# Search and Rescue with Spot the Robot Dog

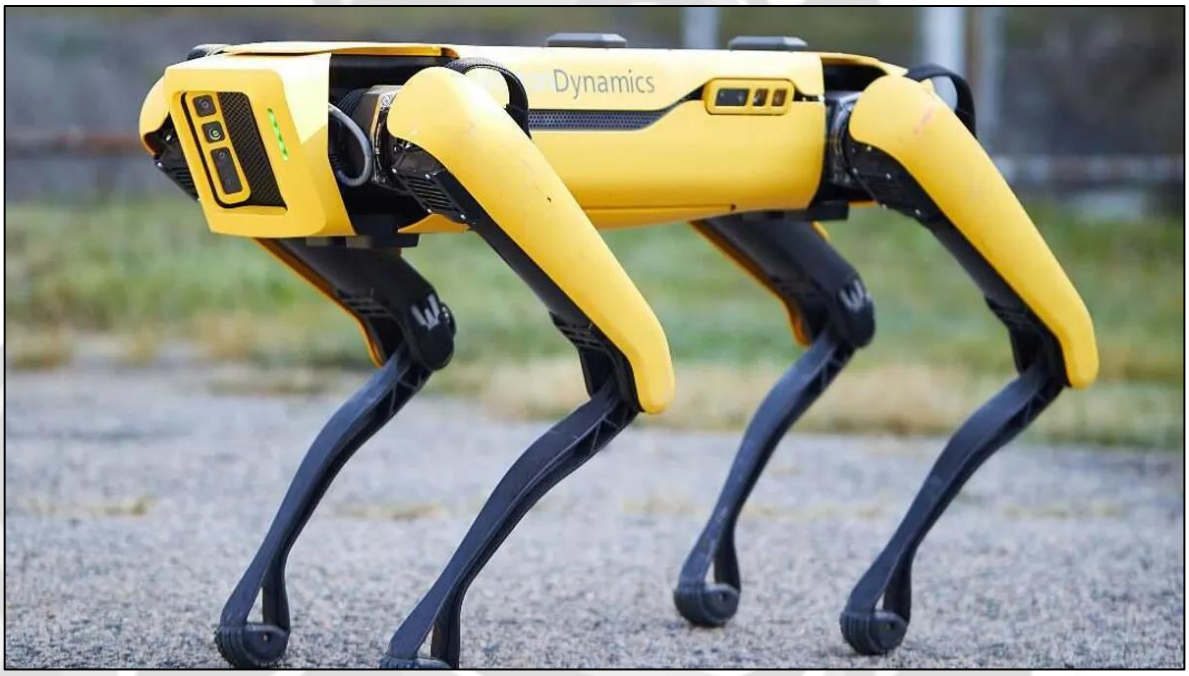



*Figure 1.1- Boston Dynamics SPOT Dog (Kooser, 2020)*

SPRINT NUMBER: 4

DOCUMENT VERSION NUMBER: 1.0

DATE: 29/10/2024

SUBMITTED BY: Liam Faulkner-Hogg

**TEAM DETAILS**


| Student Number | Full Name | Email |
|---|---|---|
| 14260816 | Justin Pavlovski | justin.l.pavlovski@student.uts.edu.au |
| 24633947 | Akkarawat Kaveewatcharanont | Akkarawat.kaveewatcharanont@student.uts.edu.au |
| 14272165 | Connor Fitzgibbon | Connor.K.Fitzgibbon@student.uts.edu.au |
| 13939475 | Liam Faulkner-Hogg | liam.faulknerhogg@student.uts.edu.au |
| 14260792 | Pravien Kugan | Pravien.kugan@student.uts.edu.au, |
| 14278055 | Yves Gayagay | yves,b,gayagay@student.uts.edu.au |

## Table of Contents

# Project Description

The ability for robots to autonomously map, localize, and navigate their environments is critical for their effective deployment in real-world applications such as search and rescue missions, industrial inspections, and environmental monitoring.

Search and rescue missions often take place in complex and hazardous environments, which pose significant challenges to human personnel. As a result, there is a pressing need for robotic systems that can independently perform these tasks, reducing risks to human operators while enhancing the efficiency and effectiveness of missions.

This project aims to develop these capabilities into the SPOT robot dog, a quadrupedal robot designed to traverse rough and unpredictable terrains. The project focuses on leveraging sensor data to implement Simultaneous Localization and Mapping (SLAM) algorithms, alongside sophisticated path planning and control systems. These advancements will enable SPOT to perform critical tasks such as obstacle avoidance, trajectory optimization, and precise motion planning in dynamic environments.

Ultimately, this project will contribute to the broader goal of making robots more intuitive and easier to use, facilitating their seamless integration into human environments and expanding their applications.

## Sprint Summary

Key priorities for the team include meeting all sprint requirements, maintaining clear communication among team members, and delivering a well-organized and efficient project workflow. By the end of this sprint, we have successfully chosen a project environment, used the mapping and localisation package, assessed constraints and risks, and written a plan for our project.

# SPRINT 1

## SLO 1

### SLO 1.1: Communicating with the stakeholder

Over the last 2 weeks we have been able to establish email communication and have an online meeting with our primary stakeholder, **Sangmim Song**, which took place on Wednesday 14th August. In this meeting, we were able to gain invaluable information pertaining to our design objectives and get feedback on how we have progressed so far. We were also able to express our interest in applying our project on a physical system and sought out the required steps we must take to get there. The appropriate minutes of meeting can be found below in Appendix A. This document details the discussions that occurred during the meeting and outlines what needs to be completed for the next meeting as well as the time and date of the next meeting. This is so that we as a team can be held accountable while also helping to focus our efforts to ensure our time is spent efficiently.

## SLO 1.2 Identifying and agreeing on the priorities, goals and system requirements

As a team we have had many discussions regarding what we want to achieve with our project as well as 41068 as a subject. Through this, we have come up with the below list of priorities, goals and overall system requirements.

### Team Priorities

1. Meeting all sprint submission requirements before the due date
2. Effectively utilise communication tools to ensure all members are on the same page
3. Attend at least 1 weekly meeting so that we can support each other
4. Keep our documents clean, concise and well organised
5. Celebrate as a team

### Team Goals

- To produce a robust system for a robot to assist in search and rescue situations
- To eventually implement our system on a real SPOT robot dog

### System Requirements

- To effectively implement and interpret sensor data to perform a task
- To utilise SLAM to navigate and map an environment
- To develop a path planning method which adds value to search and rescue operations
- To report important information to the user such as the location of people in need of assistance
- To implement a closed loop control system to manipulate and manoeuvre the robot through its environment

# SLO 2

## SLO 2.1 Problem Statement

Search and rescue operations are inherently risky, often placing personnel in dangerous situations as they work to locate and assist victims in challenging environments. The hazardous nature of these missions not only threatens the safety of rescuers but also limits the efficiency and effectiveness of the operations. As a result, there is a continuous need for improvements in how these missions are conducted to ensure the safety and success of all involved.

Currently, search and rescue efforts are heavily reliant on human personnel, frequently placing themselves in life-threatening situations. The limitations of this approach are evident:

- Rescuers are exposed to significant danger.
- The coverage area is often restricted due to the complexity and vastness of the environment.
- The operations are resource-intensive, requiring extensive coordination and manpower.

Moreover, the time it takes for human teams to assess, enter, and search these dangerous areas can delay locating and rescuing victims, which could be detrimental to their chances of survival.

Integrating SPOT, a robotic system equipped with an array of sensors capable of both manual and autonomous control, into search and rescue operations offers a promising solution. SPOT can be manually operated to search through rubble from a safe distance or autonomously navigate large and dangerous areas, thereby minimising the need for human personnel to enter these hazardous environments. This integration not only enhances the safety of rescuers but also improves the efficiency of operations by covering larger areas more quickly and thoroughly. SPOT's advanced sensors and autonomous navigation capabilities also increase the likelihood of effectively locating injured or trapped individuals, even in the most challenging conditions. By leveraging SPOT, search and rescue missions can become safer, more effective, and more resource-efficient, ultimately leading to quicker and more successful rescues.

## SLO 2.2 Functional requirements and design parameters of the project

*Table 1 - Functional Requirements and Design Parameters*

| Design Parameters | Functional Requirements | Where Requirements Came From |
|---|---|---|
| Must include adequate processing power to handle real-time data processing, SLAM computations, and path planning algorithms. | The robot must be able to navigate autonomously in a complex environment. | Personal Requirement |
| | The robot must be capable of creating a map of its environment using sensor data. | Personal Requirement |
| | The robot must be able to perform trajectory optimization to ensure efficient path planning. | Personal Requirement |
| Must include wireless communication, including both control and data transmission. | The robot must provide real-time feedback to operators on its status, including position, and any detected issues or anomalies. | Personal Requirement |
| | The robot must allow for wireless manual control up to 50m away | Personal Requirement |
| | The robot must display a constant video to its human operators. | Personal Requirement |
| The robot must have physical properties that allow it to operate efficiently within disaster areas | Must be able to operate within 50-degree Celsius | (Eser, 2024) |
| | Must be able to fit through 55 (d) x 200 (h) cm | (Perth Window and Door, 2023) |
| Design for resistance to environmental factors such as dust, water, and extreme temperatures. | Ingress Protection = IP54 (water resistance, dust resistance) | (Glamox, 2024) |
| The robot must operate effectively on different terrains and uneven surfaces | Max Slope = +- 30 degrees | Personal Requirement |
| | Max height variation required to navigate = 250 mm | (Sydney, 2015) |
| Modular design which allows optimization of robot for specific disaster situations and quick setup | Can carry up to 14kg of sensing equipment with the ability to swap sensors for specific tasks | (Boston Dynamics, 2024a) |
| | Setup of robot takes less than 8 minutes | (Commission, 2023) |
| Camera should be able to work in multiple environments | Camera should be able to record through environment such as dark or foggy weather | (intel REALSENSE, 2024) |

## SLO 2.3 Technical statement

**Workspace and Simulation Environment**

The elements of the project such as the source code, assets and configuration files will be administered and stored in a private repository through GitHub; where it is open solely to team members for collaboration. These source codes, which are crucial to the project, are developed by the C++ language with ROS2 middleware integrated for robotic system interfacing under the Linux operating system. Testing and commissioning are conducted in a physics-based simulator named Rviz, which is in-housed by ROS2. This simulator contains many customizable options to allow users to recreate environments inspired by real-life scenarios. For this project, the use of a simulator is the main mode of the project's deliverable as requested by the client. Because of this mode, they can be integrated to any compatible computer systems for wide range accessibility such as public outdoors or in personal spaces. Also, simulators provide streamlined client feedback (*Simulation-based Learning*, 2024) towards the team during development to create an efficient project workflow, whilst proving itself to be a useful tool to interact with the client on certain builds *(Simulation-based Learning, 2024)*. Finally, simulators output realistic models if handled comprehensively, adopting these models onto real-world apparatuses can create desirable results with minimal physical tuning.

**Sensors**

- Cameras
  Cameras provide the optical sense for the SPOT robot, allowing the integration of computer vision that is now readily available for setup without added costs through open-source programs like OpenCV. In this project object detection, where images are acquired by cameras and processed in software, it's crucial to complete the missions for the SPOT robot. This is because object detection immensely benefits the robot's mapping and localization capabilities within its environment in conjunction to the implementation of the SLAM algorithm. Additionally, on the extreme end, budget friendly cameras would suffice as a great sensor towards robotics systems, due to it being a product of a highly competitive market which tends to target a low-price point.

- Laser/ LIDAR
  LIDAR is specialized to calculate distance between itself and the obstacles at all spectra from near to far ranges. Even with its limited FOV, LIDAR is highly valuable because of the high accuracy of its reading that it inputs towards the robotic system. This would be noticeably favorable over its ultrasonic counterpart because

of it. LIDAR can also be used to track the SPOT robot's positions within a plane. This results in improving its localization capabilities when traversing through missions.

**Platform**

The main platform where the project will be developed is the SPOT robot dog, a flagship product developed by Boston Dynamics. According to official product specifications by Boston Dynamics themselves, SPOT has 12 DOF, allowing for complex maneuverability around difficult environments *(Boston Dynamics, 2024b)*. It also comes equipped with 5 in-built stereo cameras, which capture both images and videos in binary formats *(Boston Dynamics, 2024b)* for streamlined processing. The form factor of the robot is relatively light, only weighing 35kg. As a result, there will be less resistance to the robot's translation and rotational movement in respect to the robot's weight.

## SLO 2.4 Design Objectives & SLO 2.5 Evaluation criteria

Design objectives and how they will be measured in complacency through criteria are listed below. The list comprises of elements based on the client's interest and the team's reasoning and critical analysis.

*Table 2 - Design Objectives and Evaluation Criteria*

| Design Objective | Evaluation Criteria | Necessary or Desired |
|---|---|---|
| Implement Sensor data<br>- Ability to correctly identify objects and environment | - Will successfully identify 3 unique objects and the walls and ground of the environment<br>- Algorithm recognize objects almost instantly (<~5ms)<br>- Utilizes camera and Lidar | Necessary |
| Implement SLAM<br>- Will be able to identify it's own pose and the relative pose of the objects within the environment | - SPOT should be able approximate the location of desired objects to 100mm | Necessary |
| Path Planning<br>- Will implement priority system to create a weighted path | - Independently generates the most effective path in a pre-determined environment<br>- Hazard Avoidance built in (Doesn't kill Spot), correctly identifies hazardous environments | Necessary |
| Locating<br>- Will implement a system that reports back the location of the survivors | Scan and find survivor on the site and report back their location | Necessary |
| Implement Control System | Utilizes closed loop control to manipulate and move the robot | Necessary |
| Can Perform in Various Environments | Can meet all requirements in the following 3 environments<br>• Commercial<br>• Industrial<br>• Natural | Desired |
| Real time simulation | Can simulate with real time compatibility (does not need to include graphical sim) | Desired |
| Live transmission of information | Provides real time mission status updates | Desired |
| Can hear you scream | Spot correctly identifies locations of different sounds in a test environment | Desired |

## SLO 2.6 Timeline for the Sprint 1

The team worked on the first sprint where each member is allocated in their given tasks that must be completed by the end of the period. A completed Gantt chart below in *figure 1.2* is made to trace progression and manage productivity in completing the sprint. These tasks can also be viewed alternatively in a dashboard with respective allocations and progression statuses in *Appendix B through notion*.

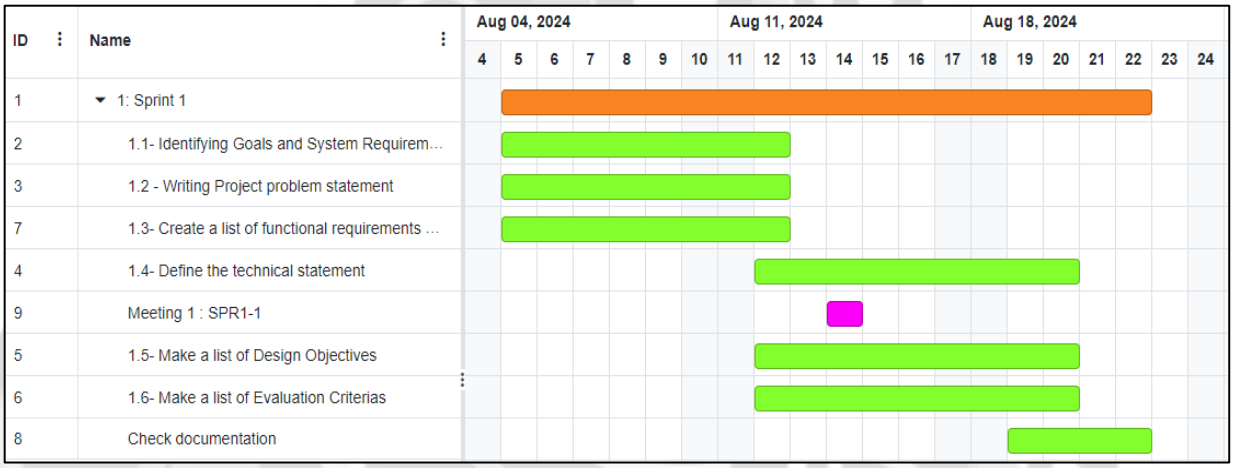



*Figure 1.2 - Team Gantt Chart*

Throughout this sprint, there are milestones and highlights that the team were able to accomplish within the period, these are:

- Meeting up with the client
- Establishing a problem statement and localizing areas of priorities and scope
- Selecting components such as mode of delivery, sensors and platforms.
- Defining design objectives and evaluation criteria for the project.

# SLO 3

## SLO 3.1 Configuring the system

Setting up Linux and ROS2 using Azure was straightforward. The ROS2 installation guide provided on the subject's Canvas page worked seamlessly during the initial setup. The inclusion of the talker and listener test code was particularly reassuring, as it confirmed that everything was configured correctly and minimized the risk of encountering issues on the first day.

However, I encountered a challenge when installing VS Code. The installation guide linked on Canvas had an incorrect step toward the end. Fortunately, the error messages displayed during installation suggested an alternative command, which resolved the issue and allowed me to install VS Code successfully.

## SLO 3.2 Set up a simple indoor office environment in Gazebo

Locating the installation paths for the worlds, models, and images used in the pre-made Gazebo environments was challenging with multiple turtlebot3_gazebo folders in the ros2_ws, which made importing the necessary materials to visualize new environments difficult. To avoid finding the folder, I began by following the README file that came with the new downloaded world, but this approach fell short as it lacked a launch file which prevented it from opening.

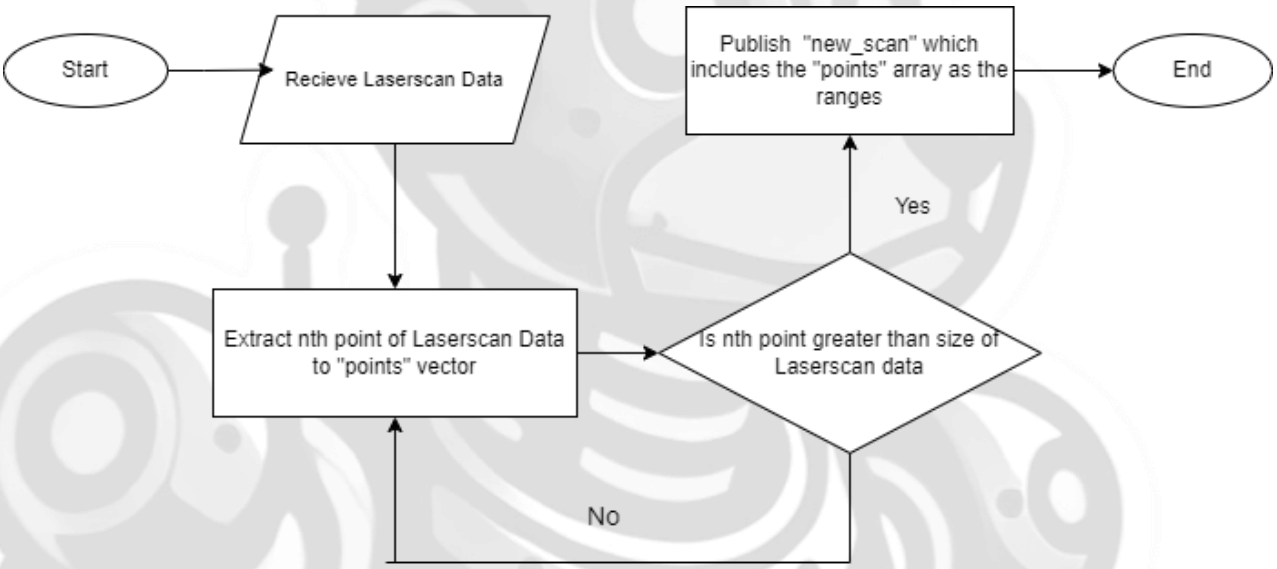
After some investigation, I found the folder containing the files for turtlebot3_gazebo. I then added the previously downloaded models and worlds to their respective directories within the correct turtlebot3_gazebo. Finally, I duplicated an existing launch file, modified it to open a different world, and renamed it. With these adjustments, all my Gazebo environments loaded correctly.

## SLO 3.3 Design and develop a subscriber/publisher

My Lidar_Points node filters the laser scan data from the Lidar sensor, selecting every nth point (as specified by the user under the “desired_points” variable) from the scan. The node subscribes to the “scan” topic, processes the incoming data to filter out unnecessary points, and then publishes the filtered data to a new topic.

1. **Subscription**: The node subscribes to the "scan" topic, which provides raw Lidar data.

2. **Filtering**: In the scanCallback function, the algorithm calculates the indices of every nth point in the ranges array using a for loop. It then creates a new Laser Scan message that includes only these filtered points.
3. **Publishing**: The filtered Laser Scan data is then published to the "filtered_scan" topic after assigning the max_angle, min_angle, and angle_increment for the new laser_scan data to ensure it is correctly visualized in Rviz.



# SPRINT 2

## SLO 1

### SLO 1.1 Identifying Constraints and Risks

| Constraints | Description |
|---|---|
| **SPOT Movement** | When the SPOT robot moves, its camera can shake, leading to potential interruptions or distortions in sensor data. This can affect the quality of the images or data collected during movement. |
| **Environmental Interference** | Factors like heavy rain, fog, dust, or smoke can degrade sensor performance. This may result in inaccurate or incomplete data, affecting the robot's ability to perform its tasks effectively. |
| **SPOT Maneuverability** | SPOT must adhere to specific size dimensions to navigate confined spaces and narrow passages. This size limitation can restrict its ability to access certain areas. |
| **LIDAR** | The Lidar sensor has a maximum effective range. Beyond this range, the accuracy and reliability of the data decreases significantly, which can limit the sensor's usefulness for distant objects or features. |
| **Mapping** | 2D Mapping:<br><br>2D maps created by the robot do not capture the full three-dimensional environment. This limitation can be particularly problematic in dynamic or changing environments, such as those affected by disasters.<br><br>Dynamic Environment:<br><br>In real-life scenarios, environments can be dynamic, with people or objects moving. This movement can affect the accuracy of the data collected and the robot's ability to navigate or complete its mission effectively. |

The following risk register was utilised to for the risk assessment. Each cell contains the risk severity and risk score (e.g. High (6)). Also included below is a breakdown, detailing each aspect of the risk register.

*Table 2.1 - Risk Register*

| | | **Likelihood** | | |
|---|---|---|---|---|
| | **Level** | **1** | **2** | **3** |
| **Impact** | **1** | Low (1) | Medium Low (2) | Medium (3) |
| | **2** | Medium Low (2) | Medium High (4) | High (6) |
| | **3** | Medium (3) | High (6) | Very High (9) |

*Table 2.2 - Risk Register Explanation*

| **Level** | **Description** | |
|---|---|---|
| | **Likelihood:** Chance of the risk occurring during the project | **Impact:** Effect the risk will have on the project |
| **1** | Very low to low chance, but exists or can be easily avoided/mitigated | Very low to low effect, project can continue with minimal changes |
| **2** | Medium chance, and/or cannot be easily avoided/mitigated | Medium effect, project may be able to continue with some changes |
| **3** | High to very high chance, and/or it cannot be avoided/mitigated | High to very high effect, project cannot continue without major changes |

**Risk Matrix and treatment plan:**

Below in Table 3 is the complete risk matrix and treatment plan. This outlines the risk ID, description, likelihood, impact, risk score, risk response and reasoning. It is also further broken up into risk categories including team management, simulation, shared simulation and real world, and real-world deployment. The possible responses to risks are:

- **Acceptance** – The risk does not pose a large threat to the project or is almost completely out of the teams control, therefore the risk will be accepted
- **Mitigation** – The risk poses a threat to the project; however, actions can be taken to ensure that the likelihood of the risk occurring is reduced.
- **Avoidance** – The risk poses a threat to the project; therefore, the project plan is to be changed to eliminate the risk.
- **Transferring** – The risk poses a threat to the project; therefore, if possible, risks will be passed to a third party for management.

*Table 2.3 - Risk matrix and treatment plan*

| Risk ID | Risk Description | Likelihood | Impact | Risk Score | Response | Reason |
|---|---|---|---|---|---|---|
| **Team management risks** | | | | | | |
| **1** | Project priorities goals and system requirements are poorly defined | 1 | 3 | 3 | Mitigation | The project priorities, goals and system requirements need to be well-defined to achieve successful completion of the project. Extensive stakeholder feedback can mitigate this risk |
| **2** | Functional requirements and design parameters are poorly defined | 2 | 2 | 4 | Mitigation | If the project scope or requirements change, it could lead to missed deadlines and poor execution. Mitigation involves establishing clear and documented requirements early on. |
| **3** | Team miscommunication or conflicts | 1 | 2 | 2 | Avoidance | Clear roles, frequent check-ins, and proper documentation help avoid misunderstandings or disagreements that may slow down progress. |
| **4** | Issues with team members availability | 2 | 2 | 4 | Mitigation | Team members may have conflicting schedules, leading to delays. Mitigation involves setting clear deadlines and assigning backup roles if a member is unavailable for a certain period. |
| **5** | Resource or budget constraints | 1 | 2 | 2 | Acceptance | These constraints are typically outside the team's control, but proper planning, prioritising key features and considering any financial issues will help keep the project on track. |

| | | | | | | |
|---|---|---|---|---|---|---|
| **Simulation risks** | | | | | | |
| **6** | Simulation failure or errors | 2 | 3 | 6 | Mitigation | Simulation issues can disrupt testing and development. Debugging, peer code review, and testing of the simulation environment can help to reduce the chance of disruptive errors. |
| **7** | Coding bugs and delays | 2 | 3 | 6 | Mitigation | Coding errors are very common but can be mitigated through systematic testing, pair programming and version control. Regular sprints for progress monitoring can help avoid major delays |

| Risk ID | Risk Description | Likelihood | Impact | Risk Score | Response | Reason |
|---|---|---|---|---|---|---|
| **Shared simulation and real-world risks** | | | | | | |
| **8** | Failure to meet performance benchmarks | 2 | 2 | 4 | Acceptance | The robot may not perform to expected benchmarks (e.g., speed, accuracy). This may be accepted if trade-offs are made for mission success, but performance issues must be monitored. |
| **9** | Failure in localisation and mapping | 2 | 3 | 6 | Mitigation | Failure to correctly localise and map the environment will likely lead to mission failure. Mitigation involves robust localisation and mapping algorithms and frequent testing. |

| | | | | | | |
|---|---|---|---|---|---|---|
| **10** | Failure in obstacle detection and avoidance | 2 | 3 | 6 | Mitigation | Obstacles might be missed or not properly avoided, likely leading to mission failure. Mitigation involves robust obstacle detection algorithms and frequent testing. |
| **11** | Failure to locate victim in simulation | 2 | 3 | 6 | Mitigation | Enhanced algorithms for image processing and object detection, combined with extensive testing across a variety of scenarios should reduce the likelihood of failure when locating the victim. |
| **12** | Sensor inaccuracies | 3 | 2 | 6 | Mitigation | Sensor inaccuracies could impact search and rescue algorithms. Improving the calibration and accuracy of sensors within the simulation environment and the real robot can mitigate this issue. |
| **13** | Software or firmware updates causing issues | 2 | 2 | 4 | Mitigation | Unforeseen software or firmware updates could create compatibility issues. Keeping a stable version for the project and thoroughly testing any updates can mitigate this risk. |
| **14** | Data corruption or loss | 1 | 2 | 2 | Transference | Regular backups, using cloud storage for documentation and Git for code will transfer the risk of data loss to third-party providers and minimise the direct risk to the project |
| **15** | Hardware failure | 2 | 3 | 6 | Mitigation | Computers used to run simulations or real robot malfunctions could halt operations, but backup systems and preventative maintenance and testing can help mitigate this risk. |

| | | | | | | |
|---|---|---|---|---|---|---|
| **16** | Incompatibility between simulation and real robot | 2 | 3 | 6 | Mitigation | Differences between the simulated environment and the real robot's hardware could lead to unexpected behaviour. Mitigation is possible through careful integration planning and testing. |

| Risk ID | Risk Description | Likelihood | Impact | Risk Score | Response | Reason |
|---|---|---|---|---|---|---|
| **17** | Real world data not reflected in simulation | 2 | 2 | 4 | Mitigation | Data used in simulations might not fully reflect real-world conditions. This can be mitigated through gathering real-world data from similar missions to create the simulation model. |
| **18** | Failure in communication between robot and operator | 2 | 3 | 6 | Mitigation | Communication failures could halt the mission or prevent obtaining the needed data. Reliable communication protocols and backup communication methods can mitigate this risk |
| **19** | Time constraints during mission simulations | 2 | 2 | 4 | Mitigation | Limited time to simulate multiple mission scenarios could reduce the quality of testing and planning. Mitigation involves prioritising critical simulations and optimising the simulation process. |
| **Real-world deployment risks** | | | | | | |
| **20** | Limited access to real robot for testing | 3 | 3 | 9 | Mitigation | If access to the real robot is restricted, it could impact validation of the simulation. Mitigation through |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | extensive simulation testing before any limited real-world trials is necessary for success. |
| **21** | Safety risks during real world testing | 1 | 3 | 3 | Avoidance | Ensure safety protocols and risk assessments are in place to avoid injuries during real-world testing, especially in unpredictable environments. |
| **22** | Battery depletion during mission | 1 | 3 | 3 | Acceptance | Given that battery life is a known issue in robots, planning missions around the battery limits or including spare power packs may acceptably manage this. |
| **23** | Environmental challenges (terrain) | 2 | 3 | 6 | Mitigation | Real-world search and rescue missions often encounter rough terrain that could cause issues for the robot. Simulating various environments and planning for contingencies can mitigate the risk. |
| **24** | Physical damage during deployment | 1 | 3 | 3 | Acceptance | Robots are utilised for search and rescue missions to reduce the risk to humans and improve efficiency. Whilst damage should be avoided if possible, damage is acceptable if it is to save a human. |
| **25** | Unanticipated legal or ethical issues | 1 | 3 | 3 | Acceptance | Real-world search and rescue using autonomous robots could involve legal or ethical issues, such as liability for damage or harm. Acceptance of these challenges should include risk planning. |

## SLO 1.2 Identify any cultural, ethical, environmental, legislative and economic perspectives

Our project has major implications for the cultural, ethical and economic aspects of our society. It has both great opportunities to enhance our societies perspective on robotics but also great risk and liability in situations of malfunctions.

**Cultural Perspectives:**

Our autonomous system will greatly enhance the accessibility of first responders as it can operate in environments that might be deemed inaccessible or dangerous. This allows the development of a new workplace culture that allows emergency services to work collaboratively with SPOT to effectively save lives.

First responders are often highly respected culturally and for our project to be successful, SPOT must be seen the same way. Lin et al. (2017, Chapter 10) discusses how cultural perspectives on robots can change especially when in situations of saving human lives. They reveal that the key to ensuring positive human-robot interactions is trust. To maintain trust between humans and robots, our robot design must include a strong positive intention, clear and understood risks, and good will.

**Ethical Perspectives:**

Lin et al. (2017, Chapter 5) explores the ethical issues that arise between robotics and liability. With regards to liability, most levels of robotics can fall under the same regulations as other technologies for example, misuse of a technology implies user liability and poor build quality implies manufacturers liability. However, if the robot is running autonomously and causes damage, who becomes liable? There are arguments to be made that liability should be split between the user and designer but in more advanced systems such as AI, it could be argued that the robot should be given legal personhood thus making it liable.

The use of sensors such as cameras brings up the issue of privacy, especially when the intended use space of our robot is within private spaces like homes or healthcare facilities. Emergency services have their own privacy policies which involve storing information that is exclusively related to the services activity (Fire and Rescue NSW, 2021). Storage of data from robotics is essential for the continued development of both the software and possible use cases of the technology (Marchang & Di Nuovo, 2022). For this purpose, emergency services and their collaborators should have access to camera footage providing it complies with government privacy regulations.

**Economic Perspectives:**

Robotics is regularly discussed in the world of economics as it presents challenges and opportunities for things like job creation, business practices and economic policy (Dirican, 2015). Our product could be seen as taking jobs from first responders. However, it could also be seen as redefining the position of first responders in order to enhance their safety. Both perspectives have strong merit, but the inclusion of this technology would undoubtedly change the economic landscape as it is intended to enhance the capabilities of first responders. This impact would also be restrained by the ability to reskill the workforce in order to utilise and co-operate with SPOT.

# SLO 2

## SLO 2.1 Present the overall project idea

SPOT starts by processing a 2D top-down floor plan through a path-planning algorithm to determine its optimal route. As SPOT navigates the environment, its LiDAR sensor measures distances to surrounding objects, while an IMU tracks its translational and rotational velocities. A SLAM algorithm continuously logs SPOT's position, enabling it to adjust its route if an obstacle is encountered.

SPOT's camera captures RGB images, which are analysed by a YOLO object detection algorithm to identify people in danger. Upon detecting a person, SPOT transmits their location to the operator. The operator can see real-time visuals from SPOTs camera of the injured persons and can opt for manual control to further assess the situation over camera. After all locations have been searched, SPOT returns to its starting point, ready for collection or new instructions.

SPOT movement style is rooted from sensor readings and calculated metrics from SLAM to estimate the location of the robot in future point of time whilst manipulating the robot as well. The algorithm will include elements of probability randomness through normal distribution to simulate real-world elements which is deduced from the Kalman filter.

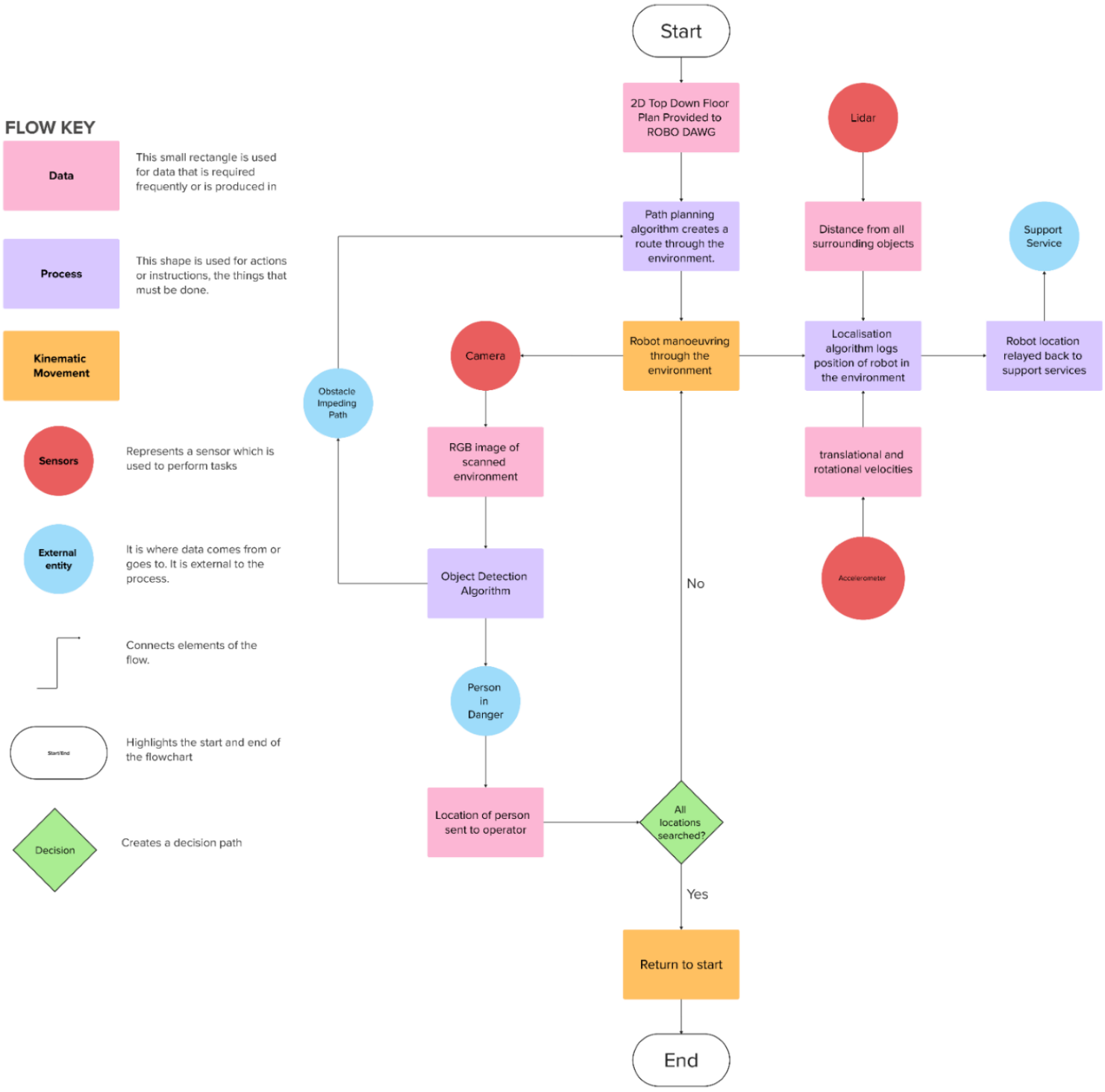



*Image 2.1 Process flow chart for project idea*

## Pseudo Codes

### Movement

import cmath library
import ROS msg libraries - twist, odometry, Quaternion , tf2_geometry_msgs
import probabilities (random) library
import ROS node library

***** Define subscription and publications of topics *****
(subscribe to teleop)
publish to /cmd_vel
publish to /noisy_odom

***** Initialize instructions into node *****
check status
if teleop
publish teleop input -> /cmd_vel;
if algorithm
Initialize parameters such as linear speed, angular speed , distance and direction;

***** Initialize Time frame =0 (T0) parameters *****
Create a quaternion matrix that are made from the x,y,z orientation of the pose msgs;
Extract the yaw angle from the quaternion and set it as the initial angle of the robot in time 0;
Extract the initial x and y position from the pose msgs;

***** Estimate final position and publish to cmd_vel *****
Capture T and use T0 or previous iteration as T-1 to get elapsed time;
Use Elapsed time to find calculate robot's location on x and y coordinates through polar to cartesian calculation;
Use elapsed time to find angle by multiplying with angular speed;

Calculate the travelled distance between the current location of the robot and the initial or previous iteration location of the robot through hypot;

Check Distance travelled
if >= Distance (parameter)
publish to /cmd_vel

if < Distance (parameter)
publish a gaussian noise through normal distribution to be refined by algorithm in next iteration;
publish to /cmd_vel and set t-1 as time at this point;
Loop back

### Object Detection

***** Import msg types and libraries used for node *****
Import YOLO Library
import OpenCV
Choose YOLO model v8

***** Define subscription and publications of topics *****
Create Object_Detection node
subscribe to "camera/image_raw",
create publisher to "camera/image_modified"

*****Camera Callback Function*****
Takes image from camera
runs image through YOLO algorithm
stores results in matrix
returns results

*****Predict and detect function*****
extracts information from YOLO results matrix
runs detected items through a switch statement
**switch (results.detections)**
if **person detected**
run "visualization" function
store location in "injured_personnel_location" matrix
send location data to support personnel
if **obstacle detected**
run "visualization" function
store location in "obstacle_location" matrix
send data to support personnel
create new path around obstacle
if **hazard detected**
run "visualization" function
store location in "hazard_location" matrix
send data to support personnel
create new path around hazard
if **none of the above**
continue movement through enviornment as determined by SLAM algorithm

*****Visualization Function*****
recieve results from YOLO algorithm
for result in results
create box around detected objects/ people using OpenCV
label boxes with type of object and confidence level
publish new image to "camera/image_modified"

### Localization

***** Pseudo Code for Cartographer Localization Node *****

***** Include Required Libraries *****
necessary libraries

***** Node Initialization, Subscription to Topics, and Publisher Setup *****
initialize_node("cartographer_localization")

***** Subscribe to laser scan, odometry data, and submap topics *****
subscribe_to_topic("/scan", LaserScanCallback)
subscribe_to_topic("/submap_list", SubmapCallback)

***** Create Publisher for Robot Estimated Pose *****
create_publisher("/estimated_pose", PoseData)

***** Load Parameters and Setup Frames *****
load_parameters()
setup_frame("map_frame")
setup_frame("tracking_frame")

***** Sensor Data Topics and Time Synchronization Setup *****
setup_sensors()
synchronize_sensor_data()

***** Callback Function for Laser Scan Data *****
function LaserScanCallback(laser_data):
cartographer_process_laser_data(laser_data)
update_map_and_pose()
***** Publish Estimated Pose *****
function publish_estimated_pose():
pose_data = get_estimated_pose()
publish_to_topic("/estimated_pose", pose_data)

### Mapping

*****Initialize ROS Node*****
initialize ROS node ("slam_toolbox_node")

*****Set up Subscribers*****
subscribe to "/scan" topic for laser scan data

*****Set up SLAM Toolbox Parameters*****
set parameters for SLAM Toolbox
- map_update_interval
- resolution
- max_range
- use_sim_time
- localization_mode

*****Start the SLAM Process*****
- launch "slam_toolbox" with the specified parameters

*****Process Sensor Data*****
while node is running:
receive laser scan data from "/scan"
*****Perform SLAM update*****
update map using incoming sensor data
update robot position on the map using odometry

*****Publish updated map*****
publish the updated map to "/map" topic

### SLO 2.2 Robot navigating in the environment while displaying sensor data

Setting up the project environment was much smoother this time around, as the process is now familiar compared to the challenges faced in Sprint 1. The environment being used from this point onward is a small warehouse layout, downloaded from Ahtsan's "aws_robotics/aws-robomaker-small-warehouse-world" GitHub page. This environment also came with a premade map, making the mapping task straightforward. However, an issue arose when plotting `odom` vs. `amcl_pose` on an `rqt_plot`. The results were wildly inconsistent, with the plot jumping around erratically and not reflecting the TurtleBot's actual behavior. This issue was resolved by using an alternate version of the map provided with the environment. The problem is believed to have been caused by a misalignment between the map's centering and the Gazebo environment, which led to inaccurate results with the initial map.

## SLO 3

### SLO 3.1 Provide a short description of a mapping task

Mapping involves creating "spatial models of physical environments" by processing sensor data collected by the robot (Stanford, 2002). This data is used to generate representations of the robot's surroundings, allowing for an understanding of the environment. These models are then used to improve robot navigation and localization within the environment. Common sensors used for mapping include:

**LiDAR**: Measures the time of flight of a self-emitted laser to determine the distance from objects (HowToRobot, 2023).

**Sonar Sensor**: Measures the time of flight of a sound wave (typically in water) to determine the distance from objects (popular for mapping the sea floor) (MaxBotix, 2024).

**Stereo Vision Camera**: Measures the distance to objects by calculating the difference in an object's location as seen by two cameras, similar to human vision (e-con Systems, 2024).

Robotic mapping presents challenges such as filtering sensor noise, navigating dynamic environments, and managing processing power (just to name a few). All sensors inherently produce some degree of error in their readings. Filtering this noise to achieve accurate data requires complex algorithms, which still can't guarantee perfect accuracy. Over time, this can lead to robot drift, gradually introducing errors into the map. Mapping dynamic environments is also challenging, as moving objects can create inconsistent data, necessitating even more complex algorithms to maintain accurate maps. These advanced algorithms demand significant processing power to achieve real-time mapping, requiring

more expensive hardware and larger batteries. This may not be feasible for smaller or budget-conscious robots, generally making mapping an expensive task to perform.

## SLO 3.2 Demonstrating the mapping task in gazebo environment

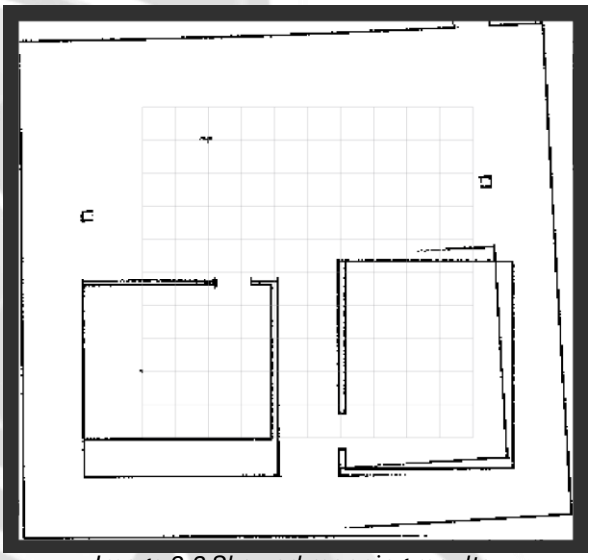


*Image 2.2 Skewed mapping results*

While using the NAV2 and slam toolbox to generate some maps, I encountered errors where the scan would slightly rotate midway through the process (drift). This caused walls that were previously mapped to be remapped at an angle, resulting in a distorted and inaccurate map as seen in image 2.1. I discovered that this issue arose when I moved the TurtleBot around bends too quickly and with inconsistent turning circles, which disrupted the robot's localization and caused it to perceive the scans at an incorrect angle. To address this, I attempted to redo the mapping multiple times, focusing on more precise movements by guiding the robot in a square pattern, turning 90 degrees at a time. While this approach yielded better results, there were still inaccuracies toward the end of the mapping process. Ultimately, I switched to mapping a smaller area, which required fewer movements and allowed me to complete the mapping without errors accumulating

## SLO 3.3 Demonstrate displaying sensors

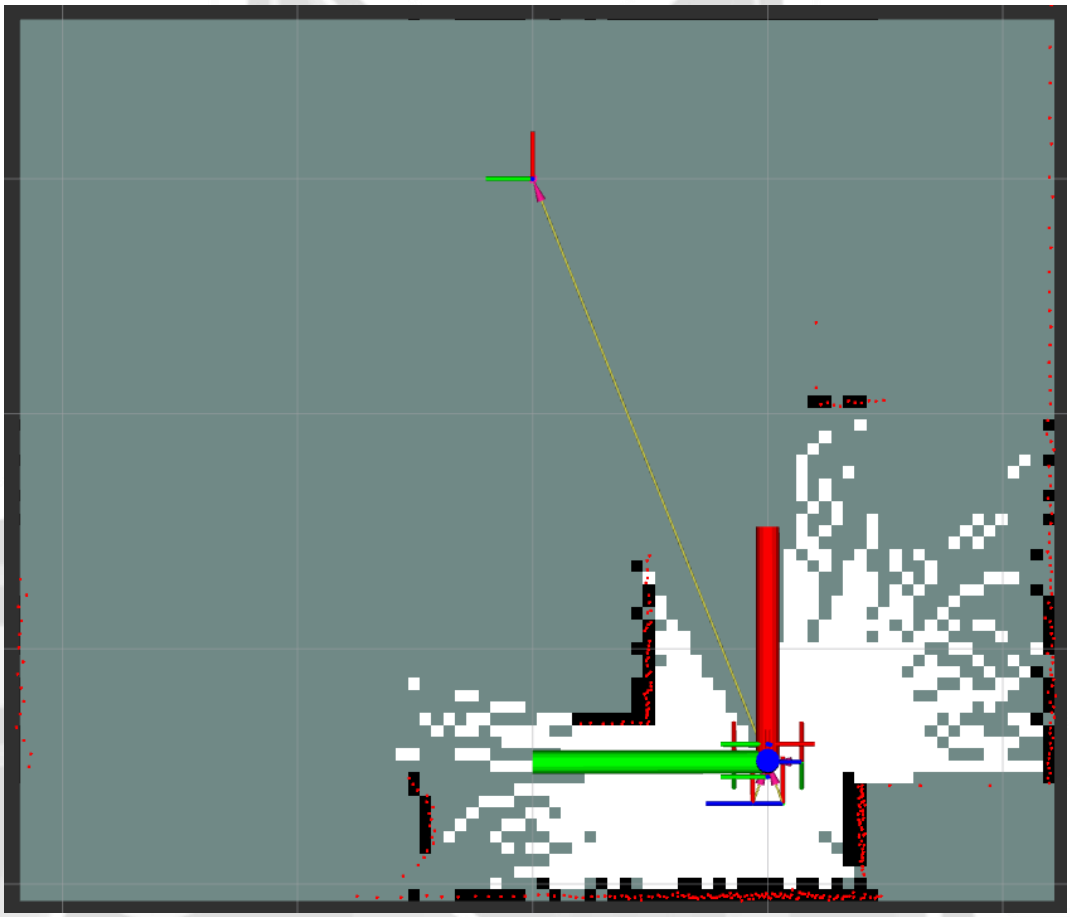


*Image 2.3 Displaying laserscan data and IMU data*

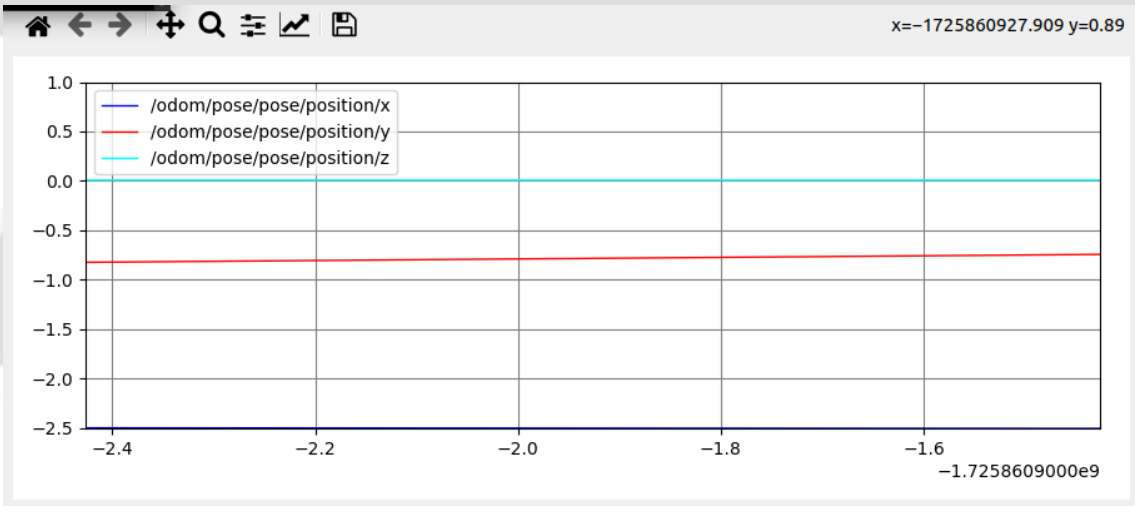



*Image 2.4 Displaying odometry data*

## SLO 3.4 Provide a short description of the localization task

Localization is the process of continuously estimating a robot's position and orientation within its environment. As the robot moves, it adjusts for drift by recognizing and matching features in its surroundings, helping to maintain an accurate understanding of its location. This is crucial for real-time obstacle avoidance and effective interaction with the environment. Common sensors used for localization include:

**- IMU (Inertial Measurement Unit)**

**- Accelerometer**

**- LiDAR**

Like mapping, localization faces challenges such as sensor noise, dynamic environments, and the need for significant processing power. However, localization also has unique concerns, such as dealing with feature-poor environments and the "kidnapped robot" problem. In environments with few distinctive features, robots may struggle to accurately match sensor data with their map, leading to difficulties in determining their location. Additionally, if a robot is powered on in an unknown location within its mapped environment, it may have trouble identifying its initial position.

## SLO 3.5 Demonstrating the localization task in the Gazebo environment

Setting up the localization algorithm was straightforward using the ROS 2 `nav2_amcl` package. With a new environment created in SLO 3.2, the localization process was smooth, requiring only minimal movement through the environment to achieve accurate results. Running the automatic path goal navigation tool was quick, only requiring a few button clicks. An example of the path navigation can be seen in image 2.5. The entire package worked flawlessly on the first attempt, without any issues.

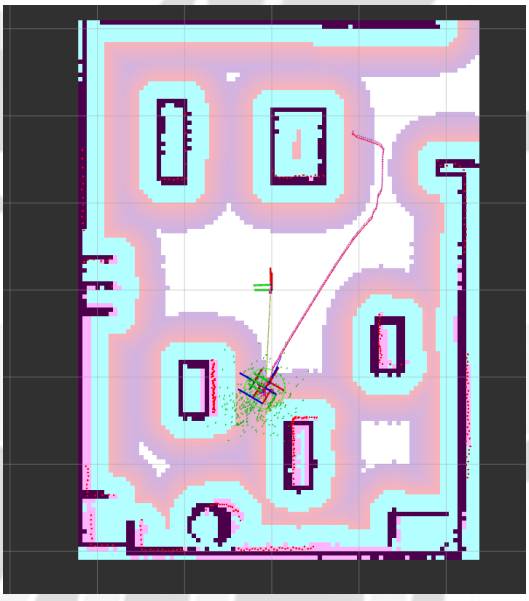


*Image 2.5 Localization & Navigation in Gazebo*

## SLO 3.6 Scan matching localizer

Completed in the lab.

## SLO 3.7 Compare the localization accuracies against the ground truth

The plot in Image 2.5 illustrates a comparison between the x and y positions of the ‘odom’ topic (representing ground truth) and the ‘amcl_pose’ topic (representing localization prediction). In this example, the robot was moving at a speed of 0.2 meters per second in a straight line within the project environment. The plot reveals that the x and y positions estimated by `amcl_pose` tend to lag behind the ground truth `odom` data. This discrepancy is likely due to sensor errors and drift. Notably, the error in the x position is significant, with a difference of approximately 0.2 meters, while the y position error is more minor, at around 0.1 meters. The orientation estimate, however, aligns perfectly with the ground truth (though it's difficult to see due to similar line colors), as the robot was not rotating. This allowed the robot to fully localize its orientation as it was not constantly changing.

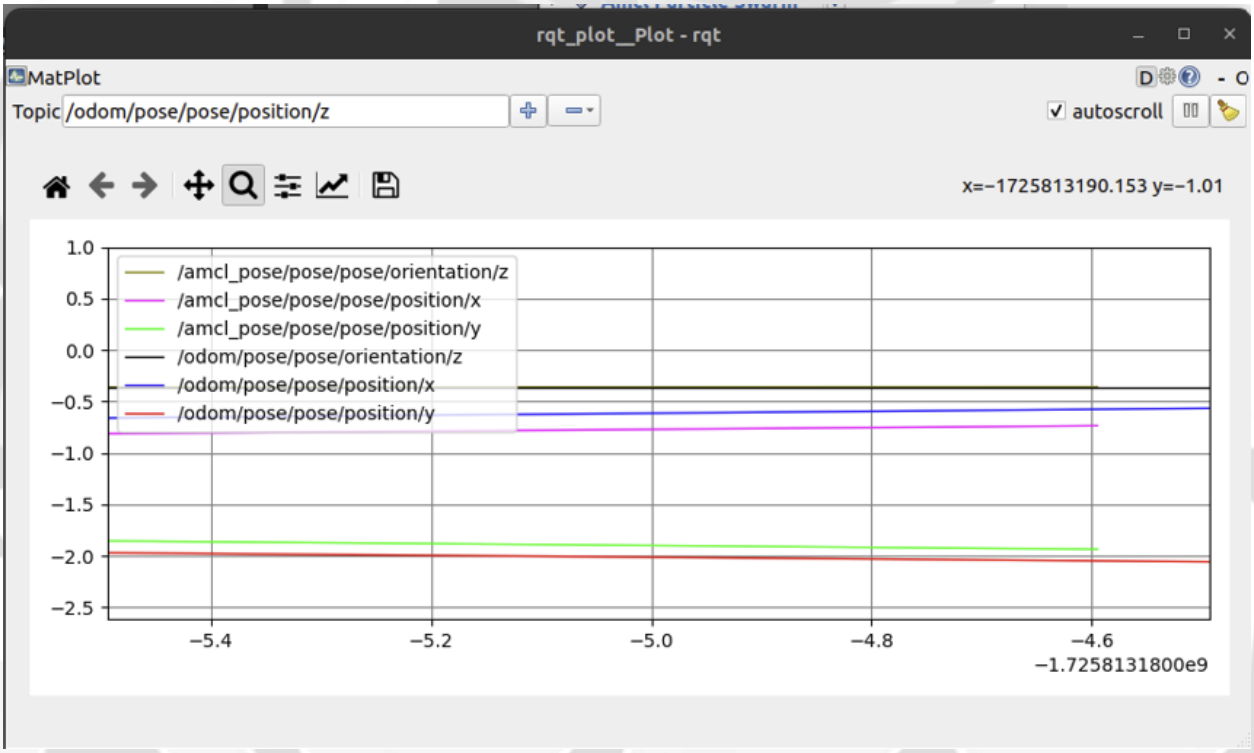



*Image 2.6 Odometry vs amcl_pose*

# SLO 4

## SLO 4.1 Time management

Completed in the lab.

## SLO 4.2 Effective Collaboration

All the graphs below (Image 2.7 to 2.12) were generated by moving the TurtleBot3 forward at a speed of 0.2 meters per second within the project environment. Across the graphs, a consistent error trend of 0.1 to 0.2 meters can be observed between the `odom` (ground truth) and `amcl_pose` (localization estimate) x and y positions. The only outlier is Win's (Akkarawat Kaveewatcharanont) and Yves plot, which show a significant discrepancy between the robot's localized yaw position and the ground truth. This anomaly occurred because Win's and Yves robots were the only one that rotated while moving, suggesting that accurately determining rotation is significantly more challenging than linear movement.

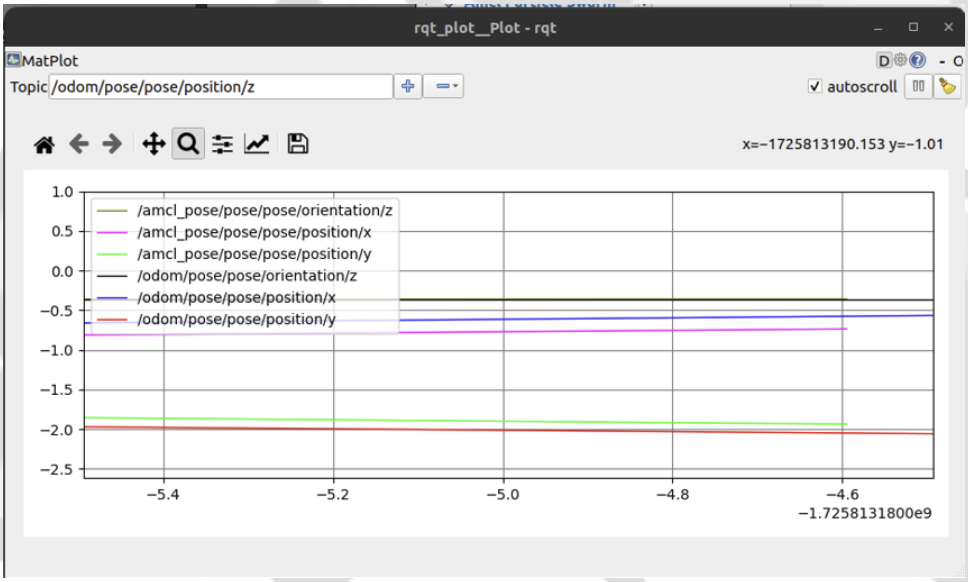



*Image 2.7:*

*Liam Faulkner-Hogg*

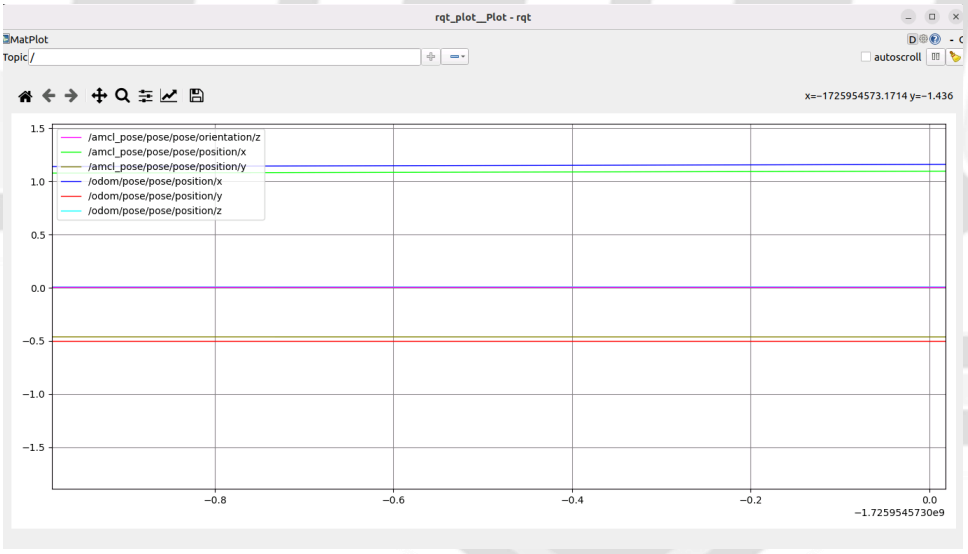



*Image 2.8:*

*Pravien Kugan*

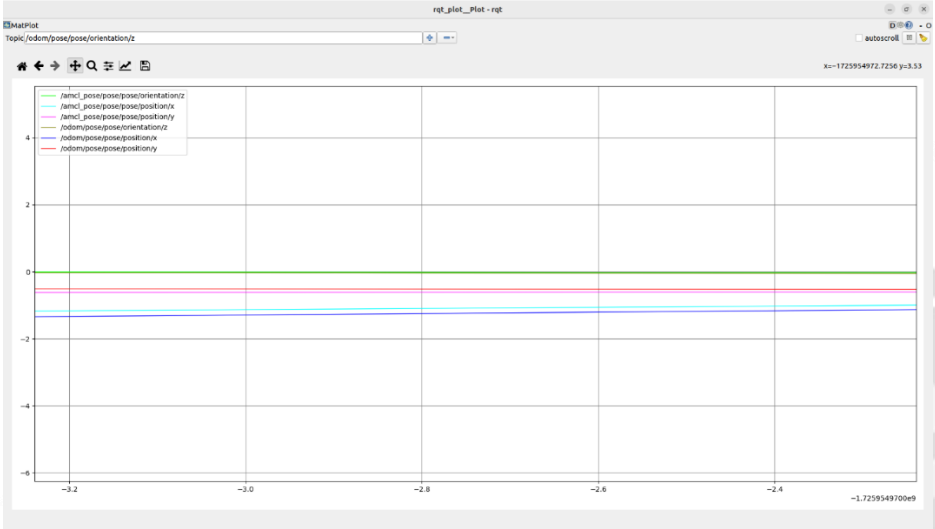



*Image 2.9:*

*Justin Pavlovski*

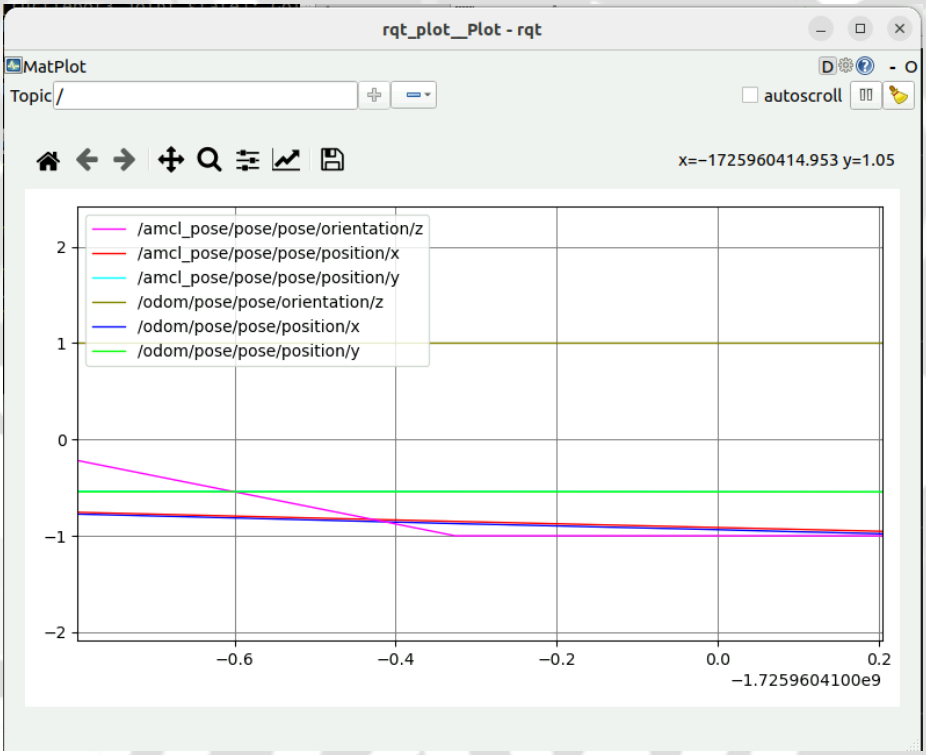



*Image 2.10:*

*Akkarawat Kaveewatcharanont*

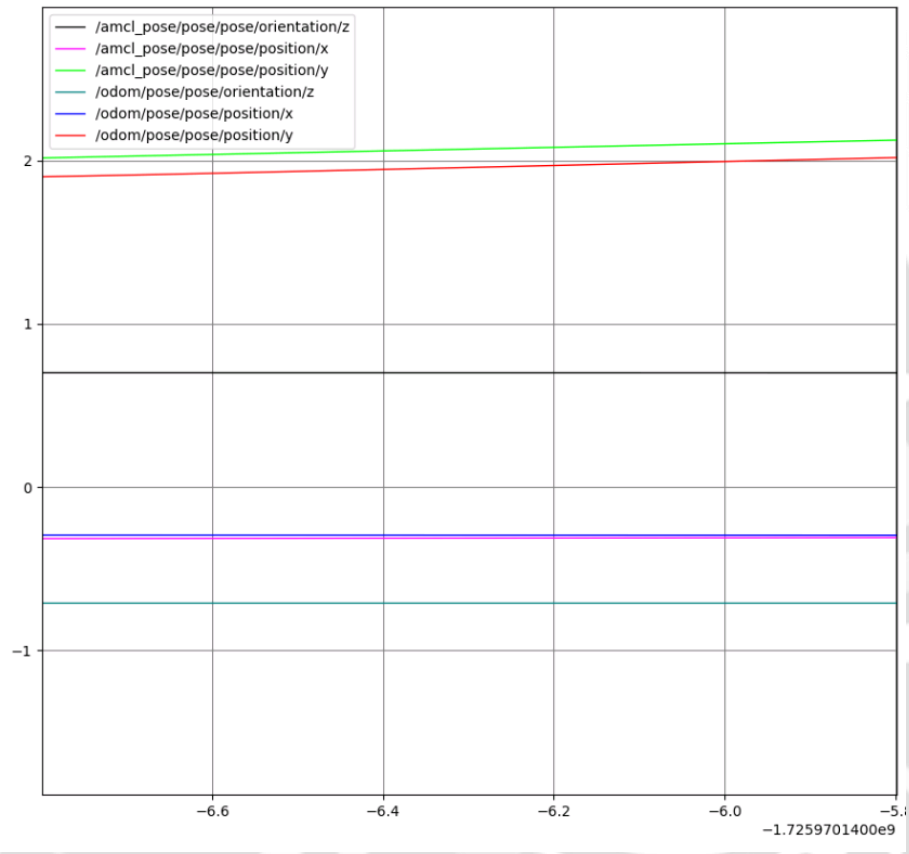



*Image 2.11:*

*Yves Gayagay*

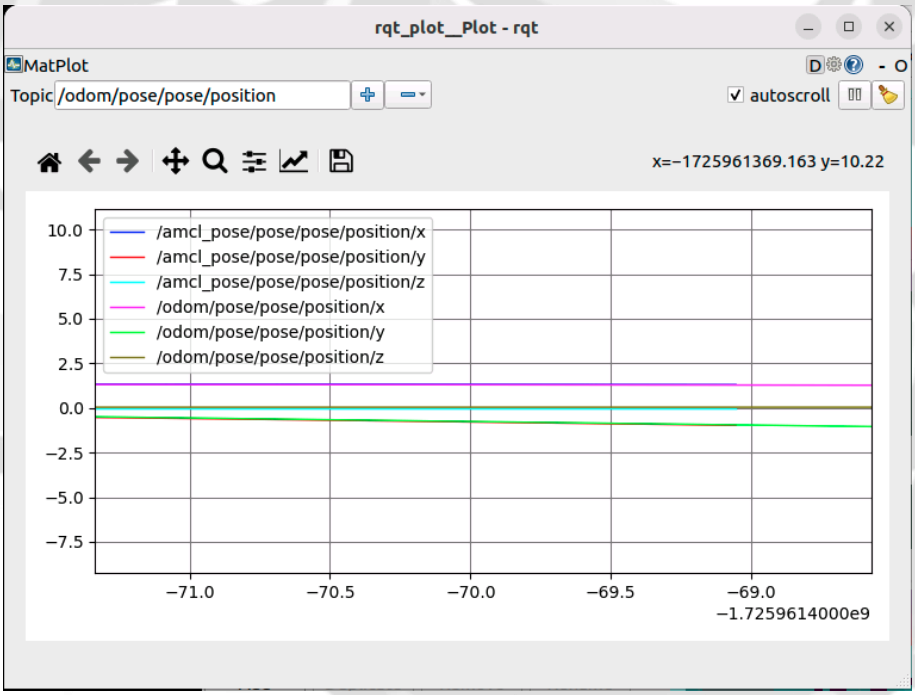



*Image 2.12:*

*Connor Fitzgibbon*

# SPRINT 3

## SLO 1

### SLO 1.1 Progress update with stakeholder engagement

**Date:** August 11, 2024

**Attendees:**

- Akkarawat Kaveewatcharanont
- Connor Fitzgibbon
- Justin Pavlovski
- Liam Faulkner-Hogg
- Pravien Kugan
- Yves Gayagay
- Sangmim Song ( Client )

**Agenda:**

1. Reviewing System mapping (flowchart)
2. Discussing the issues with spot quadpedal robot vs wheel
3. Localization and mapping discussion

**Discussion Points:**

- **Reviewing System Mapping**

The team explains the client the flow chart and how it works to the client. *The client wants to know if the environment would be complex in which the team assures the complexity of the environment.* Like AWS gazebo environment.

- **Discussing the issues with spot quadrupedal robot vs wheel**

The client advises to use wheel based robot. To assure that the camera is stabilized in every motion (in the mean time)

- **Localization and mapping discussion**

The client asks on what happens if there are a sudden change of the robot's environment compared to the floor map it was given. The client points to a localization method called Cartographer. He advises whatever is easy and accurate

**Action Items:**

- Work on team task for Sprint 2
- Look into suggested points like cartographer and AWS gazebo environment
- Look into using turtle Bot in environment first and interfacing with its sensors

## SLO 2

### SLO 2.1 Present the idea of SLAM and path planning in the context of your project

SLAM is essential to our search and rescue mission as it will enable SPOT to autonomously navigate and report the dynamic environment. SLAM will be the main tool in generating an accurate map for SPOT to traverse. This will allow for live update of path planning to ensure all areas are investigated. By generating real time maps, SPOT can also accurately and effectively report the locations of potential hazards and endangered civilians back to the first responders. This will allow first responders to more safely traverse the dangerous environment and target their efforts to effectively save lives.

Path planning is crucial aspect when utilising Spot the Robot Dog for a search and rescue mission. The features of path planning include:

- Enables efficient navigation through complex environments
- Helps avoid obstacles, ensuring safe movement
- Optimises path selection to reduce time and energy consumption
- Ensures thorough coverage of the search area to avoid missing critical zones
- Adapts to dynamic environments (e.g., moving debris, changing terrain)
- Improves the robot's ability to locate survivors quickly
- Enhances decision-making during real-time missions
- Supports safe return with crucial data for rescue teams

# SLO 3

## SLO 3.1 Provide a description of SLAM

SLAM (Simultaneous Localisation and Mapping) is a process in which robots create maps of unknown environments while simultaneously localising themselves within those maps. It is an important process which allows robots to navigate and interact with unknown environments. Some common robots using SLAM are self-driving cars and Roombas.

Common sensors used alongside SLAM algorithms include:

- **LiDAR:** LiDAR is a TOF sensor which emits and detects a laser. The time between emission and detection is used to determine the distance between objects.
- **Depth Camera:** Depth cameras use two cameras to determine the distance from objects in the robot's environment by calculating the distortion between the two images seen.
- **Inertial Measurement Unit (IMU):** IMU are able to measure the motion of an object. Commonly used in SLAM algorithms to record the linear and angular motion.

The core challenge of SLAM lies in simultaneously mapping an environment while localizing within it. How can a robot accurately map an area without knowing its precise location, or localize itself without an existing map? Sensor inaccuracies can introduce errors, affecting both object positions and robot location. Dynamic environments add complexity, creating situations where a robot may incorrectly mark objects that have moved. These factors illustrate just a few difficulties faced in SLAM. The following two SLAM packages are popular for use in ROS with their accompanying sensors:

**Cartographer:** Can perform SLAM in real time, generating 2D and 3D maps, with the ability to match scans and poses to correct graphing and localization errors. Widely used in outdoor and complex environments.

- LiDAR
- IMU
- Radar

**SLAM Toolbox:** Optimised for 2D SLAM, includes algorithms for loop closure and multi session mapping. Most notably used for lifelong mapping where refining and expanding maps over long periods of time is required.

- LiDAR
- IMU
- Radar
- Camera

## SLO 3.2 Demonstrating the SLAM task in Gazebo environment

There was no difficulty setting up SLAM using Cartographer. The required packages were already installed on my system with ROS and the online documentation was sufficient in describing how to run it. My only frustration with it was the speed in which I had to map the environment. The online tutorial recommends moving slowly and never turning while moving linearly, doing this leads to a large amount of time spent mapping the environment.

Image 3.1 shows the ground truth map made using gazebo map creator (red) overlaid onto the cartographer generated map (black). It is clear some detail such as the gaps between boxes is missed when using cartographer, but ultimately the SLAM map is highly accurate for all accessible areas.

*Image 3.1: Overlay of ground truth map and SLAM map*

## SLO 3.3 Analyse computational aspects in simulations

Creating a map of my team's project environment took a total of 47 minutes using Cartographer as seen in image 3.2, this was due to the recommendation "drive as slow as possible" found on the ROS 2 Cartographer tutorial.

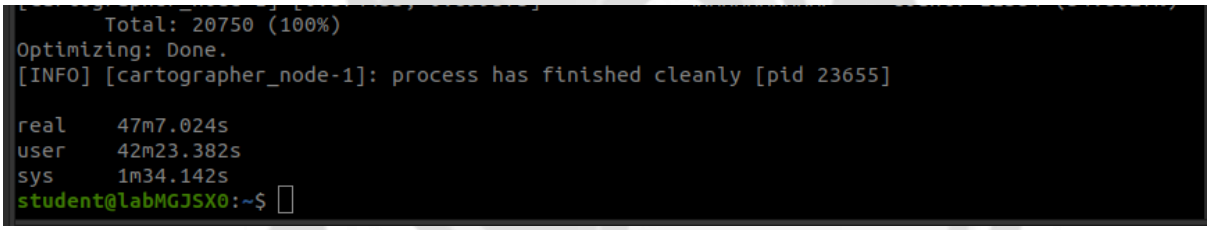



*Image 3.2: Cartographer simulation time*

## SLO 3.4 Provide a short description of the Robot Path Planning task

Path planning involves generating a route for a robot to follow which enables it to reach the point/s of interest. The generated path should be optimized to the user's desires, usually to minimize total travel distance and time. Path planning is especially important for tasks that have multiple locations or possible routes, like postal services and google maps.

Path planning can be performed using the following 3 algorithms:

**Dijkstra's Algorithm:** Guarantees finding the shortest path by exploring all nodes adjacent to the starting node and keeping track of the shortest distances until the goal is found.

**Greedy Best First Search:** Similar to Dijkstra's, travels through adjacent nodes but uses a heuristic to select nodes towards the goal, often finding a route faster but not always the optimal path.

**A* Algorithm:** Combines Dijkstra's accuracy with Greedy Best First's speed, using both path cost and heuristic to find the shortest path efficiently. It is the most popular choice for pathfinding.

More information can be found for all 3 methods at *Introduction to A** by Stanford and Navone, "Dijkstra's shortest path algorithm" (can be found in references).

Path planning faces numerous challenges, including navigating complex environments, dealing with dynamic obstacles, and handling real-time computations. In complex environments, there may be numerous obstacles, narrow passages, or varying terrains, making it challenging to find feasible paths. Dynamic obstacles, such as moving vehicles or pedestrians, require continuous updates to the planned route. Managing all of this information also requires significant processing power, which is not always available, especially on cost effective robots. These factors together make it difficult for path planning algorithms to consistently determine optimal paths.

## SLO 3.5 Demonstrating the path planning task in the Gazebo environment

The following path planning parameters for my nav2 were changed:

Planner Server:

- tolerance: 0.5 → 0.1
- use_astar: False → True

Global Costmap & Local Costmap:

- resolution: 0.05 → 0.01
- inflation_radius (Local): 1 → 0.35
- inflation_radius (Global): 0.55 → 0.35
-

all use_time_sim variables were also changed from ‘False’ to ‘True’

The main challenge with path planning for my TurtleBot was its difficulty in navigating tight spaces between boxes in our warehouse. To address this, I initially reduced the inflation_radius for both local and global planners to 0.2, which allowed the robot to plan paths into tight areas it previously couldn't reach. However, this led to collisions when making tight turns around obstacles, which was unacceptable. To prevent these collisions, I increased the inflation_radius to 0.35. While this adjustment resolved the collision issue, it made some of the tight spaces inaccessible again, though not all of them. Another issue arises with the robot's localization when navigating tight spaces, the laser readings begin to drift, causing a misalignment with the obstacles displayed on the map. Although this is not ideal, it hasn't caused any path planning issues so far, and the localization corrects itself once the robot leaves these areas.

### SLO 3.6 Exclude a randomly placed known object from mapping

Lab Work

## SLO 4

### SLO 4.1 Time management

Evidence of all sprint 3 report tasks being complete can be found by reading through all sprint 3 sections of this report. For evidence of the lab tasks being complete, please see the below images.

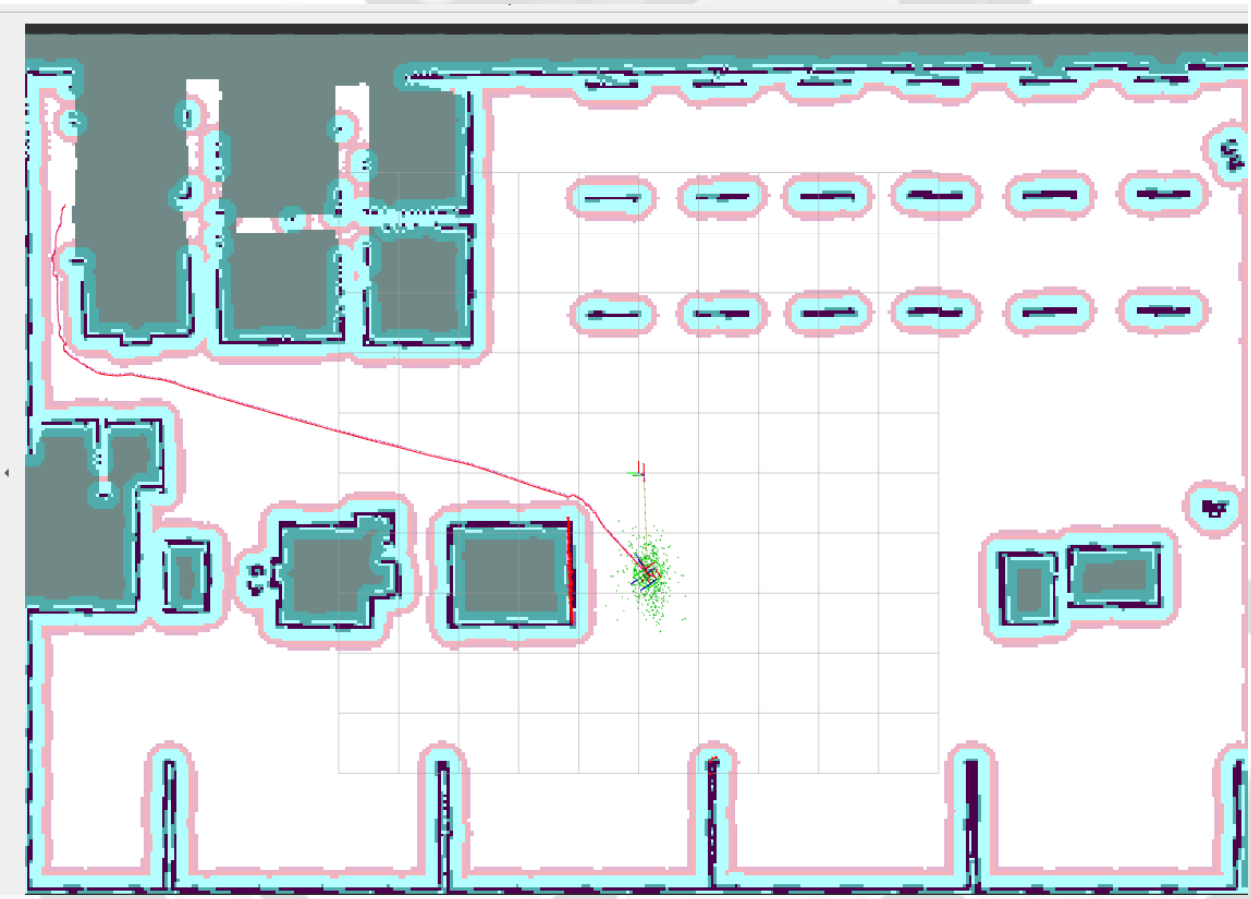


*Image 3.3: Path Planning being done for 3.5.*

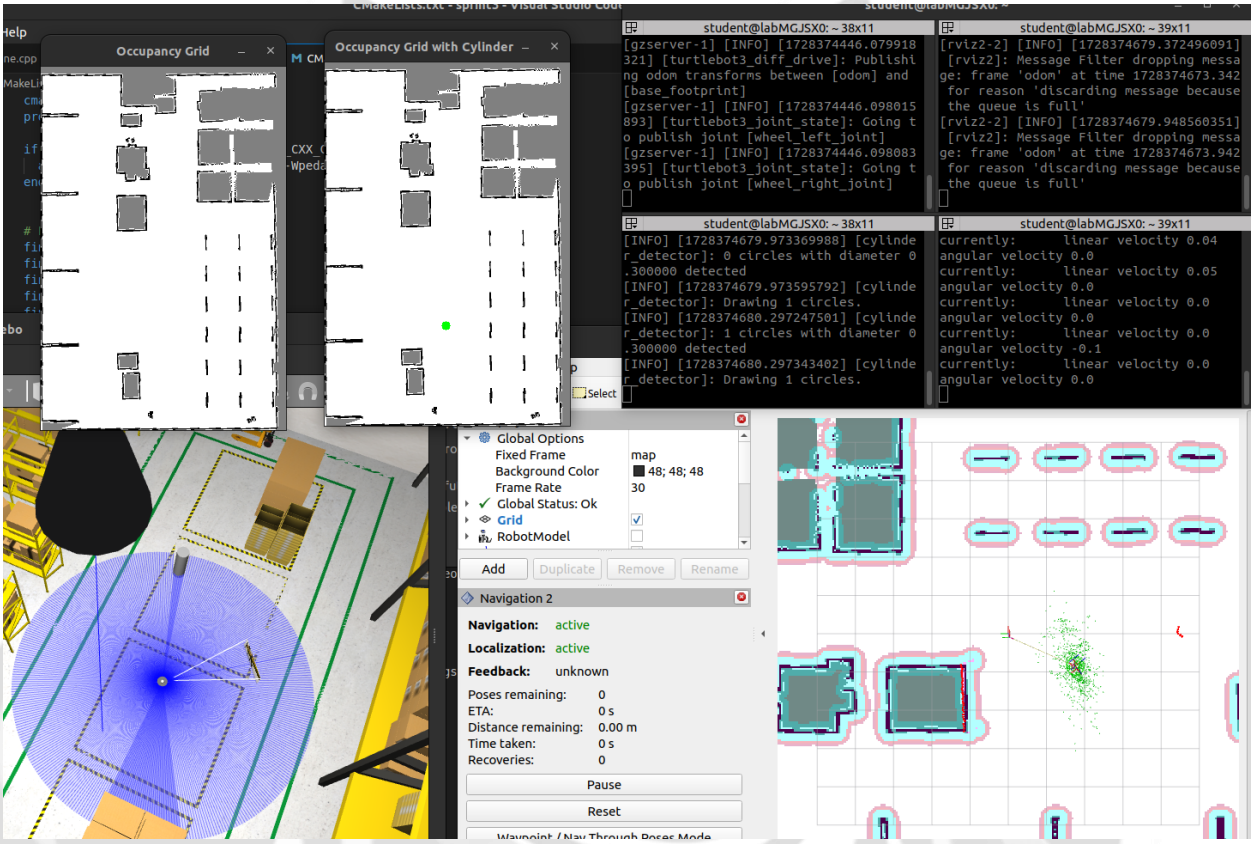



*Image 3.4: TurtleBot detecting cylinder using laser scan data for 3.6.*

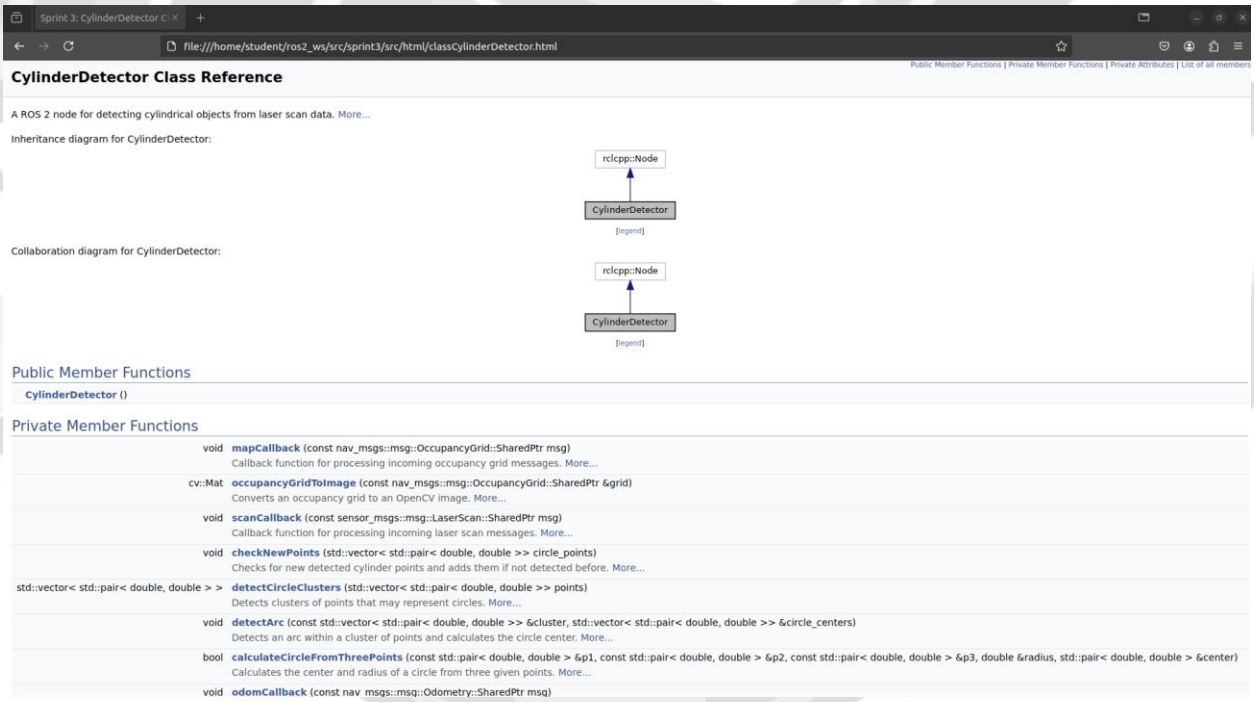



*Image 3.5: Doxygen documentation for 4.3.*

### SLO 4.2 Create a github repository

https://github.com/LiamFaulkner-Hogg/Robotics-Studio-1.git

### SLO 4.3 Produce professionally formatted portfolio and code documentation with Doxygen

Lab Task

# SPRINT 4

## SLO 1

### SLO 1.1 Stakeholder Engagement on project completion

**29/10/24 Handover Meeting [SPT4]**

**Attendees:**

- Akkarawat Kaveewatcharanont
- Connor Fitzgibbon
- Justin Pavlovski
- Liam Faulkner-Hogg
- Pravien Kugan
- Yves Gayagay
- Sangmim Song (Client)
-

**Agenda:**

1. Going through survivor detection
2. Going through Spot integration
3. Going through sustainability and finance of the project
4. Going through the GUI concept
5. Final Discussion about the project scope

**Discussion Points:**

**Going through survivor detection**

The client was pleased with the system's ability to detect accurate level of detail of survivors and their location.

However the client was concerned about the level of accuracy not extending to the different orientation of survivors and the amount of false-positives i.e. survivors laying down or an orientation where it cannot be detected by laser scans.

**Going through Spot integration**

The client was very pleased with the team's ability to extend its capability in integrating the SPOT robot into the deliverable as it was known that many teams did not integrate this.

**Going through sustainability and finance of the project**

There was a thorough discussion over the finance behind the project and its application through a deliverable state. The client was unsure at first with the finance and sustainability, however the team were able to address his concerns which returns a positive outlook on the deliverable.

**Going through the GUI concept**

The team presented a prof of concept of a GUI which shows how future users can operate the system without the knowledge of linux , ROS, SPOT etc. The client was quite pleased with the concept as no questions were asked.

**Final Discussion about the project scope**

The team has an engaging and thorough discussion in reiterating the project scope with the client. As these discussion pointers transforms into future plans for future iterations of the project.

**Action Items:**

- Proceed to demonstrate in peer marking and Sprint 4 demonstration
- Sign off all of the remaining documentations
- Close the git and hand over to the client.

**Notes:**

This is the final meeting with the stakeholder. Hence all formal communications with the client will be terminated and documentation will end at this MoM. However , if the client is concerned with the project all forms of communication will now be labelled 'informal'

# SLO 2

## SLO 2.1 Develop an evaluation criteria.

| Technical Aspect | Description | Evaluation Criteria |
|---|---|---|
| Localisation | Uses ROS navigation2 package to localise the robot within a known warehouse environment using a pre-saved map. The AMCL pose is used to accurately pinpoint the robot's position within the environment. | **Accuracy:**<br><br>Calculate the RMSE between the robot's estimated positions and the known reference positions (ground truth) within the warehouse.<br><br>**Target:** Aim for an RMSE of less than X cm to indicate acceptable localisation performance.<br><br>**Stability:**<br>The AMCL pose should consistently maintain accuracy even in areas with limited features, without significant fluctuations in the estimated position. |
| Path Planning | Utilises different waypoints in ROS Navigation2 for the robot to move around the warehouse and covers all areas to detect injured people. The robot's path should update to dynamic environments. | **Object Avoidance:**<br>The robot should avoid obstacles moving to each of the waypoints tasked especially with dynamic environments.<br><br>**Coverage:**<br>The robot ensures complete coverage of the environment for detecting injured individuals. |

| **Client Assessment** | **Yves** | **Win** | **Connor** | **Justin** | **Pravien** | **Liam** |
|---|---|---|---|---|---|---|
| Completeness Criteria | The reliability and effectiveness of the technical solutions implemented.<br><br>The project's quality and its performance outcomes. | The reliability and effectiveness of the technical solutions implemented.<br><br>The project's quality and its performance outcomes. | The reliability and effectiveness of the technical solutions implemented.<br><br>The project's quality and its performance outcomes. | The reliability and effectiveness of the technical solutions implemented.<br><br>The project's quality and its performance outcomes. | The financial feasibility and sustainability of the project. | The reliability and effectiveness of the technical solutions implemented.<br><br>The project's quality and its performance outcomes. |
| Key elements included in the project to meet the completeness criteria. | Addition of a GUI for easier control and management of the project. Creation of two other environments (domestic and outdoor) with corresponding maps to demonstrate adaptability, ensuring | Incorporating simulated people into the project for the detector node to recognise injured individuals, validating detection | Integration of the SPOT robot instead of the TurtleBot to enhance the project's realism and demonstrate robust operation in a | Developing a detector node specifically designed to detect injured individuals in the environment, ensuring consistent and high-quality | Ensuring that the project remains financially feasible while demonstrating its capability to be used in real-life applications, thereby | Implementing path planning algorithms to ensure the robot covers the entire environment, achieving effective search and rescue capabilities in |

| | | | | | | |
|---|---|---|---|---|---|---|
| | solutions work in different scenarios. | reliability and performance. | realistic scenario. | detection outcomes. Combining the entire project with one launch file. | showing its sustainability. | various conditions. Providing the shortest route for paramedics to reach the injured personnel. |
| Completion | Incorporate the pgm and png files of the maps for the warehouse, house and outdoor in a separate directory under Main_project/maps . So that it can be imported over to the SLAM and NAV2 mapping. Create a graphic user interface (GUI) that best proves as a proof of concept. The GUI allows you to input choose the map and robot model to | Import human models and editing it making sure that the nodes that detects it works with the models. Made sure it works on each environment outdoors and indoors. Edited readme.me file for an easier implementation of the project. | Spot was integrated into one of the environments (outdoor). Work was done to ensure SPOT had identical functionality to the turtlebot (ability to navigate and detect injured persons)<br><br>Circle detection algorithm | Detector node works well for detecting injured civilians throughout the environment. The node also publishes their location for access from supporting nodes. All nodes were combined into one launch file so that we can quickly and | I created a table to compare cost analysis, sustainability, operational efficiency, and financial risks and mitigation strategies. To illustrate the cost analysis, I developed a graph comparing the real-life scenario of an operation with | Developed a navigation node which excutes full navigation through an environment which ensures all locations are searched and all injured personnel found. Once an injured person is found, the shortest path to the injured personnel will be visualised on |

| | | | | | | |
|---|---|---|---|---|---|---|
| | launch the simulator with. | | needs fine tuning for new laser scan. Working on implementing into single launch file | efficiently run our project. | one that incorporates our project using SPOT over a five-year period. | the Rviz map for support workers to follow. This node has been optimised to work in all environments |
| Unable to Meet | Functioning GUI as there is a large knowledge gap of the qt framework. and functioning usages of the maps as the project incorporates to only a limited selection of maps. | Multiple models that are not human | Further time and testing is required to implement spot in other environment | The detector node could be supported with camera images to verify that all the detected points are actually people and not just environmental features. | Provide financial information for the other different environment but it is roughly the same. | Audio recognition was not implemented. Path planning around environmental hazards such as fires was not implemented |

# SLO 3

## SLO 3.1 Project Evaluation and Testing: Overall

Robo Dawgs solution to developing a search and rescue dog utilizing SPOT, has been successful in achieving many of the required functions initially laid out in our design objectives.

**Implement SLAM:** Effectively uses ROS Nav2 package to implement SLAM. Because of this, SPOT is able to accurately identify and localise itself within the environment (this can be seen in image 4.1) to ensure that all injured personnel have their locations correctly relayed to support personnel. This also allows for obstacle obstructions to be recognised and mapped in real time to aid in identifying blocked pathways and dead ends for support personnel.

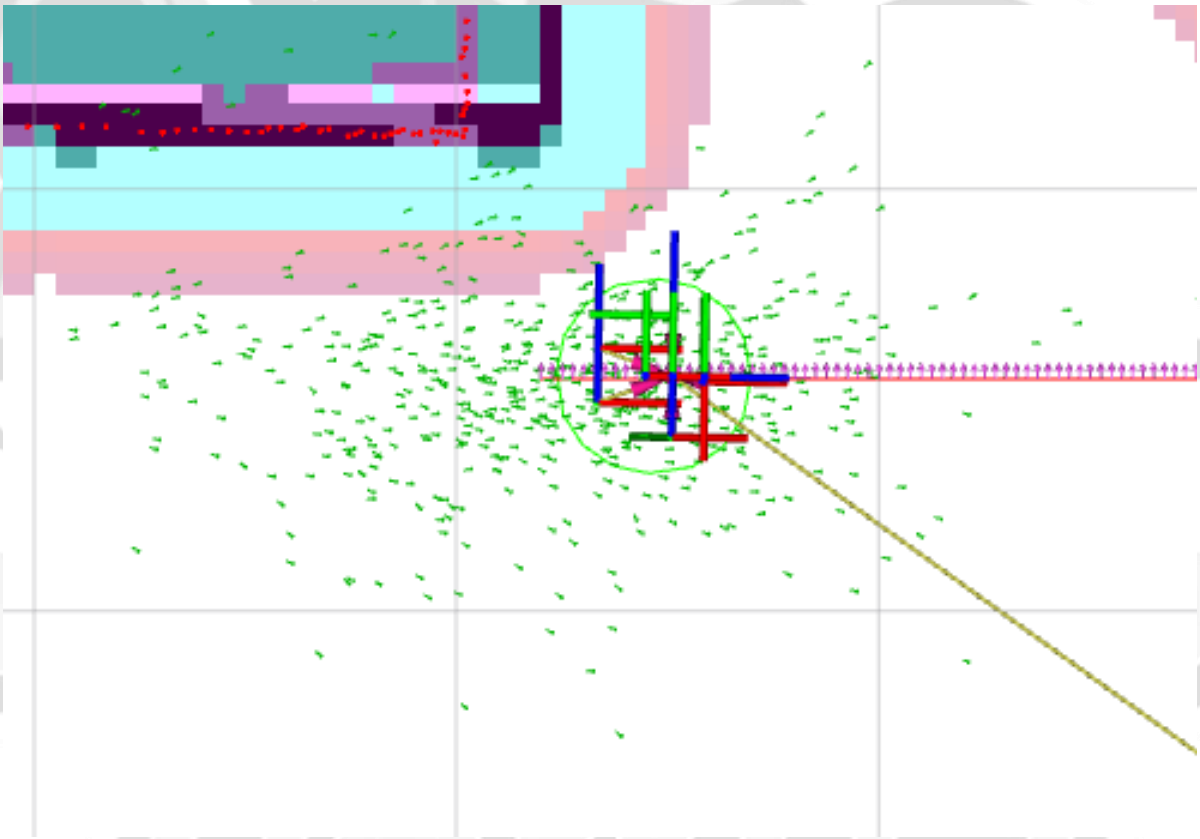


*Image 4.1: Shows SLAM in the form of green arrows which represent the possible pose of the robot while also scanning its environment to update the map in case of injured people and dynamic obstacles*

**Implement Path Planning:** Our path planning solution was only able to meet a few of our expectations. The following is all of the expectations for path planning:

1. Independently generate the most effective path for exploration in a pre-determined environment to ensure all areas are searched for injured personnel.

2. Path plan around environmental hazards to ensure SPOT does not get destroyed.

3. Travel to desired locations as selected by the operator in the shortest path possible.

4. Visualize the shortest route between injured individuals and support personnel to ensure the quickest and most efficient access to their target.

Only solutions 3 and 4 were fully implemented, while solution 2 was not implemented, and solution 1 was only partially completed. Generating the most effective exploration path requires the operator to select locations on the map, allowing SPOT to then generate the shortest path between those points in sequence, a significant difference from autonomous, independent exploration. These solutions can be seen in image 4.2.

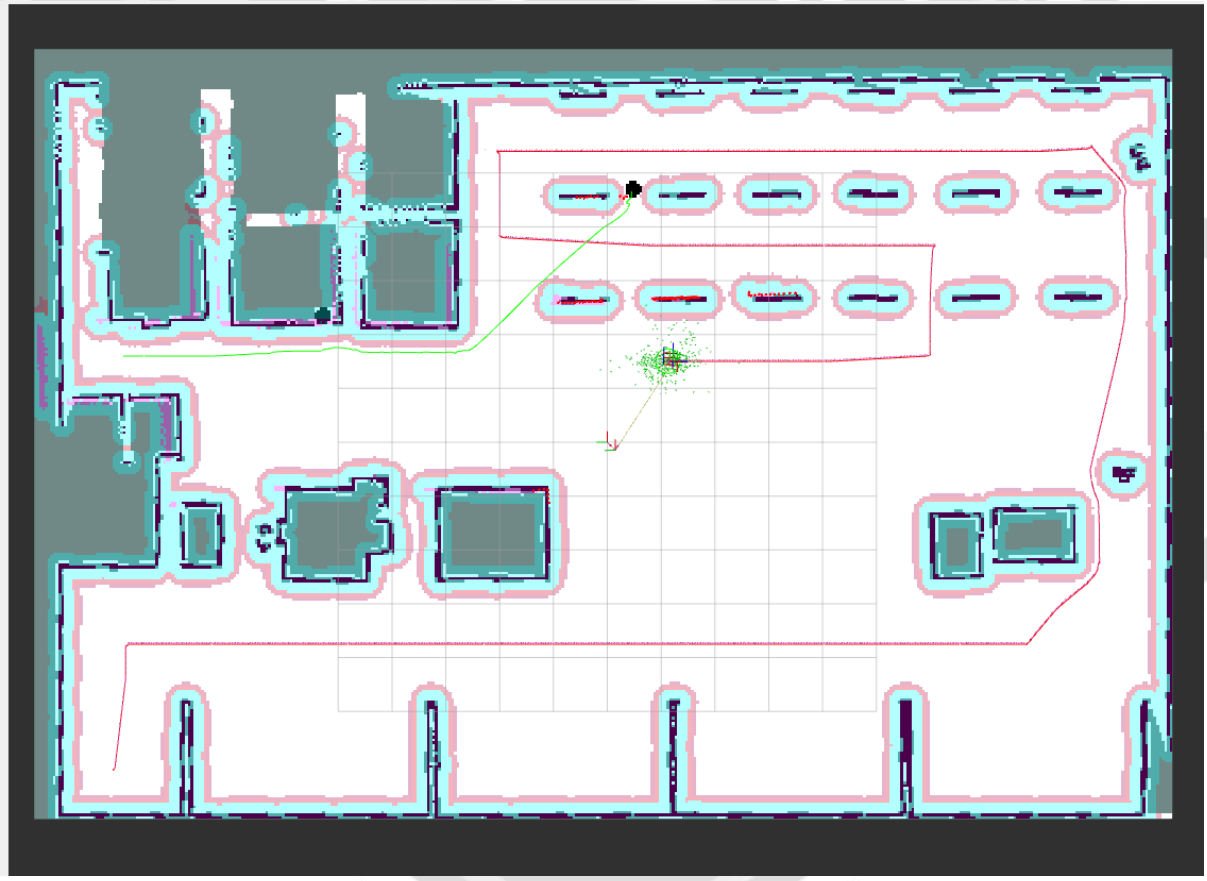


*Image 4.2: Shows the robots search path (red) and path to the injured person it has found (green)*

**Implement Sensor Data:** Our solution implements Camera, LiDAR, IMU and Accelerometer data to effectively detect injured people and report their location back to operators with a high degree of accuracy. This can be seen through all of the above solutions as they all utilize one or more of these sensors to properly operate.

Because of the above implemented functionality, our search and rescue solution is able to correctly identify, locate and report the positions of all injured personnel within a disaster environment (seen in image 4.3). There are still improvements which will need to be made for future versions such as hazard detection, noise recognition and a more efficient/ independent environment exploration solution.

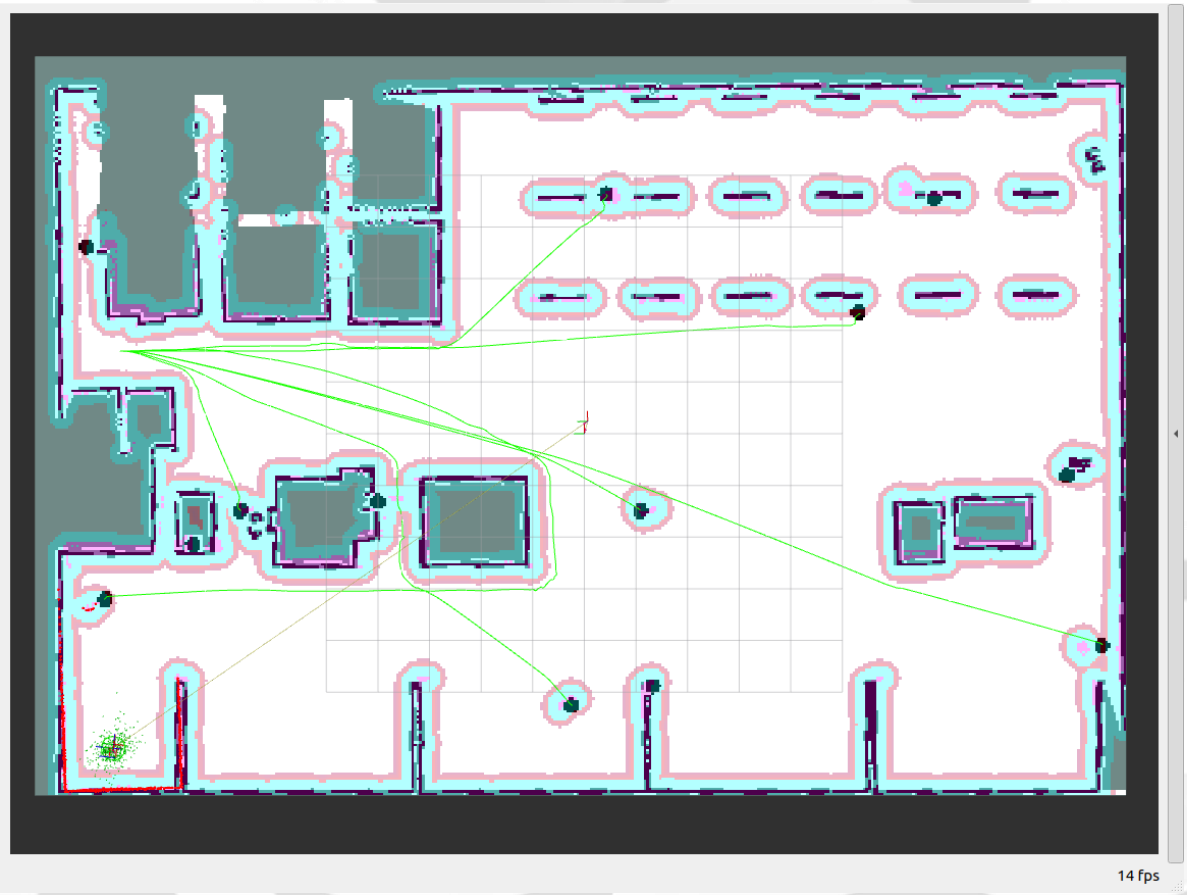



Image 4.3: Robot has reached the end of its search and successfully located all injured people with paths to reach all.

## SLO 3.2 Project Evaluation & Testing: Aspect 1

**Relevance:** Effective path planning is essential to our search and rescue solution for searching disaster environments. Path Planning is what allows SPOT and Turtlebot to autonomously navigate through the environment efficiently and allows it to respond appropriately in dynamic environments. Without this functionality, SPOT and Turtlebot would continuously get stuck due to unknown obstacles, dead ends and complicated terrains, requiring the operator or surrounding personnel to manually reset its path, taking their focus away from saving lives. Furthermore, this functionality allows for visualization of the quickest path to injured persons, aiding support personnel in reaching their targets as quickly as possible. This is especially necessary for large/ dangerous areas with complicated terrain, such as forests and fires.

**Challenges:** We faced three main challenges in developing our path-planning solution:

- Understanding Actions: None of us, including myself, had prior knowledge of "actions" in ROS. We were only familiar with "topics" and creating publishers and subscribers. As a result, we struggled to figure out how to interact with the Nav2 behavior trees, spending significant time inspecting topic lists before learning about actions and creating action clients.

- Identifying Available Actions: The NAV2 package offers multiple actions, but the documentation on these actions was hard to find, incomplete, or vague. This led to considerable time spent trying to understand what each action did and how to incorporate them into our project.

- Extracting Data from Actions: Accessing action data differs significantly from topics, and the limited documentation made it challenging to use "path" data computed through the “compute_path_through_poses” and “navigate_through_poses” actions effectively.

These challenges complicated the visualization of paths to injured personnel since all paths were published on the same /path topic, making it difficult to distinguish between exploration and rescue paths. Additionally, setting up the exploration path was challenging due to unknown path-smoothing parameters embedded deep within the Nav2 package framework, which we couldn’t locate or modify.

**Available Path Planning ROS2 Packages:**

| **Name** | Navigation 2 | Clearpath |
|---|---|---|
| **Pros** | • Modular<br>• Open Source<br>• Large library of functions | • Easily integrate with Clearpath hardware<br>• Commercial Support<br>• Tailored for specific platforms |
| **Cons** | • Complex setup compared to Clearpath<br>• Performance less optimized<br>• Less support | • Limited to Clearpath hardware<br>• Some software costs |

Most path planning packages available on ROS 2 are used by Nav2, this includes the TEB local planner, SMAC planner, and behaviour tree library. Because of this, there is little reason to use any package other than Nav2, unless your path planning solution is being strongly optimised for use with specific planners, in which case the packages within Nav2 can be used independently. The Clearpath navigation package is specifically developed for Clearpath hardware and has open-source software as well as paid commercial packages. This package has less functionalities than Nav2 and would only be used for interfacing with Clearpath products.

## SLO 3.3 Project Evaluation & Testing: Aspect 2

**Relevance**: Effective localization is essential for our search and rescue dog solution to operate in disaster environments. Localization allows SPOT to accurately determine its position within a complex environment, enabling reliable navigation and decision-making. Without accurate localization, SPOT would lose track of its location, causing it to get disoriented, miss critical paths, and potentially require manual intervention to reset its position. This would take away focus from rescuers in critical moments. Precise localization is particularly crucial in large, dynamic, and dangerous areas, like forests or fire-prone zones, where misjudging location could lead to delayed rescues or unsafe maneuvers.

**Challenges**: Our localisation solution used SLAM toolbox (which uses Adaptive Monte Carlo Localization), which was easy to setup as it seamlessly integrates with the previously mentioned Nav2 package. The only problems I faced were issues with effective localising without an accurate initial pose estimate as our warehouse environment lacked enough environmental features. To overcome these issues, we moved our TurtleBot starting position to an area which included significantly more obstacles and walls than its original starting position. This improved its ability to localise without an accurate initial pose estimate.

**Available Localisation ROS2 Packages**:

| **Name** | SLAM Toolbox | Cartographer |
|---|---|---|
| **Pros** | • Online & Offline SLAM<br>• Multi-Session Mapping<br>• Seamlessly fits into Nav2 stack | • Support 3D environments<br>• Allows for fusion of multiple different sensors<br>• Supports multi-robot mapping |
| **Cons** | • Limited to 2D Environments<br>• Less effective than cartographer in dynamic environments<br>• Limited support for sensors other than LiDAR and odometry sensors | • Resource Intensive<br>• Difficult to finely tune<br>• Less mature than SLAM toolbox for use in ROS2 |

## SLO 3.4 Project evaluation: Clients assessment of completeness of the project.

Complete with Sangmim Song.

## SLO 3.5 Programming Aspect.

Lab Based Task

# SLO 4

## SLO 4.1 Time Management.

All sprint 4 tasks can be seen as complete by reading through the Sprint 4 section of this report.

Section 3.1 of Sprint 4 includes photos demonstrating the lab task for 3.1, 3.2 and 3.3, showing that they have been completed and can be shown in the lab. Completion of 3.4 can be seen from the MoM in section 1.1. We also have a Gannt Chart (image 4.4) showing the completion of all parts (It contains all sprint tasks from sprint 1 to 4 and is too big to visualize properly in word).

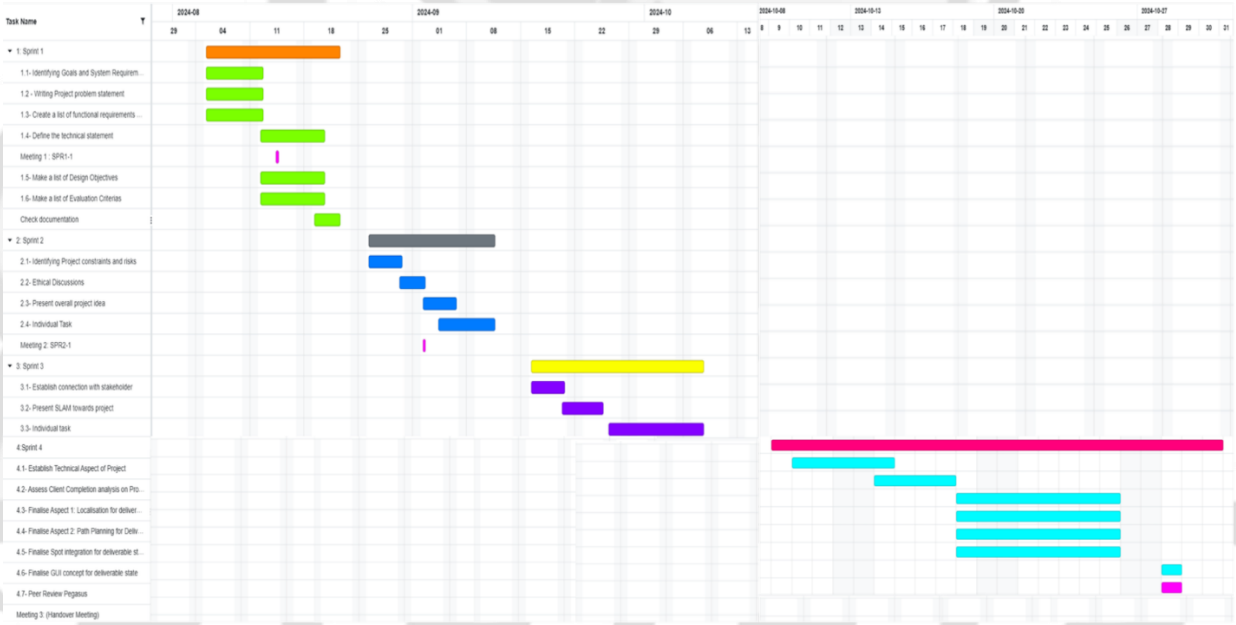



*Image 4.4: Gannt Chart showing completion of all tasks from sprint 1 to 4.*

# SLO 5

## SLO 5.1 Critical reflection on the subject learning

**Establish priorities, uncertainties and risks to identify and scope a problem**

We set clear team priorities, engaged with our stakeholder Sangmim Song to align objectives, and reduced uncertainties. We assessed risks in our problem statement, defined functional requirements like autonomous navigation and sensor integration in sprint1, and outlined design parameters, ensuring our solution's reliability and effective project scope to meet team and stakeholder needs.

**Apply design/systems thinking to the analysis of a mobile robot problem**

We tackled the challenge of disaster navigation and detection for SPOT by breaking it down into clear functional requirements and design goals in SLO 2.2 and 2.4 from Sprint 1. We focused on key capabilities like SLAM, sensor data processing, and path planning. By considering real-world constraints like rough terrains and obstacles, we established design parameters to ensure reliability and adaptability for multiple scenarios.

**Apply technical skills to develop, model and/or evaluate an estimation solution.**

I used technical skills such as programming in C++ and python to develop a solution which allows SPOT or a TurtleBot to effectively navigate through a disaster environment and locate injured people. My solution provides the ability to start the robot navigation and wait for it to search an entire disaster environment without needing a human presence, while doing this it will return the quickest paths to all injured people in the environment.

**Demonstrate effective collaboration and communication skills as an effective member or team leader of team/s.**

All people in my group, including myself maintained effective communication through teams, messenger and GitHub. I showed up to every class and ensured all my team members knew what parts of the project I was doing and when they were complete. When team members needed help or their were uncertainties about tasks, I was always ready to help, and so was everyone else. Evidence of this can be seen from previous sprints.

**Conduct critical self, peer and team reflection for performance evaluation**

In SLO 5.3 for sprint4, my team and I performed a peer reflection for group Pegasus. In this assessment we went over their functional criteria and their project abilities to provide meaningful feedback. This feedback was sent to them, and theirs was sent to us. I have also conducted critical self and team reflections throughout all of the sprints including this SLO 5.1 and 5.2.

**Challenges and how they were overcome**

Many challenges have been faced during this project, most of them have been talked about in detail during this and previous sprints. One major challenge which has not been mentioned was the integration of SPOT into our project. For the entire semester we used Turtlebots as our robotic platform, this worked well to teach us the basics of the SLAM and Path Planning packages, but did not help when we needed to use SPOT for our major project. Because of this, there were many challenges such as updating launch files, urdef files, rviz configurations and world files which needed to be done for SPOT (which is actually champ coloured in yellow) to work with our packages and solutions. In the end we got it working in a limited capacity by changing these files and installing extra packages, but it would have been helpful to be given some direction in how to do this during the course.

## SLO 5.2 Statement of what you will do differently

The biggest issue with our project was its weak object detection algorithm which worked in a limited capacity by detecting cylinders rather than actual people. The algorithm would also create false positives where it would detect corners and other objects similar in size and shape to the cylinders it was designed to detect. Because of this it would not be viable in a real life situation as a search and rescue solution, and instead only works as a concept in our simulated environments to show what could be possible with more robust object detection methods.

If I had the chance to do this again, or were given more time to refine the project, I would implement a CNN such as YOLO to perform object detection. This would greatly improve our detection algorithm and allow the robot to detect people of all shapes/ sizes, whether they are laying down, standing up, partially occluded due to debris or located somewhere outside the range of the laser scanner. This would also give the ability to detect other features in the environment like fires or collapsed structures and relay this information to support personnel.

Furthermore, our solution fails to independently generate a path through the environment. It requires a list of pre-determined waypoints to be sent to SPOT by the operator for it to explore the environment. In a real-world situation, it is unlikely that a map would be available for the operator to choose waypoints from, and this would also be very time consuming in large environments like forests and big warehouses. Because of this, I would have preferred our solution to use some sort of frontier exploration that allows SPOT to independently navigate and map the environment for support personnel while also relaying

information about injured people. This would be a much more realistic situation and would improve the autonomy of our solution.

## SLO 5.3 Peer Assessment

**Peer Review Project Pegasus- From RoboDawgs**

**Project Overview and Objective:**

The firefighting search and rescue robot system is designed to identify fire hazards and potential survivors in disaster environments. Their robot can locate areas that do not present on the pre-existing map and relay critical information to an operator as rumble or person. Future implementation for the robot includes an attachment of a hose capable of extinguishing fires and carrying medical and food supplies for survivors.

**Functional Criteria:**

- **Rumble Detection:** The system accurately detects rumbles in the environment and maps these areas. Potentially identifying persons trapped under debris which is conveyed back to operators for faster response.

- **Fire Detection:** This is achieved through a camera that identifies specific colours associated with fire, providing a basic solution. However, incorporating infrared could improve detection, especially in smoke-filled scenarios.

**Demonstration of Abilities:**

The firefighting robot system shows thoughtful integration of practical features:

- **SLAM for Rumble Mapping:** By comparing the occupancy map with ground data, the system accurately delineates areas affected by rumble, which can help locate people or assess hazard zones.

- **Waypoint Publication for Fire Location:** When a fire is detected, the robot publishes a waypoint, enhancing coordination with other responders.

**Conclusion:**

Using a thermal camera for fire detection is a good idea. Still, the TurtleBot's lack of an in-built thermal camera and reliance on a standard camera will pose a challenge. Since real-world fire detection typically requires infrared sensing, relying on colour detection may be

less reliable, especially in smoke-filled environments. Additionally, the fire suppression capability through a hose is innovative, but concerns arise regarding the TurtleBot's stability due to the physical forces of water pressure. The weight of medical supplies was mentioned however, we think that more is needed to hold, so addressing these forces with a stabilising mechanism or compensatory algorithm could improve reliability. It is crucial to assess the TurtleBot's fireproofing and its ability to withstand high temperatures and smoke. This project shows promise with innovative rumble and fire detection capabilities. This firefighting system could be an asset in real-world disaster scenarios with enhancements like stabilising mechanisms for hose deployment, improved fireproofing, and smoke resilience.

---

**Peer Review RoboDawgs - From Project Pegasus**

**Project Overview and Objective:**

The RoboDawgs team presented a Search and Rescue robot, aiming to assist in emergency scenarios by combining LIDAR and 2D visual camera technology for effective detection and navigation around objectives in hazardous environments. The project includes an intuitive GUI for ease of use, which is highly valuable for real-world applications where rapid interpretation is essential for people without much experience with robotics. The team’s ambition to implement a hierarchy for prioritizing survivors adds a thoughtful layer of decision-making that could significantly enhance the efficiency of rescue missions in future iterations and really increase the effectiveness of the system.

**Functional Criteria:**

The project meets several crucial search-and-rescue criteria:

- Human Detection: RoboDawgs used a well-thought-out approach for detecting people by identifying concave shapes and matching them against typical human contours.

- Environment Localization: The robot can reliably localize itself within complex environments, Allowing it to navigate itself without colliding with anything.

- Obstacle Avoidance and Path Planning: Path planning around new obstacles, especially people, shows the team’s foresight in dealing with dynamic environments that may be present in a search and rescue environment.

- Use of Quadrupedal Robot: Leveraging a quadruped for rugged terrain navigation is a smart choice that would improve stability and reach compared to a wheeled robot as it could navigate over complex terrain.

**Demonstration of Abilities:**

RoboDawgs demonstrated versatility with their system:

- Integrating Turtle Bot and Quadruped robots provides flexibility across different terrains, highlighting the project's adaptability to various rescue scenarios.

- The GUI functionality is particularly nice, enabling operators with minimal robotics experience to easily monitor robot status and access critical information. This capability aligns well with real-world operational needs, where simplicity and accessibility are key to ensuring success.

- Their 30m LIDAR scan radius is practical for larger area coverage, providing ample time for detection and decision-making in dynamic situations.

- Running the simulation fully in RViz with Gazebo in the background was a clever choice, offering smooth performance and enhanced visualization, both crucial for development and demonstration purposes.

- The team's path optimisation from the entrance to survivors stands out as a great feature that ensures rapid response by guiding first responders through the quickest routes.

## SLO 5.4 Concluding Statement

By the conclusion of this project, my team and I had developed a software package which allows for autonomous navigation of disaster environments with SPOT and the Waffle Pi TurtleBot. During navigation the robots are able to detect injured people and fallen debris while relaying this information back to support personnel. The shortest path between the injured people and support personnel can be visualised through Rviz to assist in reaching the injured faster. We also created a custom GUI, which streamlines controlling SPOT and utilizing the package's functions efficiently.

To create the above solution, a lot had to be learnt about available Ros2 packages like Nav2, Cartographer and OpenCV. Navigating their functions, data types, and ensuring they worked seamlessly together were the most difficult challenges that had steep learning curves. Beyond these packages, I also learnt a lot more about launch files, configurations files, and how to tailor these to my specific needs. I think this is the most important thing I learnt as it is an important aspect of coding to understand, regardless of the platform or packages being used.

The skills and knowledge gained from creating this solution will be helpful for future courses or jobs where coding is necessary (especially in ROS). The experience of integrating multiple software packages and solving countless errors will make it easier in future situations.

Moreover, gaining a basic understanding of customizing launch and configuration files will help me adapt to a range of robotics or software projects. Moving forward, I feel more confident tackling complex robotics projects due to this experience.

# References

*AWS RoboMaker Small Warehouse World*. (2022, August 13). GitHub. https://github.com/aws-robotics/aws-robomaker-small-warehouse-world

Boston Dynamics. (2024a). *Payload*. Boston Dynamics. https://bostondynamics.com/products/spot/payload/

Boston Dynamics. (2024b). *Spot Automate sensing and inspection, capture limitless data, and explore without boundaries. ® Operate with Ease*. https://bostondynamics.com/wp-content/uploads/2020/10/spot-specifications.pdf

Carlson, C. (2023, March). *How Ultrasonic Sensors Work*. MaxBotix; MaxBotix. https://maxbotix.com/blogs/blog/how-ultrasonic-sensors-work?srsltid=AfmBOopywTS9627DQmM5BFWmzndlGtH9ZsVUGVTvHB6UimYTE-6HeyAl

Collett, D. (2024, February 27). *Simulation-based learning*. Learning Environments. https://le.unimelb.edu.au/news/articles/simulation-based-learning#:~:text=Simulations%20can%20enrich%20teaching%20and

Commission, corporateName:Productivity. (2023, January 31). *9 Emergency services for fire and other events - Report on Government Services 2023*. Www.pc.gov.au. https://www.pc.gov.au/ongoing/report-on-government-services/2023/emergency-management/emergency-services

Dirican, C. (2015). The impacts of robotics, artificial intelligence on business and economics. *Procedia - Social and Behavioral Sciences*, *195*, 564–573. https://doi.org/10.1016/j.sbspro.2015.06.134

Eser, A. (2024). *Average House Fire Temp Statistics: Market Data Report 2024*. Worldmetrics.org. https://worldmetrics.org/average-house-fire-temp/

Fire and Rescue NSW. (2021, October 15). *FRNSW Privacy Policy*. Fire and Rescue NSW. https://www.fire.nsw.gov.au/page.php?id=9117

Glamox. (2024). *IP classifications*. Glamox Prod. https://www.glamox.com/en/pbs/knowledge-centre/ip-classification/

intel REALSENSE. (2024). *Intel® RealSense™ LiDAR Camera L515*. Intel® RealSense™ Depth and Tracking Cameras. https://www.intelrealsense.com/lidar-camera-l515/

*Introduction to A**. (n.d.). Theory.stanford.edu. https://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

Kooser, A. (2020, June 16). *Boston Dynamics' four-legged Spot robot dog finally goes on sale for $74,500*. CNET. https://www.cnet.com/science/boston-dynamics-robot-dog-spot-finally-goes-on-sale-for-74500/

Kumar, P. (2022, September 30). *What are depth-sensing cameras? How do they work?* E-Con Systems; e-con Systems. https://www.e-consystems.com/blog/camera/technology/what-are-depth-sensing-cameras-how-do-they-work/?srsltid=AfmBOopndu37GB0cUe4F5WmyEzjbMdHBtz6bHObygP5tsvJfbNcQlL3X

Lin, P., Abney, K., & Jenkins, R. (2017). Robot Ethics 2.0: From Autonomous Cars to Artificial Intelligence. Oxford University Press.

Marchang, J., & Di Nuovo, A. (2022). Assistive Multimodal Robotic System (AMRSYS): security and privacy issues, challenges, and possible solutions. *Applied Sciences*, *12*(4), 2174. https://doi.org/10.3390/app12042174

Navone, E. C. (2020, September 28). *Dijkstra's Shortest Path Algorithm - A Detailed and Visual Introduction*. FreeCodeCamp.org. https://www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual-introduction/

Perth Window and Door. (2023, January 9). *The Ultimate Guide to Door Sizes in Australia*. Perth Window & Door Replacement; Perth Window & Door Replacement. https://www.perthwindowreplacement.com.au/latest-news/door-sizes-guide/#:~:text=The%20typical%20ranges%20of%20this

*Robotic sensors: The interface between robots and the world*. (2024). HowToRobot. https://howtorobot.com/expert-insight/robotic-sensors-interface-between-robots-and-world#:~:text=LIDAR%20is%20a%20remote%20sensing

Selby, W. (2019, October 9). *Building Maps Using Google Cartographer and the OS1 Lidar Sensor*. Ouster.com. https://ouster.com/insights/blog/building-maps-using-google-cartographer-and-the-os1-lidar-sensor

Sydney, D. W. (2015, August 20). *SketchUp wall thickness*. Graphic Design Courses. https://www.designworkshopsydney.com.au/sketchup-wall-thickness/#:~:text=These%20will%20vary%20according%20to

Thrun, S. (2002). *Robotic Mapping: A Survey*. http://robots.stanford.edu/papers/thrun.mapping-tr.pdf

*Understanding SLAM in Robotics and Autonomous Vehicles*. (n.d.). Www.flyability.com. https://www.flyability.com/blog/simultaneous-localization-and-mapping

# Appendix

## Appendix A – Minutes of Meeting (14/08/24)

**Attendees:**

- Akkarawat Kaveewatcharanont
- Connor Fitzgibbon
- Justin Pavlovski
- Liam Faulkner-Hogg
- Pravien Kugan
- Yves Gayagay
- Sangmim Song ( Client )

**Agenda:**

1. [Agenda item 1] Project Design
2. [Agenda item 2] Design Specifics
3. [Agenda item 3] Design Objectives

**Discussion Points:**

- **[Agenda item 1]: Project Design**
    - Problem statement including risks for people and their safety
    - Reducing the need for danger of rescuer
    - Manually or Autonomously search from remote control
- **[Agenda item 2]: Design Specifics**
    - We are allowed to make up our own environment
        - Pre-made environment or Build Your Own environment
            - Pre-made examples given by client include Amazon warehouse and hospital
        - We or the Client could choose only one environment
            - We will be choosing the environment

- **[Agenda item 3]: Design Objectives**
    - Summary of project scope is in canvas
    - Clients thought's on our design objectives:
        - What sensor should we use? and how many sensors should we include?
            - Lidar, RGBD camera for indoor environment
            - Radar and camera for outdoor environment because if there is smoke/fire to overcome we use radar
            - 2-3 sensors should be used for Localisation, Mapping and Perception
            - Microphone can be used but "Desired"
        - Client suggests we do indoor as outdoor is way more complicated and sensors we listed suit best indoors
        - Outdoor/both as a "Desired"
        - What path planning strategies should we use?
            - Client suggests have a brief 2d map since some danger could occur after and to plan the path quickly "Necessary"
            - Frontier style exploration is good but harder to implement "Desired"
        - Anything important/missed in Design Objectives :
            - Expect to find people in disaster, find their location and report back their locations "Necessary"
            - Go in, scan, report back the location of people break into desired or required
        - Constraint:
            - Up to us to choose how many people will be on site and the robot locates
            - Robot will not know initially how many people are on site
        - Physical robot implementation for higher marks

- Using actual spot robot? nah any hardware with the required sensors.
- Client will inquire and let us know what are details for hardware implementation, the requirements leading to the hardware and if we can make a scaled down model
    - 1. Establish sensor how it works
    - 2. Not many hardware available
    - 3. Working simulation
    - 4. Will let us know what to do prior to hardware

**Action Items:**

- [Action Item 1] Update the Design Objectives
- [Action Item 2] Plan meeting; keep as 10am Wednesday minimum of 3 meetings throughout semester or request for more meetings
- [Action Item 3] Focus on sprint 1

**Next Meeting:**

- **Date:** 28 August 2024
- **Time:** 10AM
- **Location:** Online via Zoom

**Notes:**

- [Additional notes]

# Appendix B – Sprint 1 Task Dashboard

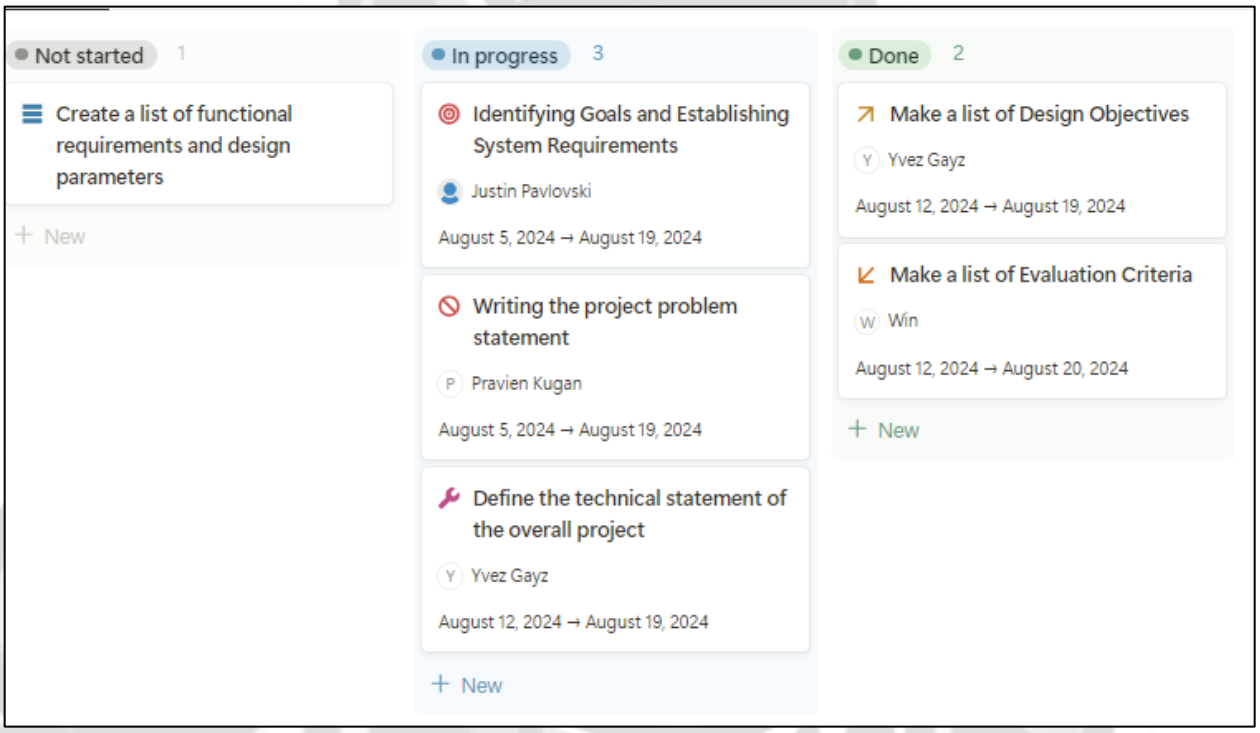



*Figure 5.1 - Team Notion Task Allocation*

# Appendix C – Team Charter

**Team Name: Robo Dawgs**

**Team Members:**

**Connor Fitzgibbon** **Justin Pavlovski** **Pravien Kugan**

**Akkkarawat Kaveewatcharanont** **Liam Faulkner-Hogg** **Yves Gayagay**

## Team Vision/Purpose:

Our team is committed to creating a coherent and comprehensive solution to allow the SPOT robot to become an aid in disaster situations to assist in saving lives.

## Members Skills/Knowledge/Expertise:

*Table 3 - Team Charter Table*

| Name | Skills/ Strengths | Group Skills I Want to Work On | Stressors | Group Role(s) |
|---|---|---|---|---|
| **Akkarawat Kaveewatcharanont** | - proof reading<br>- research preparation | **- listening to others prior to inputting ideas** | **- multiple deadlines** | **- Localisation** |
| **Connor Fitzgibbon** | - Programming<br>- Leadership<br>- Talking | **- Collaboration with a large number of people** | **- Missed Deadlines**<br>**- People not showing up** | - Team organisation<br>- Path Planning |
| **Yves Gayagay** | - Programming<br>- Application<br>- Problem Thinking | **- Effective and efficient time management** | - Missed deadlines | **- Mapping** |
| Justin Pavlovski | **- Programming**<br>**- Implementation** | - Localisation and Mapping<br>- Leadership | **- Stress (σ)**<br>**- Strain (ε)** | - King of the nerds (coding lead)<br>- Programming |
| **Pravien Kugan** | - Oral Communication<br>- Critical Thinking<br>- Problem Solving | **- Working in a big group**<br>**- Express my ideas and opinions in projects** | - Missed Deadlines<br>- many projects or deadlines on the same date or close together | **Follower of the nerds (coding assistant or servant)**<br>**- Programming** |
| Liam Faulkner-Hogg | - Modelling<br>- Management<br>- Problem Solving<br>- Always keen to try new things | Efficient File Sharing | - Multiple deadlines close together<br>- Bad communication | 6th Member<br>- Path Planning |

## Guiding Principles and Norms - How will you operate together for the duration of the project?

**Communication:** Outline how the team will communicate — include frequency and methods (e.g. email, Facebook, team meetings). What is the maximum expected response time?

Our primary communication tool will be through Facebook Messenger where we will discuss meeting times and our weekly agenda. In conjunction with that, we will use notion to track progress and share files. We will commit to an in person, weekly meeting time which will occur on a Monday afternoon where we will discuss what tasks need to be completed and with what priority.

**Decision-Making:** How will decisions be made in this team? How will you stay on track?

Decisions will be made democratically and will occur in our team meeting or, if they are urgent, will occur over teams where we will create a poll for members to vote.

**Conflict Resolution:** How will you resolve differences?

Indecisions will be decided by a democratic vote. All team members will given a chance to have their opinions heard. If a final decision cannot be reached, an external mediator will be sought to help resolve any issues.

**Commitments:** How will you handle different levels of participation and commitment? What process will you follow if someone does not live up to his or her responsibilities? What are the consequences for poor performance?

Everyone will be encouraged to follow what they set out to do in the charter. Firstly, support will be offered to help with any blockers or issues that team members may be facing. We will discuss as a team if there are possible alternatives to distribute work if needed. If there are repeated breaches of the team charter and a resolution with team members cannot be reached, the issue will be taken up with our tutor and/or subject coordinator.

**Diversity:** How will you accommodate different learning and working styles?

We will extensively discuss strengths, weaknesses and working preferences with the aim of distributing work based on preference and ability. We all understand that everyone has different working styles and will attempt to use this to our advantage. Attempts to provide support will be given by team members whenever possible. If different learning and working styles develop into an issue within the team, a meeting will be held to discuss possible solutions.

**Other Procedures:** Are there any other commitments, goals, processes or responsibilities, roles that your group has agreed upon? Consider the scope of work, work schedule, rituals, or other information that you may want to include.

We have all shown interest in being able to actually implement our project on the SPOT robot to see how well our code and designs function in a real application rather than simply through simulation.

**Signatures**

Member 1 *Connor Fitzgibbon*_____ Date: *05/08/2024*_____

Member 2 *Akkarawat Kaveewatcharanont*____Date: *05/08/2024*_____

Member 3 *Justin Pavlovski*_____ Date: *05/08/2024*_____

Member 4 *Pravien Kugan*_____ Date: *05/08/2024*_____

Member 5 *Yves GayaGay*_____ Date: *05/08/2024*_____

Member 6 *Liam Faulkner-Hogg*_____Date: 05/08/2024_____