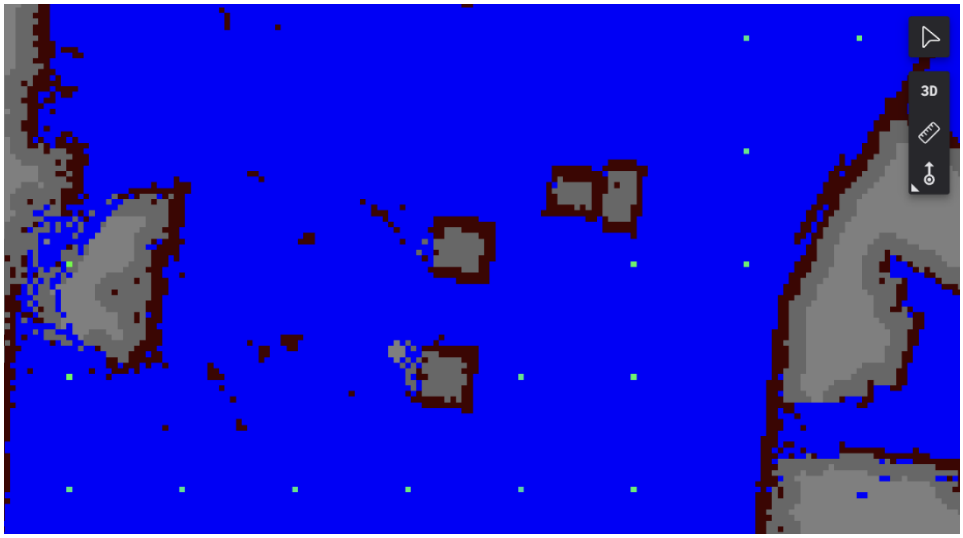
Now, I can easily determine the current robot's actions and execute either the network coverage movement or the mapping movement as required. The next step is to determine how the robot will navigate to cover the entire area with connection tests. To achieve this, I decided to, based on a user-defined variable (named X)**, start from the origin point** (where the robot begins mapping) **and place a new test point every X meter to the left and right until I reach the map's boundary. Then, additional test points are placed every X meter higher and lower from all these points. Only the accessible points are kept, while those falling into obstacles or unknown spaces are removed.**



Example of the generation of point to test (Green G).

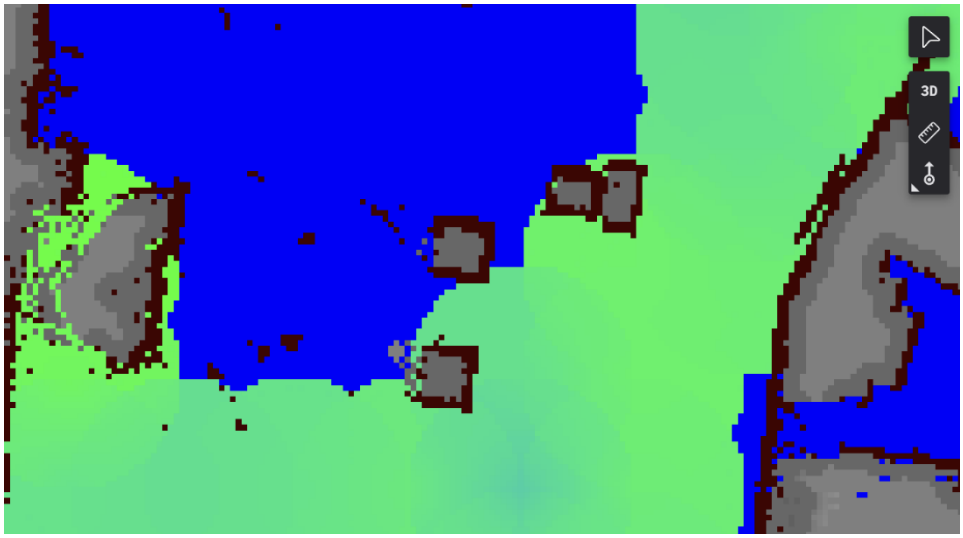
Now that I know where to go, I simply needed to navigate to the closest point, call the previously created service, retrieve the data, and repeat this process until every point has been tested. However, I had nothing to determine if the service call had been completed, as service calls are asynchronous. Consequently, once the robot reached a point, it immediately proceeded to the next closest point. Nonetheless, with the previously made substate, I only needed to create a new substate indicating that a connection test was being performed. If the robot is in this state, it remains still until receiving the service response.

Using the data received from the service, I can finally generate a map that displays the network speed at the tested points. I integrated the results of the service calls into the map generated by the SLAM node. Since the map can only display values between 0 and 100, I identified the highest value obtained from all service calls and set it to 100, scaling all other values proportionally.



Map with the connection point (Green Point).

However, this method was not very effective for identifying areas with slower connections. To improve this, I applied a smoothing filter to the map while restricting the smoothing effect within a certain distance from the tested points. This prevented the entire map from becoming totally green due to the smoothing. Now, the map displays a gradient from 0 (blue) **to 100** (green)**, effectively highlighting areas with varying connection strengths.**



The same map as before but with the connection point smoothed.

Now that I can display the connection tests in real-time on the map, I needed to save them for sending to the storage server, reviewing later, and using in the graphical user interface. To do this, I formatted the data into a YAML file, a configuration format already used by ROS for node-specific configurations and by the Nav node for saving maps. In the file, I specified the exact position where each service call was made, a rounded version based on the distance between each test for easier human readability, and the response from the service call.

```yaml
# Network coverage map info
# Format: rounded_pos, pos, test1_result, test2_result, signal_info
data:
  - rounded_x: 1.0
    rounded_y: 0.0
    x: 0.8907193879766668
    y: -0.10777010872416035
    test1_result:
      download_speed: 0.0
      upload_speed: 0.0
    test2_result:
      download_speed: 79.46923653372 74
      upload_speed: 80.96999724011727
    signal_info:
      rsrp: -74.0
      rsrq: -11.0
      snr: 38.0
  - rounded_x: 1.0
    rounded_y: -1.0
    x: 0.8929330848589443
    y: -0.9684815626229082
    test1_result:
      download_speed: 0.0
      upload_speed: 0.0
    test2_result:
      download_speed: 71.24495831524617
      upload_speed: 72.92932223682196
    signal_info:
      rsrp: -76.0
      rsrq: -11.0
      snr: 37.0
map_name: map_2
map_path: /home/buxg/ros_ws/freerabbitagv/maps/map_2_23-05-2024-12-17-50
```

The save file of the connection points.

3. Conclusion

In this project, I successfully developed a system for the robot to autonomously navigate and perform network connection tests across its environment. By implementing new nodes and services, and creating custom ROS message types, the robot can effectively map network coverage, generate a gradient map indicating connection speeds in real-time, and save the data into a file for future use.

However, due to time constraints, I was unable to complete the final steps of sending the data to a remote storage server and developing the graphical user interface. Consequently, the data visualization and analysis remain limited to the robot's local environment. Despite these limitations, the project represents a significant advancement in autonomous network coverage mapping.

G. Conclusion

During my internship at Free Pro, I had the chance to work on diverse projects that significantly enhanced my skills in telecommunications and robotics. I started with deploying a 5G core network, where I tested and improved internal documentation and evaluated network performance. Despite initial challenges, this project taught me valuable lessons in problem-solving and collaboration.

My next project introduced me to robotics with the AGV robot and ROS, where I developed the robot's basic movement and obstacle avoidance capabilities. This phase sparked my interest in robotics and led to a more complex task: enabling the robot to autonomously map its environment using advanced SLAM techniques.

Finally, I focused on autonomous network coverage mapping, where I integrated navigation algorithms and connection testing functionalities into the robot. While I couldn't complete the final steps of data transmission and GUI development due to time constraints, this project marked a significant achievement in assessing network performance autonomously.

Throughout my time at Free Pro, I benefited from a supportive and innovative work environment. I would like to extend my sincere gratitude to the LAB team at Free Pro Marseille for welcoming me and providing an enriching and collaborative atmosphere. Their guidance and support were instrumental in the successful completion of my projects.

II. Section 2 (Proposal for Future Projects to the Manager)

A. Introduction

Dear Narcisse Kamtchoum,

I hope this message finds you well. I am writing to express my interest in contributing to Free Pro's ongoing artificial intelligence (AI) **projects. Based on the experiences and skills I have developed during my internship, I believe I can add significant value to these initiatives.**

B. Context and Background

During my internship at Free Pro, I engaged in multiple projects that expanded my technical knowledge and allowed me to develop key skills directly applicable to AI projects. My first major project involved deploying a 5G core network. I followed detailed documentation to set up and test the network, which improved my understanding of network infrastructure and performance optimization, which are crucial elements for data-intensive AI applications. Additionally, I worked on developing an Automated Guided Vehicle (AGV) **using the Robot Operating System** (ROS)**. This project required me to program and control the robot, implementing navigation algorithms, obstacle avoidance mechanisms, and autonomous mapping capabilities using SLAM** (Simultaneous Localization And Mapping)**. These experiences required advanced problem-solving skills and a deep understanding of sensor integration and real-time data processing.**

C. Skills and Experiences

Throughout my internship, I demonstrated adaptability by overcoming various challenges and refining processes to ensure successful project outcomes. The projects I worked on required innovative thinking to develop new solutions and improve existing systems. For instance, optimizing the robot's autonomous mapping and network coverage assessment involved creative problem-solving and continuous iteration to enhance performance and accuracy.

These experiences not only improved my technical competencies but also allowed me to develop strong analytical skills. My work on network performance evaluation and autonomous mapping demonstrated my ability to analyze complex data and derive actionable insights, which are crucial for developing effective AI models and algorithms. Additionally, my innovative mindset, demonstrated through my ability to think creatively and develop novel solutions, is valuable in addressing the complex challenges inherent in AI projects.

D. Proposal for AI Project Involvement

Free Pro's commitment to innovation is evident in its ongoing projects, which focus on advanced 5G technologies, artificial intelligence, and edge computing. Given my background and experiences, I am particularly interested in contributing to the development and implementation of AI-driven solutions that leverage these cutting-edge technologies.

My experience with 5G networks and robotics aligns well with the requirements of AI projects that necessitate robust data handling and real-time processing capabilities. I believe I can make significant contributions to Free Pro's AI projects. For instance, I could leverage AI to enhance network performance and reliability, building on my 5G core network deployment experience. Additionally, I could further develop the AGV's capabilities with AI-driven navigation and decision-making systems, improving its efficiency and functionality. Moreover, using AI for advanced data analysis and predictive modeling could drive strategic insights and improve service offerings.

E. Conclusion

My internship at Free Pro has been a transformative experience, providing me with the skills and knowledge necessary to contribute effectively to AI projects. I am excited about the idea of working on these innovative initiatives and am confident that my background aligns well with the goals of our AI projects.

I am looking forward to discussing how my experiences and skills can be used to further our AI initiatives and contribute to the ongoing success of Free Pro.

Thank you for considering my request.

Sincerely,

Kenan Blasius