University of California Berkeley

Dr. Hannah Stuart

Final Project - Prometheus

Group# 27

Members:

Shawn Chan

Aldo Jardines

Dennis Tercero

Angel Valdivia

Prometheus: Submersible Explorer

Opportunity

Mapping or exploring bodies of water has always been a difficult task that requires the use of underwater drones. Purchasing a personal underwater drone has proven to be expensive, and this acts as a barrier to entry for lower-budget researchers who would like to use one for exploratory purposes. Our team took this opportunity to develop and construct an underwater drone that can be remotely piloted using an intuitive controller.

High-Level Strategy

To develop a remotely controlled submersible system, the task could be broken down into two smaller subtasks, buoyancy control and thrust control. Accomplishing these subtasks would enable the system to vary its depth within the water, as well as control where it is going.

For the buoyancy control system, syringes controlled by motors were used to vary the overall density of the system. These work by taking in and releasing water when the plungers of the system are moved inwards and outwards respectively. Initially, the original design was to be able to displace 785 cm^3 of air. However, due to space constraints, only 589 cm^3 of air could be displaced.

For the thrust control, two pairs of horizontal and vertical-facing propellers were used to provide thrust in their respective directions. The speed of these propellers was controlled via brushless motors and electronic speed controllers (ESCs).

Device Diagram

The mechanical design of the buoyancy system is detailed in Figures 1 and 2. The entire system is composed of two smaller, identical, subsystems that control the position of the plungers of the syringe.

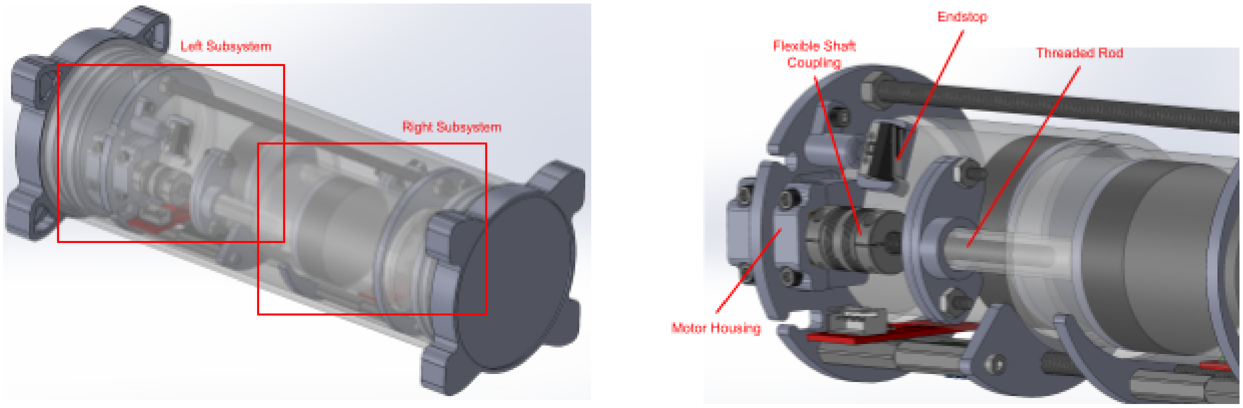
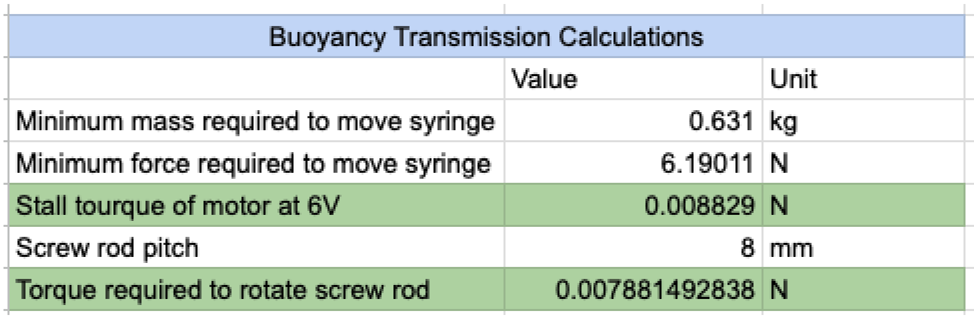


Figure 1: Buoyancy System Overview and Detailed View

Function-critical Decisions

A key function-critical calculation that had to be made was the force required by the motors controlling the buoyancy system. These calculations are shown in Figure 1.

$$Torque\ Required\ =\ \frac{Screw\ Pitch}{2\pi} \times Minimum\ force\ to\ move\ syringe$$



Buoyancy Transmission Calculations		
	Value	Unit
Minimum mass required to move syringe	0.631	kg
Minimum force required to move syringe	6.19011	N
Stall tourque of motor at 6V	0.008829	N
Screw rod pitch	8	mm
Torque required to rotate screw rod	0.007881492838	N

Figure 2: Calculations for buoyancy transmission

Based on our calculations, our chosen motors for the transmission are sufficient to move the syringes for the buoyancy system.

Circuit and State Transition Diagrams

Circuit Diagram

The main elements controlling the motors are the ESP32 and the motor driver. The motors controlling the buoyancy system, motors 1 and 2, were given an appropriate level of voltage as dictated by the motor controller. In turn, the motor controller is controlled via the ESP32 module, which receives inputs from the remote control operated by the user.

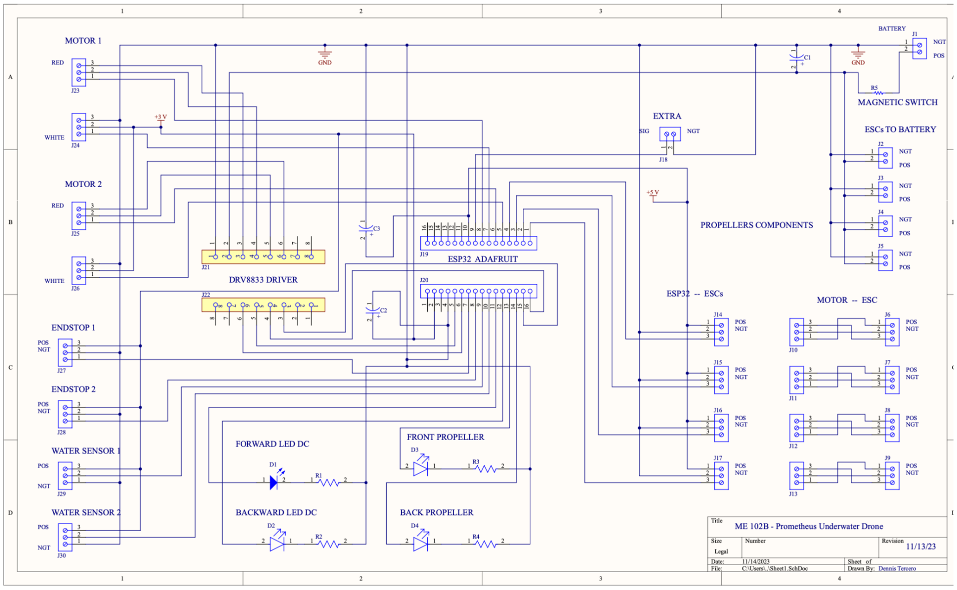


Figure 3: Circuit diagram of the entire system

State Transition Diagram

The system relies on two ESP32s, where each controls one of the two subsystems. The thrust system will be referred to as Core 0, while the buoyancy system will be referred to as Core 1.

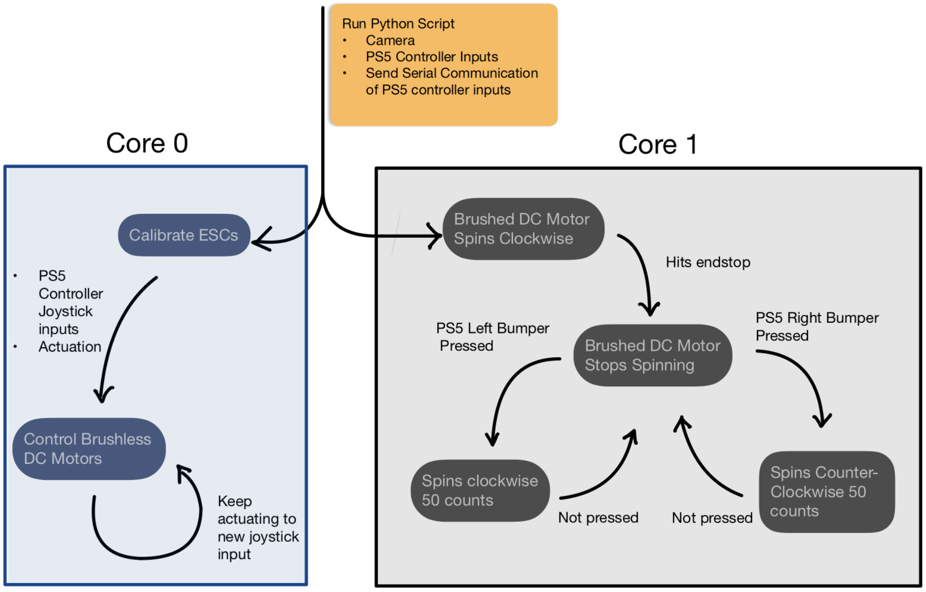

Run Python Script
- Camera
- PS5 Controller Inputs
- Send Serial Communication of PS5 controller inputs
Core 0
Core 1
Calibrate ESCs
- PS5 Controller Joystick inputs
- Actuation
Control Brushless DC Motors
Keep actuating to new joystick input
Brushed DC Motor Spins Clockwise
Hits endstop
Brushed DC Motor Stops Spinning
PS5 Left Bumper Pressed
PS5 Right Bumper Pressed
Spins clockwise 50 counts
Spins Counter-Clockwise 50 counts
Not pressed
Not pressed

Figure 4: State Transition Diagram

In Core 0, the ESCs are first calibrated. If the controller's joysticks are moved, the brushless DC motors are spun. The speeds of the motors are dependent on the input amount of the joysticks and continue to spin as long as the input is given. When the joysticks are released, and hence when there is no longer input, the motors stop spinning.

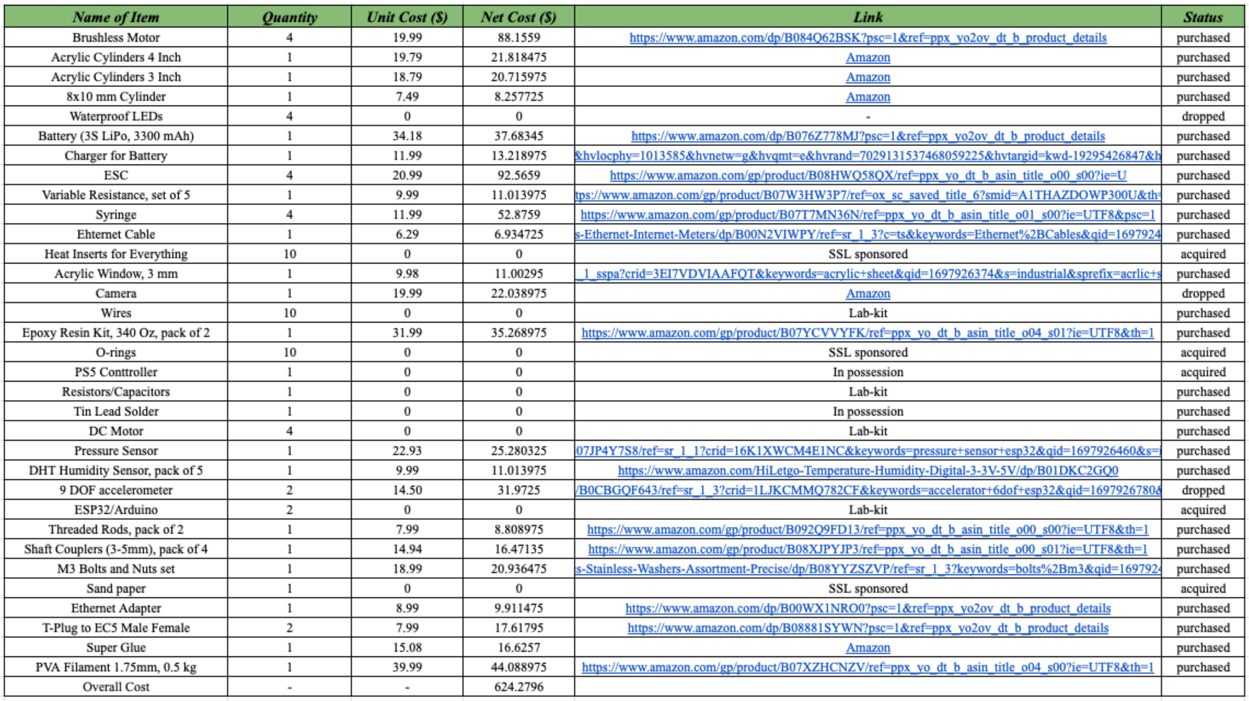
In Core 1, the DC motors attached to the syringe are first spun to enable the system to submerge. When the syringes reach their maximum displacement, end-stops are engaged, telling the ESP32 that the syringes are unable to move any further. When the remote control's button is pressed, there is a switch in the screw rod's direction of rotation, causing it to displace the syringes in the opposite direction and return them to their original position.

Reflection

Overall, the project taught us significantly about how to integrate mechanical systems with electronics through the lessons learned in the course's labs. Communication was key to achieving the project's final state, as well as dividing up the tasks to meet the project's deliverables on time. We learned how to compromise on part selection when it came to cost and functionality, as well as how to implement and write clean code that can be easily debugged and reused in the future.

Appendix

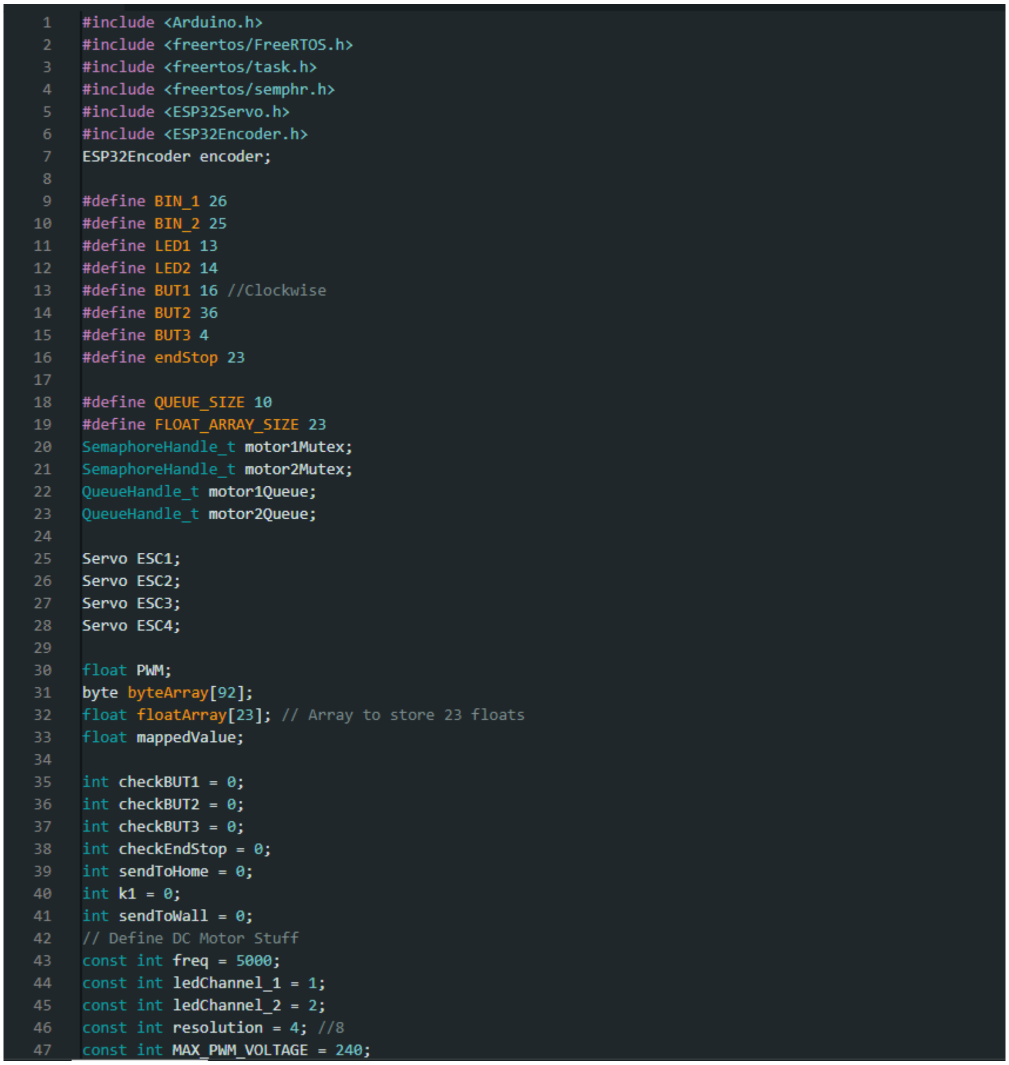
Bill Of Materials



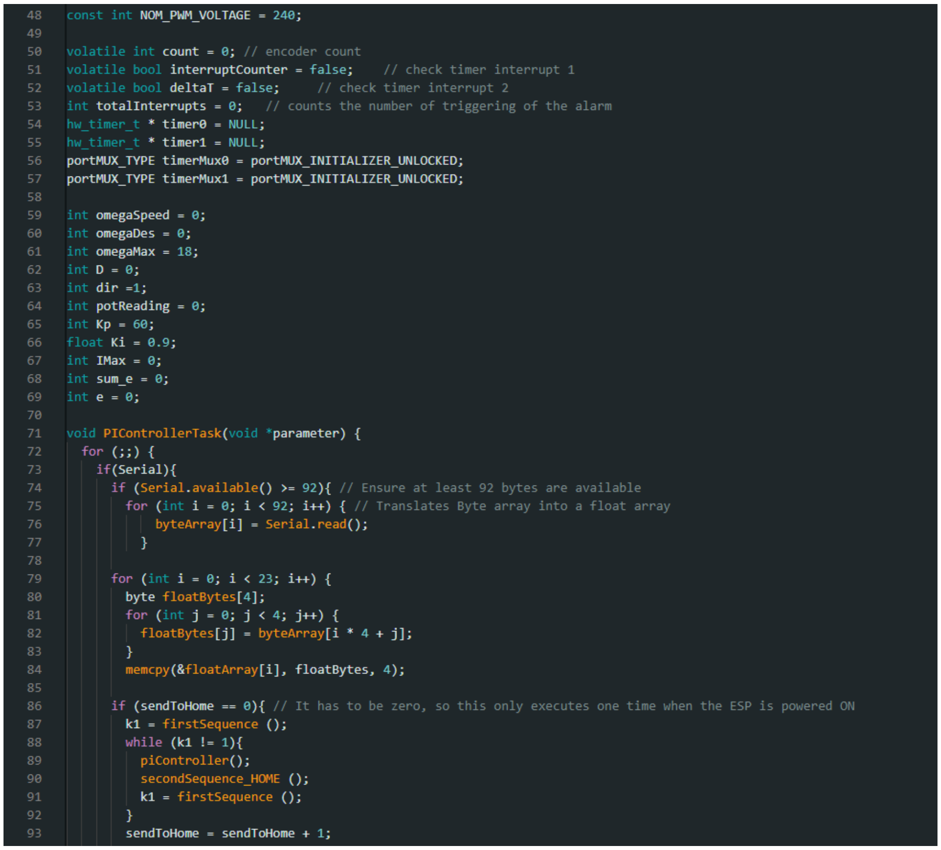
Name of Item	*Quantity*	*Unit Cost ($)*	*Net Cost ($)*	*Link*	*Status*
Brushless Motor	4	19.99	88.1559	https://www.amazon.com/dp/B084Q62BSK?psc=1&ref=ppx_yo2ov_dt_b_product_details	purchased
Acrylic Cylinders 4 Inch	1	19.79	21.818475	Amazon	purchased
Acrylic Cylinders 3 Inch	1	18.79	20.715975	Amazon	purchased
8x10 mm Cylinder	1	7.49	8.257725	Amazon	purchased
Waterproof LEDs	4	0	0	-	dropped
Battery (3S LiPo, 3300 mAh)	1	34.18	37.68345	https://www.amazon.com/dp/B076Z778MJ?psc=1&ref=ppx_yo2ov_dt_b_product_details	purchased
Charger for Battery	1	11.99	13.218975	&hvlocphy=1013585&hvnetw=g&hvqmt=e&hvrand=7029131537468059225&hvtargid=kwd-19295426847&h	purchased
ESC	4	20.99	92.5659	https://www.amazon.com/gp/product/B08HWQ58QX/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=U	purchased
Variable Resistance, set of 5	1	9.99	11.013975	tps://www.amazon.com/gp/product/B07W3HW3P7/ref=ox_sc_saved_title_6?smid=A1THAZDOWP300U&th=	purchased
Syringe	4	11.99	52.8759	https://www.amazon.com/gp/product/B07T7MN36N/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&psc=1	purchased
Ehternet Cable	1	6.29	6.934725	s-Ethernet-Internet-Meters/dp/B00N2VIWPY/ref=sr_1_3?c=ts&keywords=Ethernet%2BCables&qid=1697924	purchased
Heat Inserts for Everything	10	0	0	SSL sponsored	acquired
Acrylic Window, 3 mm	1	9.98	11.00295	_1_sspa?crid=3EI7VDVIAAFQT&keywords=acrylic+sheet&qid=1697926374&s=industrial&sprefix=acrlic+s	purchased
Camera	1	19.99	22.038975	Amazon	dropped
Wires	10	0	0	Lab-kit	purchased
Epoxy Resin Kit, 340 Oz, pack of 2	1	31.99	35.268975	https://www.amazon.com/gp/product/B07YCVVYFK/ref=ppx_yo_dt_b_asin_title_o04_s01?ie=UTF8&th=1	purchased
O-rings	10	0	0	SSL sponsored	acquired
PS5 Conttroller	1	0	0	In possession	acquired
Resistors/Capacitors	1	0	0	Lab-kit	purchased
Tin Lead Solder	1	0	0	In possession	purchased
DC Motor	4	0	0	Lab-kit	purchased
Pressure Sensor	1	22.93	25.280325	07JP4Y7S8/ref=sr_1_1?crid=16K1XWCM4E1NC&keywords=pressure+sensor+esp32&qid=1697926460&s=i	purchased
DHT Humidity Sensor, pack of 5	1	9.99	11.013975	https://www.amazon.com/HiLetgo-Temperature-Humidity-Digital-3-3V-5V/dp/B01DKC2GQ0	purchased
9 DOF accelerometer	2	14.50	31.9725	/B0CBGQF643/ref=sr_1_3?crid=1LJKCMMQ782CF&keywords=accelerator+6dof+esp32&qid=1697926780&	dropped
ESP32/Arduino	2	0	0	Lab-kit	acquired
Threaded Rods, pack of 2	1	7.99	8.808975	https://www.amazon.com/gp/product/B092Q9FD13/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&th=1	purchased
Shaft Couplers (3-5mm), pack of 4	1	14.94	16.47135	https://www.amazon.com/gp/product/B08XJPYJP3/ref=ppx_yo_dt_b_asin_title_o00_s01?ie=UTF8&th=1	purchased
M3 Bolts and Nuts set	1	18.99	20.936475	s-Stainless-Washers-Assortment-Precise/dp/B08YYZSZVP/ref=sr_1_3?keywords=bolts%2Bm3&qid=169792	purchased
Sand paper	1	0	0	SSL sponsored	acquired
Ethernet Adapter	1	8.99	9.911475	https://www.amazon.com/dp/B00WX1NRO0?psc=1&ref=ppx_yo2ov_dt_b_product_details	purchased
T-Plug to EC5 Male Female	2	7.99	17.61795	https://www.amazon.com/dp/B08881SYWN?psc=1&ref=ppx_yo2ov_dt_b_product_details	purchased
Super Glue	1	15.08	16.6257	Amazon	purchased
PVA Filament 1.75mm, 0.5 kg	1	39.99	44.088975	https://www.amazon.com/gp/product/B07XZHCNZV/ref=ppx_yo_dt_b_asin_title_o04_s00?ie=UTF8&th=1	purchased
Overall Cost	-	-	624.2796		

Code

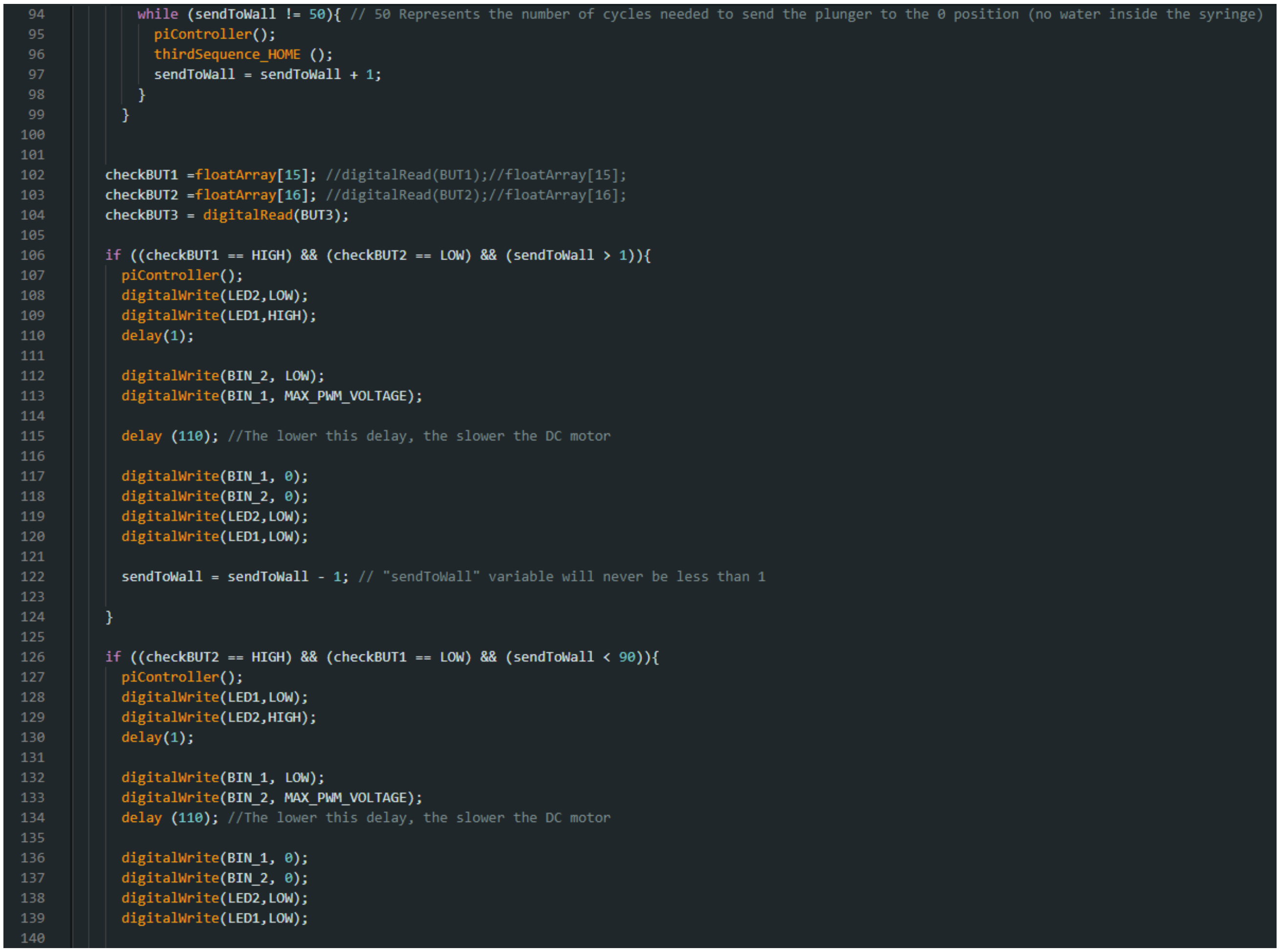
Code for buoyancy system



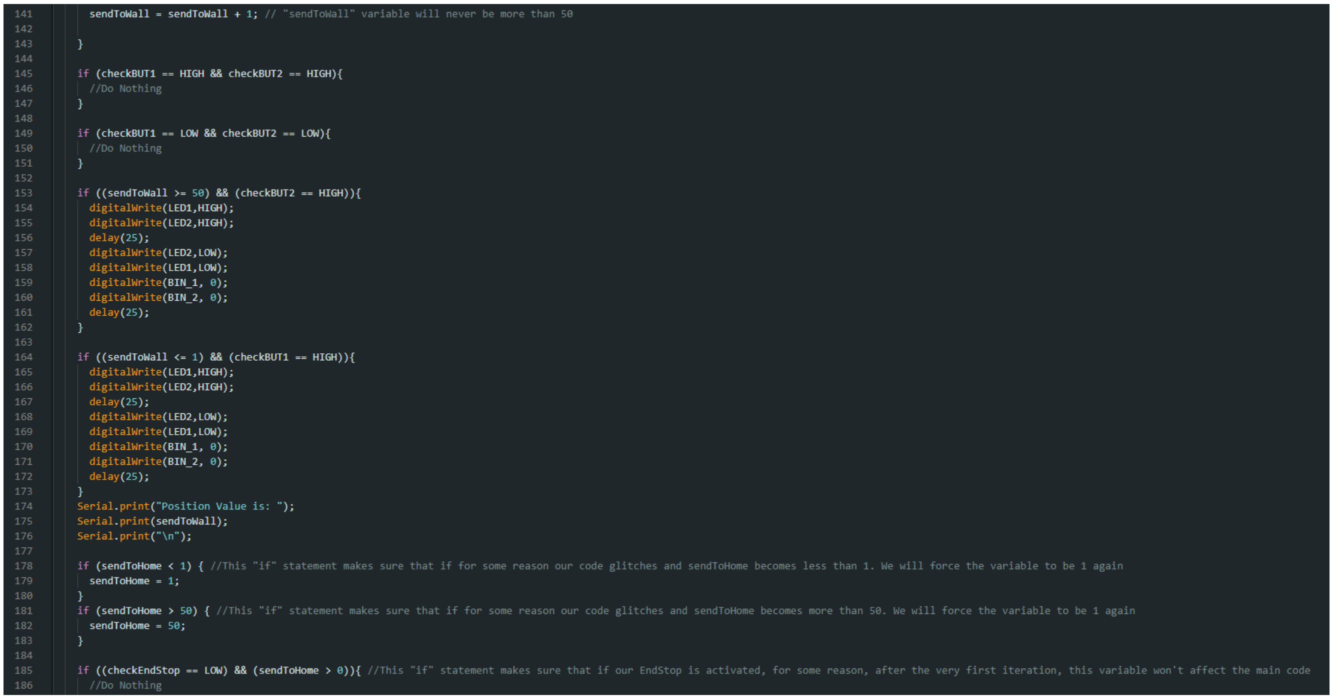
```
#include <Arduino.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include <freertos/semphr.h>
#include <ESP32Servo.h>
#include <ESP32Encoder.h>
ESP32Encoder encoder;

#define BIN_1 26
#define BIN_2 25
#define LED1 13
#define LED2 14
#define BUT1 16 //Clockwise
#define BUT2 36
#define BUT3 4
#define endStop 23

#define QUEUE_SIZE 10
#define FLOAT_ARRAY_SIZE 23
SemaphoreHandle_t motor1Mutex;
SemaphoreHandle_t motor2Mutex;
QueueHandle_t motor1Queue;
QueueHandle_t motor2Queue;

Servo ESC1;
Servo ESC2;
Servo ESC3;
Servo ESC4;

float PWM;
byte byteArray[92];
float floatArray[23]; // Array to store 23 floats
float mappedValue;

int checkBUT1 = 0;
int checkBUT2 = 0;
int checkBUT3 = 0;
int checkEndStop = 0;
int sendToHome = 0;
int k1 = 0;
int sendToWall = 0;
// Define DC Motor Stuff
const int freq = 5000;
const int ledChannel_1 = 1;
const int ledChannel_2 = 2;
const int resolution = 4; //8
const int MAX_PWM_VOLTAGE = 240;
```



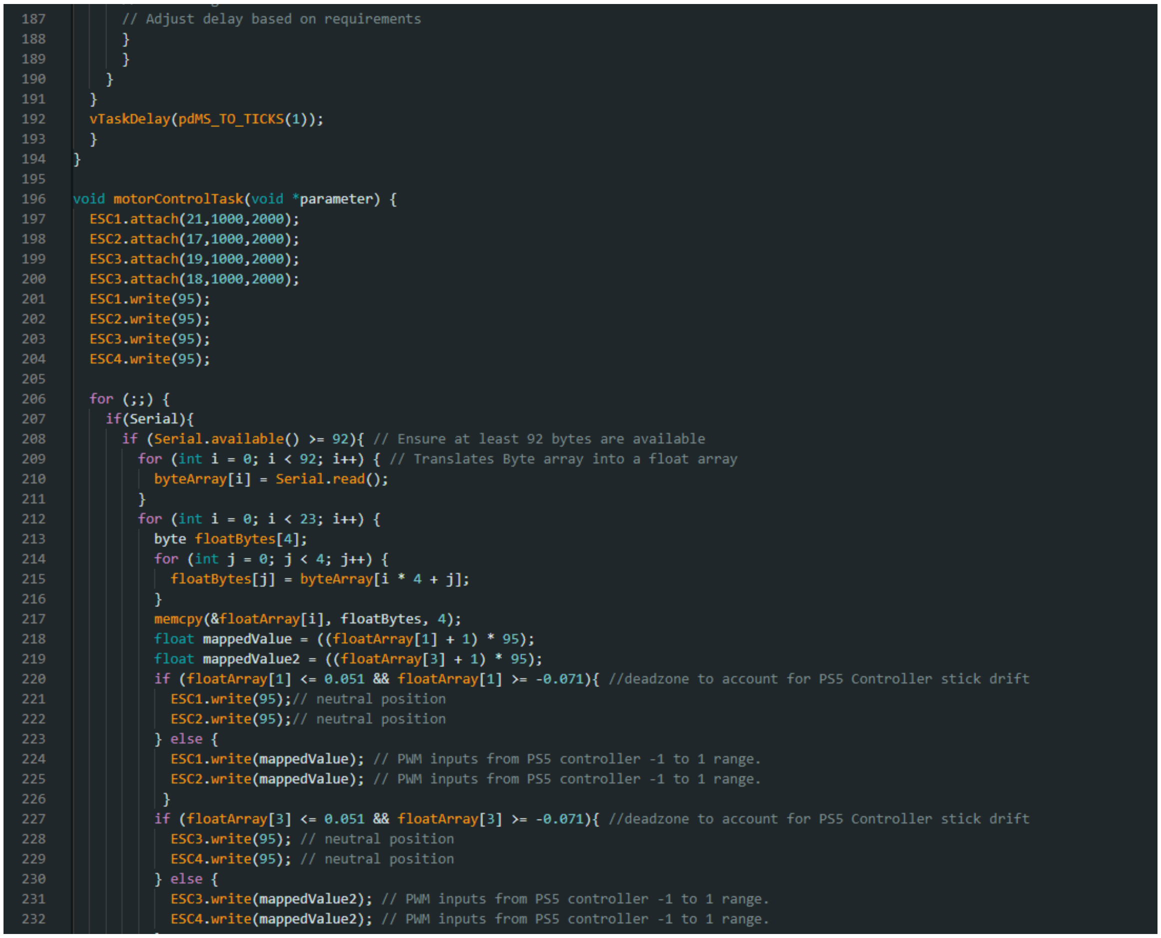
```
const int NOM_PWM_VOLTAGE = 240;

volatile int count = 0; // encoder count
volatile bool interruptCounter = false;    // check timer interrupt 1
volatile bool deltaT = false;     // check timer interrupt 2
int totalInterrupts = 0;   // counts the number of triggering of the alarm
hw_timer_t * timer0 = NULL;
hw_timer_t * timer1 = NULL;
portMUX_TYPE timerMux0 = portMUX_INITIALIZER_UNLOCKED;
portMUX_TYPE timerMux1 = portMUX_INITIALIZER_UNLOCKED;

int omegaSpeed = 0;
int omegaDes = 0;
int omegaMax = 18;
int D = 0;
int dir =1;
int potReading = 0;
int Kp = 60;
float Ki = 0.9;
int IMax = 0;
int sum_e = 0;
int e = 0;

void PIControllerTask(void *parameter) {
  for (;;) {
    if(Serial){
      if (Serial.available() >= 92){ // Ensure at least 92 bytes are available
        for (int i = 0; i < 92; i++) { // Translates Byte array into a float array
            byteArray[i] = Serial.read();
          }

      for (int i = 0; i < 23; i++) {
        byte floatBytes[4];
        for (int j = 0; j < 4; j++) {
          floatBytes[j] = byteArray[i * 4 + j];
        }
        memcpy(&floatArray[i], floatBytes, 4);

      if (sendToHome == 0){ // It has to be zero, so this only executes one time when the ESP is powered ON
        k1 = firstSequence ();
        while (k1 != 1){
          piController();
          secondSequence_HOME ();
          k1 = firstSequence ();
        }
        sendToHome = sendToHome + 1;
```



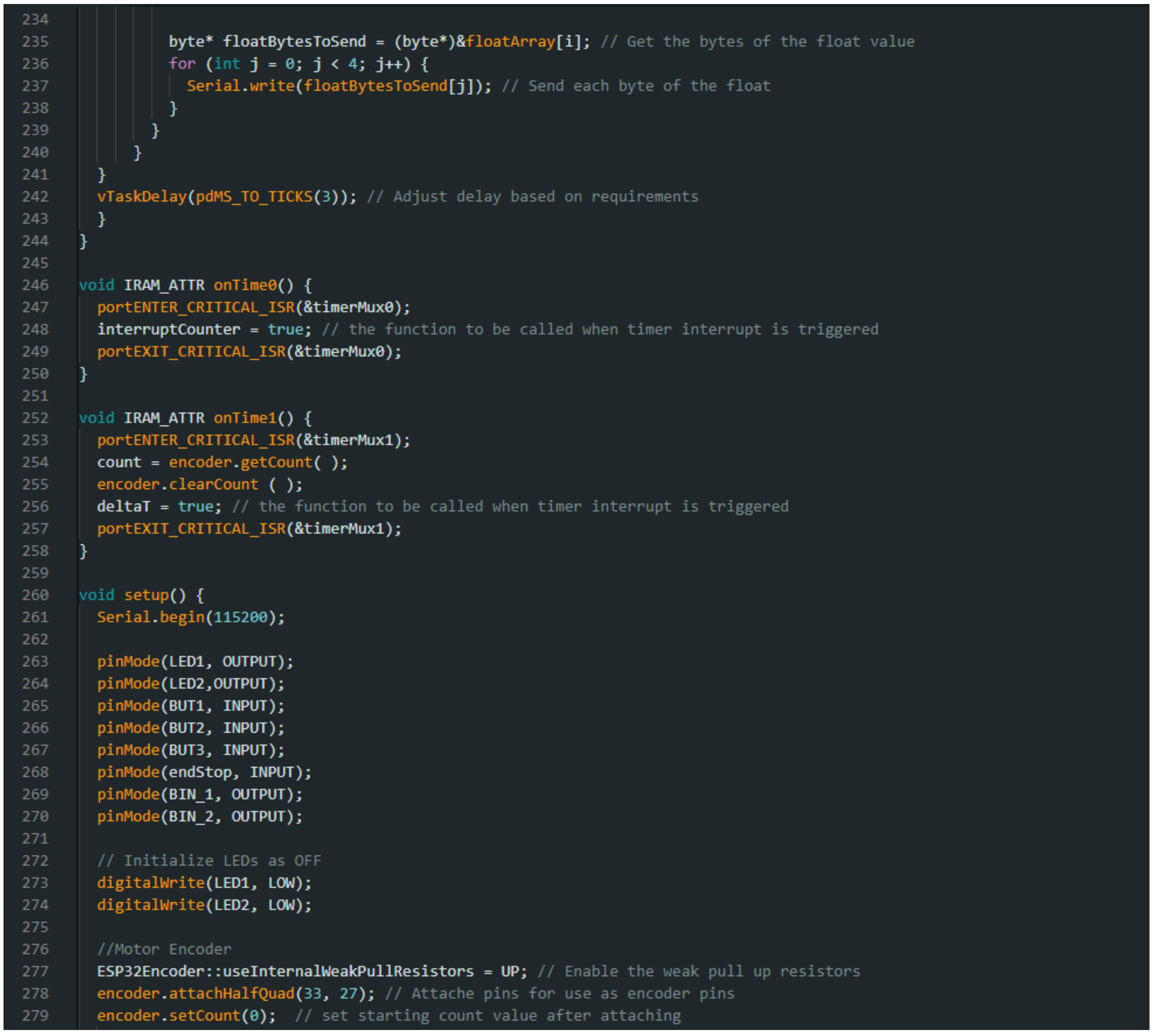
```
    while (sendToWall != 50){ // 50 Represents the number of cycles needed to send the plunger to the 0 position (no water inside the syringe)
      piController();
      thirdSequence_HOME ();
      sendToWall = sendToWall + 1;
    }
  }


checkBUT1 =floatArray[15]; //digitalRead(BUT1);//floatArray[15];
checkBUT2 =floatArray[16]; //digitalRead(BUT2);//floatArray[16];
checkBUT3 = digitalRead(BUT3);

if ((checkBUT1 == HIGH) && (checkBUT2 == LOW) && (sendToWall > 1)){
  piController();
  digitalWrite(LED2,LOW);
  digitalWrite(LED1,HIGH);
  delay(1);

  digitalWrite(BIN_2, LOW);
  digitalWrite(BIN_1, MAX_PWM_VOLTAGE);

  delay (110); //The lower this delay, the slower the DC motor

  digitalWrite(BIN_1, 0);
  digitalWrite(BIN_2, 0);
  digitalWrite(LED2,LOW);
  digitalWrite(LED1,LOW);

  sendToWall = sendToWall - 1; // "sendToWall" variable will never be less than 1

}

if ((checkBUT2 == HIGH) && (checkBUT1 == LOW) && (sendToWall < 90)){
  piController();
  digitalWrite(LED1,LOW);
  digitalWrite(LED2,HIGH);
  delay(1);

  digitalWrite(BIN_1, LOW);
  digitalWrite(BIN_2, MAX_PWM_VOLTAGE);
  delay (110); //The lower this delay, the slower the DC motor

  digitalWrite(BIN_1, 0);
  digitalWrite(BIN_2, 0);
  digitalWrite(LED2,LOW);
  digitalWrite(LED1,LOW);

```



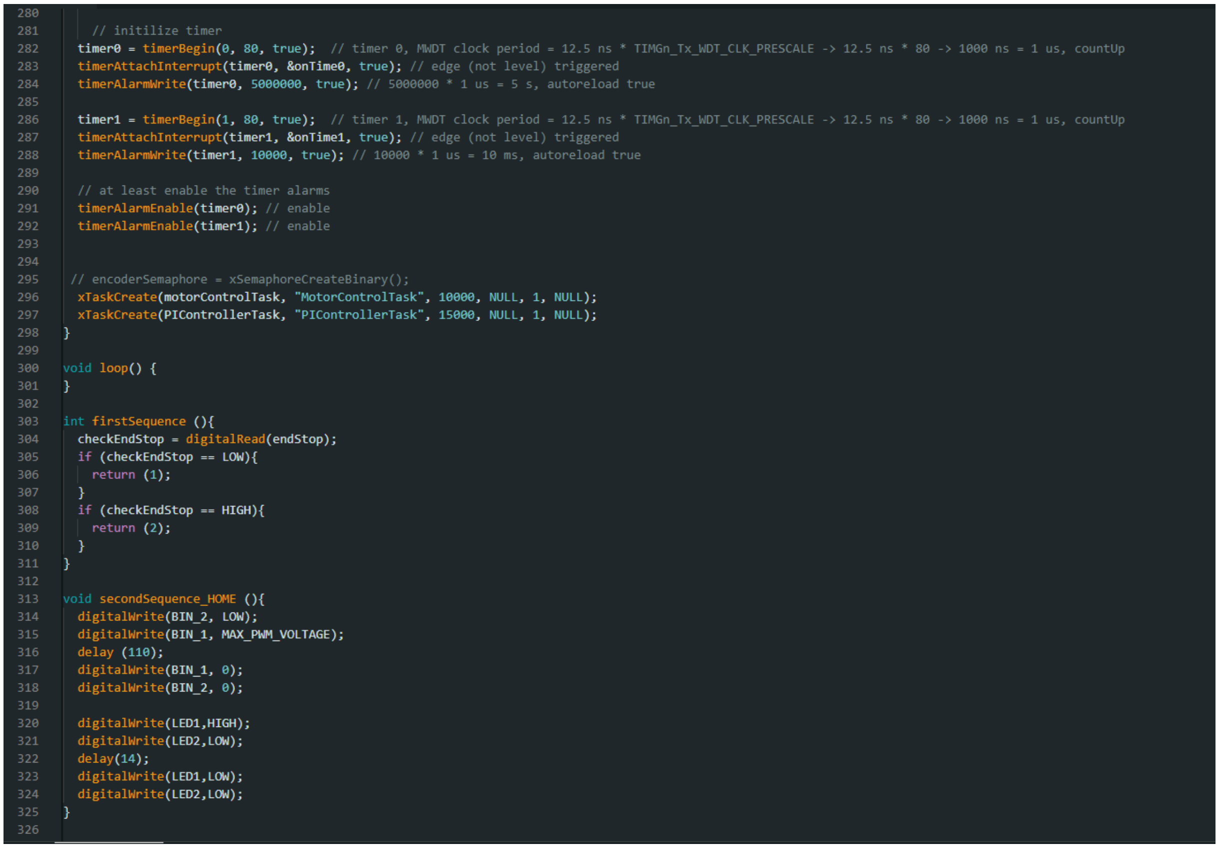
```
  sendToWall = sendToWall + 1; // "sendToWall" variable will never be more than 50

}

if (checkBUT1 == HIGH && checkBUT2 == HIGH){
  //Do Nothing
}

if (checkBUT1 == LOW && checkBUT2 == LOW){
  //Do Nothing
}

if ((sendToWall >= 50) && (checkBUT2 == HIGH)){
  digitalWrite(LED1,HIGH);
  digitalWrite(LED2,HIGH);
  delay(25);
  digitalWrite(LED2,LOW);
  digitalWrite(LED1,LOW);
  digitalWrite(BIN_1, 0);
  digitalWrite(BIN_2, 0);
  delay(25);
}

if ((sendToWall <= 1) && (checkBUT1 == HIGH)){
  digitalWrite(LED1,HIGH);
  digitalWrite(LED2,HIGH);
  delay(25);
  digitalWrite(LED2,LOW);
  digitalWrite(LED1,LOW);
  digitalWrite(BIN_1, 0);
  digitalWrite(BIN_2, 0);
  delay(25);
}
Serial.print("Position Value is: ");
Serial.print(sendToWall);
Serial.print("\n");

if (sendToHome < 1) { //This "if" statement makes sure that if for some reason our code glitches and sendToHome becomes less than 1. We will force the variable to be 1 again
  sendToHome = 1;
}
if (sendToHome > 50) { //This "if" statement makes sure that if for some reason our code glitches and sendToHome becomes more than 50. We will force the variable to be 1 again
  sendToHome = 50;
}

if ((checkEndStop == LOW) && (sendToHome > 0)){ //This "if" statement makes sure that if our EndStop is activated, for some reason, after the very first iteration, this variable won't affect the main code
  //Do Nothing
```



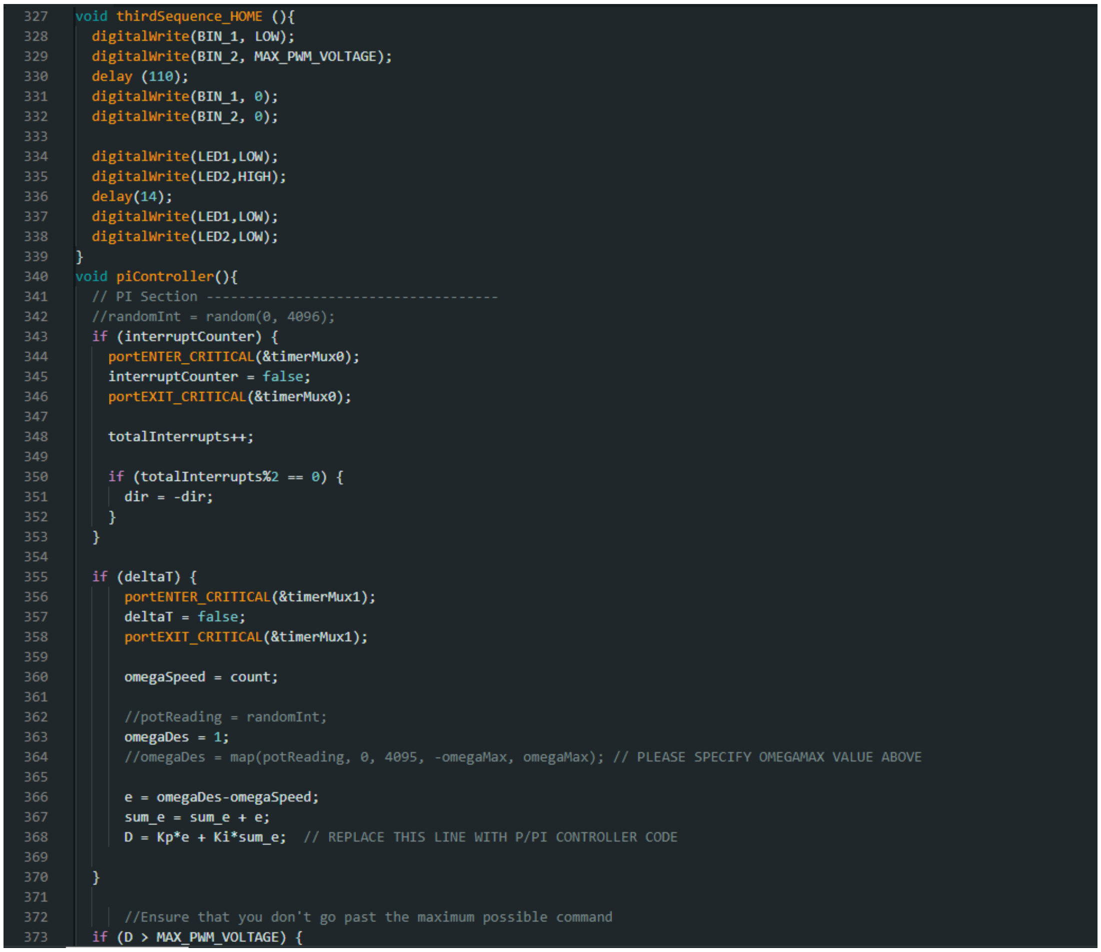
```
      // Adjust delay based on requirements
      }
      }
    }
  }
  vTaskDelay(pdMS_TO_TICKS(1));
  }
}

void motorControlTask(void *parameter) {
  ESC1.attach(21,1000,2000);
  ESC2.attach(17,1000,2000);
  ESC3.attach(19,1000,2000);
  ESC3.attach(18,1000,2000);
  ESC1.write(95);
  ESC2.write(95);
  ESC3.write(95);
  ESC4.write(95);

  for (;;) {
    if(Serial){
      if (Serial.available() >= 92){ // Ensure at least 92 bytes are available
        for (int i = 0; i < 92; i++) { // Translates Byte array into a float array
          byteArray[i] = Serial.read();
        }
        for (int i = 0; i < 23; i++) {
          byte floatBytes[4];
          for (int j = 0; j < 4; j++) {
            floatBytes[j] = byteArray[i * 4 + j];
          }
          memcpy(&floatArray[i], floatBytes, 4);
          float mappedValue = ((floatArray[1] + 1) * 95);
          float mappedValue2 = ((floatArray[3] + 1) * 95);
          if (floatArray[1] <= 0.051 && floatArray[1] >= -0.071){ //deadzone to account for PS5 Controller stick drift
            ESC1.write(95);// neutral position
            ESC2.write(95);// neutral position
          } else {
            ESC1.write(mappedValue); // PWM inputs from PS5 controller -1 to 1 range.
            ESC2.write(mappedValue); // PWM inputs from PS5 controller -1 to 1 range.
           }
          if (floatArray[3] <= 0.051 && floatArray[3] >= -0.071){ //deadzone to account for PS5 Controller stick drift
            ESC3.write(95); // neutral position
            ESC4.write(95); // neutral position
          } else {
            ESC3.write(mappedValue2); // PWM inputs from PS5 controller -1 to 1 range.
            ESC4.write(mappedValue2); // PWM inputs from PS5 controller -1 to 1 range.
```



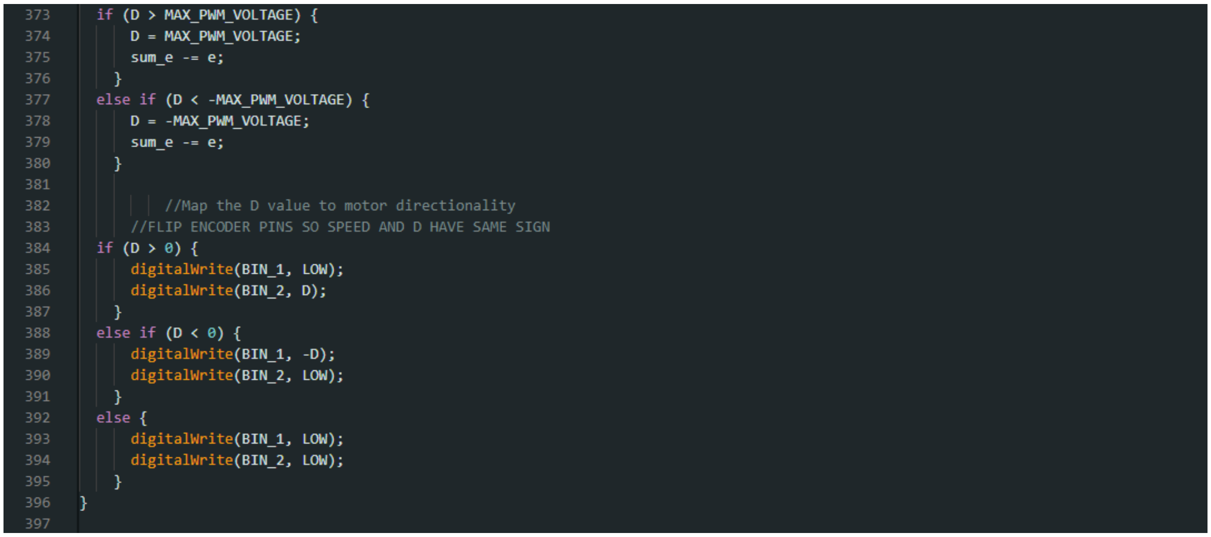
```

          byte* floatBytesToSend = (byte*)&floatArray[i]; // Get the bytes of the float value
          for (int j = 0; j < 4; j++) {
            Serial.write(floatBytesToSend[j]); // Send each byte of the float
          }
        }
      }
  }
  vTaskDelay(pdMS_TO_TICKS(3)); // Adjust delay based on requirements
  }
}

void IRAM_ATTR onTime0() {
  portENTER_CRITICAL_ISR(&timerMux0);
  interruptCounter = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux0);
}

void IRAM_ATTR onTime1() {
  portENTER_CRITICAL_ISR(&timerMux1);
  count = encoder.getCount( );
  encoder.clearCount ( );
  deltaT = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux1);
}

void setup() {
  Serial.begin(115200);

  pinMode(LED1, OUTPUT);
  pinMode(LED2,OUTPUT);
  pinMode(BUT1, INPUT);
  pinMode(BUT2, INPUT);
  pinMode(BUT3, INPUT);
  pinMode(endStop, INPUT);
  pinMode(BIN_1, OUTPUT);
  pinMode(BIN_2, OUTPUT);

  // Initialize LEDs as OFF
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, LOW);

  //Motor Encoder
  ESP32Encoder::useInternalWeakPullResistors = UP; // Enable the weak pull up resistors
  encoder.attachHalfQuad(33, 27); // Attache pins for use as encoder pins
  encoder.setCount(0);  // set starting count value after attaching
```



```
    // initilize timer
  timer0 = timerBegin(0, 80, true);  // timer 0, MWDT clock period = 12.5 ns * TIMGn_Tx_WDT_CLK_PRESCALE -> 12.5 ns * 80 -> 1000 ns = 1 us, countUp
  timerAttachInterrupt(timer0, &onTime0, true); // edge (not level) triggered
  timerAlarmWrite(timer0, 5000000, true); // 5000000 * 1 us = 5 s, autoreload true

  timer1 = timerBegin(1, 80, true);  // timer 1, MWDT clock period = 12.5 ns * TIMGn_Tx_WDT_CLK_PRESCALE -> 12.5 ns * 80 -> 1000 ns = 1 us, countUp
  timerAttachInterrupt(timer1, &onTime1, true); // edge (not level) triggered
  timerAlarmWrite(timer1, 10000, true); // 10000 * 1 us = 10 ms, autoreload true

  // at least enable the timer alarms
  timerAlarmEnable(timer0); // enable
  timerAlarmEnable(timer1); // enable


 // encoderSemaphore = xSemaphoreCreateBinary();
  xTaskCreate(motorControlTask, "MotorControlTask", 10000, NULL, 1, NULL);
  xTaskCreate(PIControllerTask, "PIControllerTask", 15000, NULL, 1, NULL);
}

void loop() {
}

int firstSequence (){
  checkEndStop = digitalRead(endStop);
  if (checkEndStop == LOW){
    return (1);
  }
  if (checkEndStop == HIGH){
    return (2);
  }
}

void secondSequence_HOME (){
  digitalWrite(BIN_2, LOW);
  digitalWrite(BIN_1, MAX_PWM_VOLTAGE);
  delay (110);
  digitalWrite(BIN_1, 0);
  digitalWrite(BIN_2, 0);

  digitalWrite(LED1,HIGH);
  digitalWrite(LED2,LOW);
  delay(14);
  digitalWrite(LED1,LOW);
  digitalWrite(LED2,LOW);
}

```



```
void thirdSequence_HOME (){
  digitalWrite(BIN_1, LOW);
  digitalWrite(BIN_2, MAX_PWM_VOLTAGE);
  delay (110);
  digitalWrite(BIN_1, 0);
  digitalWrite(BIN_2, 0);

  digitalWrite(LED1,LOW);
  digitalWrite(LED2,HIGH);
  delay(14);
  digitalWrite(LED1,LOW);
  digitalWrite(LED2,LOW);
}
void piController(){
  // PI Section ----------------------------------
  //randomInt = random(0, 4096);
  if (interruptCounter) {
    portENTER_CRITICAL(&timerMux0);
    interruptCounter = false;
    portEXIT_CRITICAL(&timerMux0);

    totalInterrupts++;

    if (totalInterrupts%2 == 0) {
      dir = -dir;
    }
  }

  if (deltaT) {
      portENTER_CRITICAL(&timerMux1);
      deltaT = false;
      portEXIT_CRITICAL(&timerMux1);

      omegaSpeed = count;

      //potReading = randomInt;
      omegaDes = 1;
      //omegaDes = map(potReading, 0, 4095, -omegaMax, omegaMax); // PLEASE SPECIFY OMEGAMAX VALUE ABOVE

      e = omegaDes-omegaSpeed;
      sum_e = sum_e + e;
      D = Kp*e + Ki*sum_e;  // REPLACE THIS LINE WITH P/PI CONTROLLER CODE

  }

      //Ensure that you don't go past the maximum possible command
  if (D > MAX_PWM_VOLTAGE) {
```



```
  if (D > MAX_PWM_VOLTAGE) {
      D = MAX_PWM_VOLTAGE;
      sum_e -= e;
    }
  else if (D < -MAX_PWM_VOLTAGE) {
      D = -MAX_PWM_VOLTAGE;
      sum_e -= e;
    }

          //Map the D value to motor directionality
      //FLIP ENCODER PINS SO SPEED AND D HAVE SAME SIGN
  if (D > 0) {
      digitalWrite(BIN_1, LOW);
      digitalWrite(BIN_2, D);
    }
  else if (D < 0) {
      digitalWrite(BIN_1, -D);
      digitalWrite(BIN_2, LOW);
    }
  else {
      digitalWrite(BIN_1, LOW);
      digitalWrite(BIN_2, LOW);
    }
}

```

CAD

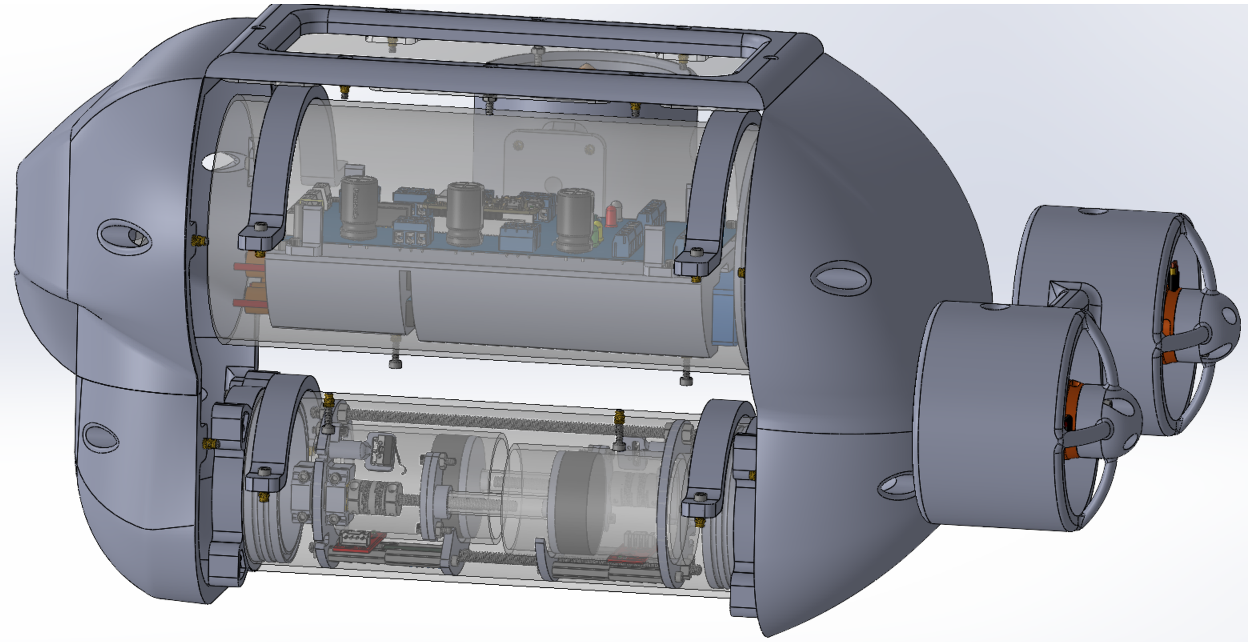


Figure 5: Prometheus Section View

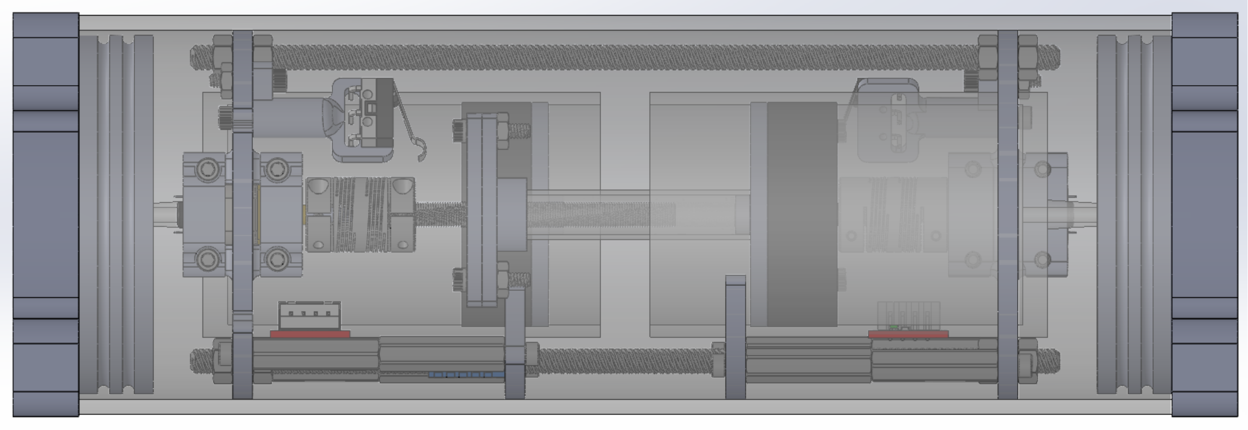


Figure 6: Buoyancy system overview

Pictures of the Device