Assignment 1: Challenge Definition

ECED 3901: Electrical Engineering Design II

For: Dr. Vincent Sieben

By: Ashton Dudley (B00934296)
Liam Legge (B00960172)
Lorne MacLean (B00912671)
Lucas Melanson (B00955897)
Megan Neville (B00969137)

January 21, 2026

1 Client Requirements

The client requires an automated system that will allow them to traverse through a coastal path and through open waters to deliver cargo. The design must operate safely as to not hit any obstacles, protect against intruders, and deliver the cargo to the required port and return in less than 3 minutes. There must be safety lights on the vehicle.

1.1 Requirements List

Requirements are set up by the initial objective of the project. There are six systems that need to be completed, and five of them are critical to goals that the project must complete. Those are:

> Safety lights, Autonomous Navigation, Cargo Delivery Safely in both Ports, Rescue System and Pirate Deterrent System

1.2 Constraints List

Constraints are set up by the project goal as well. We want to limit what is possible to do in the project, to make it realistic to complete in an efficient and timely manner. These constraints are what allows a project to be feasible. These include:

> $300 Budget, only 3D printed parts, Only Suppliers are Digikey for parts, and PCBway for PCBs, Deliverables must be completed within 3 months, Deliver cargo in 3 mins, No pneumatics/hydraulics, and No gas/aerosol/liquid systems

1.3 Future Planning

R&D needs to be done on the ROS2 system, and how each command can be used in sync with the LIDAR to autonomously control the vehicle. We will need to research how to use Gazebo with ROS2 so that we can simulate the control of the vehicle to improve the program. Training with the Linux OS will be required, as well as with the ROS2 software to generate a navigation map for the vehicles decision-making process.

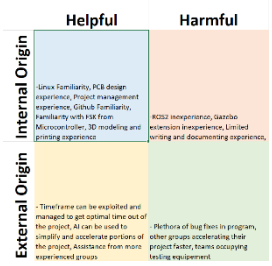


Figure 1. SWOT

As said before, the resources we have are the device itself, $300 that can be used to purchase components from Digikey, and PCBs from PCBway. There is access to the Electrical Design shop, where soldering irons can be used and oscilloscopes and multimeters are available to test other systems that may be developed. The team has skills and experience that will provide beneficial to the project. Experience in programming in languages of Python, C, C++. Experience in soldering and designing PCBs and testing them, will provide useful for external systems. Project management experience is helpful when planning and outlining a timeline for the project.

2 Risk Analysis

A risk analysis will be completed to identify problems that may come up in the future, to be prepared for the issue and have solutions set up. This is to fix the problem quickly to continue work efficiently.

2.1 Stakeholders

> Team Members: Ashton, Liam, Lorne, Lucas, and Megan

Each team member is responsible for a system of the design. It is expected that the R&D will be done comprehensively, to ensure that the best method is chosen through decision matrices. As well, a working system is expected at the end of the project lifespan.

Dr. Vincent Sieben

Dr. Vincent Sieben expects a fully functional AOV that is capable of completing the challenge that has been laid out. The budget of $300 cannot be exhausted, as that is the max that our sponsor has given us.

Lab Technicians: Justin Stanley, Tyler Seize, Jordan Wright, and Alex Pudsey

The lab technicians are responsible for ensuring that the challenge can go ahead, as they have the course set up, and are responsible for the 3D printing of some components and will receive all shipments of components and materials.

2.2 Risks for Stakeholders

Team Members – Failure to complete individual tasks, Delay in completion of independent task

Dr Vincent Seiben – Non-functional prototype, Overbudget on funds

Lab Technicians– Excess time spent on project, Limited availability for assistance

2.3 Risks Matrix and Tornado

A risk matrix was assigned to determine for each project, how sever a problem would be if it arose. The three severities are set, those being cosmetic damage, an impairment of one or more modules, and then a complete failure of the AOV.

A risk tornado is a graphing tool used to measure the risk and the impact of said risk on the project. The y-axis being the task that has to be complete, and the x-axis being the good and bad effect of the task. Below is a half-tornado graph of the project.

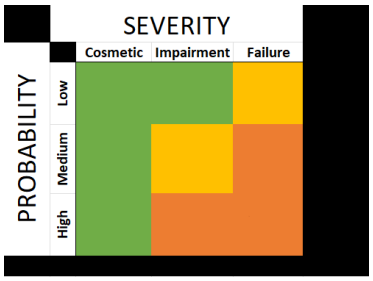
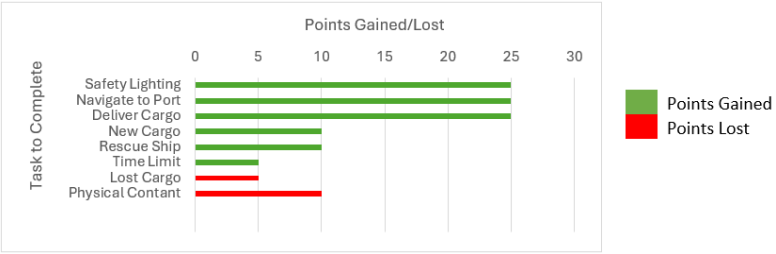


Figure 2. Risk Matrix

Table 1. Half-Risk Tornado



All of these assessments will be used to determine the best course of action to follow while working on the project, to reduce risks which will reduce delays.
Modular testing of all the systems before integration will be crucial for keeping a quick timeline on the project, and to ensure that all systems are functional with no critical bugs or issues.
Virtual Navigation testing using Gazebo will be important to allow the navigation system to operate properly without risking damage to the AOV. It will also be quicker to fix the bugs in the code virtually instead of manually restarting the AOV.
Battery Stress testing will be important to ensure that all the systems can confidently run off of the power supply. This can be conducted by running open circuit and load tests, as well as capacity verification tests to ensure that the AOV systems will not die mid-run.

3 ROS2 Architecture

This project will make use of Robot Operating System 2 (ROS2). This section covers the basics of ROS2 and how it will be used within the project.

3.1 ROS2 Overview

Robot Operating System 2 (ROS2) is a collection of tools and software that can be used to build the controls for a robot. Despite its name, it is not a true operating system. Rather, it is a collection of middleware that works closely with the host operating system to allow robot components to communicate efficiently and be controlled by a user. ROS2 is also open source, allowing for users to adjust the code base based on their needs. It provides the groundwork for basic autonomous robot operation and solves the problem typically encountered in the past, where users would essentially need to build the middleware for sensors and control from scratch.

3.2 Useful ROS2 Features

ROS2 will provide many features that will be useful to our design. Since ROS2 uses modular nodes, each of the features required for a functioning autonomous robot can be developed individually. This allows for easier team development, where each node could be worked on by individual team members and the node will integrate with the system easily. This will also help with debugging, since each component of the robot will be self-contained. This allows editing the problem node without affecting the other components that will be contained in a separate node.

ROS2 also has pre-built tools that will be very useful in development of our robot. This includes a navigation stack that will make course planning, object avoidance, and mapping much easier. It also includes software for visualization called RViz. This provides visualization of what the LiDAR is scanning and will be useful in the development for autonomous navigation. The simulation package, Gazebo, is included with ROS2. This will be very useful in development, allowing us to run simulations of the robot without needing physical access to the course.

3.3 ROS2 Software Architecture

As mentioned previously, ROS2 makes use of nodes that package individual robot actions into their own software executable. Each of these nodes are then able to communicate and transfer data to each other. These nodes use a publisher/subscriber architecture to transfer information across the system. ROS2 uses topics to accomplish this. A topic essentially serves as a connection point between nodes that receives and distributes data. The publisher in this architecture is a node that is sending information to a topic to be distributed to another node. The subscriber is a node that is receiving the information from a topic. This architecture allows communication between multiple nodes easily, and many nodes can be “subscribed” to a topic and receive the information sent by the publishing node. It also allows multiple nodes to publish data to the topic, and the subscribers will be able to get information from both publishers. A simple node to node connection is shown in Figure 3.

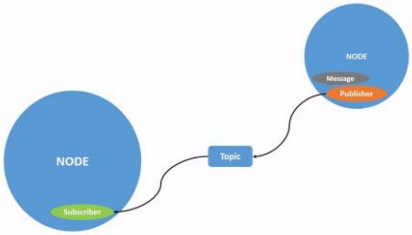


Figure 3. Node to node connection in a publisher/subscriber architecture. Adapted from ROS Documentation.

4 ROS2 Interface and Topics

ROS2 uses three main types of interfaces in order to communicate, messages, actions, and services. The following section will dive deeper into the uses and definitions of the different ROS2 interfaces.

4.1 Messages

Messages are stored in the /msg section of a ROS2 package and all end in .msg. Message files are made up of constants and fields. Constants are standard definitions that will be repeatedly used in different sections of the code, they are fixed values. Fields are similar to constants, but their values are not fixed. A field is a declared variable with a defined name and type, which can be reused repeatedly with different values being assigned to it.

4.2 Actions

Actions all end in .action and they describe actions. Each action file has three main parts, the goal, the result, and any feedback. Actions are used for longer tasks that may not have an immediate response. Actions use both topics and services in order to achieve the final result. An action can also be canceled while it is running, as it is often used for longer tasks, using the cancel_goal() function.

4.3 Services

Messages are stored in the /srv section of a ROS2 package and all end in .srv. A service is a communication method used by ROS2 which allows for a request and an immediate response. A request is given (asking for some sort of question), and then a response is given (an answer to the question). The request and response are separated by three dashes (- - -) in the service code. In both the request and response, you will have constants and fields.

4.4 Topics

One of the ways that ROS2 allows information to travel in between nodes is by using Topics. A node will publish data to a Topic, that Topic will then be able to send that received data to other nodes that are "subscribed" to it. ROS2 comes with a few Topics that are already active in the system. A few of these topics will be analyzed in the following section.

- /cmd_vel is the Topic used for velocity commands. Nodes will publish velocity commands to /cmd_vel and the robot or simulator (or both) will subscribe to this Topic and then act on the commands given by the nodes. These commands can be saved as messages so they can be reused or reassigned.
- /odom is the Topic that has odometry data. Odometry data includes estimated position relative to start position. This includes x and y position as well as velocity. This may not be completely accurate due friction while turning and unexpected drift from the desired route.
- /LaserScan will be the Topic where all of the Lidar data is published. Different nodes will subscribe to this topic in order to access and use the data from the Lidar. This will be used in order to create commands to send to /cmd_vel to allow for successful robot navigation.

4.4.1 Additional Topics

There are a few other Topics that will most likely be needed to complete the Design Challenge. Transform frames on /tf and /tf_static define how the robot's links and sensors relate in space. Mapping and localization topics such as /map and /amcl_pose let the robot track its potion in an an environment map. Nav2 uses this information to issue motion commands and plan safe paths. Sensor-fusion inputs such as /imu and /filtered_odom refine pose accuracy by combining multiple data sources. A URDF-based robot description provides the structural model needed for tools like Gazebo. These are a few of the additional necessary Topics that will be needed, there may be more needed as we go through the design process.

5 Current ECED3901 Robot Design Flaw

The robot is fully operational from Day 1. However, will all technology, it can be improved and upgraded with different components, by making it lighter or cheaper. A review of the schematic will be completed to ensure that the IC application is correct with no issues. After this, an overall design improvement will be suggested.

5.1 Schematic review

The notes below indicate that we have reviewed the schematic and found that there were no issues with the ICs being implemented into the system

5.1.1 9-Axis IMU

Connects to the microcontroller via a UART connection. COM1 and COM2 pins are correctly connected to TX and RX and Clock speed is correct; however, the capacitor is sized for 4pF, rather than 22pF (Datasheet p.100). This is likely fine and should not cause issues.

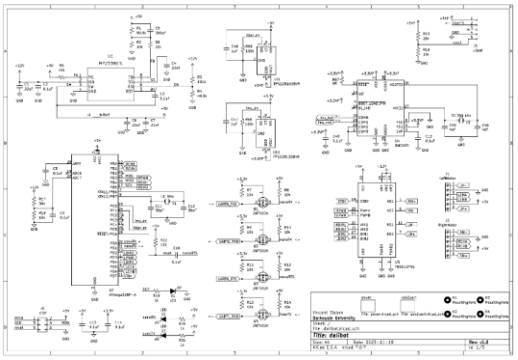


Figure 4. Microcontroller, IMU, Motor Driver and Switching Regulator

5.1.2 Switching Regulator

The board schematic matches the datasheet. Input/output range is fine, and the circuit topology is correct. This is likely fine.

5.1.3 TB6612FNG

This is the motor driver IC. The power connections are correct. Circuit topology matches datasheet typical application. Potentially missing 0.1uF and 10uF decoupling capacitors but should not cause issues.

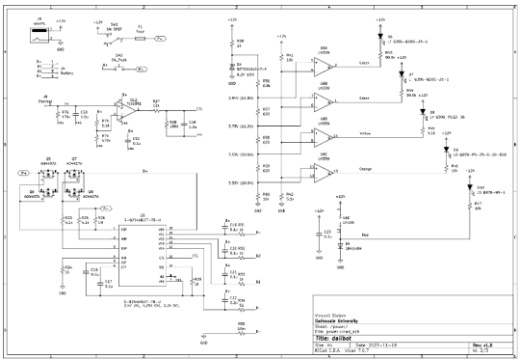


Figure 5. BMS and voltage indicator circuit

5.1.4 BMS and Voltage indicator

The diode is used for reverse polarity protection of the BMS indicator light. BMS circuit matches the 4-serial cell example. USB input looks okay. ID pin is not used, since it is not needed for USB 2.0. J7 used as an external UART connection. Crystal frequency is correct. UART1_TX is not connected to U7. However, the LiDAR does **not** have a RX pin, so this is not an issue.

5.2 Overall Improvement

The overall design of the ECED3901 robot is good. There is a change that could improve the robot, and that is by lowering the LiDAR/getting a LiDAR that can display in three dimensions. The issue with a two-dimension LiDAR that is high up on the robot is that low objects will not be scanned and the autonomous navigation may collide into those objects. This is one of the issues with picking up the Rescue boat and new Cargo, as the LiDAR will not be able to identify the position of the cargo as it is too low to the ground.

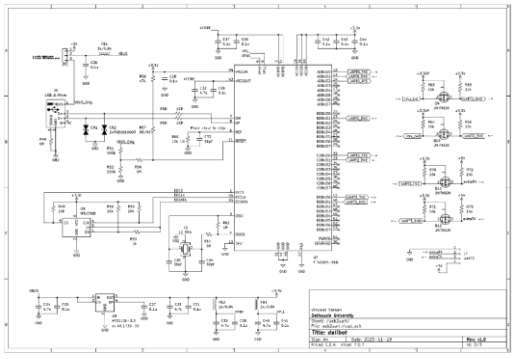


Figure 6. FT4232H-56Q Quad High-Speed USB to Multi- Purpose UART/MPSSE IC