BMW Golf Ball Collector & Sorter

ME102B Final Project

Felisha Jacobus, Joey Chen, Prenda Zhang

Dept. of Mechanical Engineering, University of California at Berkeley

(Date: 12/14/2023)

1. INTRODUCTION

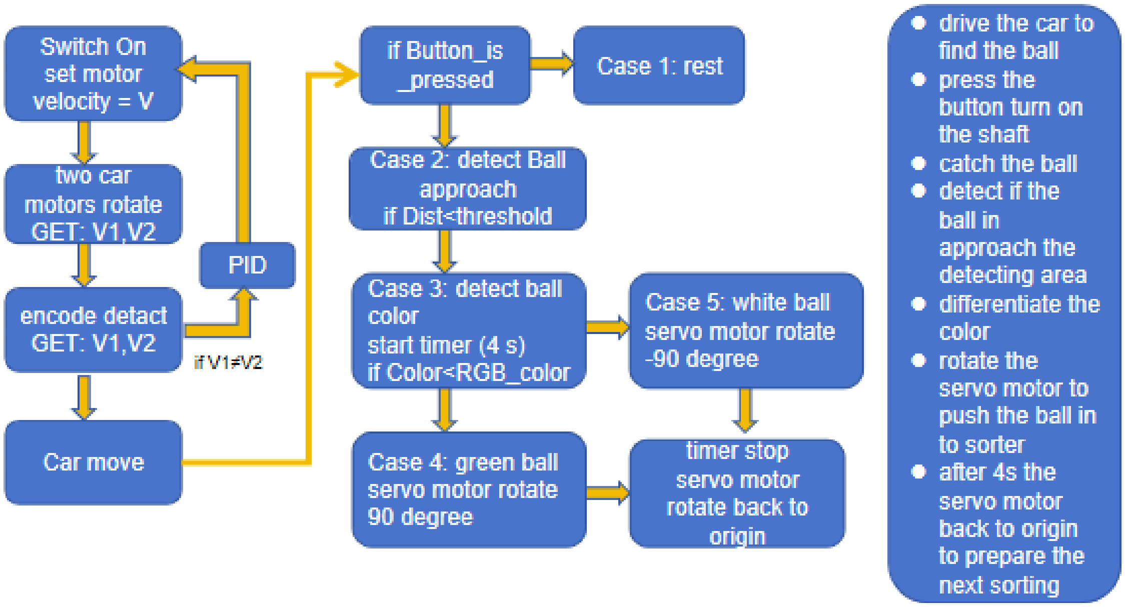
1.1 Problem

Miniature golf or table tennis (ping-pong) is a game that involves the utilization of several balls. The sport entails the likelihood of the balls being dispersed on the ground, making it redundant for players to retrieve them after each game. It's worth mentioning that in miniature golf, the sport employs balls of various colors.

1.2 Chosen Opportunity: Building a robot car that helps pick up ping-pong balls in the table tennis room after each game and sort the ball based on colors.

2. METHODS

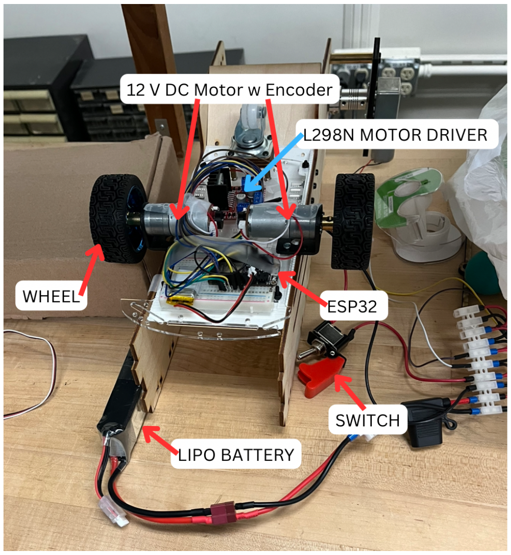
2.1 Device High Level Strategy

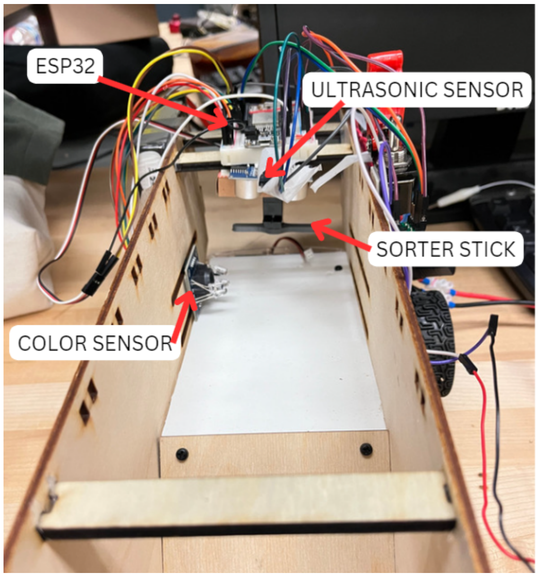

Switch On set motor velocity = V
two car motors rotate GET: V1,V2
encode detect GET: V1,V2
PID
if V1≠V2
Car move
if Button_is _pressed
Case 1: rest
Case 2: detect Ball approach if Dist<threshold
Case 3: detect ball color start timer (4 s) if Color<RGB_color
Case 4: green ball servo motor rotate 90 degree
Case 5: white ball servo motor rotate -90 degree
timer stop servo motor rotate back to origin
drive the car to find the ball
press the button turn on the shaft
catch the ball
detect if the ball in approach the detecting area
differentiate the color
rotate the servo motor to push the ball in to sorter
after 4s the servo motor back to origin to prepare the next sorting

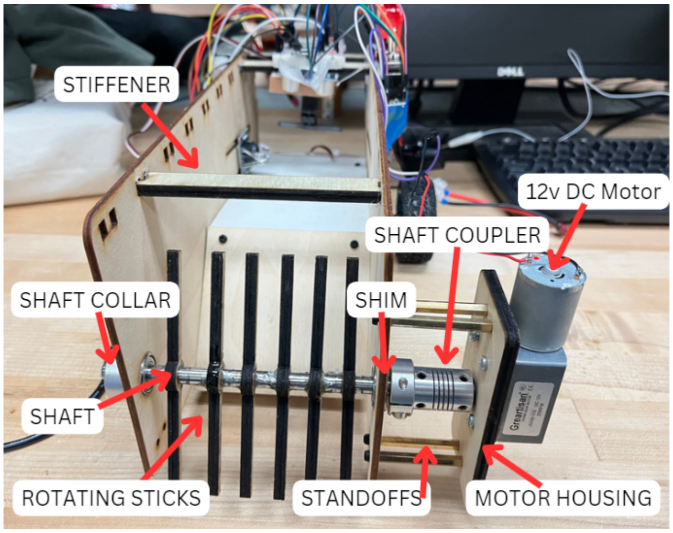
In the practical execution of the project, the set threshold for the color sensor had to undergo several tuning and adjustments. Initially, our goal was for the color sensor to detect any two selected colors. However, due to the physical constraints of the color sensor, it became necessary to establish distinct thresholds for the two balls, choosing colors from opposite spectrums—one light and one dark.

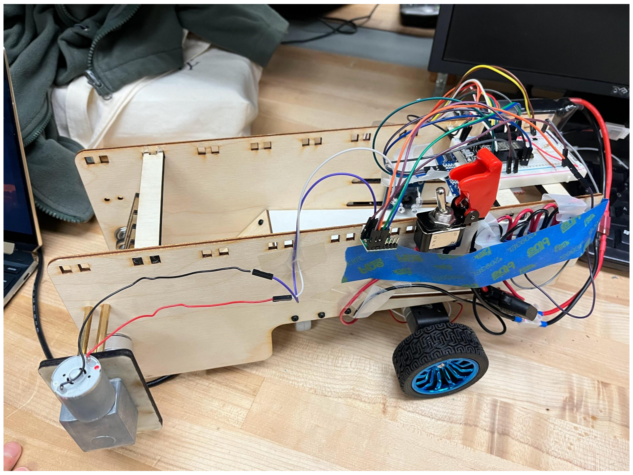
3. PHYSICAL RESULTS

3.1 PHYSICAL DEVICE









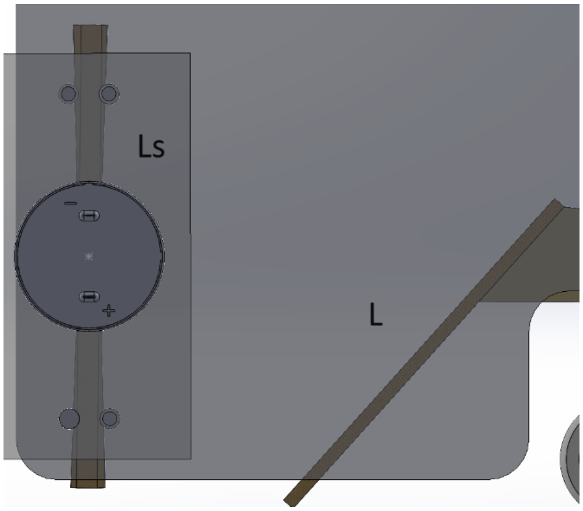
4. RELEVANT DECISIONS / CALCULATIONS / DIAGRAMS

4.1 FUNCTIONAL-CRITICAL DECISIONS

A critical decision involved selecting the right motor for the shaft to gather balls from the ground. Without the right motor, the project would have been unsuccessful, highlighting the importance of calculating torque and motor speed. Additionally, we considered the forces acting on the bearings connected to the shaft.

4.2 RELEVANT CALCULATIONS

Motor Torque / Speed Calculations for Shaft:



Given:

Mass of ball (m)	= 0.0027 kg
Length of ramp (L)	= 100 mm
Angle of ramp (θ)	= 30 deg
Length of Rotating Stick (Ls)	= 100 mm

By Conservation of Energy:

WORK = mgh

F.d = mgh

F = $\frac{mgh}{d} = \frac{0.0027\,kg\,(9.81)\,(0.1\,sin\,(30))}{0.1} = 0.019683\,N$

$Torque\ = F.\frac{Ls}{2} = 3.344\,x\,10^{-4}\,Nm$

$Actual\,Torque\ = \frac{Torque}{0.60}$ = 5.57 x 10^{-4} Nm

By Power Formula:

Given:

Measured current	= 0.72 A
Voltage	= 12 V

P = V.I = 8.64 W = 0.00864 kW

By Power-Torque-Velocity Relationship:

$Torque\ = 9.5488.\frac{Power\,(kW)}{\omega\,(RPM)}$

$\omega\ = 9.5488.\frac{Power\,(kW)}{Torque}$

$\omega\ = 9.5488.\frac{0.00864\,kW}{5.57\,x\,10^{-4}\,Nm}$ = 148.12 RPM

Actual $\omega\ = \frac{\omega}{0.6}$ = 246.86 RPM

Selected Motor:
Torque 5 kg.cm >>> 5.57 x 10^{-4} Nm
250 RPM ≈ 246.86 RPM

Calculation for Bearings:
The parameter is same as previous:

$$\sum F_X = \mu(m_{ball} + F\,sin60) \le F\,cos\,60$$

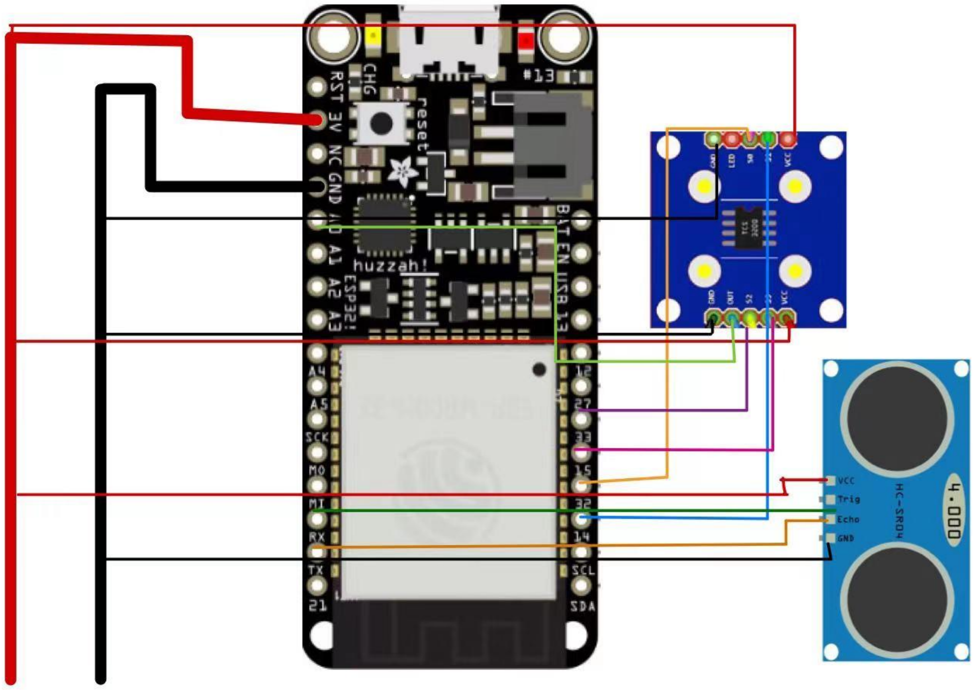
$$\sum F_Y = m_{ball}g + F\,sin60 - N = 0$$

$$\sum M = m_{ball}g + F\,sin60\,L_{shaft} + M_{bearing} = \underline{\boldsymbol{0}}$$

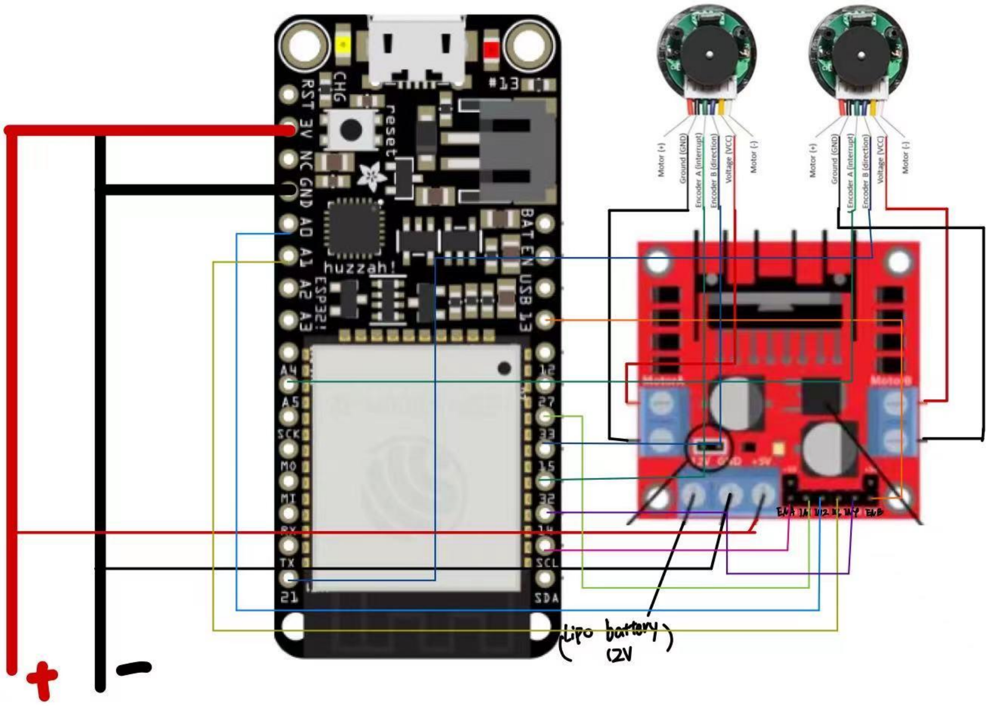
In the result, F in the bearing should larger than 1 N

4.3 CIRCUIT DIAGRAM

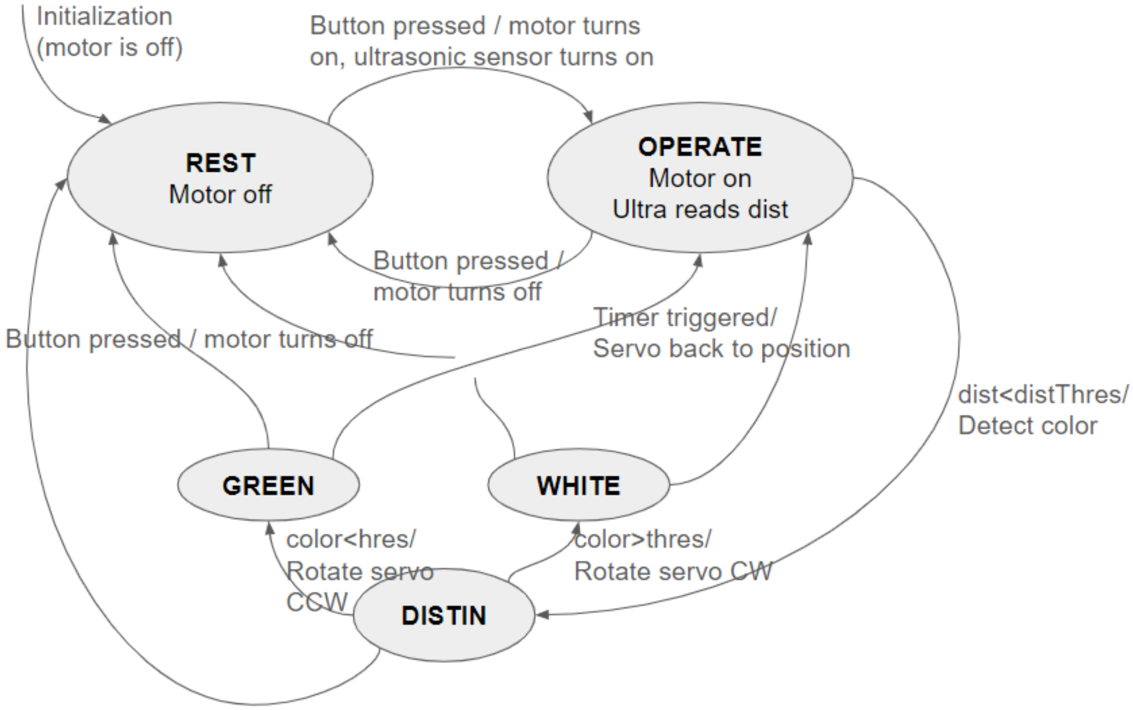
4.3.1 ULTRASONIC & COLOR SENSOR DIAGRAM



4.3.2 MOTOR & WHEEL TRANSMISSION DIAGRAM



4.4 STATE TRANSITION DIAGRAM



5. REFLECTION

The team thoroughly enjoyed working on this project and gained valuable insights throughout the process of designing, manufacturing, and building. Our successful strategy involved incorporating fasteners and pre-designed holes, simplifying the assembly of our car. A key design lesson learned was to always consider ample hand room in the design phase to facilitate smoother assembly. Reflecting on the overall project, we acknowledge that investing in more reliable, albeit potentially pricier, parts, especially the color sensor, would have been beneficial. Integrating the color sensor posed challenges due to its limited ability to detect color at a distance beyond our initial expectations. Additionally, we needed to modify the car's design to accommodate our shaft motor. Selecting the appropriate motor and conducting meticulous calculations from the outset can significantly save time. Despite this adjustment, the overall project proceeded smoothly, and we would recommend it.

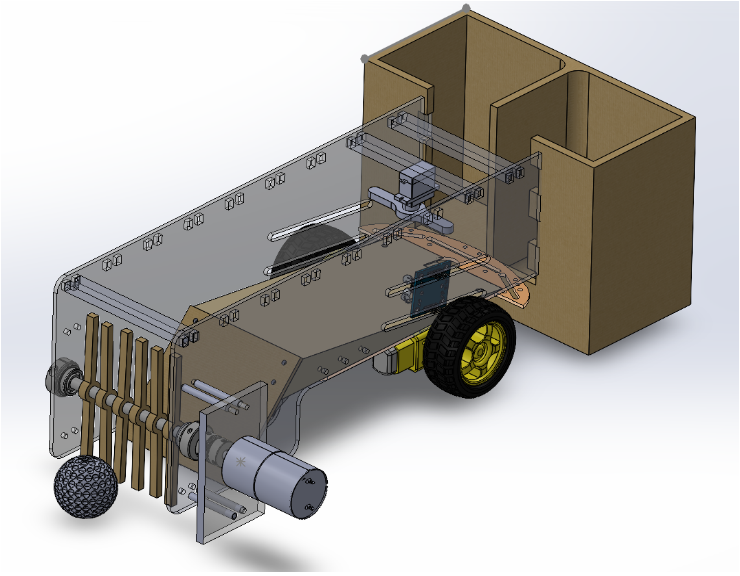
6. Appendix

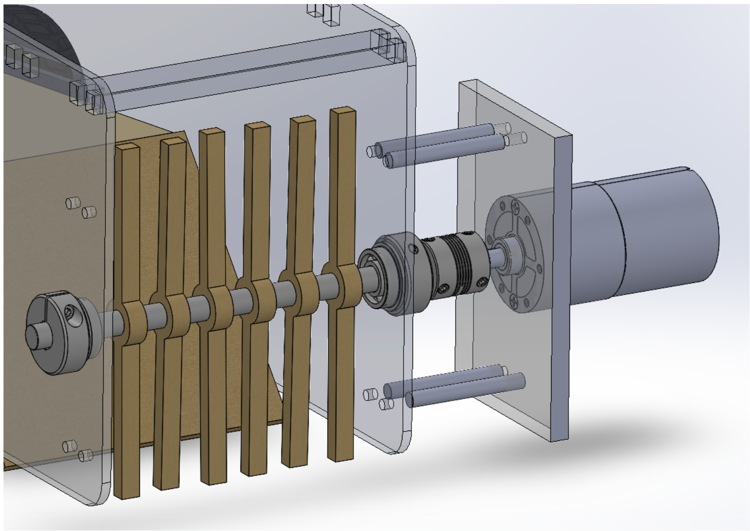
Appendix A: BILL OF MATERIALS

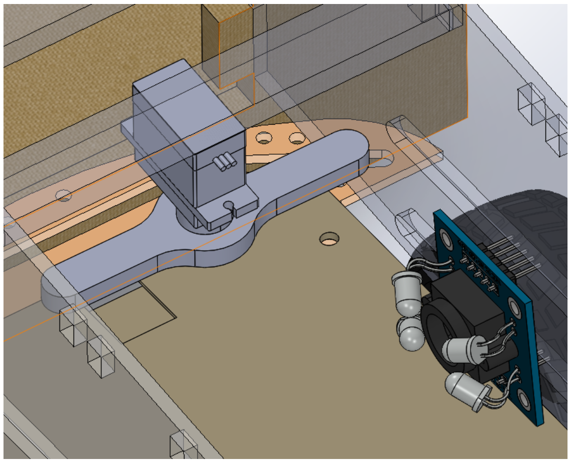
ITEM	MODEL	QUANTITY	COST	LINK
Robot Car Chassis Kit	YIKESHU 2WD Smart Robot Car Chassis Kit with Speed Encoder Battery Box 2 Wheels	1	Previously owned	Link
Shaft	8mmX150mm Linear Motion Rod Shaft	1	$8.99	Link
Bearings	uxcell F698ZZ Flanged Ball Bearing 8x19x6mm	1	$6.49	Link
Flexible Shaft Coupler	Hahiyo 6mm to 8mm Inner Diameter Shaft Couplings Flexible	1	$7.99	https://a.co/d/5jRkx6k
Shim	TRB RC M8x10mm Dia. Steel Shim Pack 10ea 0.1, 0.2, 0.3, 0.5mm Width	1	$9.69	Link
Shaft Collar	HARFINGTON 2pcs Shaft Collar 8mm Bore, 25mm OD, 10mm Width	1	$12.98	Link

Standoff Spacers	Female to Female Brass Hex Nut Spacer Screws Brass Hex Standoff M3 x 40mm, Pack of 20	1	$11.01	https://a.co/d/2GKxsVI
Color Sensor	Teyleten Robot GY-31 TCS3200 TCS230 Color Sensor Module	1	$9.88	Link
Motor (Shaft Transmission)	Greartisan DC 12V 250RPM Turbo Worm Geared Motor High Torque Turbine Worm Gear Box Reduction Motor 6mm Shaft JSX40-370	1	$14.99	https://a.co/d/1Jhl51T
Motor (Wheel Transmission)	DC 12V DIY Encoder Gear Motor...	1	$47.68	Link
Colored Balls	28 Pcs Colored Ping Pong Balls, 40mm Table Tennis Balls,Ping Pong Balls for Game or Arts, Pong Balls for Kids,Pet Toys	1	$8.81	https://a.co/d/6MSlGAT
M3 Screw Set	yddmyo Wall Anchor and Screw Kit M8*40 60 and M6*30 40 60 80 Lengths for Drywall Sort Kit Drywall Drywall Mounting Screws (M3 Outer Corner Nylon Screws) https://a.co/d/7Tq0UMu	1	$5.50	https://a.co/d/7Tq0UMu
Jacobs Plywood		2	$14.57	
Lipo Battery	OVONIC 3S Lipo Battery 50C 3000mAh 11.1V Lipo Battery with Dean-Style T Connector for RC Airplane Helicopter Quadcopter RC Car Truck Boat(2 Packs)	1	Previously owned	https://a.co/d/334of9a
Glue		2	$3.76	
Servo Motor		1	Lab Kit	
Ultrasonic Sensor		1	Lab Kit	
Breadboard		2	Lab Kit	
ESP32		2	Lab Kit	
Jumper Wires		multiple	Lab Kit	

Appendix B: CAD Images

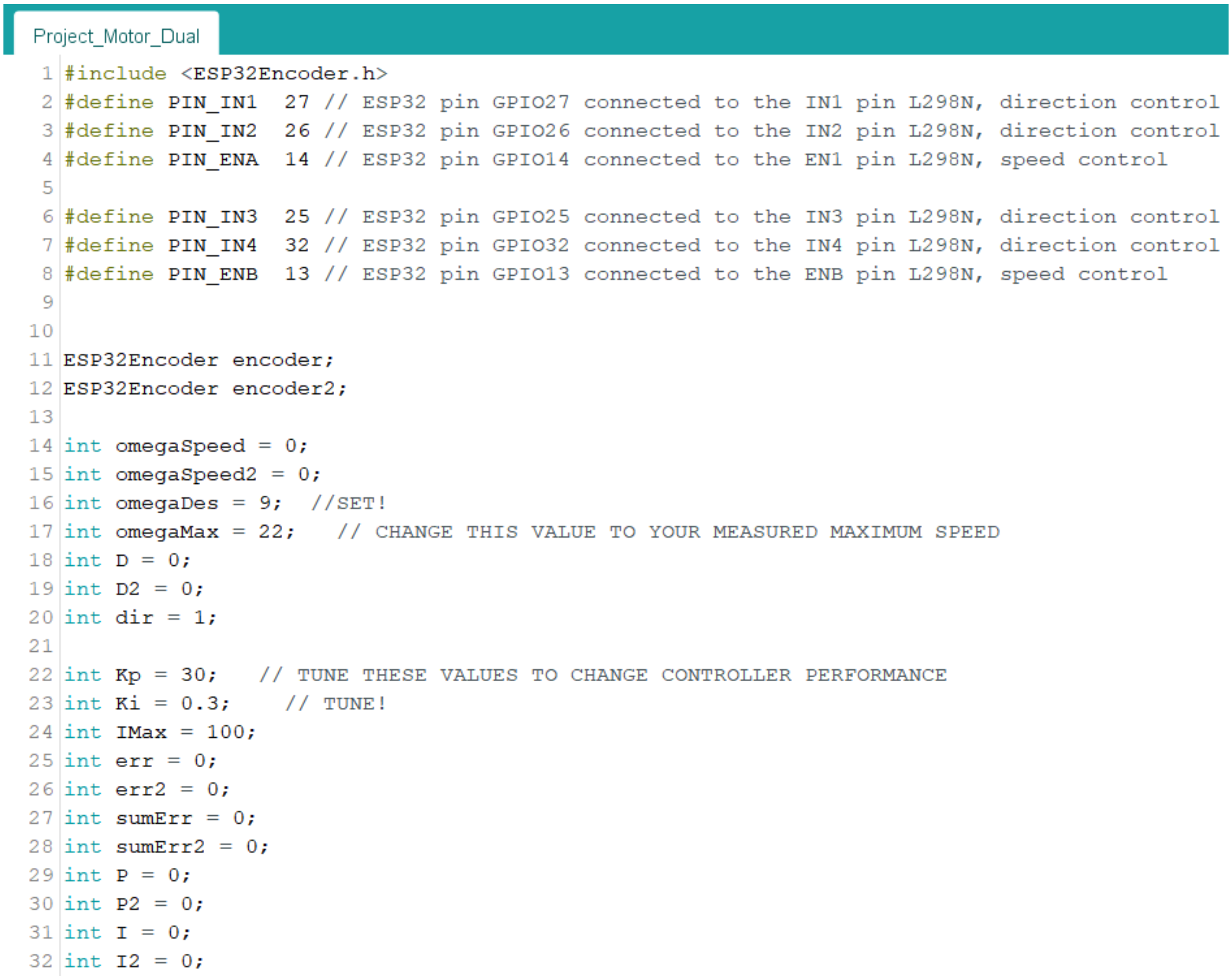




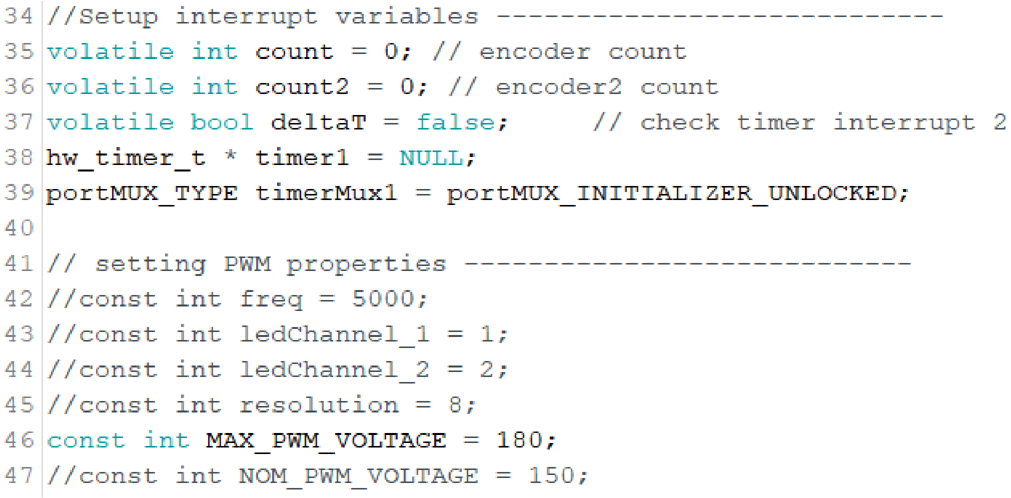


Appendix C: Arduino Code

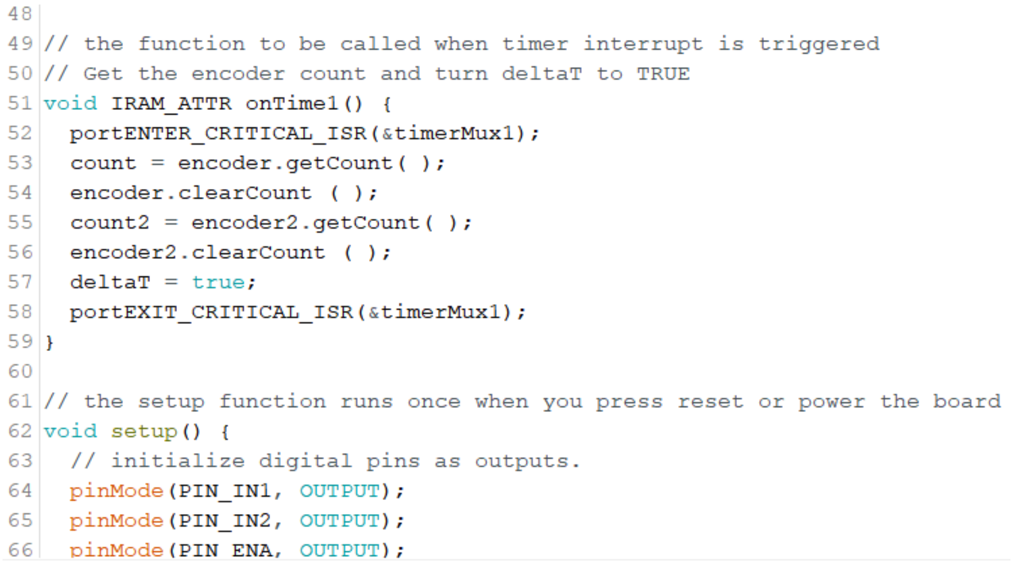
Motor Wheel Code



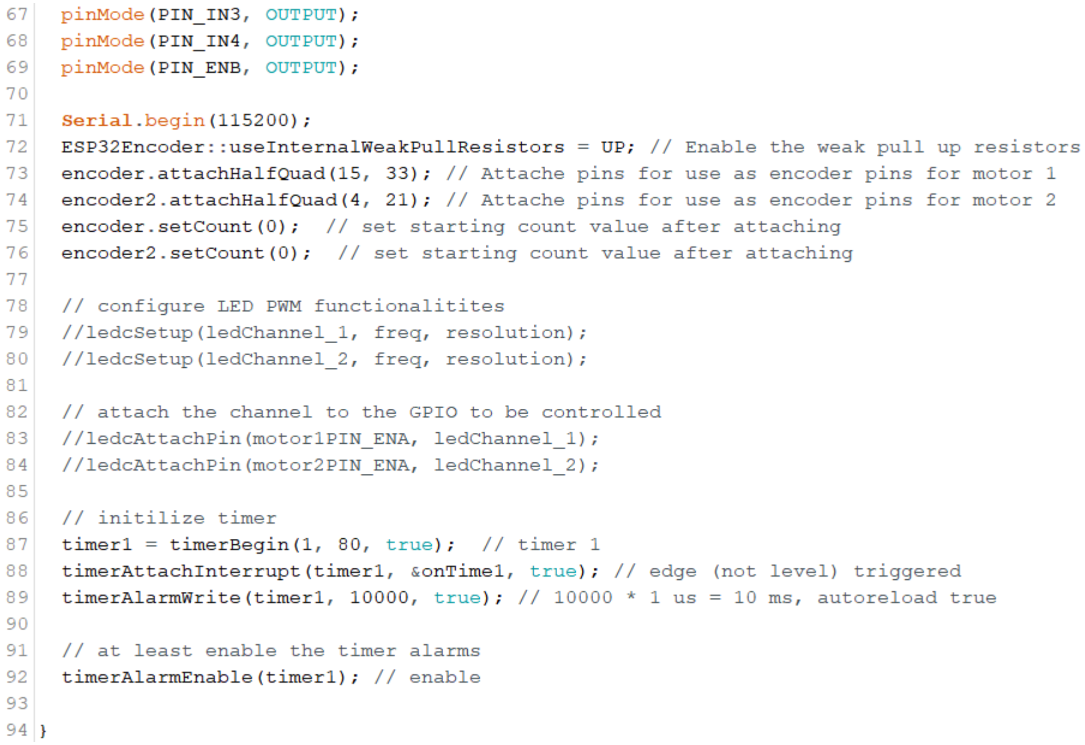
```
#include <ESP32Encoder.h>
#define PIN_IN1  27 // ESP32 pin GPIO27 connected to the IN1 pin L298N, direction control
#define PIN_IN2  26 // ESP32 pin GPIO26 connected to the IN2 pin L298N, direction control
#define PIN_ENA  14 // ESP32 pin GPIO14 connected to the EN1 pin L298N, speed control

#define PIN_IN3  25 // ESP32 pin GPIO25 connected to the IN3 pin L298N, direction control
#define PIN_IN4  32 // ESP32 pin GPIO32 connected to the IN4 pin L298N, direction control
#define PIN_ENB  13 // ESP32 pin GPIO13 connected to the ENB pin L298N, speed control


ESP32Encoder encoder;
ESP32Encoder encoder2;

int omegaSpeed = 0;
int omegaSpeed2 = 0;
int omegaDes = 9;  //SET!
int omegaMax = 22;   // CHANGE THIS VALUE TO YOUR MEASURED MAXIMUM SPEED
int D = 0;
int D2 = 0;
int dir = 1;

int Kp = 30;   // TUNE THESE VALUES TO CHANGE CONTROLLER PERFORMANCE
int Ki = 0.3;   // TUNE!
int IMax = 100;
int err = 0;
int err2 = 0;
int sumErr = 0;
int sumErr2 = 0;
int P = 0;
int P2 = 0;
int I = 0;
int I2 = 0;
```



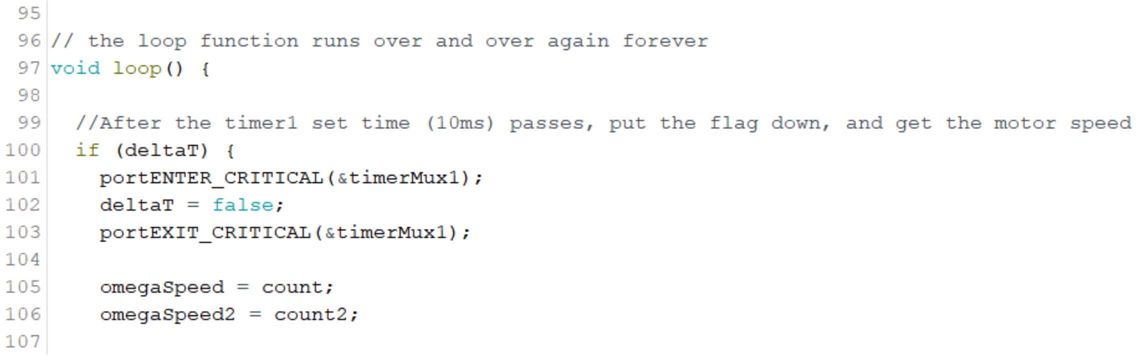
```
//Setup interrupt variables ----------------------------
volatile int count = 0; // encoder count
volatile int count2 = 0; // encoder2 count
volatile bool deltaT = false;     // check timer interrupt 2
hw_timer_t * timer1 = NULL;
portMUX_TYPE timerMux1 = portMUX_INITIALIZER_UNLOCKED;

// setting PWM properties ----------------------------
//const int freq = 5000;
//const int ledChannel_1 = 1;
//const int ledChannel_2 = 2;
//const int resolution = 8;
const int MAX_PWM_VOLTAGE = 180;
//const int NOM_PWM_VOLTAGE = 150;
```



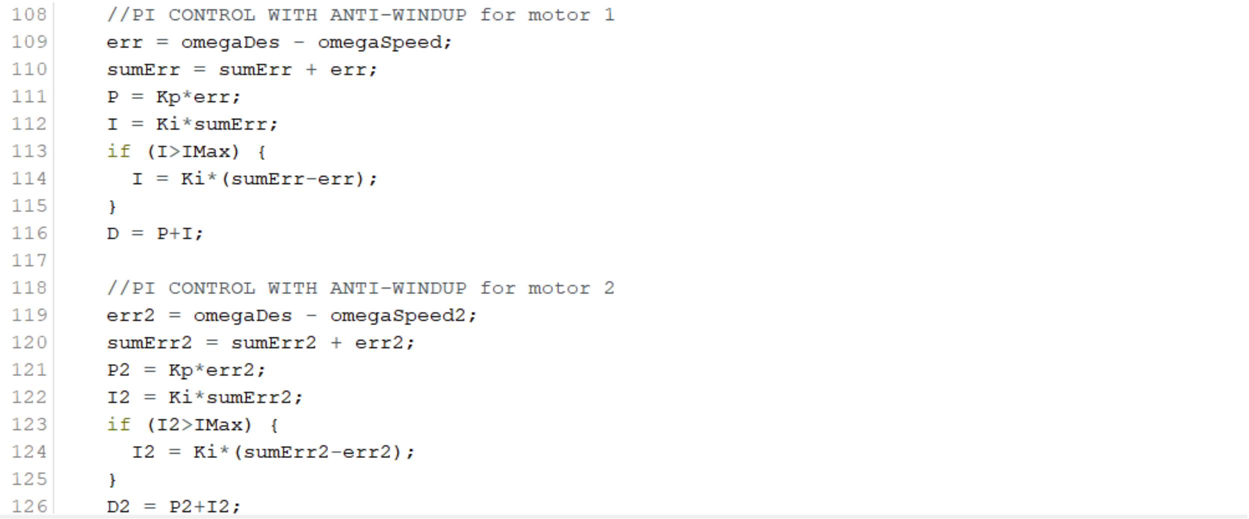
```
// the function to be called when timer interrupt is triggered
// Get the encoder count and turn deltaT to TRUE
void IRAM_ATTR onTime1() {
  portENTER_CRITICAL_ISR(&timerMux1);
  count = encoder.getCount( );
  encoder.clearCount ( );
  count2 = encoder2.getCount( );
  encoder2.clearCount ( );
  deltaT = true;
  portEXIT_CRITICAL_ISR(&timerMux1);
}

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pins as outputs.
  pinMode(PIN_IN1, OUTPUT);
  pinMode(PIN_IN2, OUTPUT);
  pinMode(PIN_ENA, OUTPUT);
```



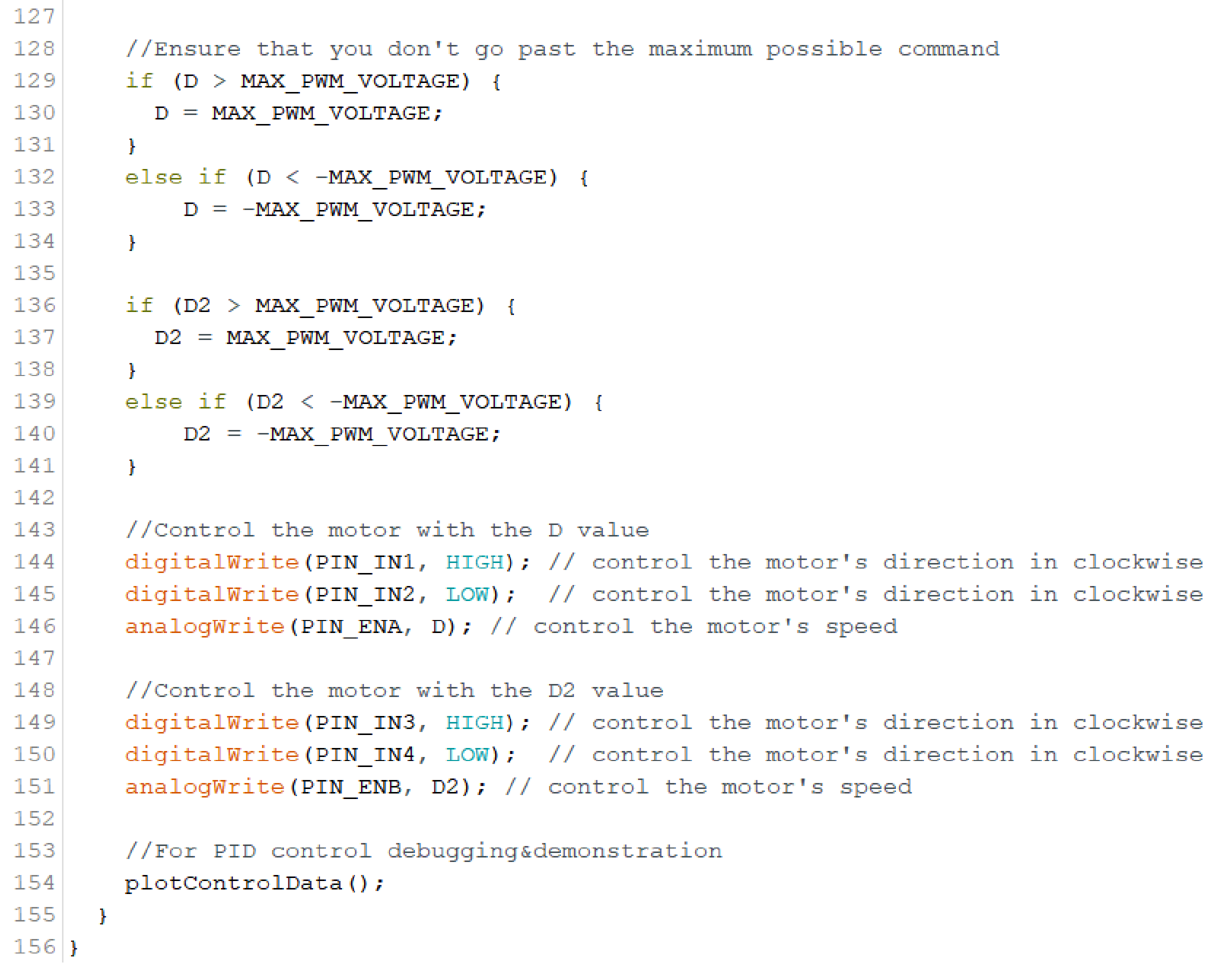
```
  pinMode(PIN_IN3, OUTPUT);
  pinMode(PIN_IN4, OUTPUT);
  pinMode(PIN_ENB, OUTPUT);

  Serial.begin(115200);
  ESP32Encoder::useInternalWeakPullResistors = UP; // Enable the weak pull up resistors
  encoder.attachHalfQuad(15, 33); // Attache pins for use as encoder pins for motor 1
  encoder2.attachHalfQuad(4, 21); // Attache pins for use as encoder pins for motor 2
  encoder.setCount(0);  // set starting count value after attaching
  encoder2.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(motor1PIN_ENA, ledChannel_1);
  //ledcAttachPin(motor2PIN_ENA, ledChannel_2);

  // initilize timer
  timer1 = timerBegin(1, 80, true);  // timer 1
  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(timer1); // enable

}
```



```

// the loop function runs over and over again forever
void loop() {

  //After the timer1 set time (10ms) passes, put the flag down, and get the motor speed
  if (deltaT) {
    portENTER_CRITICAL(&timerMux1);
    deltaT = false;
    portEXIT_CRITICAL(&timerMux1);

    omegaSpeed = count;
    omegaSpeed2 = count2;

```



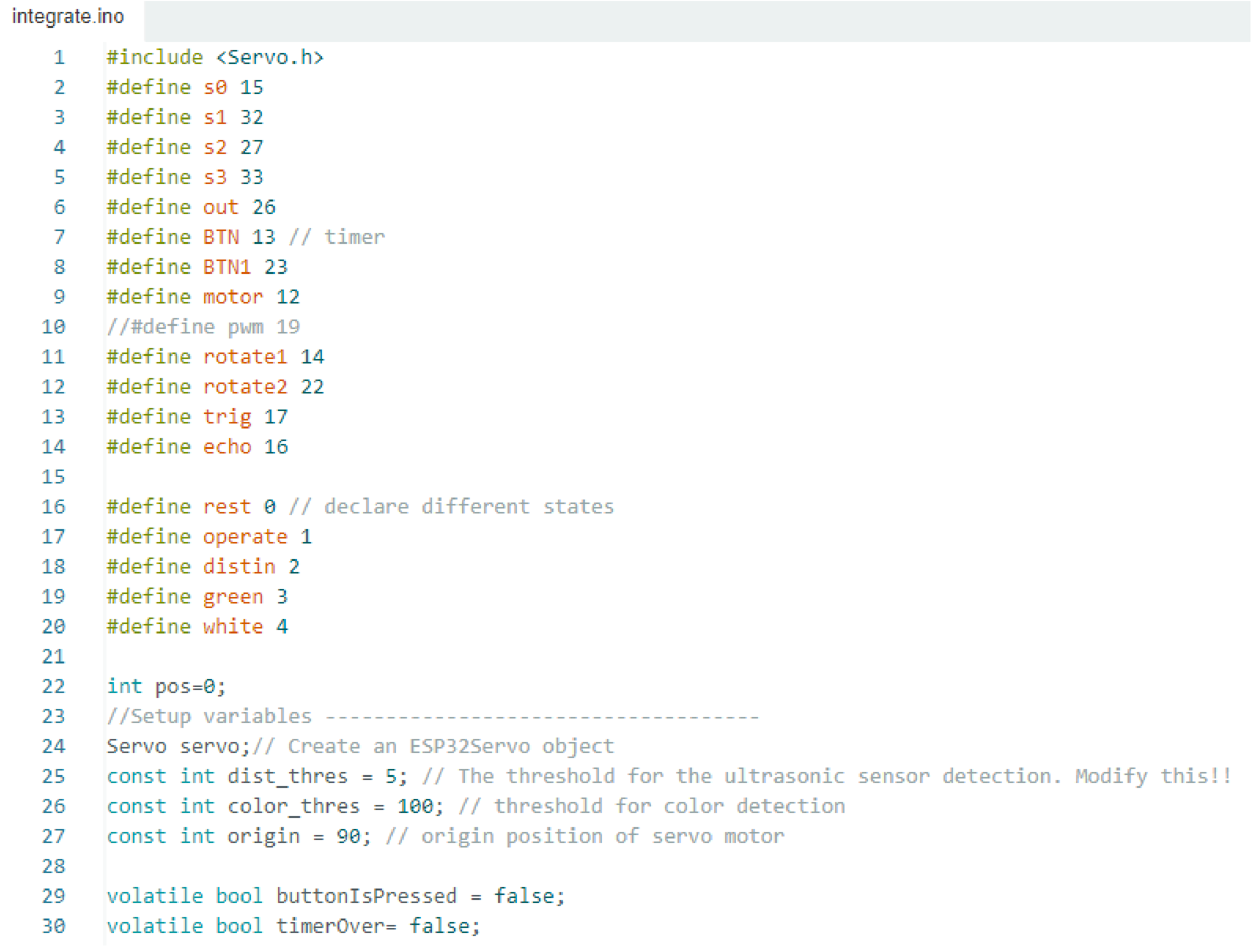
```
    //PI CONTROL WITH ANTI-WINDUP for motor 1
    err = omegaDes - omegaSpeed;
    sumErr = sumErr + err;
    P = Kp*err;
    I = Ki*sumErr;
    if (I>IMax) {
      I = Ki*(sumErr-err);
    }
    D = P+I;

    //PI CONTROL WITH ANTI-WINDUP for motor 2
    err2 = omegaDes - omegaSpeed2;
    sumErr2 = sumErr2 + err2;
    P2 = Kp*err2;
    I2 = Ki*sumErr2;
    if (I2>IMax) {
      I2 = Ki*(sumErr2-err2);
    }
    D2 = P2+I2;
```



```

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

    if (D2 > MAX_PWM_VOLTAGE) {
      D2 = MAX_PWM_VOLTAGE;
    }
    else if (D2 < -MAX_PWM_VOLTAGE) {
        D2 = -MAX_PWM_VOLTAGE;
    }

    //Control the motor with the D value
    digitalWrite(PIN_IN1, HIGH); // control the motor's direction in clockwise
    digitalWrite(PIN_IN2, LOW);  // control the motor's direction in clockwise
    analogWrite(PIN_ENA, D); // control the motor's speed

    //Control the motor with the D2 value
    digitalWrite(PIN_IN3, HIGH); // control the motor's direction in clockwise
    digitalWrite(PIN_IN4, LOW);  // control the motor's direction in clockwise
    analogWrite(PIN_ENB, D2); // control the motor's speed

    //For PID control debugging&demonstration
    plotControlData();
  }
}
```

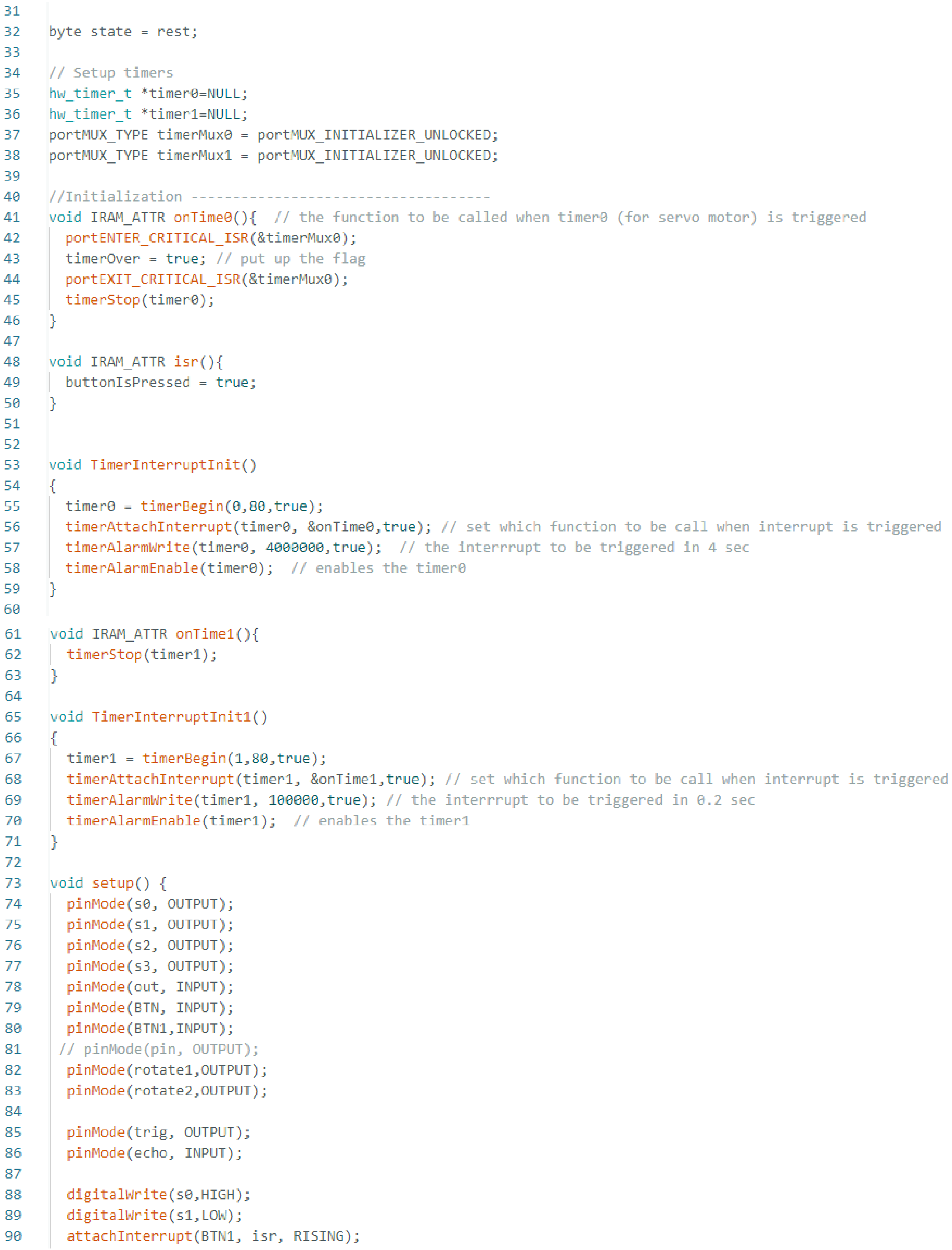
Shaft Motor Code

Integrate code (using button to control sensors and motors)

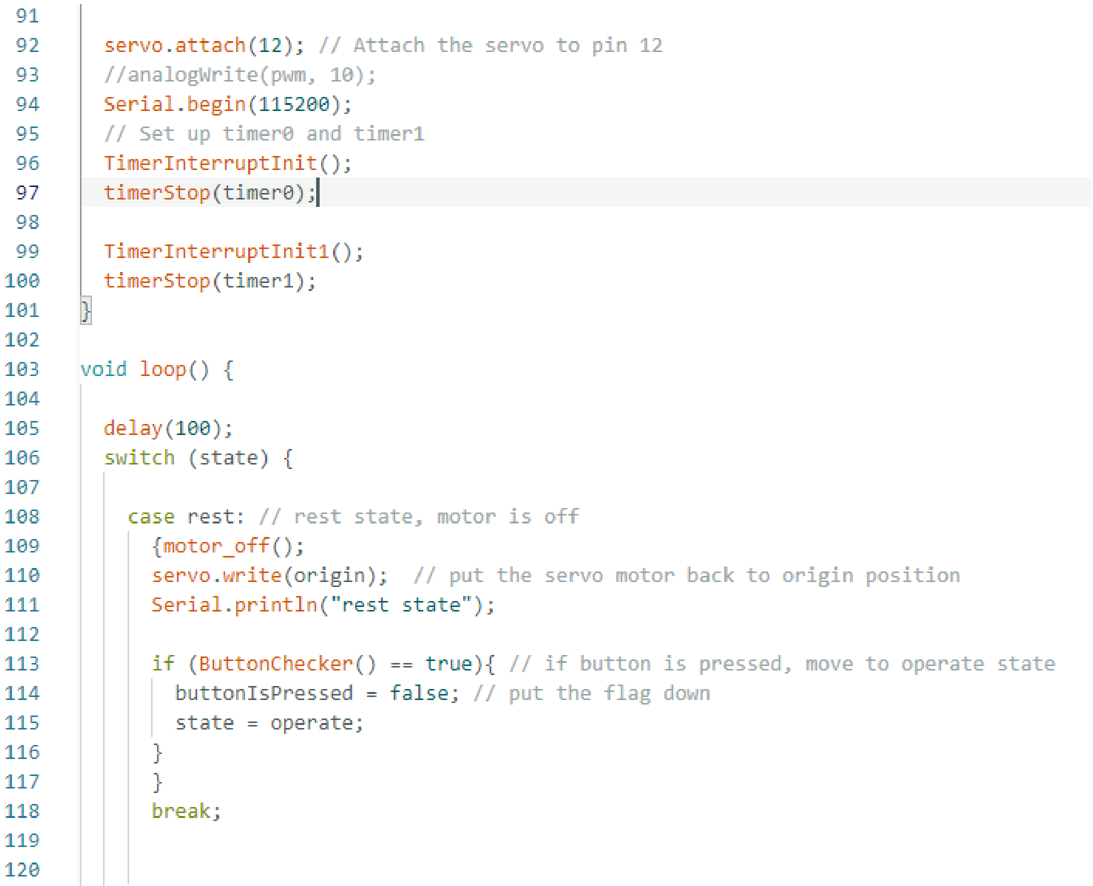


integrate.ino

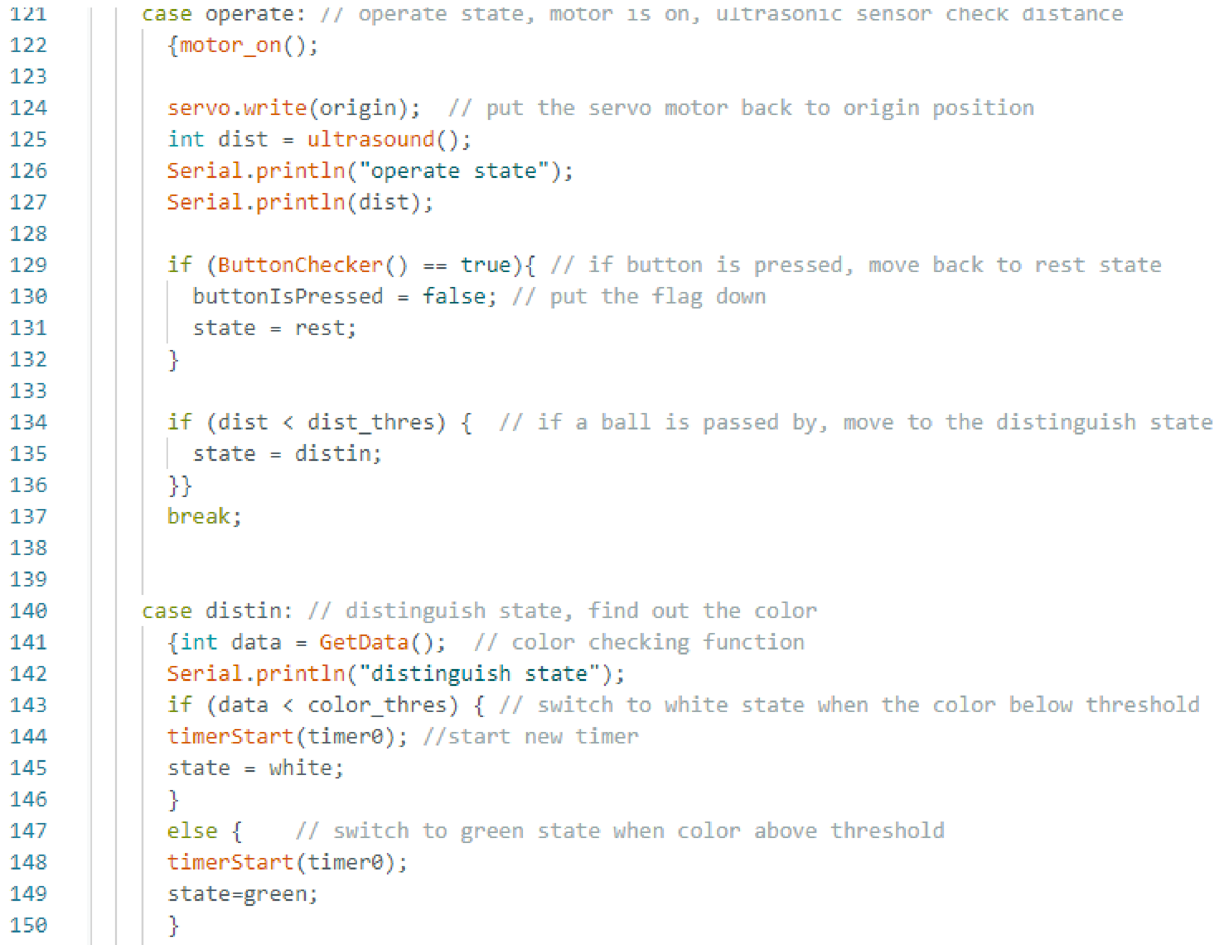
```
#include <Servo.h>
#define s0 15
#define s1 32
#define s2 27
#define s3 33
#define out 26
#define BTN 13 // timer
#define BTN1 23
#define motor 12
//#define pwm 19
#define rotate1 14
#define rotate2 22
#define trig 17
#define echo 16

#define rest 0 // declare different states
#define operate 1
#define distin 2
#define green 3
#define white 4

int pos=0;
//Setup variables ------------------------------------
Servo servo;// Create an ESP32Servo object
const int dist_thres = 5; // The threshold for the ultrasonic sensor detection. Modify this!!
const int color_thres = 100; // threshold for color detection
const int origin = 90; // origin position of servo motor

volatile bool buttonIsPressed = false;
volatile bool timerOver= false;
```



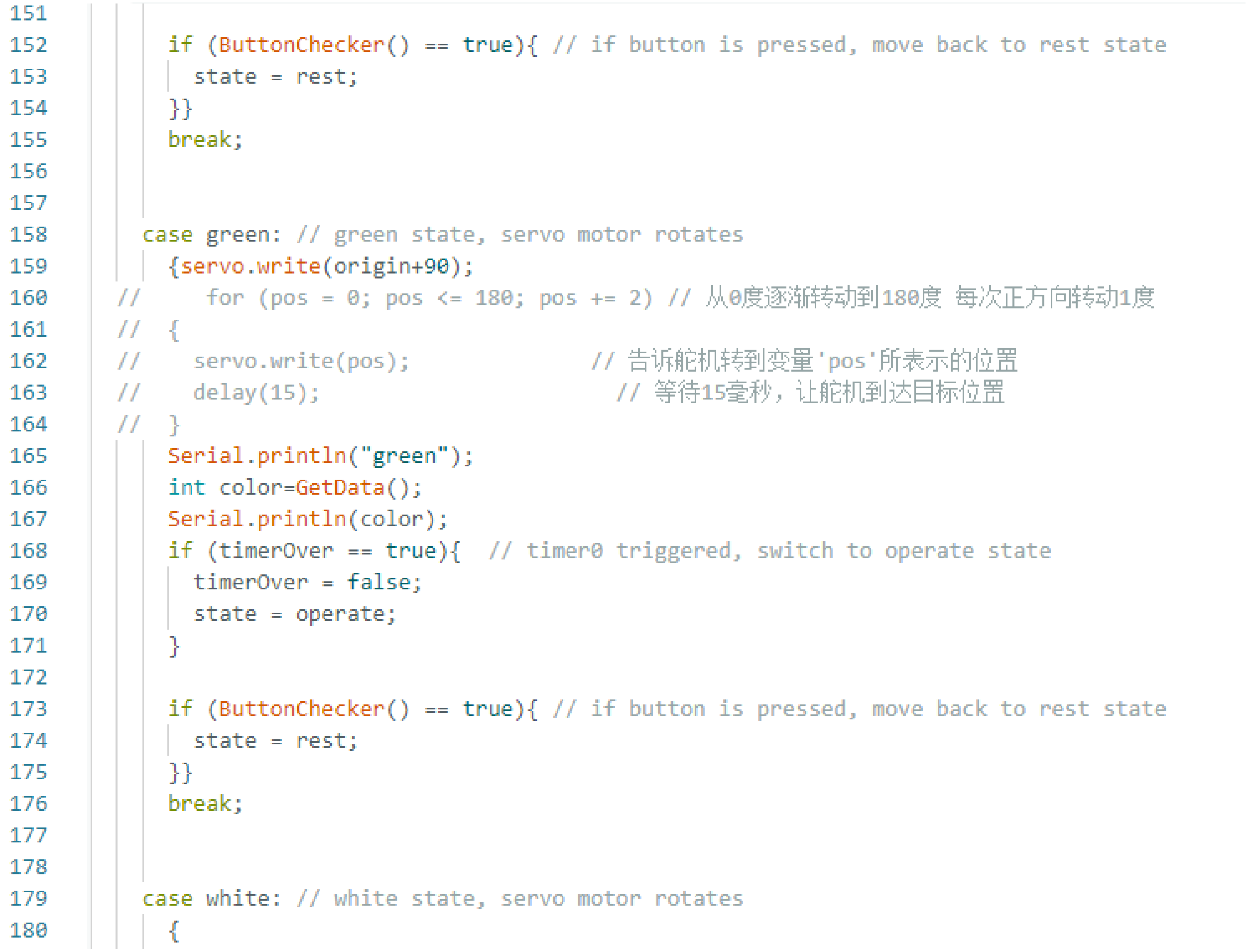
```
byte state = rest;

// Setup timers
hw_timer_t *timer0=NULL;
hw_timer_t *timer1=NULL;
portMUX_TYPE timerMux0 = portMUX_INITIALIZER_UNLOCKED;
portMUX_TYPE timerMux1 = portMUX_INITIALIZER_UNLOCKED;

//Initialization ------------------------------------
void IRAM_ATTR onTime0(){  // the function to be called when timer0 (for servo motor) is triggered
  portENTER_CRITICAL_ISR(&timerMux0);
  timerOver = true; // put up the flag
  portEXIT_CRITICAL_ISR(&timerMux0);
  timerStop(timer0);
}

void IRAM_ATTR isr(){
  buttonIsPressed = true;
}


void TimerInterruptInit()
{
  timer0 = timerBegin(0,80,true);
  timerAttachInterrupt(timer0, &onTime0,true); // set which function to be call when interrupt is triggered
  timerAlarmWrite(timer0, 4000000,true);  // the interrrupt to be triggered in 4 sec
  timerAlarmEnable(timer0);  // enables the timer0
}

void IRAM_ATTR onTime1(){
  timerStop(timer1);
}

void TimerInterruptInit1()
{
  timer1 = timerBegin(1,80,true);
  timerAttachInterrupt(timer1, &onTime1,true); // set which function to be call when interrupt is triggered
  timerAlarmWrite(timer1, 100000,true); // the interrrupt to be triggered in 0.2 sec
  timerAlarmEnable(timer1);  // enables the timer1
}

void setup() {
  pinMode(s0, OUTPUT);
  pinMode(s1, OUTPUT);
  pinMode(s2, OUTPUT);
  pinMode(s3, OUTPUT);
  pinMode(out, INPUT);
  pinMode(BTN, INPUT);
  pinMode(BTN1,INPUT);
 // pinMode(pin, OUTPUT);
  pinMode(rotate1,OUTPUT);
  pinMode(rotate2,OUTPUT);

  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);

  digitalWrite(s0,HIGH);
  digitalWrite(s1,LOW);
  attachInterrupt(BTN1, isr, RISING);
```



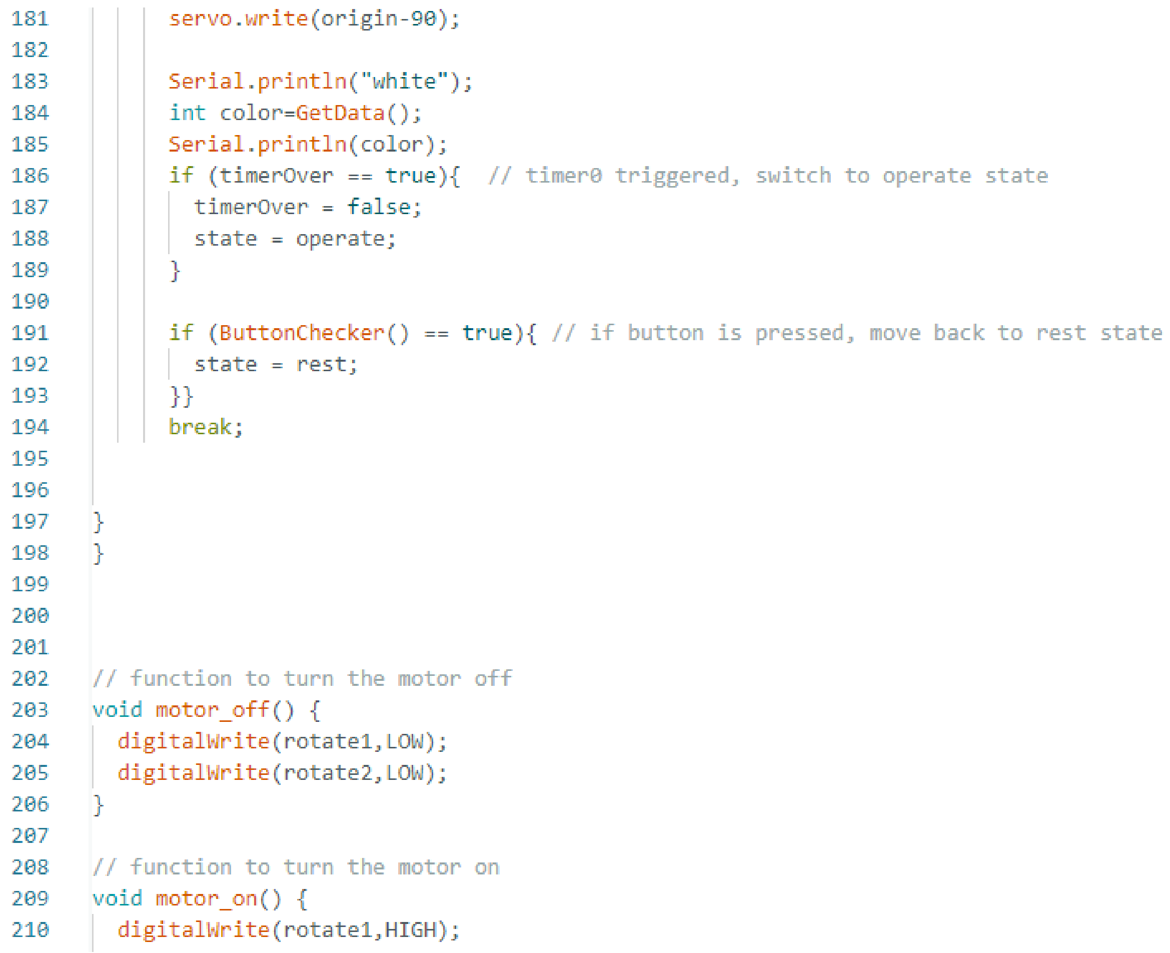
```

  servo.attach(12); // Attach the servo to pin 12
  //analogWrite(pwm, 10);
  Serial.begin(115200);
  // Set up timer0 and timer1
  TimerInterruptInit();
  timerStop(timer0);

  TimerInterruptInit1();
  timerStop(timer1);
}

void loop() {

  delay(100);
  switch (state) {

    case rest: // rest state, motor is off
      {motor_off();
      servo.write(origin);  // put the servo motor back to origin position
      Serial.println("rest state");

      if (ButtonChecker() == true){ // if button is pressed, move to operate state
        buttonIsPressed = false; // put the flag down
        state = operate;
      }
      }
      break;


```



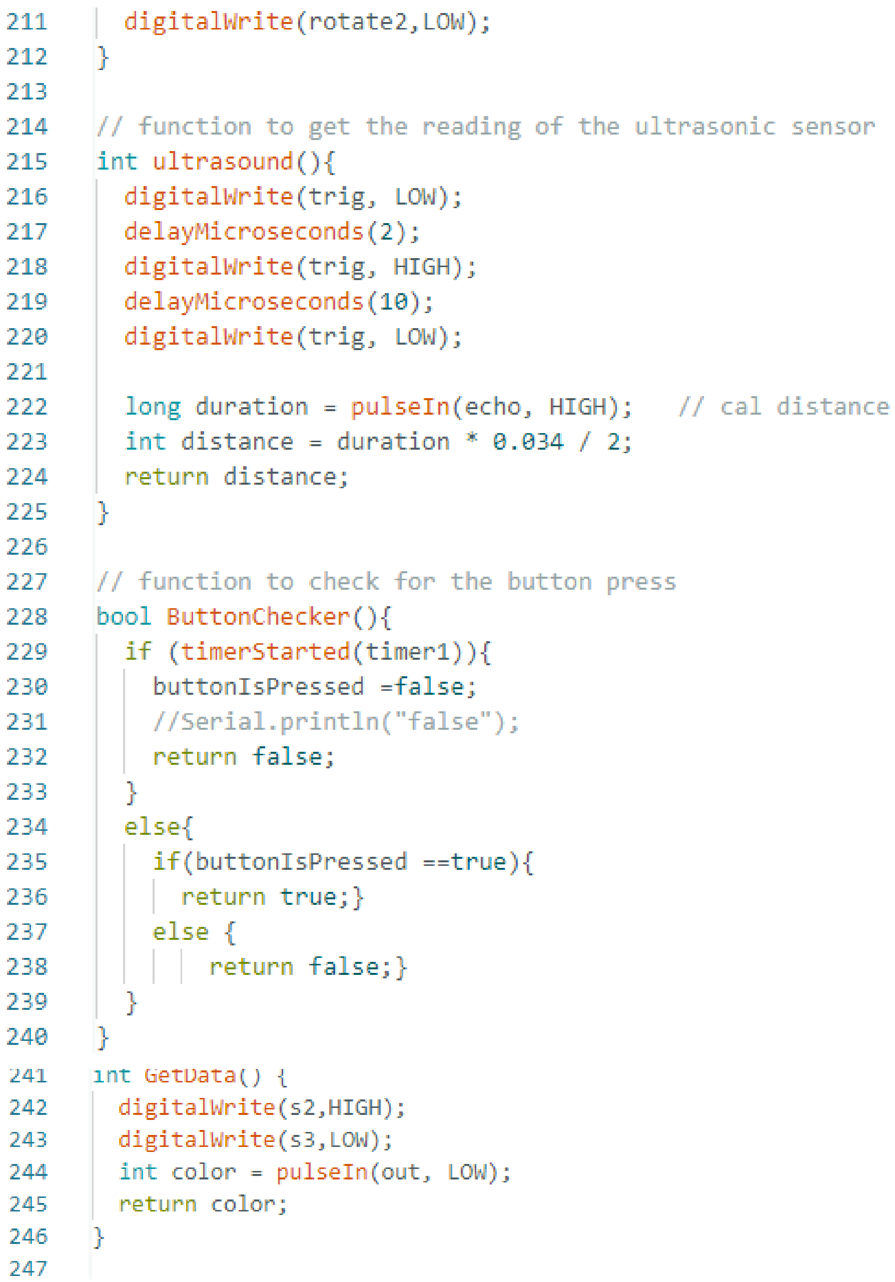
```
    case operate: // operate state, motor is on, ultrasonic sensor check distance
      {motor_on();

      servo.write(origin);  // put the servo motor back to origin position
      int dist = ultrasound();
      Serial.println("operate state");
      Serial.println(dist);

      if (ButtonChecker() == true){ // if button is pressed, move back to rest state
        buttonIsPressed = false; // put the flag down
        state = rest;
      }

      if (dist < dist_thres) {  // if a ball is passed by, move to the distinguish state
        state = distin;
      }}
      break;


    case distin: // distinguish state, find out the color
      {int data = GetData();  // color checking function
      Serial.println("distinguish state");
      if (data < color_thres) { // switch to white state when the color below threshold
      timerStart(timer0); //start new timer
      state = white;
      }
      else {    // switch to green state when color above threshold
      timerStart(timer0);
      state=green;
      }
```



```

    if (ButtonChecker() == true){ // if button is pressed, move back to rest state
      state = rest;
    }}
    break;


  case green: // green state, servo motor rotates
    {servo.write(origin+90);
//     for (pos = 0; pos <= 180; pos += 2) // 从0度逐渐转动到180度 每次正方向转动1度
//  {
//    servo.write(pos);               // 告诉舵机转到变量'pos'所表示的位置
//    delay(15);                       // 等待15毫秒，让舵机到达目标位置
//  }
    Serial.println("green");
    int color=GetData();
    Serial.println(color);
    if (timerOver == true){  // timer0 triggered, switch to operate state
      timerOver = false;
      state = operate;
    }

    if (ButtonChecker() == true){ // if button is pressed, move back to rest state
      state = rest;
    }}
    break;


  case white: // white state, servo motor rotates
    {
```



```
      servo.write(origin-90);

      Serial.println("white");
      int color=GetData();
      Serial.println(color);
      if (timerOver == true){  // timer0 triggered, switch to operate state
        timerOver = false;
        state = operate;
      }

      if (ButtonChecker() == true){ // if button is pressed, move back to rest state
        state = rest;
      }}
      break;


}
}



// function to turn the motor off
void motor_off() {
  digitalWrite(rotate1,LOW);
  digitalWrite(rotate2,LOW);
}

// function to turn the motor on
void motor_on() {
  digitalWrite(rotate1,HIGH);
```



```
  digitalWrite(rotate2,LOW);
}

// function to get the reading of the ultrasonic sensor
int ultrasound(){
  digitalWrite(trig, LOW);
  delayMicroseconds(2);
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);

  long duration = pulseIn(echo, HIGH);   // cal distance
  int distance = duration * 0.034 / 2;
  return distance;
}

// function to check for the button press
bool ButtonChecker(){
  if (timerStarted(timer1)){
    buttonIsPressed =false;
    //Serial.println("false");
    return false;
  }
  else{
    if(buttonIsPressed ==true){
      return true;}
    else {
        return false;}
  }
}
int GetData() {
  digitalWrite(s2,HIGH);
  digitalWrite(s3,LOW);
  int color = pulseIn(out, LOW);
  return color;
}

```