ME 102B Final Project Report: Tears of Ocean

Tongmiao Xu, Siyuan Chen, Shangtao Li, Nafees Ahamad

Opportunity

Initially, our design tried to solve the problems including the technology of communication-phones, social media, the internet, building the bridge between mechanical design and message convey, expressing concept of protecting water and finally making an example of the mechanical designs intended to convey a message. So, to be more specific, our goal of technology and communication represents the idea of utilizing power of communication which could promote our art design to every corner around the world and therefore it could be able to be seen through the social media. All these concepts in the final version are remaining the same as our first goals.

High-level strategy

In our realized product, we remain most of the functionalities that we intend to do. Here are the functionalities of what have been shown in the first version: Firstly, A row of decorated turning gears symbolizing the ocean rotates from a clean, tumultuous ide to a polluted, lifeless side. Secondary, the sewage in the water storage falls from a pipe making waterfalls from the back to the front tank using a water lifting device and the motor rotates on linear accordance with water weight. Thirdly, Holes between the tanks close at first until the front tank is filled with water to let the water flow back as a recycle. Lastly, led lights turn on when the ocean is clean but turn off when the cycle ends with gears on its lifeless side.

And we revised some of the design details like shape of the gears mentioned in the first point have been switched to the round wooden plate since it's hard and not necessary to cut the wooden plate in the shape of gears.

What's more, we abandon the concept of applying water pump since it's too complicated to make everything waterproof and is doesn't show much relevance towards the courses as those are the suggestions given by professor. We tend to more focus on the mechanical and electrical design including transmission system, circuit itself and switching the water system from pumping to gravity water system.

Photo of the device

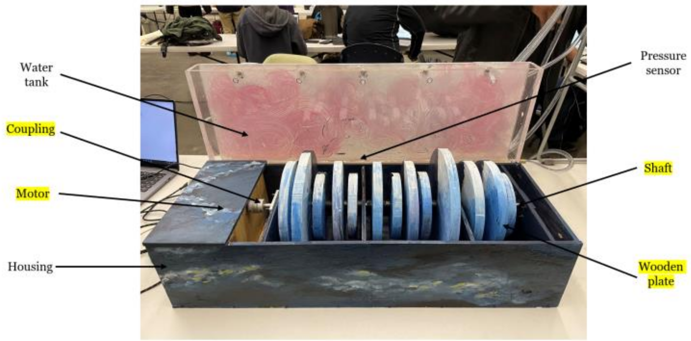


Fig1. Photo of the device

Function-critical decisions

First, we decided to use a stainless-steel shaft as the core of the entire transmission system. Considering the weight of the entire system, the stiffness of an iron shaft is necessary to meet our manufacturing requirements. In addition, we chose to drill holes in the collars to enhance the circumferential stability of the shaft system. Following this, for artistic design considerations, we did not emphasize torque and speed too much in the design. It was sufficient to meet the basic calculations for motor operation. Finally, due to the artistic features of the design, the entire system's structure and framework are entirely made of wood. Compared to iron materials and 3D printing materials, wood is easier to process and has unique artistic processing characteristics.

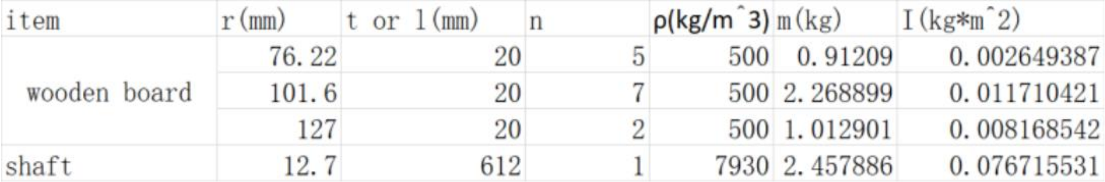
Function-critical calculations

It is known that the equation of the moment of inertia is:

$$I = \frac{1}{2}mr^2 \tag{1}$$

Where I is the moment of inertia and r is the radius of the wooden plate and steel shaft, so based on equations 1 and the design information of the wooden plates shown in Fig1, we could then calculate the moment inertia of the whole transmission system:

$$I_{totoal} = \frac{1}{2}\sum m_i r_i^2 \tag{2}$$



item	r(mm)	t or l(mm)	n	ρ(kg/m^3)	m(kg)	I(kg*m^2)
wooden board	76.22	20	5	500	0.91209	0.002649387
	101.6	20	7	500	2.268899	0.011710421
	127	20	2	500	1.012901	0.008168542
shaft	12.7	612	1	7930	2.457886	0.076715531

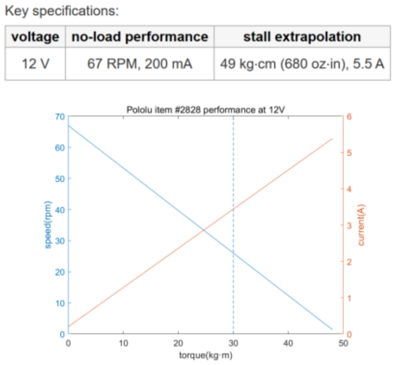
Fig 2. Design parameters of the transmission system

Then we could simply calculate the total moment inertia which is

$$I_{total} = 5 * 0.00265 + 7 * 0.00117 + 2 * 0.00817 + 0.07672 = 11.45\ \text{kg} \times cm^2 \tag{3}$$

$$P = T\omega \tag{4}$$

As an art design, the angular velocity doesn’t require an extremely high value, it’s reasonable to set it as 2rpm since the rotate represent the pollute of the sea, so based on Fig 2 we can conclude that while the speed is 2rpm, the maximum operating torque is 30kg*m which is quite safe based on the equation4. Later we calculated and found out the maximum power at 2rpm is much lower than the power of 25rpm. So, it will be quite safe based on the torque calculation.



Key specifications:

voltage	no-load performance	stall extrapolation
12 V	67 RPM, 200 mA	49 kg·cm (680 oz·in), 5.5 A

Fig 3. The speed-torque relationship of the motor

Updated circuit diagram

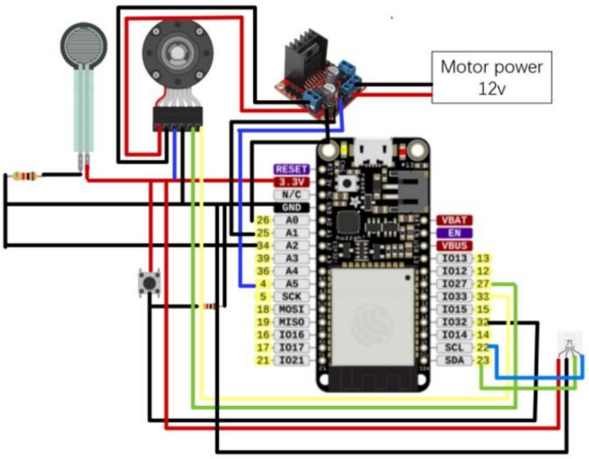


Fig4. Updated circuit diagram

State transition diagram

Variable: Position{0,5000} ; Luminosity{0,31} **Inputs**: Button: pure; Taint: R

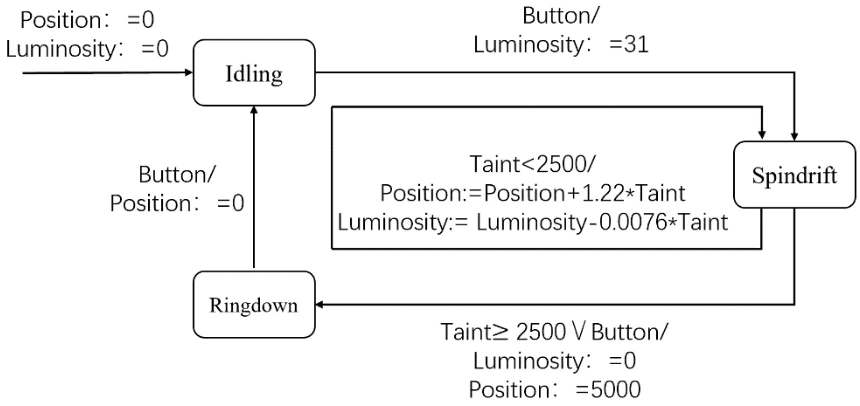
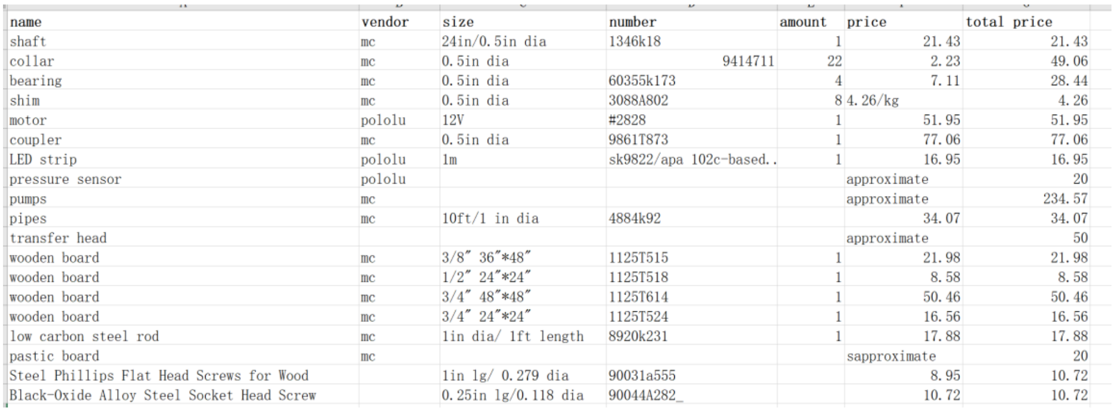


Fig5. State Diagram

Reflection

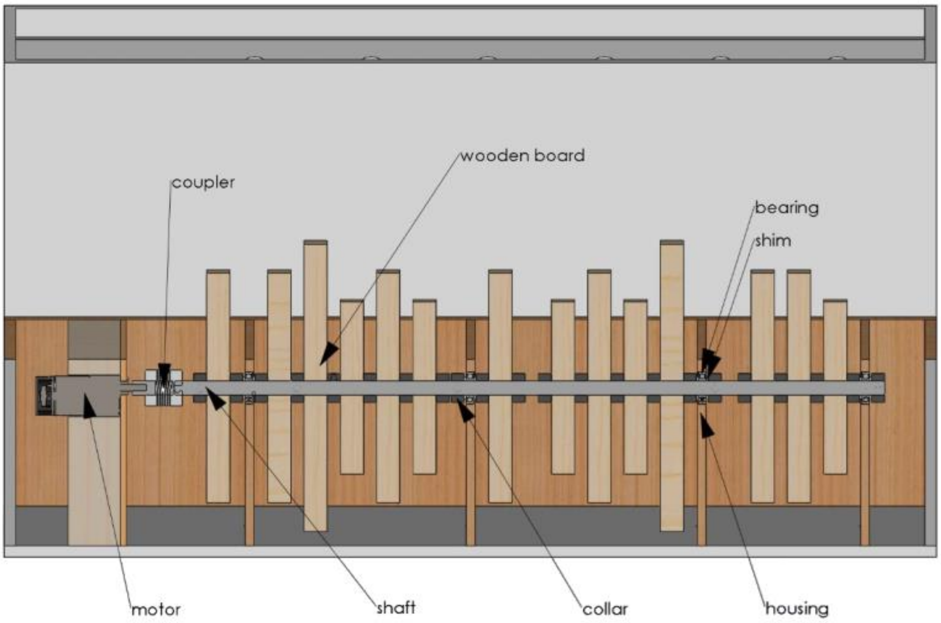
Our group functions well with effective division of tasks, but it lacks the leadership of a group leader. It is suggested that the group elects a leader in the future to enhance the efficiency of task allocation.

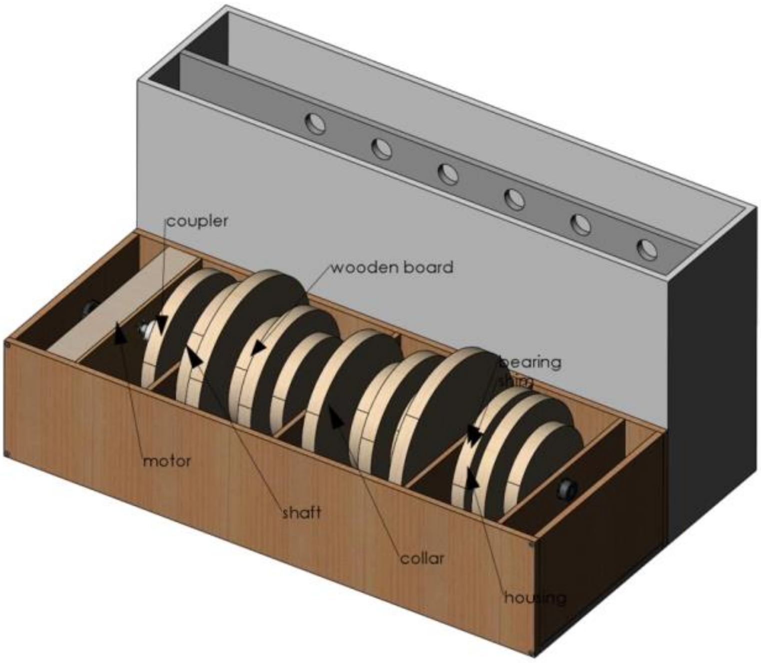
Appendix A. Bill of Materials



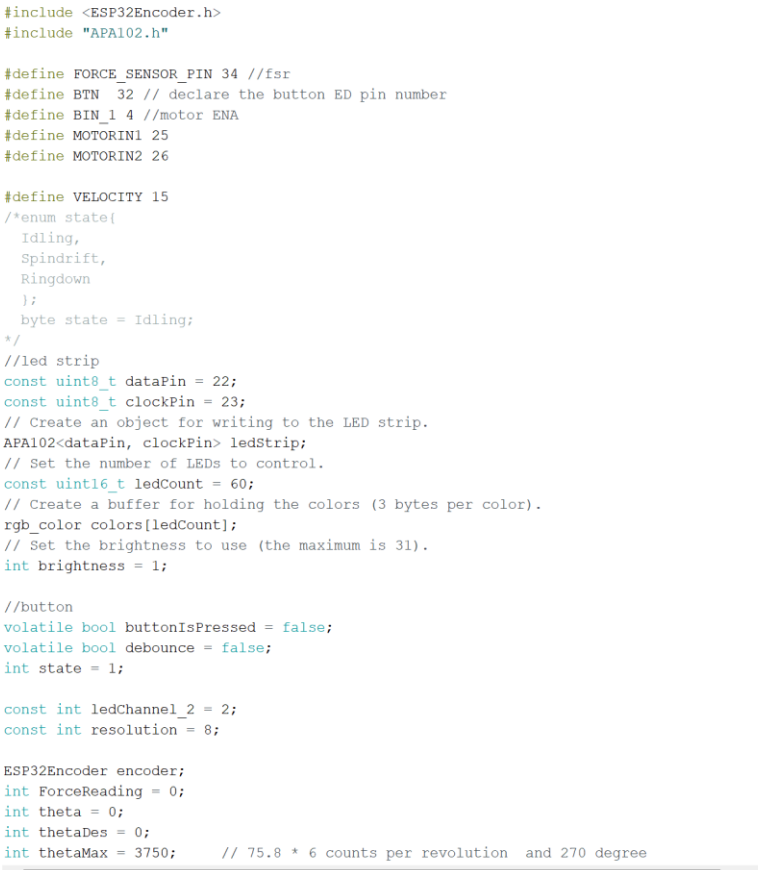
name	vendor	size	number	amount	price	total price
shaft	mc	24in/0.5in dia	1346k18	1	21.43	21.43
collar	mc	0.5in dia	9414711	22	2.23	49.06
bearing	mc	0.5in dia	60355k173	4	7.11	28.44
shim	mc	0.5in dia	3088A802	8	4.26/kg	4.26
motor	pololu	12V	#2828	1	51.95	51.95
coupler	mc	0.5in dia	9861T873	1	77.06	77.06
LED strip	pololu	1m	sk9822/apa 102c-based..	1	16.95	16.95
pressure sensor	pololu				approximate	20
pumps	mc				approximate	234.57
pipes	mc	10ft/1 in dia	4884k92		34.07	34.07
transfer head					approximate	50
wooden board	mc	3/8″ 36″*48″	1125T515	1	21.98	21.98
wooden board	mc	1/2″ 24″*24″	1125T518	1	8.58	8.58
wooden board	mc	3/4″ 48″*48″	1125T614	1	50.46	50.46
wooden board	mc	3/4″ 24″*24″	1125T524	1	16.56	16.56
low carbon steel rod	mc	1in dia/ 1ft length	8920k231	1	17.88	17.88
pastic board	mc				sapproximate	20
Steel Phillips Flat Head Screws for Wood		1in lg/ 0.279 dia	90031a555		8.95	10.72
Black-Oxide Alloy Steel Socket Head Screw		0.25in lg/0.118 dia	90044A282_		10.72	10.72

Appendix B. CAD

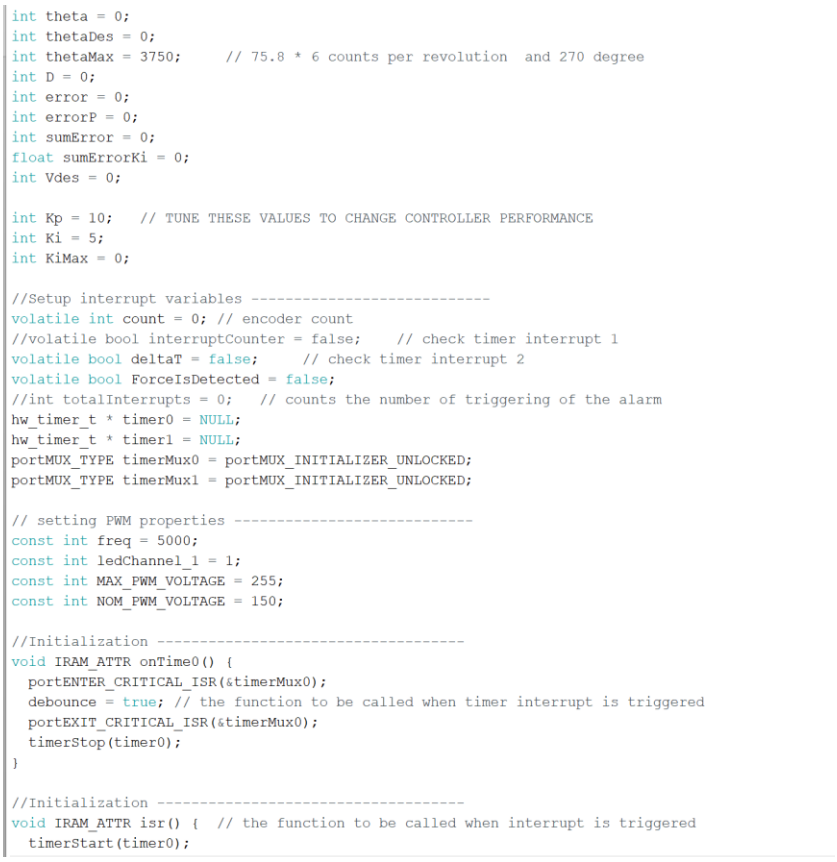




Appendix C. Code

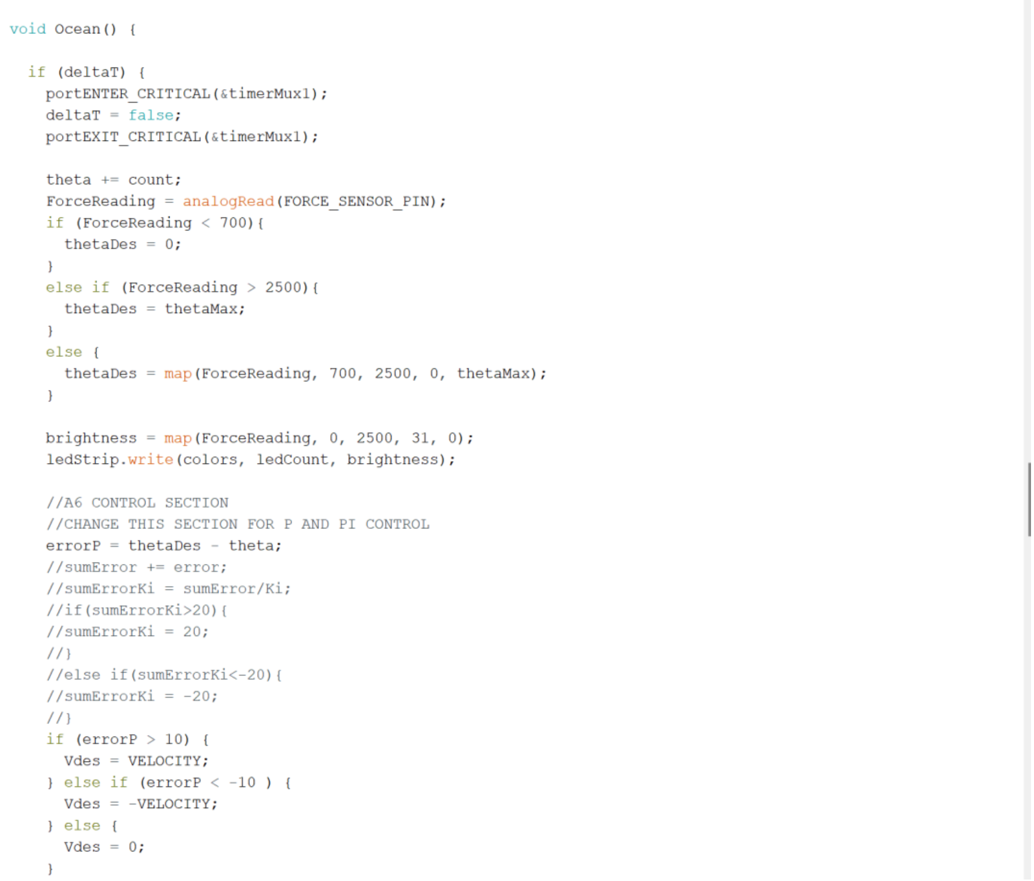


```
#include <ESP32Encoder.h>
#include "APA102.h"

#define FORCE_SENSOR_PIN 34 //fsr
#define BTN  32 // declare the button ED pin number
#define BIN_1 4 //motor ENA
#define MOTORIN1 25
#define MOTORIN2 26

#define VELOCITY 15
/*enum state{
  Idling,
  Spindrift,
  Ringdown
  };
  byte state = Idling;
*/
//led strip
const uint8_t dataPin = 22;
const uint8_t clockPin = 23;
// Create an object for writing to the LED strip.
APA102<dataPin, clockPin> ledStrip;
// Set the number of LEDs to control.
const uint16_t ledCount = 60;
// Create a buffer for holding the colors (3 bytes per color).
rgb_color colors[ledCount];
// Set the brightness to use (the maximum is 31).
int brightness = 1;

//button
volatile bool buttonIsPressed = false;
volatile bool debounce = false;
int state = 1;

const int ledChannel_2 = 2;
const int resolution = 8;

ESP32Encoder encoder;
int ForceReading = 0;
int theta = 0;
int thetaDes = 0;
int thetaMax = 3750;     // 75.8 * 6 counts per revolution  and 270 degree
```

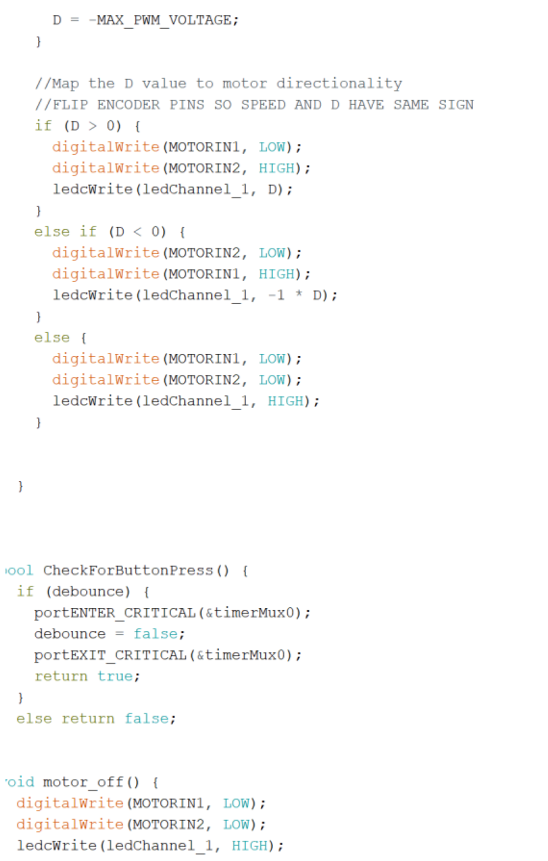


```
int theta = 0;
int thetaDes = 0;
int thetaMax = 3750;     // 75.8 * 6 counts per revolution  and 270 degree
int D = 0;
int error = 0;
int errorP = 0;
int sumError = 0;
float sumErrorKi = 0;
int Vdes = 0;

int Kp = 10;   // TUNE THESE VALUES TO CHANGE CONTROLLER PERFORMANCE
int Ki = 5;
int KiMax = 0;

//Setup interrupt variables ----------------------------
volatile int count = 0; // encoder count
//volatile bool interruptCounter = false;    // check timer interrupt 1
volatile bool deltaT = false;     // check timer interrupt 2
volatile bool ForceIsDetected = false;
//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;

// setting PWM properties ----------------------------
const int freq = 5000;
const int ledChannel_1 = 1;
const int MAX_PWM_VOLTAGE = 255;
const int NOM_PWM_VOLTAGE = 150;

//Initialization ------------------------------------
void IRAM_ATTR onTime0() {
  portENTER_CRITICAL_ISR(&timerMux0);
  debounce = true; // the function to be called when timer interrupt is triggered
  portEXIT_CRITICAL_ISR(&timerMux0);
  timerStop(timer0);
}

//Initialization ------------------------------------
void IRAM_ATTR isr() {  // the function to be called when interrupt is triggered
  timerStart(timer0);
  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() {
  // put your setup code here, to run once:
  pinMode(BTN, INPUT);
  pinMode(MOTORIN1, OUTPUT);
  pinMode(MOTORIN2, OUTPUT);
  pinMode(FORCE_SENSOR_PIN, INPUT);
  attachInterrupt(BTN, isr, RISING);

  Serial.begin(115200);
  ESP32Encoder::useInternalWeakPullResistors = UP; // Enable the weak pull up resistors
  encoder.attachFullQuad(33, 27); // Attache pins for use as encoder pins
  encoder.setCount(0);  // set starting count value after attaching

  // configure LED PWM functionalitites
  ledcSetup(ledChannel_1, freq, resolution);
  //ledcSetup(ledChannel_2, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(BIN_1, ledChannel_1);
  //ledcAttachPin(BIN_2, ledChannel_2);

  timer0 = timerBegin(0, 80, true);  // timer 0, MWDT clock period = 12.5 ns * TIMGn_Tx_WDT_CLK_PRESCALE -> 12
  timerAttachInterrupt(timer0, &onTime0, true); // edge (not level) triggered
  timerAlarmWrite(timer0, 500000, true);

  timer1 = timerBegin(1, 80, true);  // timer 1, MWDT clock period = 12.5 ns * TIMGn_Tx_WDT_CLK_PRESCALE -> 12
  timerAttachInterrupt(timer1, &onTime1, true); // edge (not level) triggered
  timerAlarmWrite(timer1, 5000, true); // 10000 * 1 us = 10 ms, autoreload true

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



rgb_color hsvToRgb(uint16_t h, uint8_t s, uint8_t v)
{
  uint8_t f = (h % 60) * 255 / 60;
  uint8_t p = (255 - s) * (uint16_t)v / 255;
  uint8_t q = (255 - f * (uint16_t)s / 255) * (uint16_t)v / 255;
  uint8_t t = (255 - (255 - f) * (uint16_t)s / 255) * (uint16_t)v / 255;
  uint8_t r = 0, g = 0, b = 0;
  switch ((h / 60) % 6) {
    case 0: r = v; g = t; b = p; break;
    case 1: r = q; g = v; b = p; break;
    case 2: r = p; g = v; b = t; break;
    case 3: r = p; g = q; b = v; break;
    case 4: r = t; g = p; b = v; break;
    case 5: r = v; g = p; b = q; break;
  }
  return rgb_color(r, g, b);
}

void loop() {

  switch (state) {

    case 1://Idling:
      PositionToFull();
      if (CheckForButtonPress() == true) {
        theta = 0;
        ledStrip_on2();
        state = 2;//Spindrift;
      }
      break;


    case 2://Spindrift:
      if (analogRead(FORCE_SENSOR_PIN) < 2500) {
        Ocean();

      if (analogRead(FORCE_SENSOR_PIN) > 2500 || CheckForButtonPress() == true ) {
        ledStrip_off();
        state = 3;//Ringdown ;
      }
      break;
    case 3://Ringdown:
      PositionToOriginal();
      if (CheckForButtonPress() == true) {
        state = 1;//Idling;
      }
      break;

  }
}
  plotControlData();

//Other functions
}

void plotControlData() {
  Serial.print("state:");
  Serial.print(state);
  Serial.print(" ");
  Serial.print("ForceReading:");
  Serial.print(ForceReading);
  Serial.print(" ");
  Serial.print("Position:");
  Serial.print(theta);
  Serial.print(" ");
  Serial.print("Desired_Position:");
  Serial.print(thetaDes);
  Serial.print(" ");
  Serial.print("D:");
  Serial.println(D);


}


void ledStrip_on2(){
  for (uint16_t i = 0; i < 10; i++)
  {
    uint16_t c = i*6;
    for (uint16_t y = 0; y < 6; y++){
      switch(y){
```

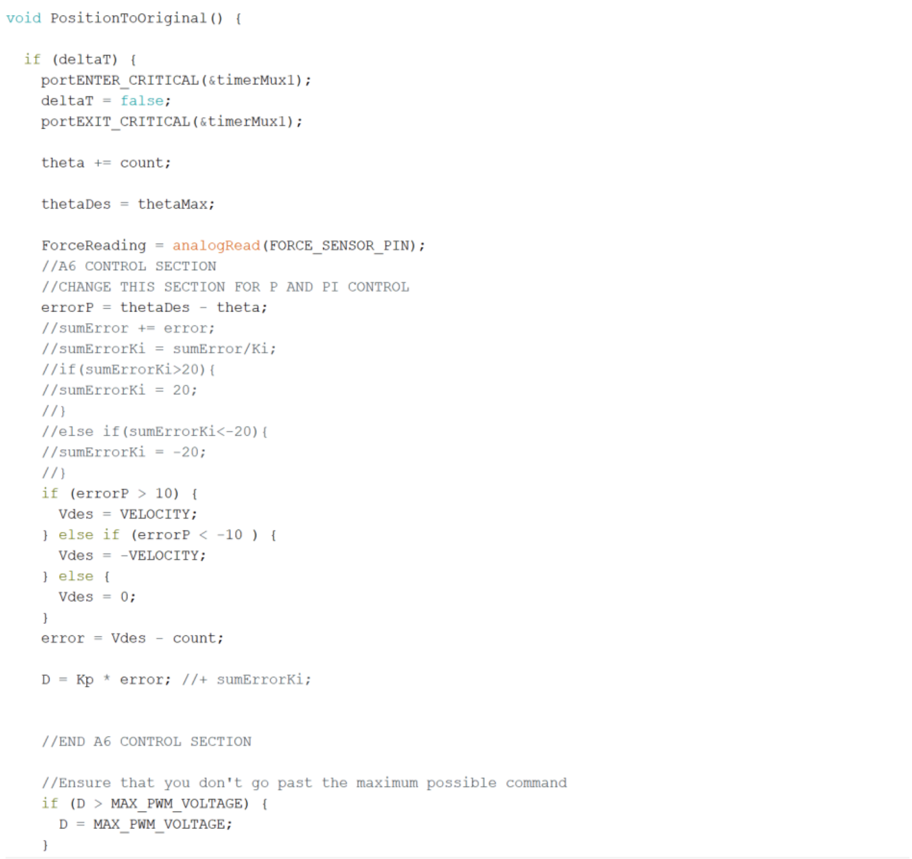
```
void ledStrip_on2(){
  for (uint16_t i = 0; i < 10; i++)
  {
    uint16_t c = i*6;
    for (uint16_t y = 0; y < 6; y++){
      switch(y){
        case 0 :
          colors[c+y]=rgb_color(255 ,245 ,245);
          break;
        case 1 :
          colors[c+y]=rgb_color(170,144,25);
          break;
        case 2 :
          colors[c+y]=rgb_color(50,20,5);
          break;
        case 3 :
          colors[c+y]=rgb_color(255,50 ,50);
          break;
        case 4 :
          colors[c+y]=rgb_color(50,20,5);
          break;
        case 5 :
          colors[c+y]=rgb_color(170,144,25);
          break;

      }

    }
  }
  ledStrip.write(colors, ledCount, brightness);
}

void ledStrip_on() {

  uint8_t time = millis() >> 4;

  for (uint16_t i = 0; i < ledCount; i++)
  {
    uint8_t p = time - i * 8;
    colors[i] = hsvToRgb((uint32_t)p * 359 / 256, 255, 255);
  }
```



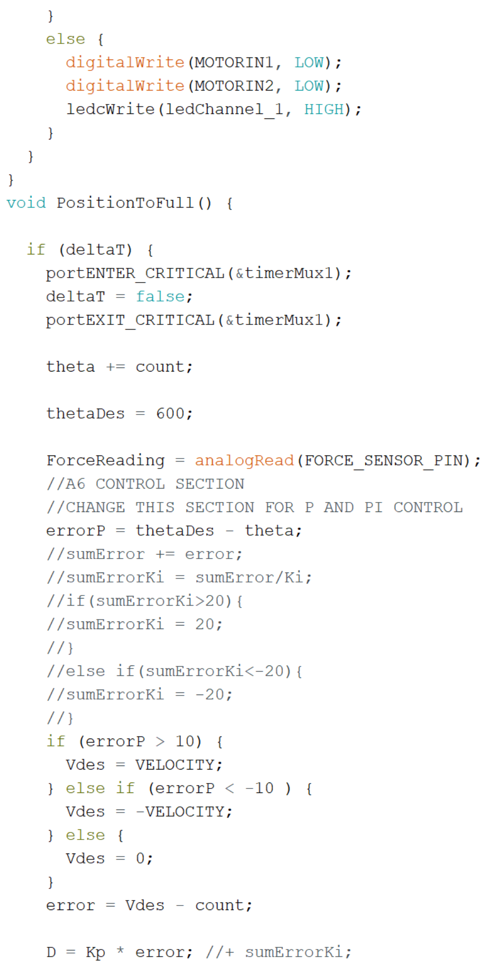
```
void Ocean() {

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

    theta += count;
    ForceReading = analogRead(FORCE_SENSOR_PIN);
    if (ForceReading < 700){
      thetaDes = 0;
    }
    else if (ForceReading > 2500){
      thetaDes = thetaMax;
    }
    else {
      thetaDes = map(ForceReading, 700, 2500, 0, thetaMax);
    }

    brightness = map(ForceReading, 0, 2500, 31, 0);
    ledStrip.write(colors, ledCount, brightness);

    //A6 CONTROL SECTION
    //CHANGE THIS SECTION FOR P AND PI CONTROL
    errorP = thetaDes - theta;
    //sumError += error;
    //sumErrorKi = sumError/Ki;
    //if(sumErrorKi>20){
    //sumErrorKi = 20;
    //}
    //else if(sumErrorKi<-20){
    //sumErrorKi = -20;
    //}
    if (errorP > 10) {
      Vdes = VELOCITY;
    } else if (errorP < -10 ) {
      Vdes = -VELOCITY;
    } else {
      Vdes = 0;
    }
```



```
    D = -MAX_PWM_VOLTAGE;
  }

  //Map the D value to motor directionality
  //FLIP ENCODER PINS SO SPEED AND D HAVE SAME SIGN
  if (D > 0) {
    digitalWrite(MOTORIN1, LOW);
    digitalWrite(MOTORIN2, HIGH);
    ledcWrite(ledChannel_1, D);
  }
  else if (D < 0) {
    digitalWrite(MOTORIN2, LOW);
    digitalWrite(MOTORIN1, HIGH);
    ledcWrite(ledChannel_1, -1 * D);
  }
  else {
    digitalWrite(MOTORIN1, LOW);
    digitalWrite(MOTORIN2, LOW);
    ledcWrite(ledChannel_1, HIGH);
  }


}



ool CheckForButtonPress() {
  if (debounce) {
    portENTER_CRITICAL(&timerMux0);
    debounce = false;
    portEXIT_CRITICAL(&timerMux0);
    return true;
  }
  else return false;


oid motor_off() {
  digitalWrite(MOTORIN1, LOW);
  digitalWrite(MOTORIN2, LOW);
  ledcWrite(ledChannel_1, HIGH);
```



```
void PositionToOriginal() {

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

    theta += count;

    thetaDes = thetaMax;

    ForceReading = analogRead(FORCE_SENSOR_PIN);
    //A6 CONTROL SECTION
    //CHANGE THIS SECTION FOR P AND PI CONTROL
    errorP = thetaDes - theta;
    //sumError += error;
    //sumErrorKi = sumError/Ki;
    //if(sumErrorKi>20){
    //sumErrorKi = 20;
    //}
    //else if(sumErrorKi<-20){
    //sumErrorKi = -20;
    //}
    if (errorP > 10) {
      Vdes = VELOCITY;
    } else if (errorP < -10 ) {
      Vdes = -VELOCITY;
    } else {
      Vdes = 0;
    }
    error = Vdes - count;

    D = Kp * error; //+ sumErrorKi;


    //END A6 CONTROL SECTION

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



```
    }
    else {
      digitalWrite(MOTORIN1, LOW);
      digitalWrite(MOTORIN2, LOW);
      ledcWrite(ledChannel_1, HIGH);
    }
  }
}
void PositionToFull() {

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

    theta += count;

    thetaDes = 600;

    ForceReading = analogRead(FORCE_SENSOR_PIN);
    //A6 CONTROL SECTION
    //CHANGE THIS SECTION FOR P AND PI CONTROL
    errorP = thetaDes - theta;
    //sumError += error;
    //sumErrorKi = sumError/Ki;
    //if(sumErrorKi>20){
    //sumErrorKi = 20;
    //}
    //else if(sumErrorKi<-20){
    //sumErrorKi = -20;
    //}
    if (errorP > 10) {
      Vdes = VELOCITY;
    } else if (errorP < -10 ) {
      Vdes = -VELOCITY;
    } else {
      Vdes = 0;
    }
    error = Vdes - count;

    D = Kp * error; //+ sumErrorKi;
```