Team 521 Operation Manual – RoboBoat

Sophia Barron, Ivanna Caballero, Michael Fitzsimmons, Andly Jean, Lucca Meyer, Nicholas Norwood, Makenzie Wiggins

FAMU-FSU College of Engineering

Project Overview

1.1.1 Project Description

The objective of this project is to design, build, and program an autonomous surface vehicle (ASV) capable of completing tasks in navigation, detection, and object avoidance, while prioritizing a robust safety system. The current team will establish a foundation for future ASV development, so future teams can build upon and enhance the capabilities of the vessel.

1.1.2 Key Goals

The primary goal for the RoboBoat team is to design and develop an autonomous surface vessel that will serve as a strong foundation for future RoboBoat teams. To meet this goal, the team aims to achieve the following key goals: have a reliable safety system, implement a modular design and code architecture, achieve accurate navigation capabilities, and design a system around modular components. The safety system will be achieved by incorporating fail-safe mechanisms and emergency protocol to ensure safe vessel operations. An accurate navigation system will enable precise waypoint navigation in open water. Implementing a modular code architecture will facilitate ease of maintenance to allow for easy troubleshooting. This approach will also simplify integration of new features for future development. Moreover, designing the system around modular components will enhance flexibility and scalability, enabling easy replacement and upgrade of individual subsystems while maintaining overall system functionality and performance.

Key characteristics of this design include having a robust safety system and maintaining the entire maritime system below 140 lbs. while fitting within six feet, by three feet, by three feet "box". The system will also have a rechargeable power source with battery lifetime of at least 30 minutes to complete the tasks.

1.1.3 Assumptions

The assumptions made for this project provide a basis for decision-making and planning to simplify the design process for the team. Previous RoboBoat evidence manuals will be available as a point of reference for this year's team and future teams. It is assumed that the vehicle will meet the expected weight and size restraints given by RoboNation of less than 140 lbs. and six feet length, by three feet height, by three feet width, respectively. The RoboBoat must also comply with this years and future year's rules and regulations to ensure fair competition and safety standards are met. The weather conditions at the time of the competition will be beyond the teams' control and therefore, all weather conditions must be accounted for. There will be a required safety inspection at the competition, meaning that the system must implement two kill switch systems: one hard-wired emergency stop directly on-board the vessel that is accessible from every angle and one wireless remote emergency stop, located off-board on a RC controller. Finally, only one task is required to be able to compete at the competition.

Module Description

2.1.1 Overall Design

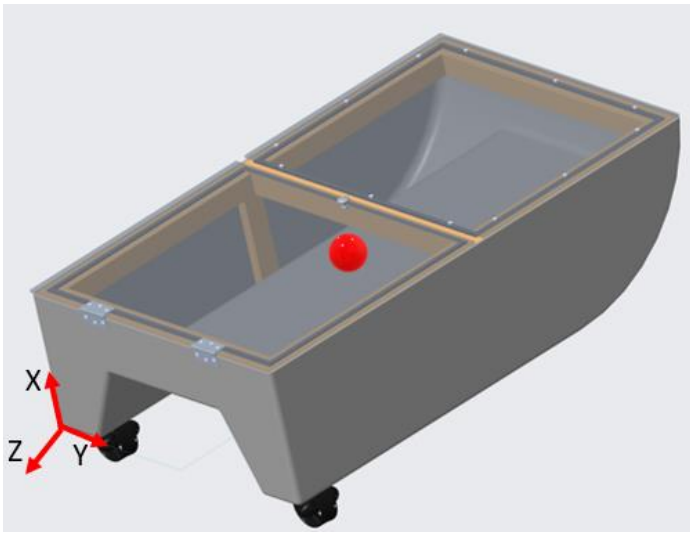


Figure 1: Center of mass

The center of mass for the hull, frame, and thruster is shown in Figure 1. The center of mass was calculated using Creo Parametric. As components are added to the hull, the center of mass will need to be recalculated. The center of mass is important for the stability of the boat as well as being the position that an IMU will be mounted, if used.

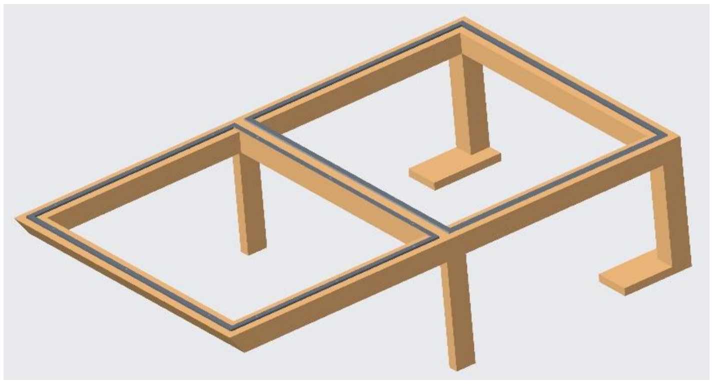


Figure 2: Frame Design

The frame was designed to reinforce the fiberglass hull. The frame consists of 2” x 2” wood screwed together to create a framework throughout the hull. Two wooden feet were added to the rear of the frame in order for the thrusters to be attached to the frame to make them sturdier. Weather stripping is added to the perimeter of where the Plexi-glass deck is attached resulting in water-sealed interior of the hull, where all the electronics will be housed.

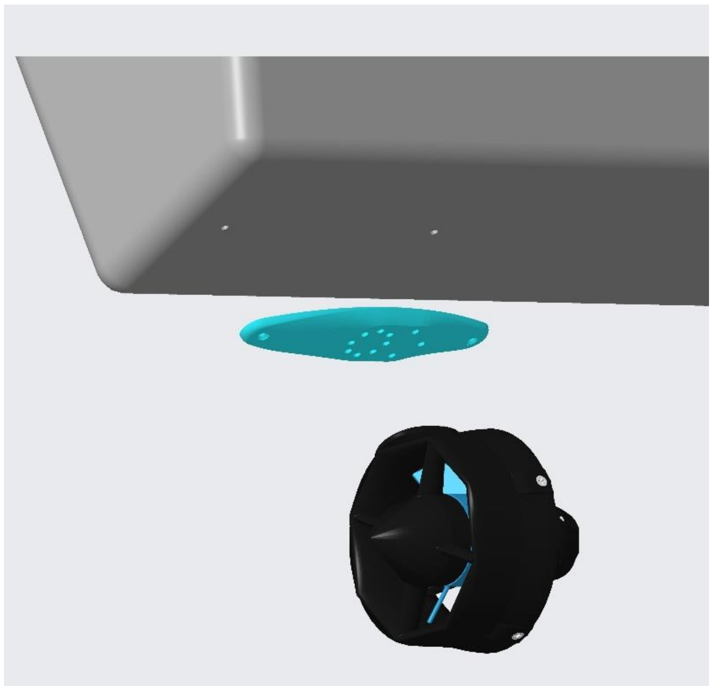


Figure 3: Thruster Assembly

The thruster is first attached at the desired angle orientation with screws to the thruster mount (shown in blue in Figure 3). The thruster mounts have an array of holes, where the thruster can be angled if needed. The thruster mount is attached through the hull and screwed into the feet of the wooden frame structural support shown in Figure 2. The electrical wires coming from the thrusters are routed up through the hull to be connected to the other electrical components.

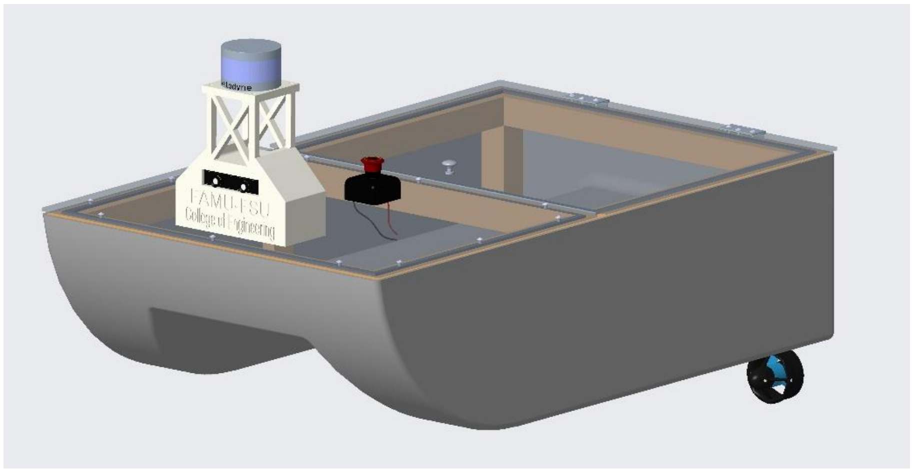


Figure 4: CAD Design

The overall CAD design of the RoboBoat shown in Figure 1, includes:

- Fiberglass hull (gifted from FSU – Panama City Campus)
- Structural wooden support frame
- Two Blue Robotics T200 Thrusters
- Two thruster mounts
- Velodyne LiDAR
- ZED Mini Camera
- Emergency Kill Switch
    - Located at a point accessible from any angle.

- Plexiglass deck
    - Front lid bolted down to frame for stability for LiDAR and camera.
    - Hinged back lid for easy access to electrical components.
- Weather stripping
- 3D Printed Camera and LiDAR mount (material: Sunlu Black PLA)

The 3D printed camera and LiDAR mount were created where both the camera and LiDAR would be elevated. It was mounted on the front of the boat for the optimal range of view.

Integration

3.1.1 Electronic Components

The following components were used to assemble the full electrical circuit of the RoboBoat:

- 14.8V Lipo Battery: Turnigy 12-24C 20Ah
- Fuse Panel: EGIS Fuse Block, Split Buss, 13 Circuit + Ground, 200 A, 12/24V
- Boost Converter: Vikye 1500W 30A Power Source Module
- Voltage Regulator: Switch Electronics 8V 1.5A ST
- Emergency Stop System: FOSHIO 24VDC 120A 4pin Relay Split Charge & OTTO Pushbutton Switches 10A

The battery is connected to the fuse panel through the emergency stop system which then distributes power across the thrusters, electronic speed controllers, boost converter, Arduino Megas, CANBUS line, and remote receiver. The fuse panel utilizes three 20A fuses and one 10A fuse. The DC-to-DC boost converter can boost any input voltage between 5V-30V and output a higher voltage value up to 90V. In this case, it takes the received voltage from its fuse panel connection, and it jumps the output voltage up to 19.5V. This will power the LiDAR and the LED light controller in the boat without having to power them using a separate system. By use of variable resistors on the boost converter, the output voltage and current are altered to convert 14.8V to 19.5V. The voltage regulator is used to power the Arduino Megas. Since they cannot take in a voltage of 14.8V from the fuse panel, voltage regulators regulate the voltage to 8V, so that the Arduinos can be powered efficiently and in a safe input voltage range. The emergency stop system is a requirement in order to compete in the competition per the Roboboat guidelines. The requirement is that the overall system must have a remote kill switch and a physical, onboard kill switch. The emergency stop system implemented below consists of a relay which is controlled by both a transistor connected to the pArduino and the button for the physical kill switch. By use of the relay, we are able to implement low current control features while opening or closing the circuit at the battery's high current terminal. The button is placed on top of the boat for easy access, so that it can be pressed at any time to kill the battery. The wiring diagrams displayed below demonstrate the physical electronic components for the Roboboat.

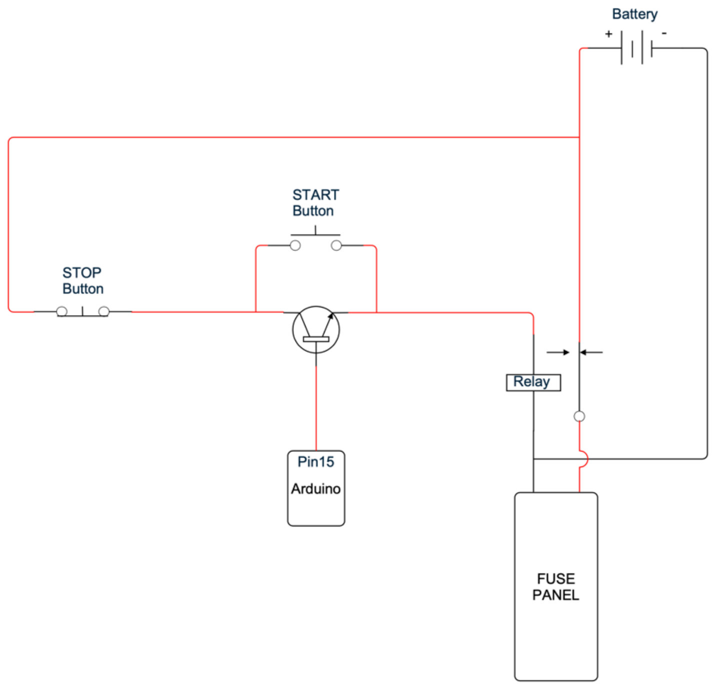


Figure 5: Emergency Stop System

In the Emergency Stop system, there are two buttons, a transistor, and a relay. If the normally closed Stop button is pressed, the relay control circuit will open, and open the main battery circuit. If the Arduino sends a low signal to the transistor, it will open, and open the main battery circuit. Because the Arduino output will stay low while the main battery circuit is open, the Start button must be pushed to allow current to flow through the relay, and therefore allow the Arduino to send a high signal to the transistor.

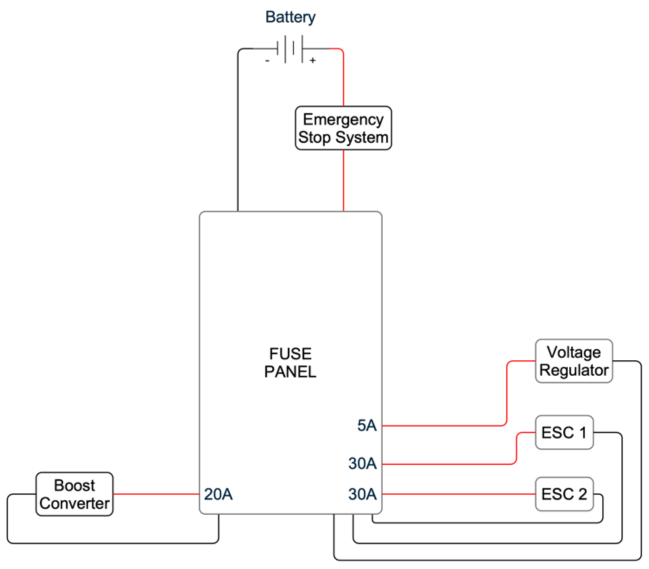


Figure 6: Fuse Panel

The fuse panel will act as an over-current safety device to protect all components from damage. As of now, only four components are connected to the fuse panel, but it allows for up to 13 outputs and has options to easily integrate extra fuse panels to increase number of outputs. Each connection has a place holder for standard mini car fuses.

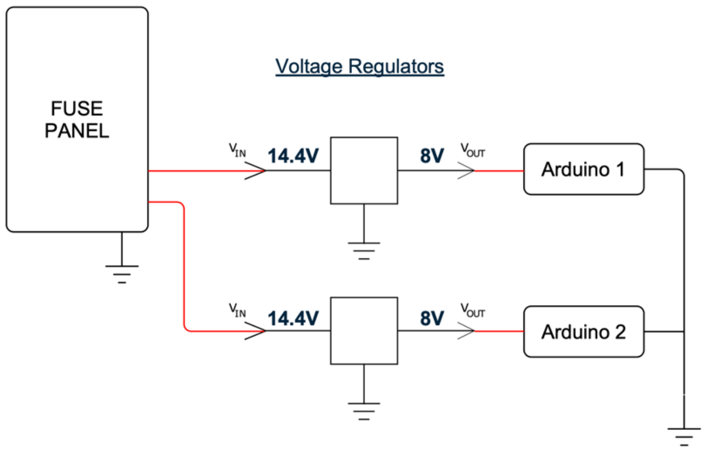


Figure 7: Voltage Regulators

Each voltage regulator takes the battery input, after the fuse panel, connects to ground through the middle pin, and outputs a regulated 8V on the outside. Currently one regulator will be used for each Arduino as they can only handle up to 1A, but a higher current regulator could be used to feed multiple controllers with one component.

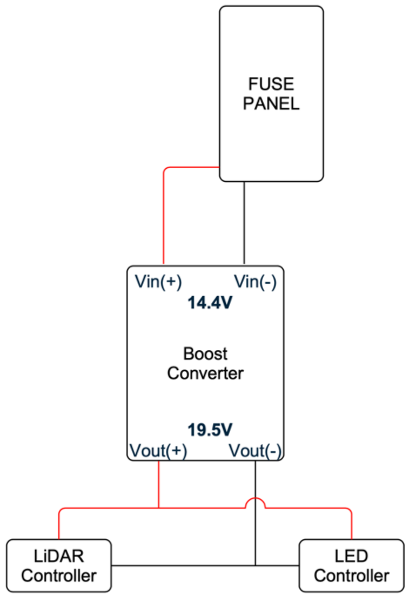
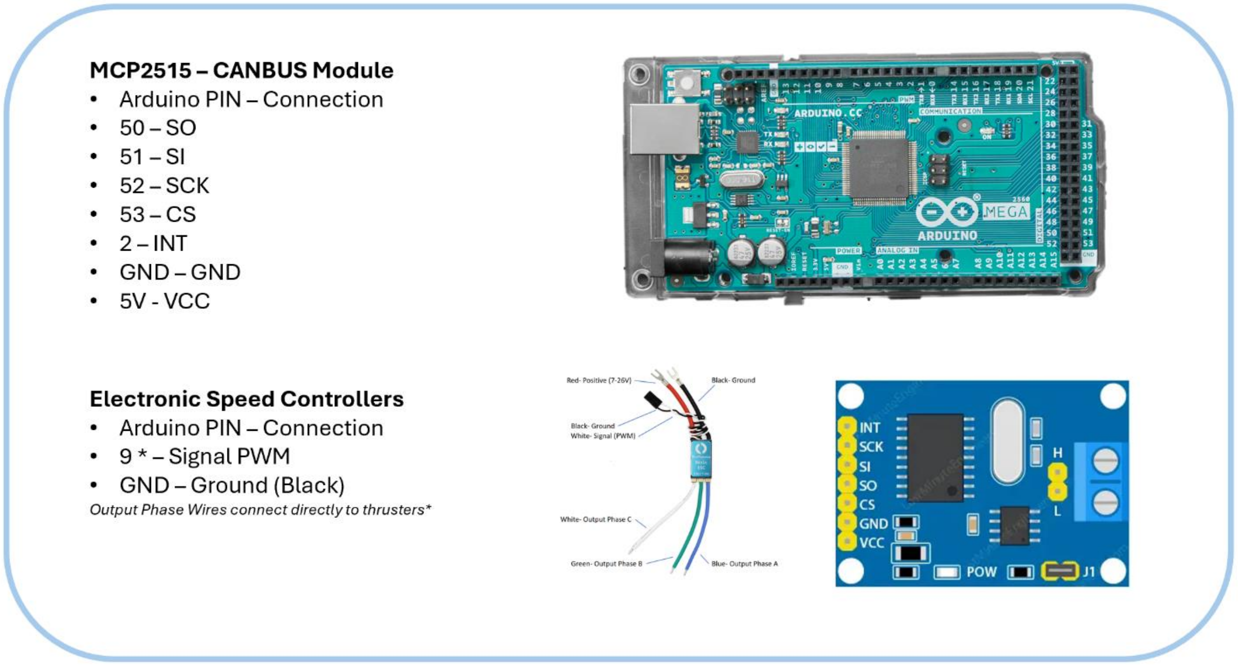


Figure 8: Boost Converter

The Boost Converter has been precisely tuned to output 19.5V with a 14.4V input, while these two components are connected to it. To adjust these values, one should connect a power supply set to their desired input voltage at the input, and a multimeter at the output. The variable resistor closest to the input terminals controls the input voltage while the variable resistor closest to the output terminals controls the output voltage.

3.1.2 Mechatronics

The communication system is simply a CANBUS line (Connected Area Network) made of two 16-gauge (Black and Red) wires with two 120-ohm resistors at either end of the line and various (connector type) connectors to onboard any new subsystem. Green and red LED lights are attached to all the different modules to help signal the status of the CANBUS line.



MCP2515 – CANBUS Module

- Arduino PIN – Connection
- 50 – SO
- 51 – SI
- 52 – SCK
- 53 – CS
- 2 – INT
- GND – GND
- 5V - VCC

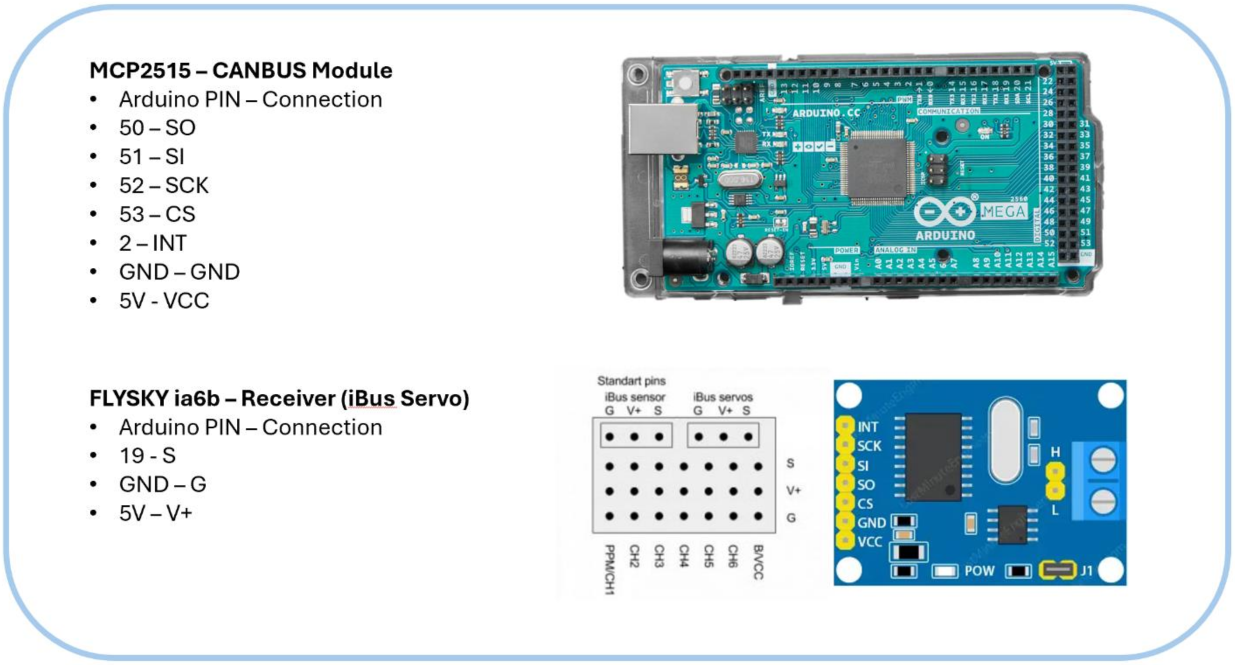
Electronic Speed Controllers

- Arduino PIN – Connection
- 9 * – Signal PWM
- GND – Ground (Black)

*Output Phase Wires connect directly to thrusters**

Figure 9: Locomotion System Connection Guide

The locomotion system is connected to the rest of the system via the CANBUS line and consists of two electronic speed controllers – controlled via an Arduino Mega 2560 - which are each connected to a thruster located at the back of the boat.



MCP2515 – CANBUS Module

- Arduino PIN – Connection
- 50 – SO
- 51 – SI
- 52 – SCK
- 53 – CS
- 2 – INT
- GND – GND
- 5V - VCC

FLYSKY ia6b – Receiver (iBus Servo)

- Arduino PIN – Connection
- 19 - S
- GND – G
- 5V – V+

Figure 10: RC System Connection Guide

The RC system is connected to the system via CANBUS line – controlled via an Arduino Mega 2560 - and consists of 1 receiver paired to our RC controller. Connected to another pin on the same Arduino is the kill switch system, which is meant to activate once connection to the RC controller is lost.

The navigation system is made up of multiple subsystems all connected via CANBUS line. The first subsystem is a GPS module hooked up on an Arduino Nano and sends longitudinal and latitudinal position to the on-board CPU. The second subsystem is an IMU connected and controlled via an Arduino Nano. Acceleration and gyroscope data from the IMU are used to estimate the boat’s local position as well as keep track of the pitch and roll of the boat as to avoid capsizing. Finally, a raspberry PI acts as the boat’s onboard CPU and directly connects to both the LiDAR and camera.

3.1.3 Waypoint Following

The algorithm to generate the motor commands to waypoint follow requires the optimization toolbox installed in Matlab. The dynamics of the simulation are powered by ode45. The state space representation is:

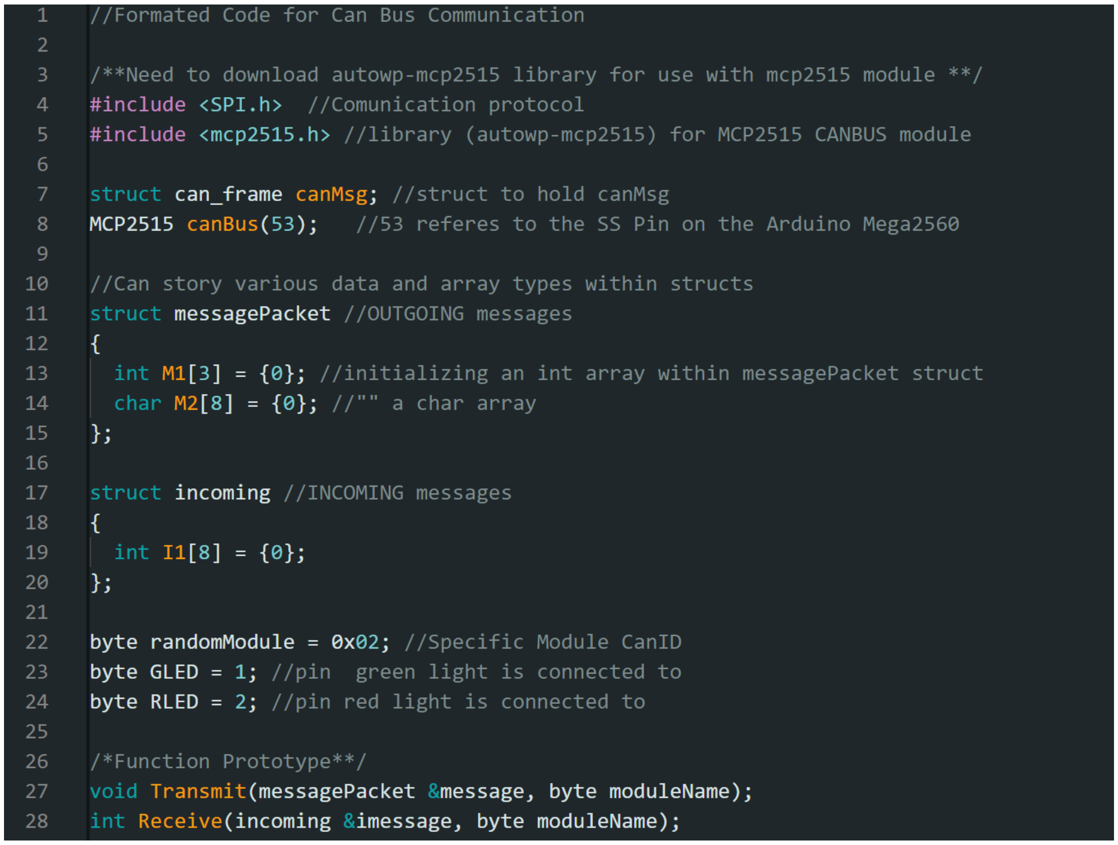
$$\begin{bmatrix} \dot{V}_x \\ \dot{V}_y \\ \dot{p}_x \\ \dot{p}_y \\ \dot{\psi} \\ \dot{w}_Z \end{bmatrix} = \begin{bmatrix} \omega_z V_{\bar{y}} + (F_R + F_L - sign(V_{\bar{x}}) K_x V_{\bar{x}}^2))/m \\ -w_z V_{\bar{x}} - sign(V_{\bar{y}}) K_y V_{\bar{y}}^2/m \\ V_{\bar{x}} \cos\psi - V_{\bar{y}} \sin\psi \\ V_{\bar{x}} \sin\psi + V_{\bar{y}} \cos\psi \\ w_z \\ ((F_R - F_L)d - sign(w_Z) K_Z w_Z^2)/J \end{bmatrix}$$

Where K_x, K_y, K_z are the coefficients of drag, m is the mass of the boat, ω is the angular velocity of the boat, ϕ is the heading of the boat, p_x, p_y are the x and y position of the boat respectively, V_x, V_y are the x and y velocities of the boat respectively, and F_R, F_L are the forces from the right and left thruster respectively.

The function fmincon from the optimization toolbox is used to generate the list of thruster inputs. The state vector is a 6x1 matrix consisting of V_x, V_y, p_x, p_y, ϕ, ω. The input vector is a list of 10 inputs for the left thruster, 10 inputs for the right thruster, and the time to get to the waypoint. The inputs for the thrusters are interpolated across the time interval to create a profile for each thruster. Q is the variable that was chosen to represent the state vector, and X represents the input vector. The cost function is the time elapsed, so that the optimizer finds the quickest path to the waypoint. The constraints passed into fmincon are that the final x and y positions equal the values of the waypoint, and that at the final waypoint, V_x, V_y, and ω are 0. The waypoint following is achieved by using the final state of the simulation for the previous waypoint as the initial condition of the next waypoint. Each waypoint needs an initial guess as a starting point to find the best solution, so if the optimizer has problems finding a solution, changing the initial guess to be more accurate can resolve the issue. The optimized solution is an array of values, 21 for each waypoint. 10 for each thruster, and 1 final value for the time elapsed. The inputs for the thrusters are then converted from the force needed from the thruster to the PWM value to achieve the desired thrust. The equation to convert from force to PWM was calculated by using the T200 thruster spec sheet and creating a line of best fit for the force values and pwm. The equation of the line of best fit is used to convert from force to PWM values.

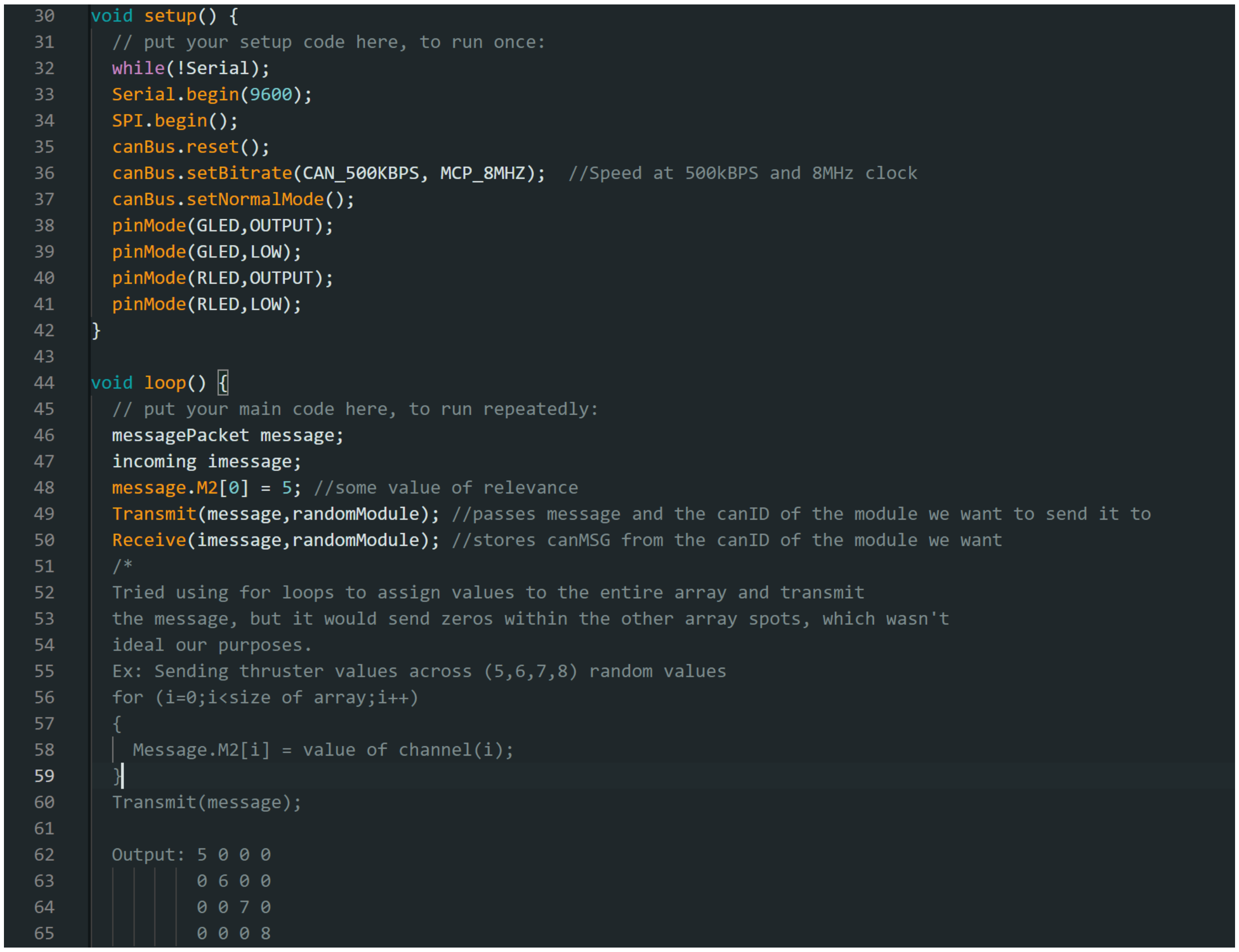
3.1.4 CANBUS communication logic

Attached in the figures below is a fully commented and formatted logic for use with the CANBUS modules attached to each individual Arduino.



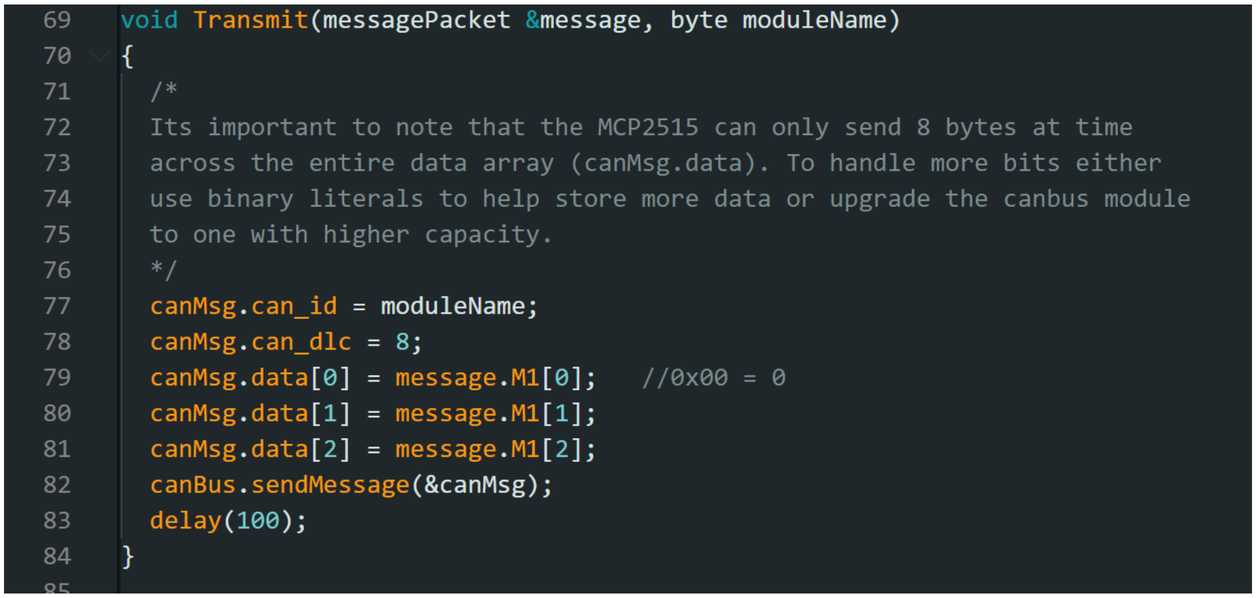
```
//Formated Code for Can Bus Communication

/**Need to download autowp-mcp2515 library for use with mcp2515 module **/
#include <SPI.h>  //Comunication protocol
#include <mcp2515.h> //library (autowp-mcp2515) for MCP2515 CANBUS module

struct can_frame canMsg; //struct to hold canMsg
MCP2515 canBus(53);   //53 referes to the SS Pin on the Arduino Mega2560

//Can story various data and array types within structs
struct messagePacket //OUTGOING messages
{
  int M1[3] = {0}; //initializing an int array within messagePacket struct
  char M2[8] = {0}; //"" a char array
};

struct incoming //INCOMING messages
{
  int I1[8] = {0};
};

byte randomModule = 0x02; //Specific Module CanID
byte GLED = 1; //pin  green light is connected to
byte RLED = 2; //pin red light is connected to

/*Function Prototype**/
void Transmit(messagePacket &message, byte moduleName);
int Receive(incoming &imessage, byte moduleName);
```

Figure 12: Part 1 of the CANBUS logic.



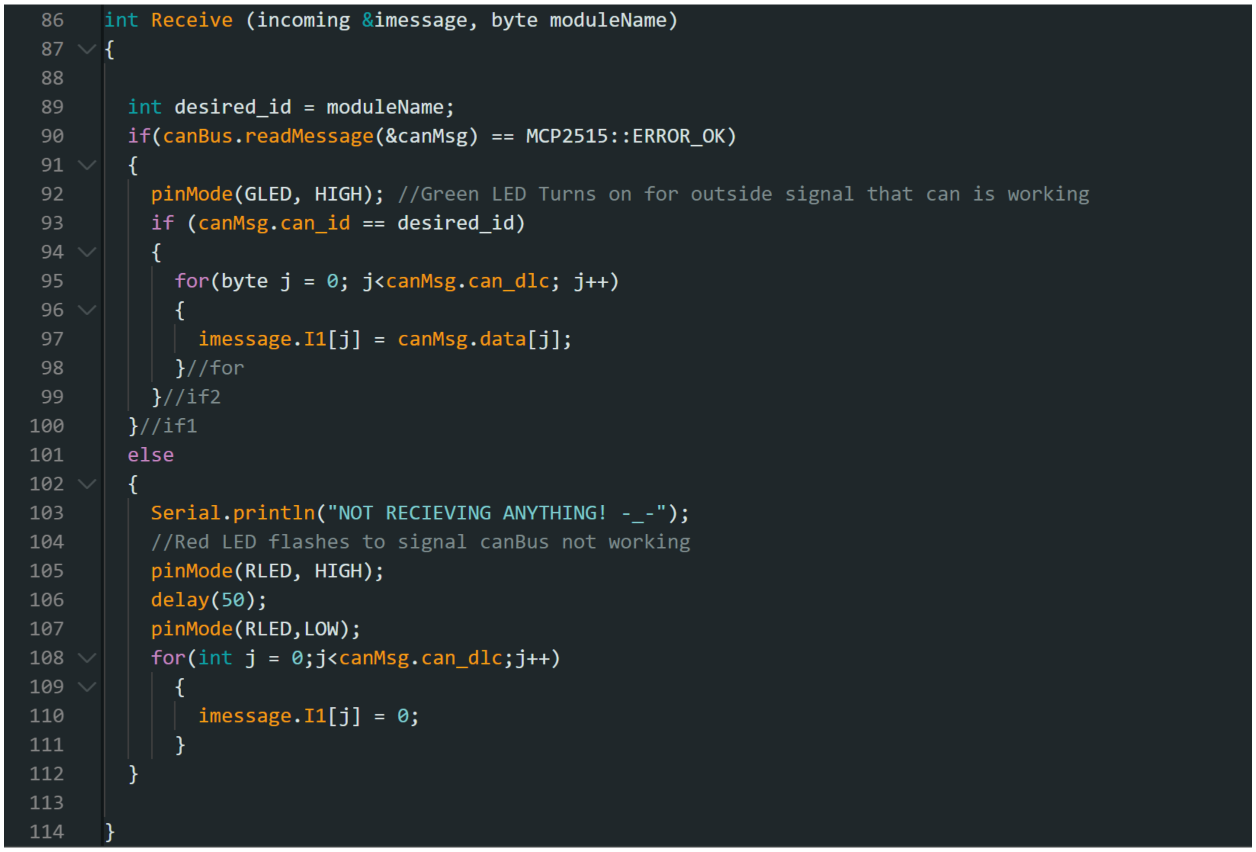
```
void setup() {
  // put your setup code here, to run once:
  while(!Serial);
  Serial.begin(9600);
  SPI.begin();
  canBus.reset();
  canBus.setBitrate(CAN_500KBPS, MCP_8MHZ);  //Speed at 500kBPS and 8MHz clock
  canBus.setNormalMode();
  pinMode(GLED,OUTPUT);
  pinMode(GLED,LOW);
  pinMode(RLED,OUTPUT);
  pinMode(RLED,LOW);
}

void loop() {
  // put your main code here, to run repeatedly:
  messagePacket message;
  incoming imessage;
  message.M2[0] = 5; //some value of relevance
  Transmit(message,randomModule); //passes message and the canID of the module we want to send it to
  Receive(imessage,randomModule); //stores canMSG from the canID of the module we want
  /*
  Tried using for loops to assign values to the entire array and transmit
  the message, but it would send zeros within the other array spots, which wasn't
  ideal our purposes.
  Ex: Sending thruster values across (5,6,7,8) random values
  for (i=0;i<size of array;i++)
  {
    Message.M2[i] = value of channel(i);
  }
  Transmit(message);

  Output: 5 0 0 0
          0 6 0 0
          0 0 7 0
          0 0 0 8
```

Figure 13: Part 2 of the CANBUS logic.



```
void Transmit(messagePacket &message, byte moduleName)
{
  /*
  Its important to note that the MCP2515 can only send 8 bytes at time
  across the entire data array (canMsg.data). To handle more bits either
  use binary literals to help store more data or upgrade the canbus module
  to one with higher capacity.
  */
  canMsg.can_id = moduleName;
  canMsg.can_dlc = 8;
  canMsg.data[0] = message.M1[0];   //0x00 = 0
  canMsg.data[1] = message.M1[1];
  canMsg.data[2] = message.M1[2];
  canBus.sendMessage(&canMsg);
  delay(100);
}
```

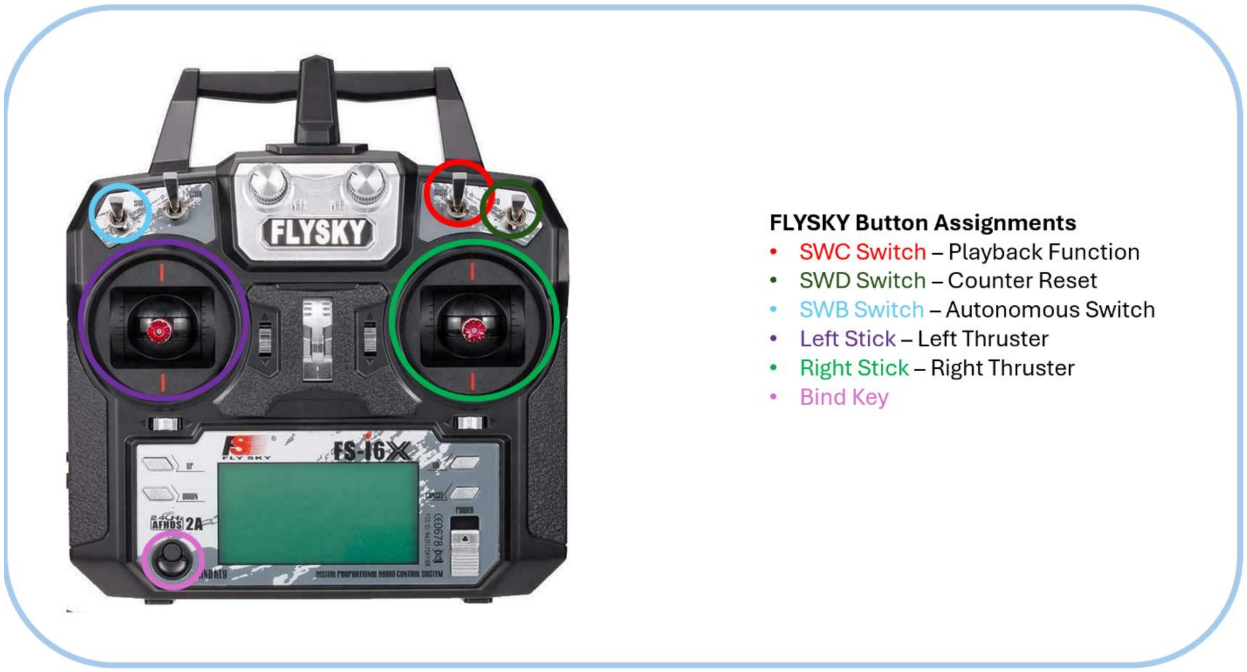
Figure 14: Part 3 of the CANBUS logic. Specifically, the transmission function.



```
int Receive (incoming &imessage, byte moduleName)
{

  int desired_id = moduleName;
  if(canBus.readMessage(&canMsg) == MCP2515::ERROR_OK)
  {
    pinMode(GLED, HIGH); //Green LED Turns on for outside signal that can is working
    if (canMsg.can_id == desired_id)
    {
      for(byte j = 0; j<canMsg.can_dlc; j++)
      {
        imessage.I1[j] = canMsg.data[j];
      }//for
    }//if2
  }//if1
  else
  {
    Serial.println("NOT RECIEVING ANYTHING! -_-");
    //Red LED flashes to signal canBus not working
    pinMode(RLED, HIGH);
    delay(50);
    pinMode(RLED,LOW);
    for(int j = 0;j<canMsg.can_dlc;j++)
    {
      imessage.I1[j] = 0;
    }
  }

}
```

Figure 15: Part 4 of the CANBUS logic. Specifically, the receiving function.

Operation


FLYSKY
FS-i6X
FLYSKY Button Assignments
● SWC Switch – Playback Function
● SWD Switch – Counter Reset
● SWB Switch – Autonomous Switch
● Left Stick – Left Thruster
● Right Stick – Right Thruster
● Bind Key

Button Functions:

Figure 16: RoboBoat RC Controller Diagram (Front)

SWC Switch: Used to move between the different playback functions. When the switch is flicked to the top position (default position) the system operates normally as an RC boat with the left and right stick being used to drive and turn the boat. In the middle position, the boat still operates as an RC boat but starts storing PWM values being sent to the thrusters via the remote controller. Finally, when switched to the bottom position, the boat begins to playback through and repeat those previously recorded PWM values.

SWD Switch: This switch is used to reset the counter back to zero. The mentioned counter is used to store the PWM values into the playback multidimensional array. Be sure to place the switch back into its default position (flicked up) before doing anything else.

SWB Switch: This switch allows the Roboboat to switch between RC mode and autonomous mode. With flicked up (default position) being the RC mode and flicked down being the autonomous mode.

Left and Right Stick: As Stated in the figure above, the left and right stick are responsible for controlling the PWM values being sent to the thrusters. In the resting position they send a value of 1500 which tells the thrusters to do nothing, as you move the sticks horizontally the thrusters start to move forwards and backwards (Move left for backwards and right for forwards).

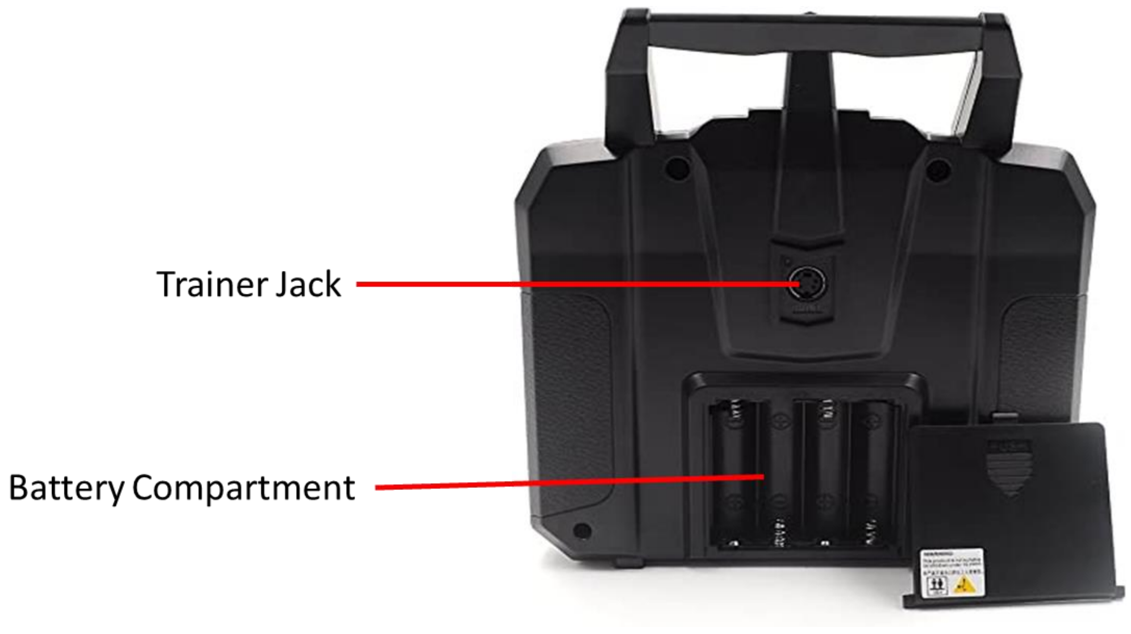


Figure 17: RoboBoat RC Controller Diagram (Back)

Steps to install the transmitter battery:

1. Open the battery compartment.
2. Insert 4 charged AA batteries into the compartment.

3. Replace the battery compartment cover.

Once the transmitter batteries are installed, power on the controller and refer to the display to ensure the receiver is connected. Refer to *Troubleshooting* if any issues arise with controller setup. The FS-ia6 RC Controller user manual can also be followed to modify the channel set up.

The IBusBM library (by Bart Mellink) is required to connect to the controller via the receiver (Refer to Mechatronics section for Arduino connection guide.

Steps to install the IBusBM library:

1. Open Arduino
2. Open Tools à Library Manager
    a. If you have an update Arduino IDE, library icon on left side of the screen will get you to the same place.
3. Search “IBusBM”

Troubleshooting

To ensure the proper functioning of the RoboBoat, the boat must be tested in a body of water. This is crucial for the safe operation of the thrusters, specifically the T200 models, which are not designed to run outside of water. Rigorous testing should only be conducted when the thrusters are submerged to prevent damage. Additionally, it is essential that the main 14.8V battery source and the RC remote are fully charged before testing to avoid the risk of the boat becoming stranded due to insufficient battery power. Upon connecting the main power source to the fuse panel, which in turn is connected to the rest of the electrical components, an auditory signal will be emitted. The first noise indicates that the thrusters are receiving power, while the second confirms their readiness to receive commands, either from the waypoint navigation system or manual control via the RC controller. The presence of both sounds from each thruster signifies readiness; the absence of either sound indicates a connectivity issue.

Furthermore, when testing new code on the Arduino microcontrollers and thrusters, a reset of the microcontroller may be required, typically achieved by pressing the red button on the device. In addition to the proper functioning of the components, it is crucial that the top plexiglass hatch is tightly secured to the boat. This ensures that the electrical components inside the hull are protected from water which could cause damage. Additionally, the rope attached to the bow of the boat must be securely fastened to prevent it from dragging in the water and potentially damaging the propellers on the T200 thrusters if it gets caught. All electrical components should be securely fastened to the boat to prevent any disconnections during maneuvers. If a thruster is damaged or breaks during testing, it can be replaced. The thruster is attached to a bracket on the bottom of the boat, which can be unscrewed and replaced, if necessary, although there is a layer of adhesive securing the connection. In addition to the previously mentioned checks, it is crucial to use a multimeter to verify that the boost converter is boosting the voltage to the appropriate level required to power the LiDAR, which is rated for 19.5 volts.

Before making any connections, ensure that the ratings of all components are checked to prevent damage. It is important to ensure that the battery is fully charged to avoid any power failures during the operation of RoboBoat. If the red LED lights on the boat starts to blink this signifies a problem in the CANBUS line, while steady lighting of the green LED tells us that the CANBUS line is operational.