GOING MERRY ROADSTER

TECHNICAL REPORT(ELTR2402)

Sponsors

Cory Thorp

Fraaz Kamal

Timothy Olusodo

December 16, 2024

Abstract

This technical report aims to discuss the process involved in the making of the Going Merry Roadster from concept to completion. This project is a realized roadster version of the distinctive Going Merry pirate ship from the One Piece anime series. It moves with the help of a RC (Remote Controlled) car type system, with the body of the ship assembled around said system. Wireless Control is achieved using bluetooth connection between the user's mobile phone and the microcontroller. Additionally, the mast can be remotely controlled to extend or retract and One Piece themed sounds can be played on a speaker built into the ship.

This project was initiated due to my passion for the One Piece anime. Initially I thought of making something that could sail on water but it was too complicated for me to achieve at the moment, so I decided to make it solely for use on land. All in all, I enjoyed making this project and it is generally for fun.

The Design process of this project is divided into three categories: Chassis Fabrication, Firmware Development, and Pirate-Ship Fabrication & Assembly.

Chassis Fabrication: The main part of the Chassis is a flat aluminum metal sheet with holes for the battery, switch, screws, and the DC motors. The battery case, standoffs(for mounting the custom PCB), motor mounting brackets, and pirate ship is tightly fixed on this metal sheet. Major components mounted on the PCB are two TB6612FNG motor drivers (Quad & Dual), HM-10 Bluetooth Module, ADXL345 Accelerometer, Arduino Mega 2560

Firmware Development: I created functions for every action taking place in the project, and in the main loop I simply used conditional statements to call the functions. These conditional statements are based on button presses from the mobile application connected to the systems bluetooth.

Pirate-Ship Fabrication & Assembly: Every other part of the ship was 3d designed and printed using the CAD software Shapr3d for designing. Some parts were later painted before assembling every piece together.

Testing each component had its ups and downs. The motors and mast mechanism performed reliably, demonstrating smooth movement and precise control. Working with the sensors(Accelerometer & Time-of-flight sensor(TOF)) and the HM-10 Bluetooth Module proved a bit challenging as the accelerometer provided inaccurate and unstable readings, the TOF sensor gave slightly inaccurate distance measurements and the HM-10 would randomly stop responding. The sound module and speaker worked as expected and delivered clear audio output.

Based on the test results, the project is functional but leaves some room for improvement. Future iterations should focus on reducing the overall size of the build by replacing huge breakout boards with smaller ones and exploring more compact battery options. Addressing Bluetooth reliability through alternative modules or better signal management would improve user control. Mechanically, increasing the durability of 3D-printed parts by using higher infill density and thicker walls will help avoid breakages. The PCB, while fullly functional, presented slight problems, including inaccurate through-hole placements for the DYHV20T sound module and shaky components due to low-quality female sockets. A highly accurate and refined PCB design with better-quality sockets, will enhance stability and reliability. These refinements will not only resolve current challenges but also elevate the project's performance, durability, and usability.

TABLE OF CONTENTS

1.0 INTRODUCTION ----- 5

1.1 PROJECT PURPOSE ----- 5

1.2 PROJECT SCOPE ----- 5

TECHNICAL CONTENT ----- 7

2.0 COMPONENTS ----- 7

2.1 Comparisons ----- 8

2.2 Hardware Block Diagram ----- 9

3.0 TESTS AND RESULTS ----- 10

3.1 Subsystem 1: Arduino -> Quad Driver -> DC Motors ----- 10

3.2 Subsystem 2: Arduino -> Dual Driver -> Linear Actuator ----- 10

3.3 Subsystem 3: Arduino -> DYHV20T -> Peerless Speaker ----- 10

3.5 Subsystem 5: VL53L0X ToF -> Arduino ----- 11

3.6 Subsystem 6: Arduino -> Adafruit NeoPixel Ring ----- 11

4.0 CIRCUIT DESIGN ----- 12

4.1 Output Transducers ----- 13

4.2 Input Transducers/Others ----- 14

4.3 Power System ----- 16

5.0 PCB DESIGN ----- 17

6.0 ELECTRONICS ASSEMBLY ----- 20

6.1 PCB Population ----- 20

6.2 Chassis Assembly ----- 22

7.0 FIRMWARE ----- 23

8.0 PRJOECT ENCLOSURE BUILD ----- 25

9.0 FINAL BUILD ----- 28

10.0 GRAPHICAL USER INTERFACE (GUI) ----- 31

11.0 CONCLUSION AND FUTURE PLANNING ----- 32

LIST OF FIGURES

Figure 1: Hardware Block Diagram ---------- 9

Figure 2: Circuit Diagram ---------- 12

Figure 3: Output Transducer Circuit ---------- 13

Figure 4: Input Transducer Circuit ---------- 14

Figure 5: Power System Circuit ---------- 16

Figure 6: Top part of PCB CAD Image ---------- 17

Figure 7: Bottom part of PCB CAD Image ---------- 18

Figure 8: Top Layer of Unpopulated PCB ---------- 18

Figure 9: Bottom Layer of Unpopulated PCB ---------- 19

Figure 10: Populated PCB ---------- 20

Figure 11: Modified PCB (Sound Module) ---------- 21

Figure 12: Battery Pack CAD ---------- 22

Figure 13: Power Switch ---------- 22

Figure 14: Chassis ---------- 23

Figure 15: Full Enclosure CAD ---------- 25

Figure 16: Front of the Ship CAD ---------- 26

Figure 17: Rear of the Ship ---------- 26

Figure 18: Body of the Ship CAD ---------- 27

Figure 19: Assorted Parts CAD ---------- 27

Figure 20: Final Build (SIDES) ---------- 28

Figure 21: Deck and Actuator Build ---------- 28

Figure 22: Top View of Assembled Build ---------- 29

Figure 23: Project's Assembled Build ---------- 30

Figure 24: Dabble App: Landing Page and Gamepad Module ---------- 31

LIST OF TABLES

Table 1: Chosen vs Alternative Components ----------8

Table 2: ADXL345 SPI Connection ---------- 15

1.0 INTRODUCTION

This project was made for the ELTR2402 course, and this section aims to shed some light on the purpose of the project and its scope.

1.1 PROJECT PURPOSE

This project was inspired by my desire to blend creativity and engineering into something fun, functional, and somewhat out of the ordinary. The idea of a pirate ship on wheels, fully controllable via Bluetooth, brought together skills in robotics, electronics, 3D printing, and programming. It wasn't just about building a cool project—it was about applying what I've learned over my years of study in real, hands-on ways. Although the project isn't intended to solve anything in particular, it fixes the challenge of making RC vehicles more engaging and interactive. Most RC vehicles are limited in features, being able to only move in high speed, but this project goes beyond with dynamic elements like a controllable mast, music playback, and lighting effects.

1.2 PROJECT SCOPE

The pirate ship project includes several key features:

- **Bluetooth-based control**: Using an HM10 module and the Dabble app's gamepad module, the ship can be remotely controlled via a smartphone.
- **Mobility system**: Four DC motors with encoders provide forward, reverse, and turning capabilities, allowing for precise control of movement.
- **Controllable mast**: A 12V linear actuator enables the mast to extend and retract, simulating the hoisting of a sail.
- **Music playback system**: A DYHV20T sound module and Peerless 20W speaker enable onboard audio functionality, allowing the ship to play music.
- **Decorative lighting**: Neo-Pixel LEDs around the mast provide customizable and programmable lighting effects.
- **Obstacle detection**: VL53L0X TOF sensor measures distance between the ship and an obstacle to deliver accurate obstacle avoidance.
- **Custom 3D-printed parts**: These give the ship its distinct pirate aesthetic and functional structure.

Each of these features served an important purpose. The Bluetooth control made it easy and intuitive to operate. The motorized mobility ensured the ship could move smoothly and accurately. The mast control added a fun, interactive element that fit the pirate theme perfectly. The sound system made it more immersive, while the lighting added visual appeal. Lastly, the custom components tied everything together, making the design both functional and visually impressive.

That said, the project wasn't without its challenges. The final build ended up larger than expected because of the big battery pack and the mounted breakout boards on the PCB. Bluetooth communication wasn't always reliable as the HM10 sometimes struggled to maintain a stable connection with the Arduino. On the mechanical side, some of the 3D-printed parts broke easily due to low infill density and thin walls, and the PCB had a few design flaws, like misaligned through-holes for the sound module and low-quality sockets that made components shaky.

These limitations came down to a few factors. The large build size was a tradeoff for using off-the-shelf components and a high-capacity battery, which prioritized functionality over compactness. The Bluetooth issues likely stemmed from the HM10 module's sensitivity to interference. The fragility of the 3D-printed parts was due to design choices meant to save material and time during printing. And the PCB problems were a mix of measurement errors and using cheaper sockets to keep costs down. Despite these challenges, the project achieved its goal of being unique and functional. It also highlighted clear areas for improvement, paving the way for future revisions. With better component selection, refined 3D printing techniques, and improved PCB design, the next iteration of the pirate ship could be even more compact, reliable, and polished.

TECHNICAL CONTENT

The technical content of this report will cover the design process of the project including components used, tests performed, circuit and PCB design, assembly and more.

2.0 COMPONENTS

This section highlights the features of every component involved in the design of this project with a table of comparisons and the hardware block diagram. Below is a list of the components chosen for this project and their respective features:

- **DYHV20T MP3 Module**
  Supports various audio formats, built-in storage, and serial communication interface.

- **DFRobot Quad Driver Shield (TB6612FNG)**
  Can control up to 4 DC motors or 2 stepper motors, integrated TB6612FNG motor driver chips.

- **DFRobot Dual Motor Driver (TB6612FNG)**
  Dual-channel motor driver, small form factor, high efficiency.

- **DFRobot DC Geared Motor with Encoder (6V, 210RPM, 10Kg.cm)**
  High torque, built-in encoder, geared for precision.

- **12V DC Linear Actuator 50mm**
  Linear motion, high force, compact design.

- **DFRobot VL53L0X TOF Distance Sensor**
  Time-of-flight measurement, long-distance range (30mm-2000mm), accurate distance measurement.

- **HM-10 BLE Module**
  Bluetooth Low Energy, easy to use, compatible with Arduino.

- **ADXL345 Accelerometer**
  Measures acceleration in three axes, low power, high resolution.

- **TXB0104 Bi-Directional Level Shifter**
  Converts voltages between different logic levels.

- **Arduino Mega 2560**
  Large number of I/O pins, multiple communication options.

- **Pololu Step-Down Voltage Regulators (D36V50F7 and D36V50F6)**
  High efficiency, compact size, wide input range.

- **Peerless by Tymphany TC7FD00-04 20W Speaker**
  High power handling, good frequency response, compact size.

- **Adafruit NeoPixel Ring - 24 x 5050 RGB LED**
  Individually addressable LEDs, high brightness, easy control.

- **CNHL 3S Lipo Battery 8000mAh 11.1V 120C**
  High capacity, high discharge rate.

2.1 Comparisons

Below is a table of the chosen components against the alternative component in the market. In the reference columns, I embedded links providing more info about each component.

Reference (Chosen)	Component	Reference (Alt)	Alternative
1	DYHV20T MP3 Module	1	DYSV5W MP3 Module
2	DFRobot Quad Driver Shield	2	L298N Motor Controller
3	DFRobot Dual Motor Driver	3	L298N Motor Controller
4	DFRobot 6V DC Geared Motor	4	Pololu 37D Metal Gearmotor 12V
5	12V DC Linear Actuator	5	Actuonix L12 Series
6	DFRobot VL53L0X ToF Sensor	6	DFRobot VL53L3CX ToF
7	HM-10 BLE Module	7	HC-05 Bluetooth Module
8	ADXL345 Accelerometer	8	MPU-6050
9	TXB0104 Level Shifter	9	BSS138 Logic Level Converter
10	Arduino Mega 2560	10	Arduino Uno
11	Pololu Step-Down Regulators	11	LM317 Linear Regulator
12	Peerless TC7FD00-04 Speaker	12	Gikfun 1.5" Speaker
13	Adafruit NeoPixel Ring	13	BTF-LIGHTING RGB LED Strip
14	CNHL 11.1V 3S Lipo Battery	14	Zeee 3S Lipo Battery

Table 1: Chosen vs Alternative Components

Reason(s) for the choice of components

The components chosen for this project were selected based on their specific features, performance, and compatibility with other components. For instance, the ***DYHV20T MP3 module*** was chosen for its reliable performance and high-power audio output. The ***DFRobot Quad Driver Shield and Dual Motor Driver*** were selected for their ability to control multiple motors efficiently, crucial for the movement of the pirate ship. The ***DFRobot DC Geared Motor*** is low cost, 6V rated and provided the necessary torque and precision, making it ideal for this application.

The ***12V DC Linear Actuator*** was chosen due to its low cost compared to the “Actuonix L12”. The ***DFRobot VL53L0X ToF*** sensor was chosen for its easy to install build as it came with a sensor cable.

The ***HM-10 BLE Module*** enabled energy-efficient wireless control and communication, while the ***ADXL345*** accelerometer was preferred because it was already on hand.
The ***TXB0104 Bi-Directional Level Shifter*** was necessary for interfacing the ADXL345 to the Arduino and was chosen because I had it beforehand. The ***Arduino Mega 2560*** was selected for its large number of I/O pins and memory, required to handle multiple transducers. The ***Pololu Step-Down Voltage Regulators*** provides high current output, stable voltage and are more efficient and compact. ***Peerless by Tymphany TC7FD00-04 Speaker*** provided high-quality audio playback.
I preferred the ***Adafruit NeoPixel Ring*** because it was essential for my design, as I planned to mount it tightly around the mast, and the **CNHL 11.1V 3S Lipo Battery** was cheaper than "Zeee 3S Lipo" and ensured sufficient power for all components due to its high capacity and discharge rate. These components were carefully chosen to meet the specific needs of the project, ensuring optimal performance

2.2 Hardware Block Diagram

The microcontroller serves as the brains of this project, controlling the flow of data. The output transducers, sound module and Arduino microcontroller are being powered by the battery, while the input transducers and other components are powered by the Arduino. Below is a block diagram of the main subsystems of this project.

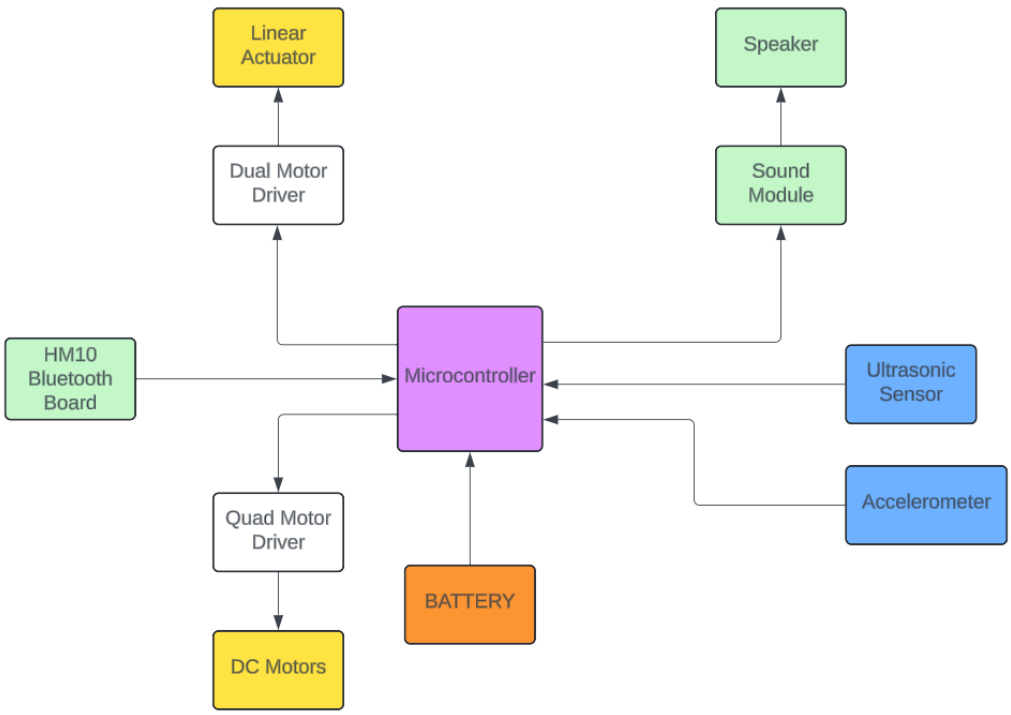

Figure 1: Hardware Block Diagram

3.0 TESTS AND RESULTS

Tests were initially performed on each subsystem of the project to ensure that the project is fully functional after complete integration of each subsystem into one optimal system. In this section, I will discuss how tests were taken on each subsystem and the corresponding results.

3.1 Subsystem 1: Arduino -> Quad Driver -> DC Motors

I connected the Quad Driver Shield to the Arduino Mega 2560 and attached the DC motors to the driver shield **[4]**. Sent PWM and direction (ON/OFF) signals from the Arduino to control motor speed and direction. Motors responded accurately to speed and direction commands, demonstrating smooth and precise movement. I adjusted PWM signal values to ensure smooth motor operation and synchronized movement. Motor performance proved to be consistent across various speeds and directions.

3.2 Subsystem 2: Arduino -> Dual Driver -> Linear Actuator

Using a breadboard, I connected the Dual Motor Driver to the Arduino Mega 2560 and the linear actuator to the driver **[5]**. Using the Arduino I was able to control the direction of the linear actuator – ON would extend it and OFF would retract it. Sending PWM signals controlled the speed at which the actuator extended & retracted. I adjusted control parameters to ensure precise and synchronized movement.

3.3 Subsystem 3: Arduino -> DYHV20T -> Peerless Speaker

The DYHV20T MP3 module was set to USART communication mode and connected to the Arduino Mega 2560 via TX1 and RX1 **[3]**. The Peerless speaker was connected to the audio output of the DYHV20T module. Using the datasheet of the module, I sent commands over the serial line from the Arduino to play, pause, and switch tracks. The module played audio files correctly, and the speaker produced clear and high-quality sound.

3.4 Subsystem 4: ADXL & Level Shifter -> Arduino

I connected the TXB0104 level shifter to the Arduino Mega 2560 and the ADXL345 accelerometer to the level shifter **[1]**. The level shifter helps to convert the 5V logic level from the Arduino to a 3.3V logic level suitable for the accelerometer. Using the provided code from a previous course (ELTR2302) I read the acceleration values from the accelerometer and monitored signal integrity through the level shifter. The accelerometer provided inaccurate readings, while the level shifter maintained signal integrity. Accelerometer proved difficult and time-consuming to calibrate, so I changed its function in the project. That is, as long as the accelerometer read values whether inaccurate or not, the neopixel ring lights up.

3.5 Subsystem 5: VL53L0X ToF -> Arduino

This was the easiest component to test as the connection to the Arduino was very straightforward. With four wires only, the VL53L0X ToF distance sensor connects to the Arduino Mega 2560, and communication is via I2C **[2]**. Using the provided library, I measured distances to various objects using the sensor and it provided accurate and consistent distance measurements.

3.6 Subsystem 6: Arduino -> Adafruit NeoPixel Ring

With only three lines – Power, Ground, and Data input, I connected the NeoPixel ring to the Arduino Mega 2560 and using the Adafruit NeoPixel library I tested various lighting patterns and colors **[6]**. The NeoPixel ring displayed colors and patterns as expected, providing vibrant lighting effects. I then calibrated the NeoPixel colors to match my desired lighting effects.

4.0 CIRCUIT DESIGN

The circuit design of this project is designed to ensure optimal functionality and seamless integration of various electronic components. For a visual reference, refer to the complete circuit diagram in *figure 2* below. This diagram offers a detailed look at our electronic system, showcasing how all the components are interconnected. To make the system work seamlessly, I divided it into three electronic subsystems, each playing a crucial role. These subsystems include output transducers, input transducers/others, and power. At the heart of everything is the Arduino Mega 2560, acting as the central control hub that coordinates all the components. In the upcoming sections, we'll explore each of these subsystems in detail, explaining their individual functions and how they are connected.

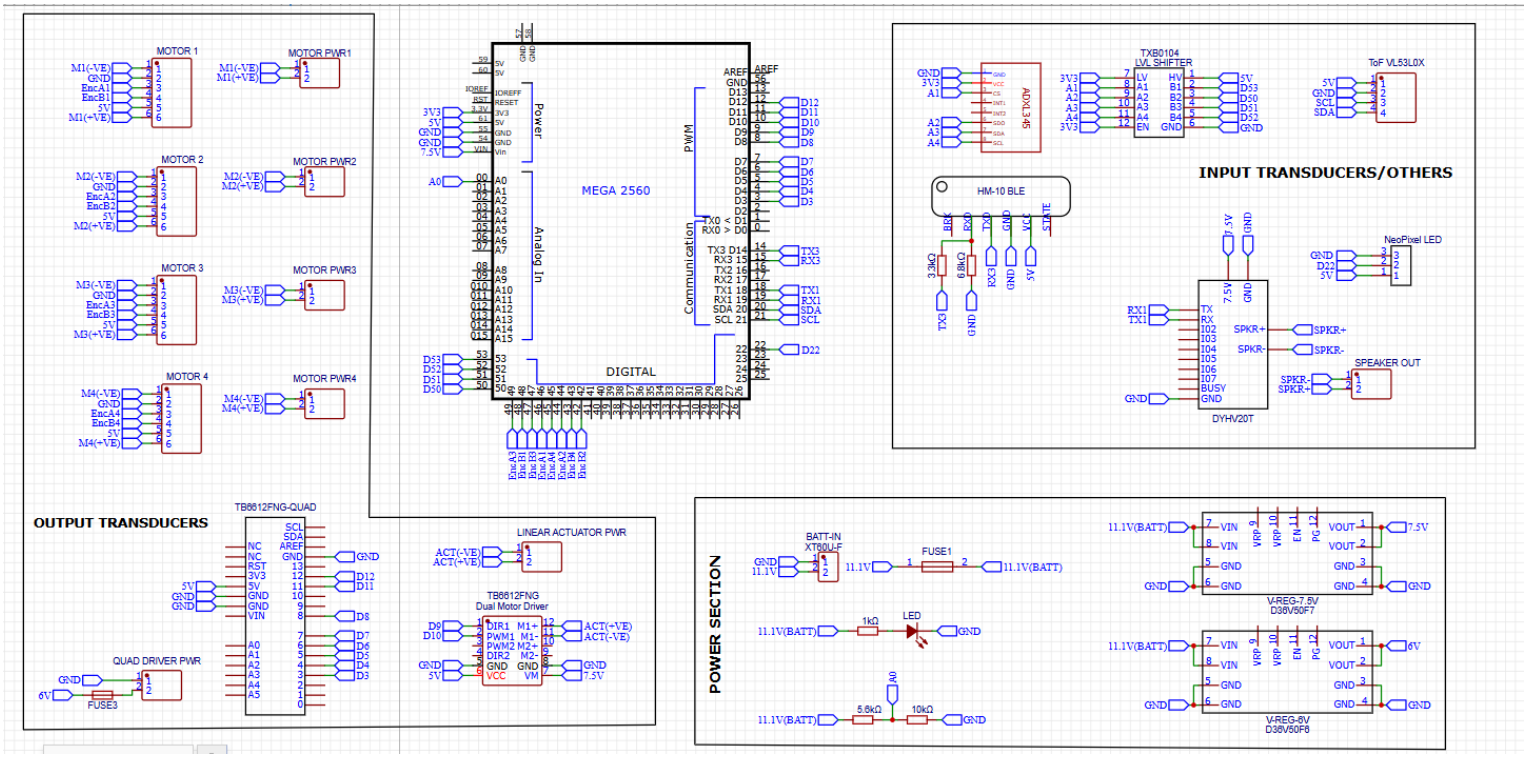

Figure 2: Circuit Diagram

4.1 Output Transducers

Refer to *figure 3* below for the circuit diagram for this subsystem.

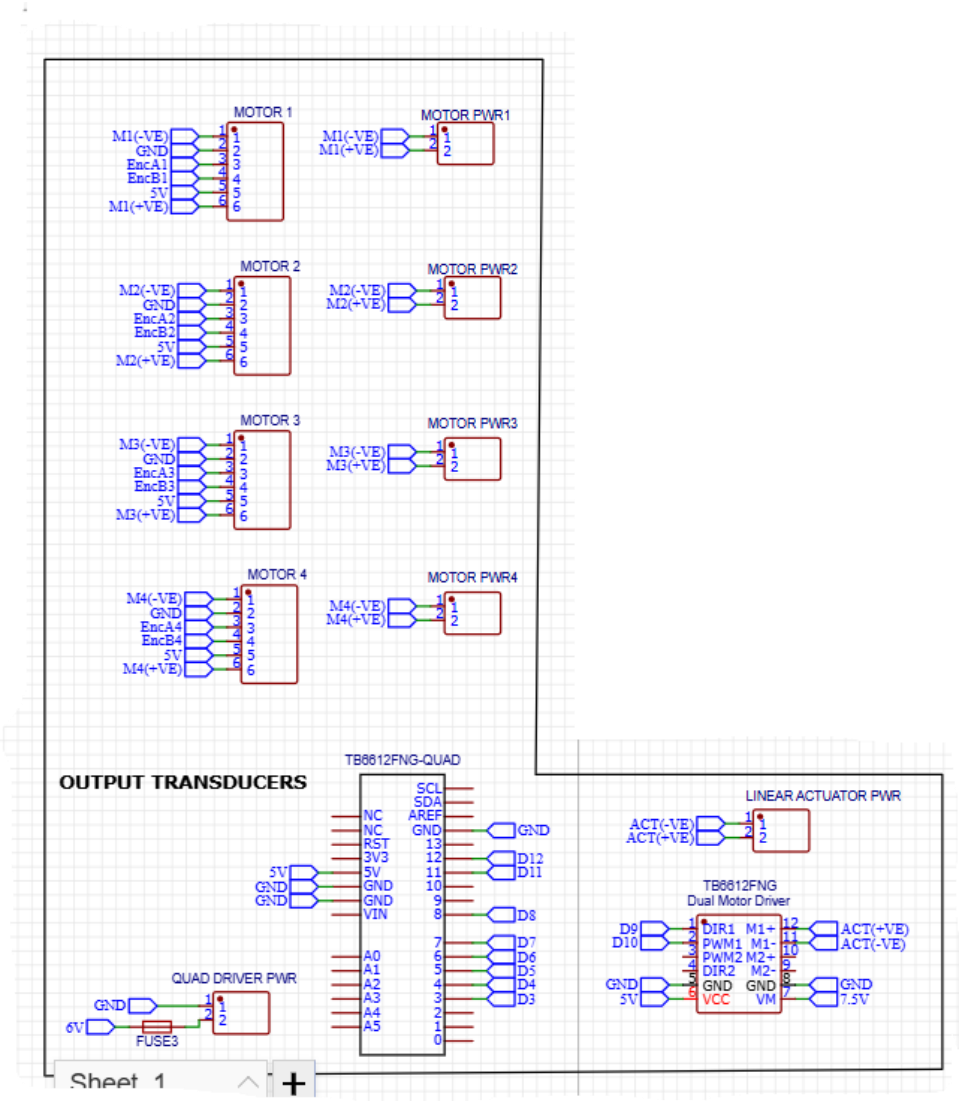

Figure 3: Output Transducer Circuit

DC Motor

The DC motors are equipped with six pin JST-XH connectors consisting of motor power(-VE/+VE), ground, two encoder data lines, and a 5V VCC line. As seen in *figure 3*, I connected the power lines for the motor to a separate two pin JST connector to ensure proper connections to the quad driver, as the quad driver comes with four motor screw terminals **[4].**

Quad Driver

The component is an Arduino mega compatible driver shield. It can drive up to four motors **[4]**. I have the PWM, and direction control pins connected to data I/O pins on the Arduino mega. Taking a closer look at *figure 3*, there is a two pin JST connector named "QUAD DRIVER PWR" it supplies power to the quad driver via a screw terminal and it is being fed by the 6V voltage regulator.

Dual Driver

This compact 2 x 6 pin breakout board drives the linear actuator and is powered by the 7.5V voltage regulator **[5]**. On the driver, direction and PWM pins for M1(motor 1) are connected to data pins on the Arduino, while a two pin JST connector which plugs into the linear actuator is connected to M1 power lines on the driver.

4.2 Input Transducers/Others

This subsystem consists of sensors, the sound module and Bluetooth. See *figure 4* below for the circuit diagram for this subsystem.

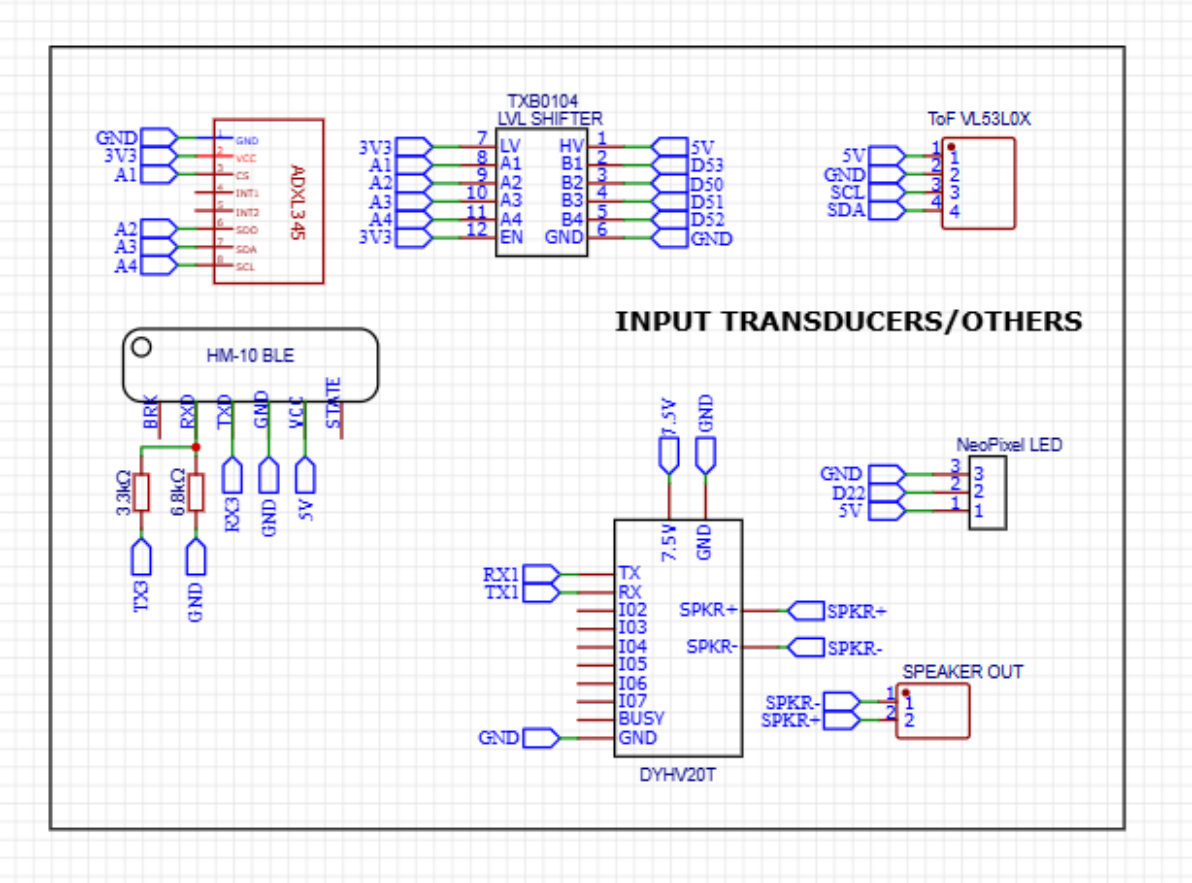

Figure 4: Input Transducer Circuit

VL53L0X TOF Sensor

The sensor comes with a JST-Dupont cable, so I added a JST-XH connector to the circuit to easily plug in the sensor. Communication is simple and via I2C.

TXB0104 Level Shifter

The component acts as an intermediary between the accelerometer and Arduino. Referring to figure 4, Pins A1-A4 correspond to B1-B4, while the enable pin(EN) should be tied HIGH to 3.3V.

ADXL345 Accelerometer

The lower logic-level side of the TXB0104 is connected to the ADXL345, while the higher logic-level side is connected to the Arduino's SPI interface. Refer to *table 2* below for a clearer view of the pin connections between the components. The SPI connection is highlighted in dark grey.

ADXL345	LV	TXB0104	HV	Arduino
GND	3.3V (VCC)		5V (VCC)	
3.3V (VCC)	EN (3.3V)		GND	
CS ->	A1	B1 ->		D53 (CS)
SDO ->	A2	B2 ->		D50 (MISO)
SDA ->	A3	B3 ->		D51 (MOSI)
SCL ->	A4	B4 ->		D52 (SCK)

Table 2: ADXL345 SPI Connection

DYHV20T MP3 Module

This is a 14 pin breakout board being supplied power from the 7.5V voltage regulator. It's communication protocol with the Arduino is USART with TX & RX from the module connected to RX1 & TX1 respectively on the Arduino. The speaker output pins on the module are connected to a two pin JST connector that plugs into the speaker.

HM-10 Bluetooth

Similar to the DYHV20T, the HM-10 interfaces with the Arduino via USART communication. As seen in *figure 4*, the RX pin on the HM-10 is connected to a voltage divider before going into the Arduino's TX3 line and this is because the HM-10 is a 3.3V device. Although the breakout board converts the 5V to a 3.3V, the module's RX pin remains rated for a 3.3V logic level **[7]**.

4.3 Power System

This subsystem delivers power to every component on the circuit. In *figure 5* below, you will get a better view for the circuit diagram for this subsystem.

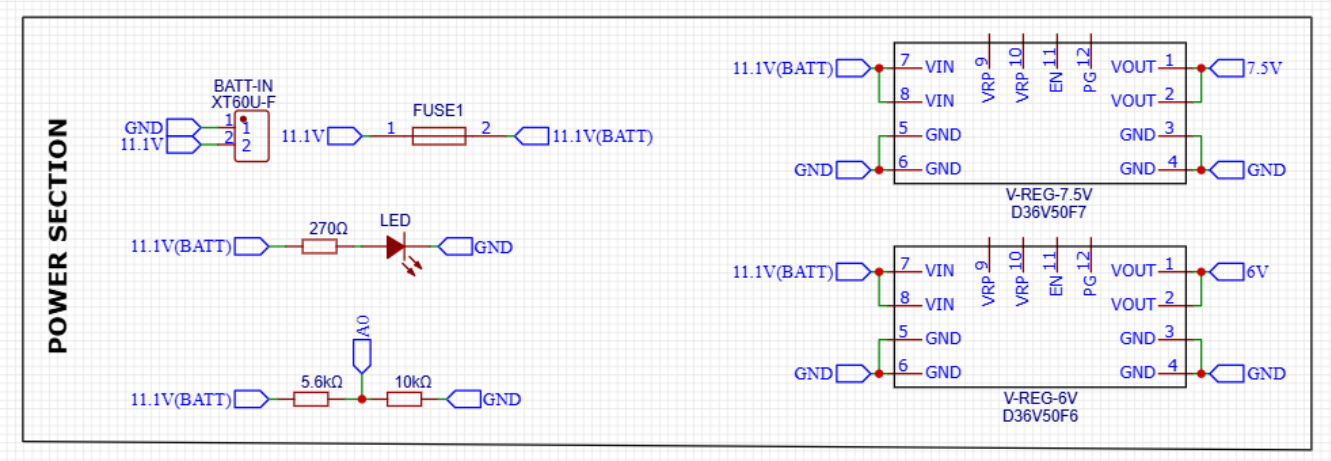

Figure 5: Power System Circuit

CNHL 11.1V Battery

This battery supplies power to all of the components via an XT60 plug. A 10A rated fuse is connected to the battery to protect the PCB traces and components, and the output end of this fuse ”11.1V(BATT)” is what feeds the rest of the power section .

Surface Mount LED

This small circuit is simply a power indicator with a 270 Ohm resistor to limit current going into the LED.

Voltage Regulators

These components are switching regulators and connection to them is very straightforward. They come with multiples of some of its pins and can handle high output currents between 2A - 9A, depending on the input and output voltage **[8]**.

Additionally, you need only three connections namely, VIN(Input Volt), VOUT(Output Volt), and GND. The 6V regulator feeds the quad driver and in turn the DC motors, while the 7.5V regulator feeds the Arduino, dual driver(linear actuator), and sound module.

Voltage Divider Circuit

I created this circuit to monitor the battery voltage. The voltage divider sends a scaled voltage value to analog pin A0 on the Arduino.

Voltage Divider Circuit Design

The battery voltage is **11.1V (nominal)** but can go as high as **12.6V (fully charged)**. The Arduino analog input pins can only handle **0V to 5V**, so you need to divide the battery voltage down to a readable range.

- **Voltage Divider Formula:**
  Vout = Vin * (R2/R1+R2)
- For **12.6V input**, to scale down to 5V:
  - **R1** = 10 kΩ
  - **R2** = 5.6 kΩ
    This will give a maximum output of:
    Vout=12.6 * (5.6/10+5.6) = 4.5V

  This 4.5V is safely within the Arduino's 5V input range, if the battery voltage drops to 11.1V then Vout would be 3.99V. The Arduino will read these scaled voltages.

5.0 PCB DESIGN

The design of the PCB for this project was carried out using EasyEDA, a user-friendly electronic design automation tool. For visuals, please refer to *figures 6, 7, 8 & 9* which includes both the CAD image and an unpopulated image of the PCB. The PCB dimensions are 134.8mm x 184.6mm, providing ample space to accommodate all necessary components while maintaining a compact and efficient layout. Most of the components used are female sockets, JST male connectors, and male headers, with a few surface mount components such as resistors, LEDs, and fuses. To ensure effective grounding, a ground plane is implemented on both the top and bottom layers of the PCB. *Figure 6* is the top part of CAD image. It's in two parts, because the image was too big.

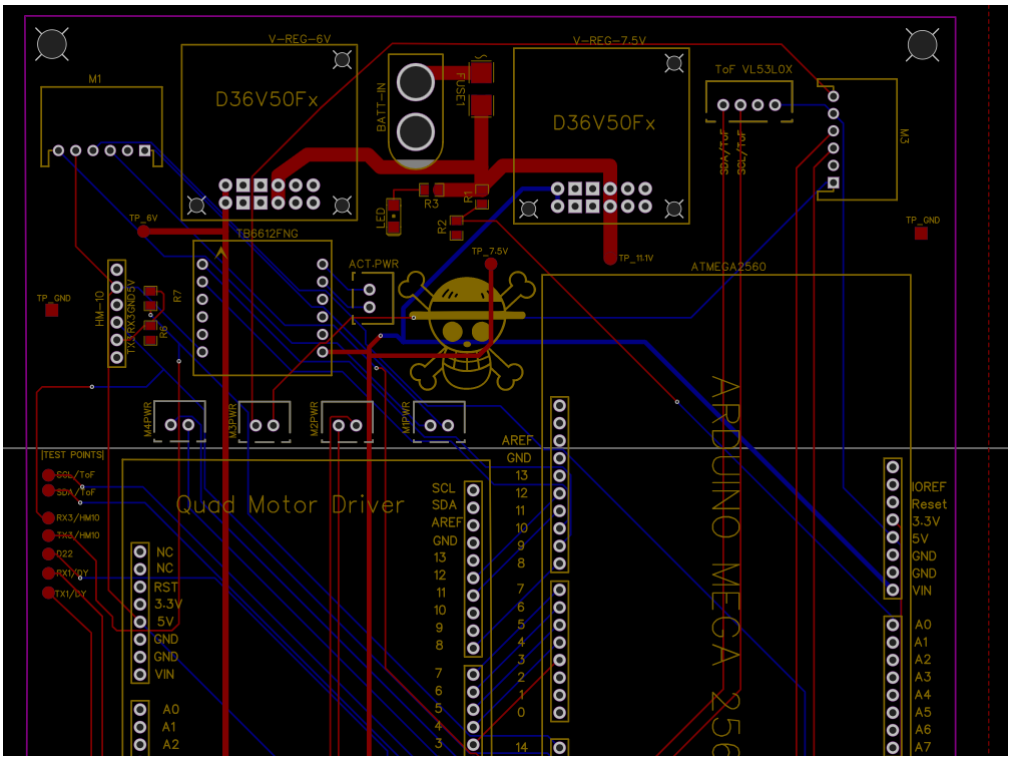

Figure 6: Top part of PCB CAD Image

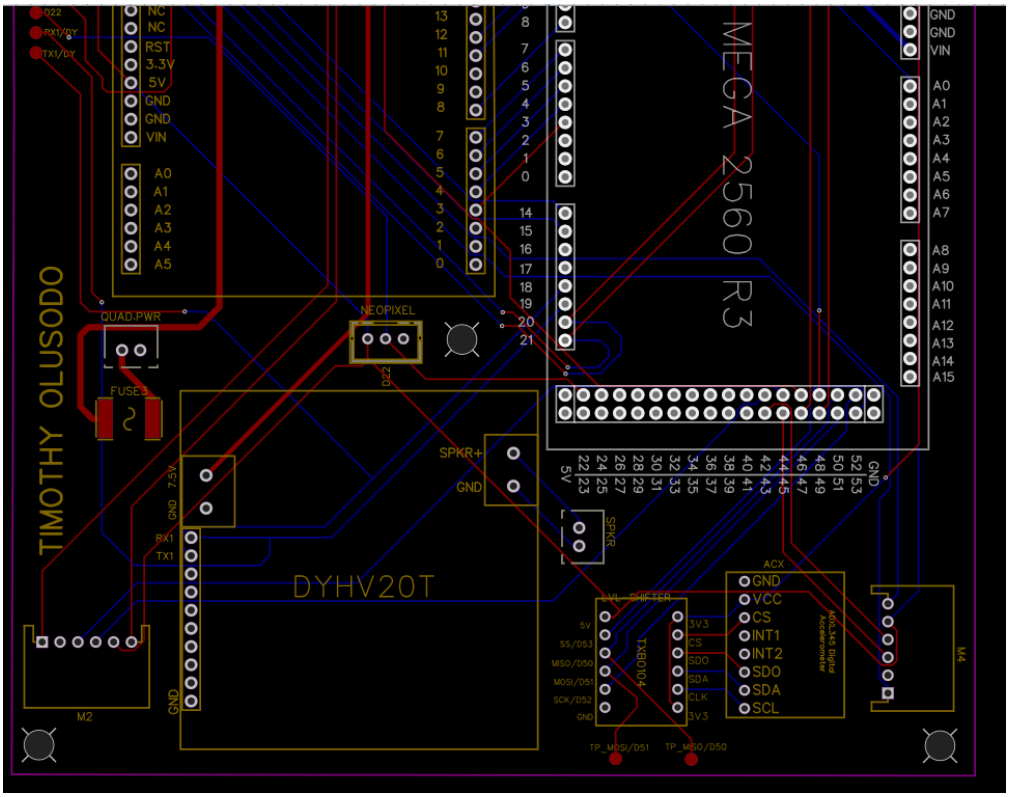

Figure 7: Bottom part of PCB CAD Image

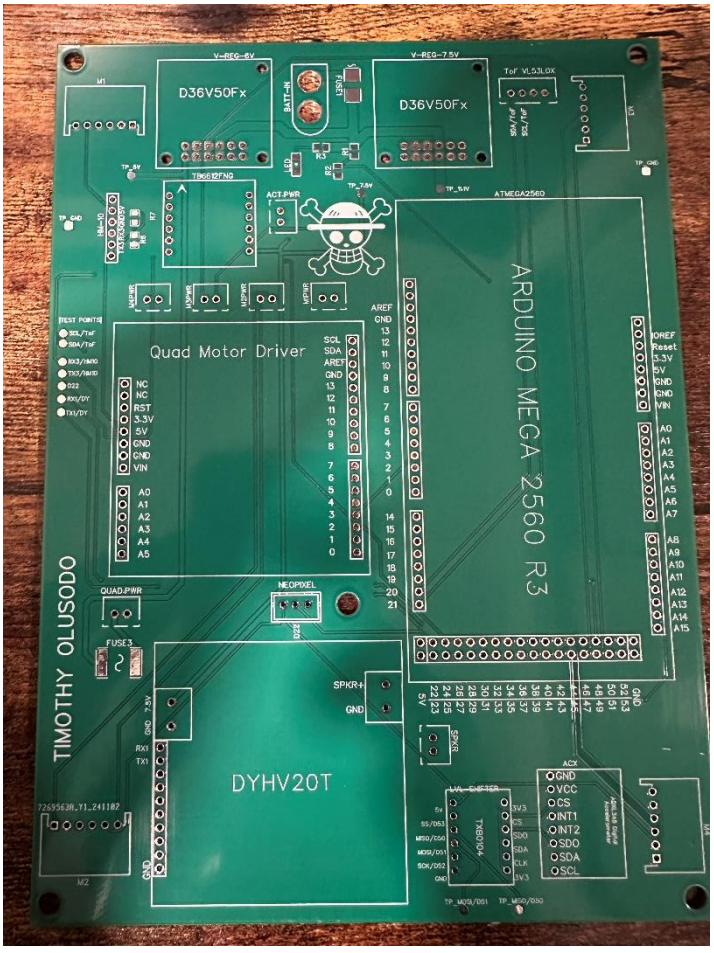

Figure 8: Top Layer of Unpopulated PCB

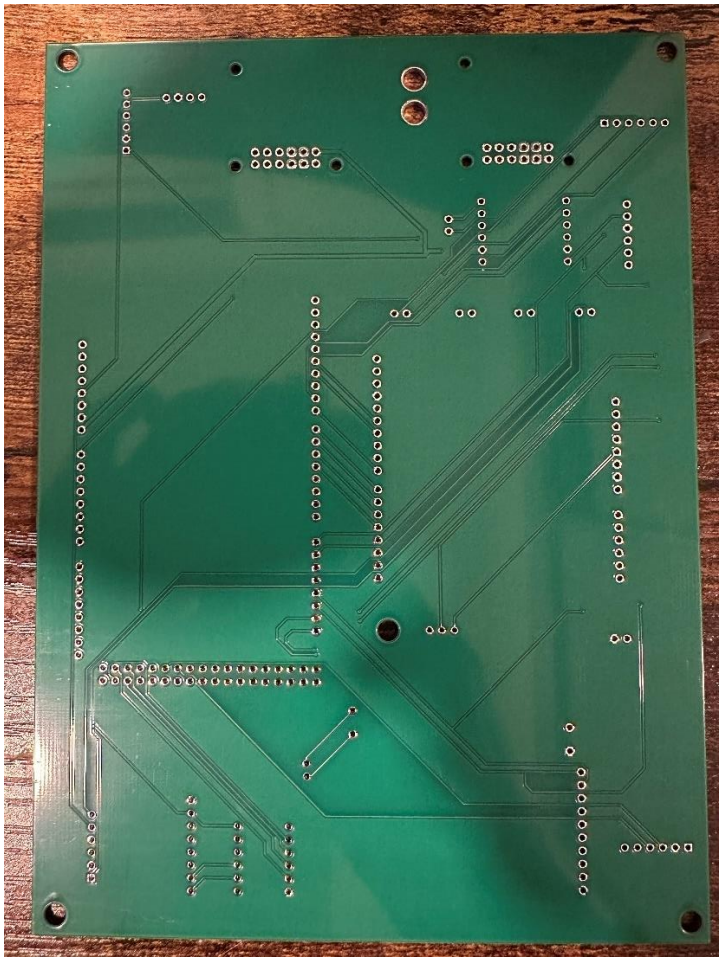

Figure 9: Bottom Layer of Unpopulated PCB

One of the significant design considerations was the use of a 2oz copper layer. This decision was influenced by the need to handle high current, as designing wide traces would interfere with other through-hole pins. The thicker copper layer helps to reduce resistance and improve current handling capacity, which in turn made me design with slimmer traces. I was able to figure out the calculations using Digikey's trace width calculator. The PCB also includes M4 screw holes in each of the four corners, designed to accommodate hexagonal standoffs, which securely mount the PCB onto the chassis.

Several factors influenced the design of the PCB, including component selection, current handling requirements, and mechanical stability. The choice of connectors and headers ensures ease of assembly and reliable connections. The placement of components was optimized to minimize signal interference and improve overall performance. Additionally, adding mounting holes and ground planes enhances the mechanical and electrical integrity of the PCB, providing a solid foundation for the project.

6.0 ELECTRONICS ASSEMBLY

This section is divided into two subsections namely: PCB Population and Chassis Assembly.

6.1 PCB Population

The PCB population process was carried out in a lead-free soldering lab to ensure compliance with environmental standards. The soldering temperature was set to 680°F for optimal results, and a PCB holder stand was used to stabilize the board during the process. The soldering process prioritized simplicity and precision, beginning with larger components and concluding with smaller, more intricate components. Although, soldering the ground pins proved difficult, likely due to the extensive ground plane that dissipates heat quickly, making solder adhesion challenging. *Figure 10* showcases the fully populated PCB.

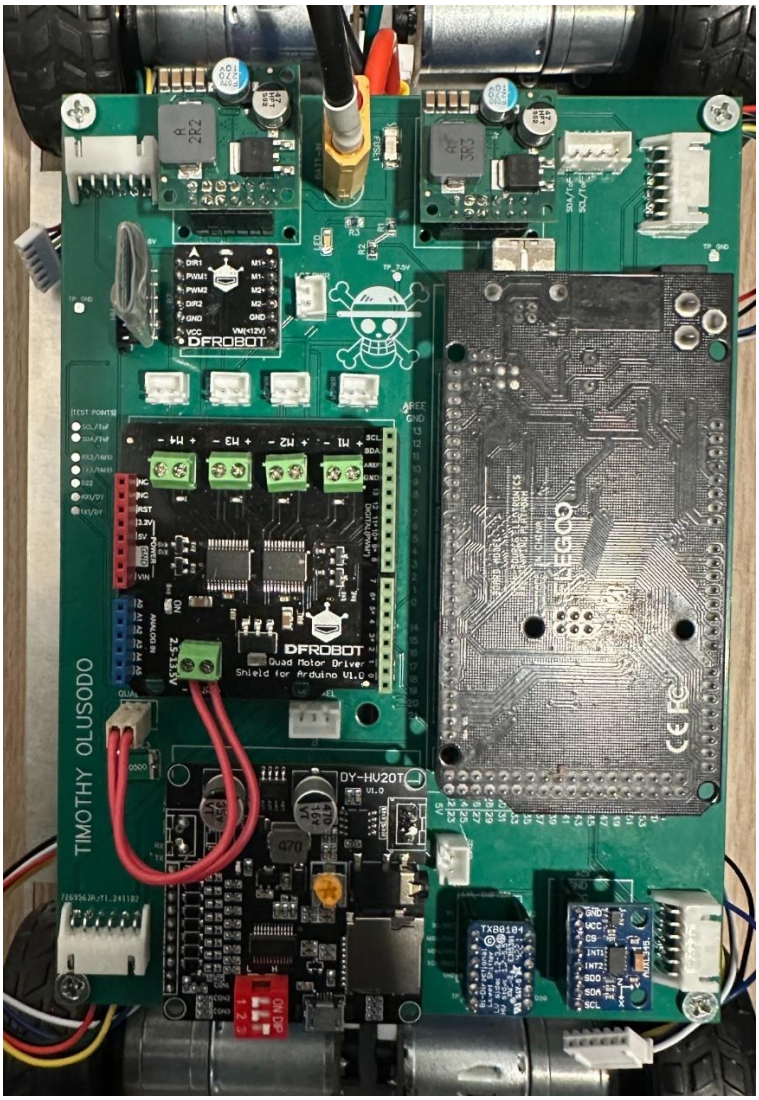

Figure 10: Populated PCB

Modifications to Address Design Errors

Level Shifter (TXB0104)

The pin placement for the level shifter was reversed on the PCB design. To rectify this, I de-soldered the pre-soldered male header pins from the TXB0104 and re-soldered them with the component flipped to match the pin orientation.

Sound Module (DYHV20T)

The power and ground pin placements for the DYHV20T were swapped on the PCB, to rectify this I had to bend the power pins on the DYHV20T module out, so as to route it elsewhere on the PCB. Later the module failed on me, which required a modification to use the DYSV5W sound module as a replacement.

The DYSV5W sound module, rated for 5V, shared a similar pin layout, so I bent its power pins as well. A custom JST connector-to-wire cable was fabricated to power the DYSV5W via the closest power source which is 5V VCC from the quad driver. Refer to *figure 11* for the modified PCB.

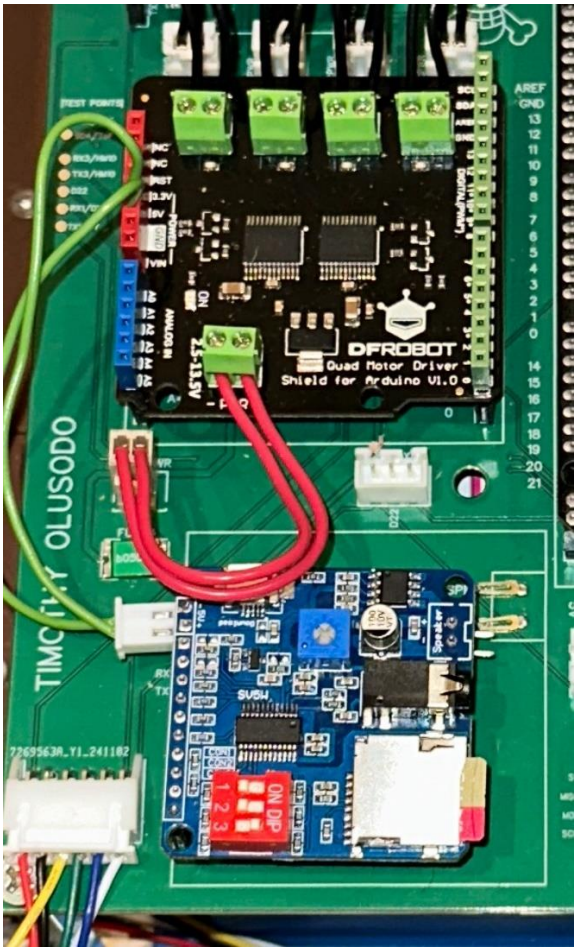

Figure 11: Modified PCB (Sound Module)

6.2 Chassis Assembly

The chassis assembly was designed to provide a solid and organized foundation for the pirate ship. An aluminum laser-cut sheet served as the base, with dimensions designed in Shapr3D and exported as a DXF file for precision laser cutting. A custom-designed battery pack, modeled in Shapr3D and 3D printed, was the first component to be mounted onto the aluminum base, see *figure 12* for the CAD image.

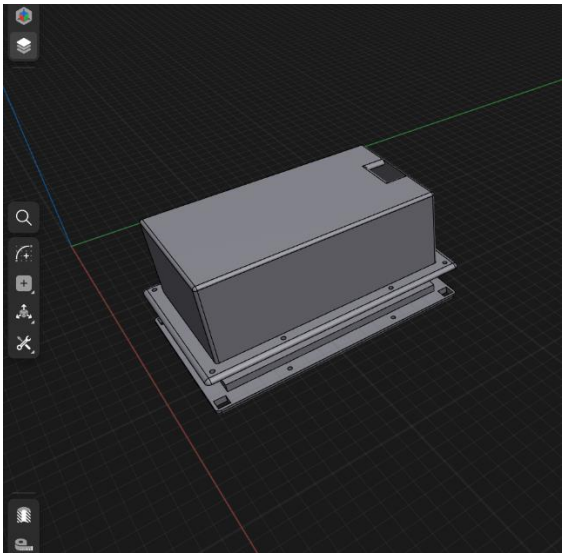

Figure 12: Battery Pack CAD

Motors were securely mounted onto the base using metal brackets, followed by the populated PCB attached using M4 hexagonal standoffs. A switch was installed beneath the metal sheet to control the main circuit power, offering convenient and reliable control (*figure 13*).

Figure 13: Power Switch

All mounting holes, except those for the PCB standoffs, were designed for M3 screws to ensure compatibility and ease of assembly. The overall chassis structure resulted in a sturdy and reliable platform for the ship (*figure 14*).

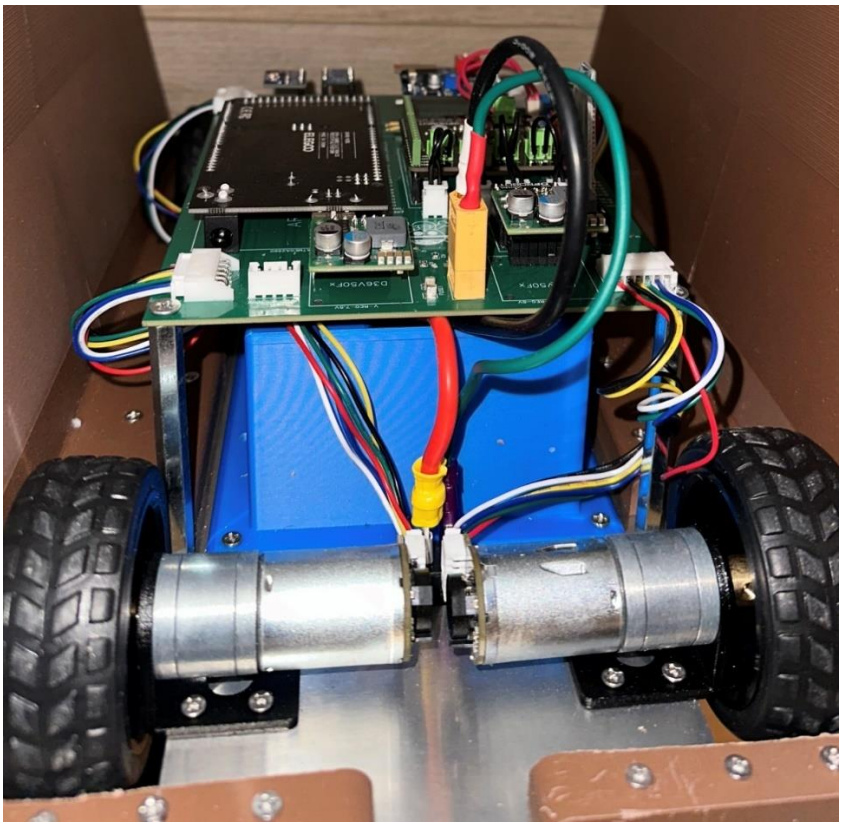

Figure 14: Chassis

7.0 FIRMWARE

The firmware for the pirate ship project was designed to coordinate various subsystems, ensuring smooth communication and operation between components. It integrates six main subsystems: Bluetooth, Time-of-Flight (ToF) sensor, ADXL345 accelerometer with NeoPixel, DC motors, linear actuator, and the sound module.

Main Subsystems

Bluetooth Communication

The Bluetooth subsystem was initialized using the Dabble app library, setting up a serial communication line at 115200 baud. This provided a reliable connection for transmitting directional and control inputs from the app to the Arduino.

ToF Sensor (VL53L0X)

Programmed using DFRobot's VL53L0X library, the sensor was configured in the setup function to operate in continuous and high-precision mode. In the loop, the sensor's distance readings controlled the ship's

movement. The ship halted whenever an obstacle was detected within 150 mm and only reversed until the obstacle was cleared. This ensured collision avoidance.

ADXL345 Accelerometer & NeoPixel

Due to unstable and inaccurate accelerometer readings, the subsystem was simplified. The NeoPixel LED ring, programmed using Adafruit's NeoPixel library, displayed a "snake" lighting animation whenever the accelerometer detected motion. The accelerometer was configured via SPI in the setup, and the NeoPixel brightness and initialization settings were also configured here.

DC Motors

The motors' PWM and direction pins were initialized and set as outputs. Functions were created for forward, backward, left, right, and stop movements. In the loop, these functions were activated based on directional button presses received via Bluetooth from the Dabble app.

Linear Actuator

This subsystem required no custom functions. PWM and direction pins were initialized and set as outputs. The loop used Boolean flags to track the actuator's position. The actuator extended or retracted based on the triangle button's state, toggling the direction pin accordingly.

Sound Module

This module was physically configured to USART mode via a red box on the board. A set of commands can control music playback. Using the datasheet, functions were created for play, pause, next/previous track, volume adjustment, and setting playback mode. Volume and playback mode were configured in the setup function, and playback functions were triggered in the loop via button presses from the Dabble app.

Order of Operations

Initialization/Configuration

All subsystems were initialized and in the setup function they were configured, including SPI and serial communication, libraries, and component-specific configurations such as NeoPixel brightness and ToF sensor mode.

Loop Function

Bluetooth commands were continuously read and processed.
Sensor data (ToF and accelerometer) influenced motor and NeoPixel behaviors.
Subsystems like the linear actuator and sound module responded to button presses in real time.

Firmware Optimization

Code Modularization

Functions were created for repetitive tasks, such as motor movement and sound module commands, reducing code redundancy and improving readability. Modularized code made it easier to identify and resolve issues during testing.

Efficient Sensor Handling and Timing

The ToF sensor was set to continuous mode, reducing the overhead of repeatedly initiating measurements. Millis() functions were used instead of delay functions to ensure efficient and reliable timing that does not interfere with other blocks of the code.

Simplified Accelerometer Integration

Due to unstable readings, accelerometer functionality was reduced to a trigger for NeoPixel lighting animations rather than precise control of the neopixel. This simplified the logic and improved reliability.

8.0 PRJOECT ENCLOSURE BUILD

The design of the project was inspired by the Going Merry, the iconic ship from the One Piece anime. While maintaining the essence of the original design, I opted for a simplified version to ensure practicality and ease of construction. The final design balances functionality with visual appeal, featuring smooth curves and distinct details that give it a recognizable yet unique appearance. The front and rear sections of the ship are designed with gentle slopes, creating a cohesive and streamlined look (see *Figure 15*). The deck and assorted parts, such as the mast, railings, and crow's nest, add intricate details, enhancing the model's overall visual impact.

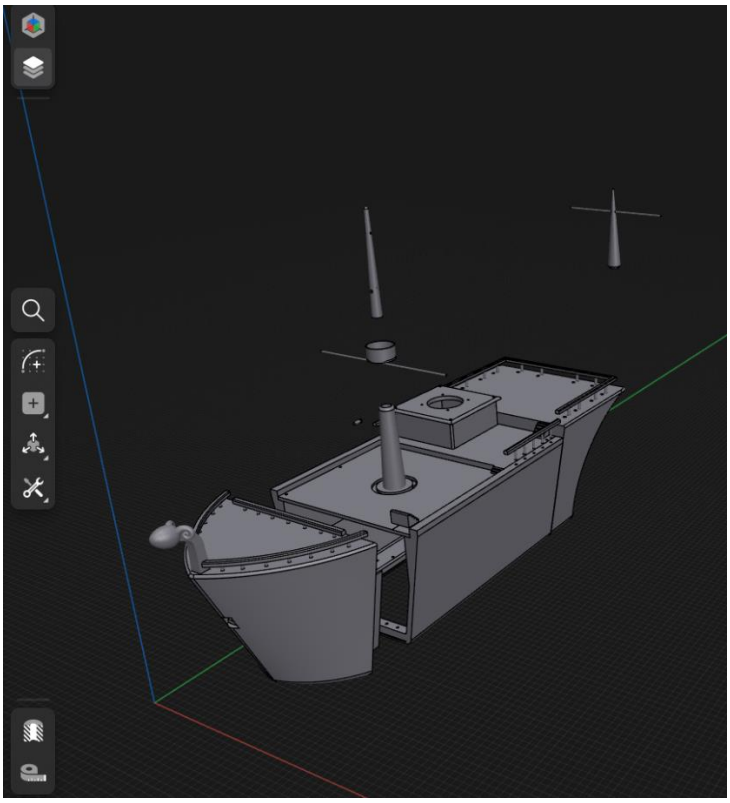

Figure 15: Full Enclosure CAD

The entire enclosure for the ship was custom designed using Shapr3D and fabricated through 3D printing. This approach allowed for complete control over the dimensions and details, ensuring all components fit together seamlessly. The enclosure is divided into three main categories: the front and rear of the ship, the body of the ship, and assorted smaller parts. To accommodate the size limitations of the 3D printer, larger parts of the ship were sliced in half. Each part was printed individually and later assembled.

Front and Rear of the Ship

The front and rear sections were the first parts to be designed. Their dimensions served as the foundation for subsequent components, ensuring consistency throughout the model. The loft tool in Shapr3D was used to create the smooth, flowing contours of these sections, which were then tweaked to achieve the desired look (see F*igure 16*). The shell tool was applied to make these parts hollow (see *Figure 17*).

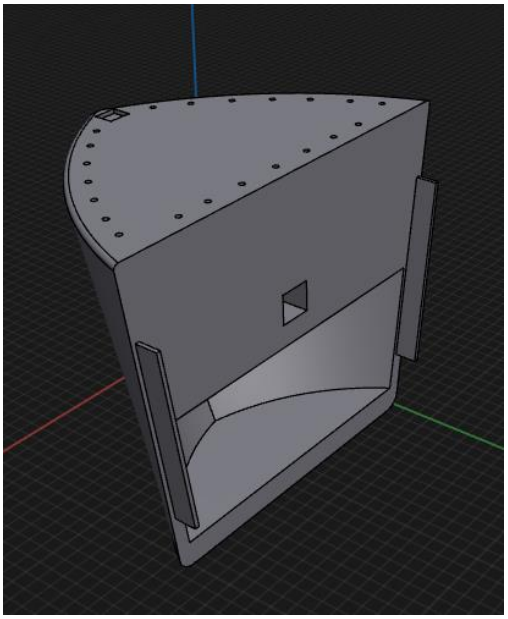

Figure 16: Front of the Ship CAD

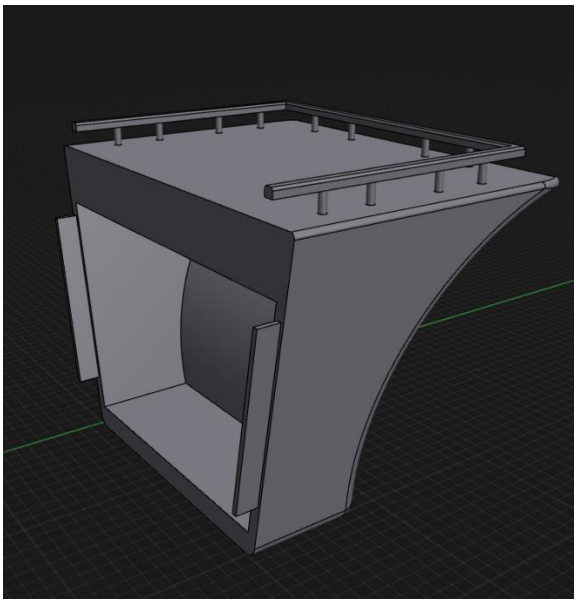

Figure 17: Rear of the Ship

Body of the Ship

The body's sides were designed with precision to align perfectly with the front and rear sections. The bottom part of the sides was tailored to fit onto the metal sheet, which acts as the structural base. The deck was designed as a flat surface with provisions for mounting various components, including a dedicated housing for the Peerless 20W speaker. The speaker housing was custom sized to securely hold the speaker in place while maintaining the ship's overall proportions (see *Figure 18*).

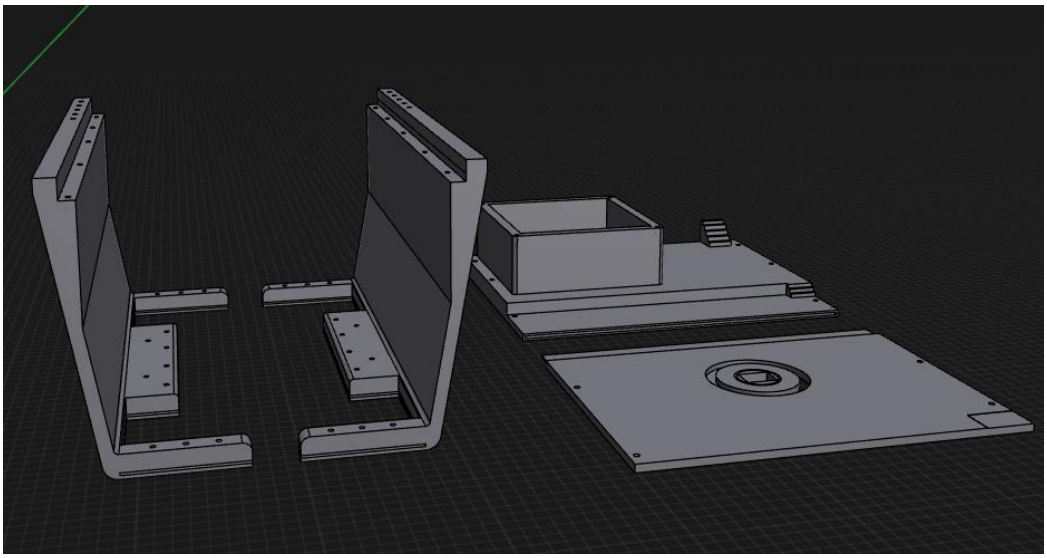

Figure 18: Body of the Ship CAD

Assorted Parts

The assorted parts, including the mast, crow's nest, railings, and figurehead, add character and functionality to the model. These smaller components required extra attention to detail, as they contributed to the ship's aesthetic appeal. The main mast's upper section was designed to attach to the linear actuator, enabling dynamic mast control (see *Figure 19*).

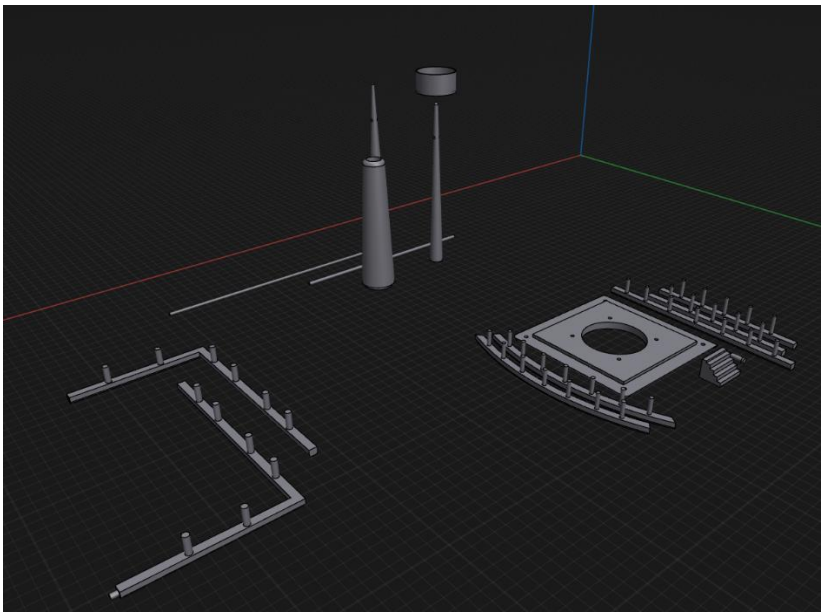

Figure 19: Assorted Parts CAD

9.0 FINAL BUILD

The assembly process involved combining all the 3D-printed parts around the metal chassis to create a sturdy structure. M3 screws were used to fasten the sides securely to the sheet metal base (see *Figure 20*).

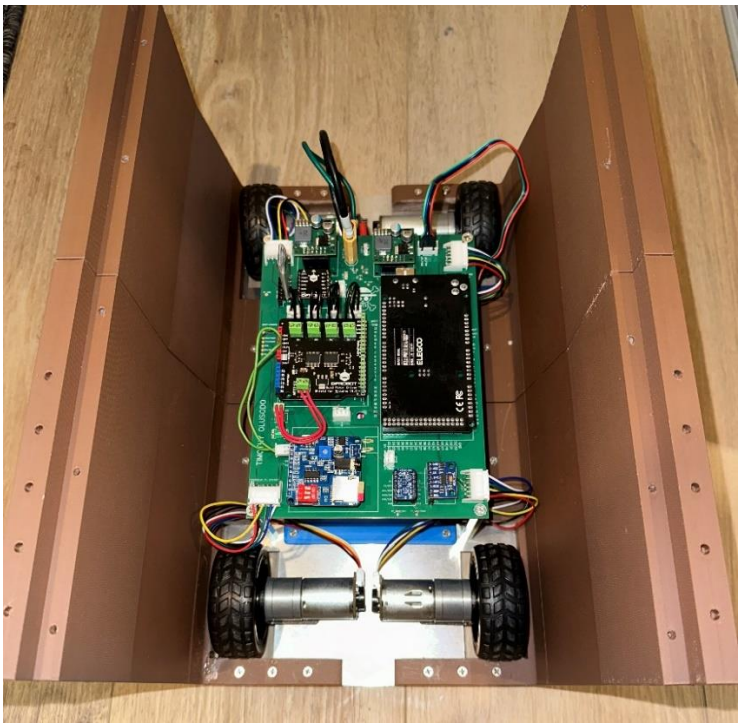

Figure 20: Final Build (SIDES)

The linear actuator was mounted onto the deck, providing the necessary functionality for controlling the mast. The attachment was carefully aligned to ensure smooth operation (see *Figure 21*). Following this, the front and rear parts of the ship were assembled and fastened to the sides, creating a complete frame for the enclosure. The deck was then securely placed and attached, forming the upper surface of the ship.

Figure 21: Deck and Actuator Build

Finally, all smaller components, such as the mast, crow's nest, railings, and figurehead, were installed, adding the finishing touches to the model. *Figure 22* provides a top view of the fully assembled enclosure.

Figure 22: Top View of Assembled Build

The final build after attaching the assorted parts and painting can be seen below.

Figure 23: Project's Assembled Build

10.0 GRAPHICAL USER INTERFACE (GUI)

I used Dabble, an existing software, to interface with the HM-10 Bluetooth module and serve as the graphical user interface for this project. On the landing page, you are presented with various modules. I employed the gamepad module to manage the project's functionality. Please refer to a screenshot of the UI below (*Figure 24).*

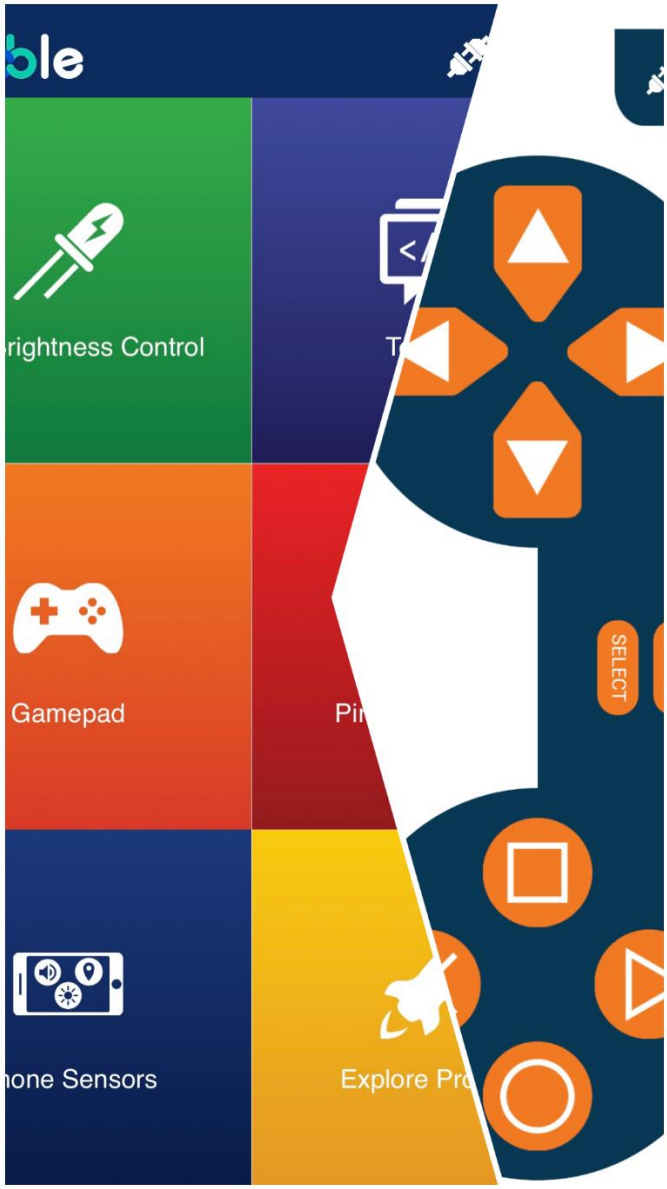

Figure 24: Dabble App: Landing Page and Gamepad Module

11.0 CONCLUSION AND FUTURE PLANNING

The project was initiated with the goal of creating a functional and aesthetically pleasing RC pirate ship that integrates various transducers, actuators, and modern electronics to showcase advanced control mechanisms.

The solution was approached through meticulous design, testing, and iterative improvement. The enclosure was designed in Shapr3D, 3D printed, and assembled around a metal chassis, while the electronics were implemented using an Arduino Mega2560 as the central controller. Multiple subsystems were developed and tested, including:

- DC motor control via the Quad Driver Shield (Subsystem 1).
- Linear actuator control via the Dual Driver (Subsystem 2).
- Audio output through the DYHV20T sound module and Peerless speaker (Subsystem 3).
- Distance sensing using the VL53L0X ToF sensor (Subsystem 5).
- LED control using the NeoPixel ring (Subsystem 6).

The results demonstrated that the project satisfied most of its intended objectives. The ship's control systems performed reliably, and the enclosure successfully balanced aesthetics and functionality. However, a few shortcomings were encountered:

- The use of large components resulted in a bulky PCB, affecting the overall build size.
- The large battery added significant weight.
- Minor PCB errors, such as pin misalignments, required additional troubleshooting and changes which brought about unexpected behavior.
- Weak 3D-printed assorted parts occasionally affected structural stability.
- The HM10 Bluetooth module exhibited unreliable connectivity.

These issues were unexpected but provided valuable insights into areas for improvement.

FUTURE PLANNING

If a Version 2.0 of the project were to be developed, several modifications and improvements would be made to address the issues and enhance the overall design:

Modifications

Redesign the PCB: Optimize the layout to reduce the overall size and improve pin alignment, ensuring better compatibility with components.
Replace the HM10 Bluetooth Module: Use a more reliable Bluetooth module to improve connectivity.
Reinforce 3D-Printed Parts: Print the assorted parts with higher infill density or use alternative materials like PETG or resin for increased strength and durability.
Optimize Power Management: Consider using a more compact battery with a higher energy density to reduce weight without compromising performance.
Streamline the Enclosure: Redesign the enclosure to reduce the overall size while maintaining its visual appeal and increasing strength.
Improve PCB Population: Use a more organized approach for PCB assembly to minimize errors.

REFERENCES

[1] SFUptownMaker, “TXB0104 level shifter Hookup Guide,” TXB0104 Level Shifter Hookup Guide - SparkFun Learn, https://learn.sparkfun.com/tutorials/txb0104-level-shifter-hookup-guide/all (accessed Dec. 15, 2024).

[2] DFRobot, “Gravity__vl53l0x_tof_laser_range_finder_sku_sen0245,” DFRobot, https://wiki.dfrobot.com/Gravity__VL53L0X_ToF_Laser_Range_Finder_SKU_SEN0245 (accessed Dec. 15, 2024).

[3] “DY-HV20T high power digital Amplifier Module TF CARD SOCKET MONO 20W 8ohm MP3 WAV UART controller for Arduino,” ICStation, https://www.icstation.com/voice-playback-module-hv20t-high-power-digital-amplifier-module-card-socket-mono-8ohm-uart-controller-arduino-p-14437.html (accessed Dec. 15, 2024).

[4] “Quad_motor_driver_shield_for_arduino_sku_dri0039,” DFRobot, https://wiki.dfrobot.com/Quad_Motor_Driver_Shield_for_Arduino_SKU_DRI0039 (accessed Dec. 15, 2024).

[5] “2X1.2a DC motor driver (TB6612FNG) wiki,” DFRobot, https://wiki.dfrobot.com/2x1.2A_DC_Motor_Driver__TB6612FNG__SKU__DRI0044 (accessed Dec. 15, 2024).

[6] Adafruit, “Adafruit/ADAFRUIT_NEOPIXEL: Arduino Library for controlling single wire led pixels (Neopixel, WS2812, etc..),” GitHub, https://github.com/adafruit/Adafruit_NeoPixel (accessed Dec. 15, 2024).

[7] “HM-10 Bluetooth 4 BLE Modules,” Martyn Currey, https://www.martyncurrey.com/hm-10-bluetooth-4ble-modules/ (accessed Dec. 15, 2024).

[8] “Pololu - 6V, 5.5a step-down voltage regulator D36V50F6,” Pololu Robotics & Electronics, https://www.pololu.com/product/4092 (accessed Dec. 15, 2024).