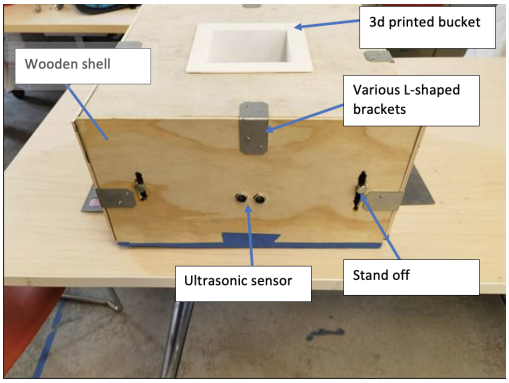
ME 102B Project Report: Snail Bot

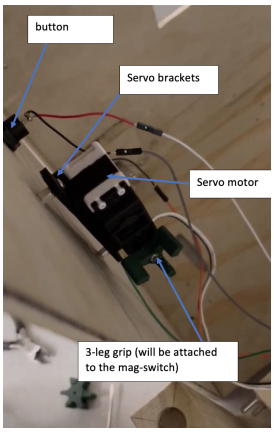
Kevin Lam, Gumer Ho, Guanhua Fu
Dept. of Mechanical Engineering, University of California at Berkeley
(Date: 12/7/2022)

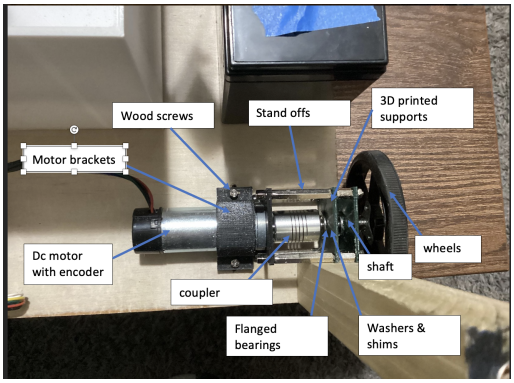
Opportunity:

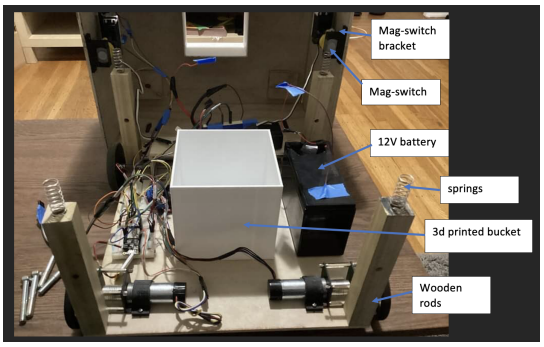
Food delivery robots can often be the target of harassment, whether it be damage or stolen. Our goal is to address this problem by creating a food delivery robot that is designed to reduce potential for critical damage and remain outdoors at all times without being stolen. As such this will reduce the need for repairs or private housing facilities.

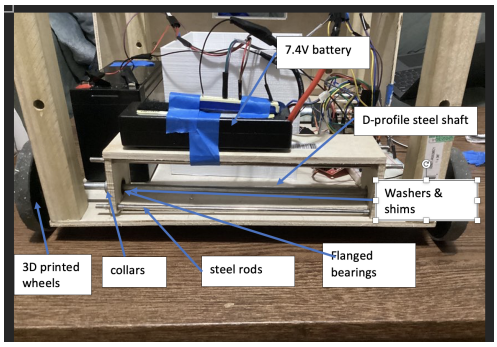
Photographs:











High Level Strategy:

- To mimic a robot's food delivery process, our snail bot is designed to move along a straight track and stop at various locations. There are ultrasonic sensors to detect obstacles and stop the bot when a reading goes under a specific threshold. There are 3D printed grips connected to the servo motors which rotate the magswitch which then compresses the shell to the bottom plate and locks itself to the ground. There are also

springs and slots to ensure the shell moves only in the vertical direction. It protects the snail bot from being lifted up to 120 pounds

- The green button on the side is pressed. The snail bot enters into travel mode. The motors are turned on and the magswitches are turned off. In this phase, when the ultrasonic sensor senses an obstacle in front of it by 15 inches, it will stop (motors are turned off). It will only start moving once the obstacle disappears. When encoder reaches a certain number of counts (10000) or 52 meters equivalent, it stops and locks itself to the ground (motors are on and magswitches are on).
- In this idle state, users can put food or get food out of the robot. Then, there are two available options. Press the green button again to let the snail bot enter the travel mode again (food delivery or going back to the station). The other option is to press the red button, which resets itself to the initial mode (motors off and magswitches off). Notably, there are no effects on the motion when the red button is pressed during the travel mode due to switch case coding structure.

Comparison to initial desired functionaility:

Reflectance sensors were to be installed which will be used with PI control to motors to move the bot along a desired black line. This idea was abandoned due to functional issues with our ordered sensor, as the uploaded code gave inconsistent data. We also opted to avoid using wireless charging as the motors required 12v and 6a which we found would be better accomplished by a smaller rechargeable battery. However, most of the functions and the main essence (locking mechanism) are included in this project.

Function-Critical Decision

- For our transmission system we utilized two brushed DC motors that acted as our front wheel drive. In the event that we wanted to design the bot to steer we decided it would be best to adjust the speeds of each motor in order to steer instead of implementing a dedicated steering system. Additionally we opted for front wheel drive instead of all wheel drive as it was more cost effective and we did not expect to encounter any issues where all wheel drive would be important, such as slippage
- For our analog input we utilized an ultrasonic sensor in order to prevent the bot from crashing into objects or possible pedestrians. We opted for this method of distance measurement over an IR sensor because in practicality this bot will travel outdoors, where an IR sensor can be affected by weather conditions such as haze. Additionally IR sensors are typically shorter range.
- For our digital input we used two buttons. These buttons are to be activated by the users who are loading food or unloading food. This was the most efficient way to implement a way for the bot to re-enter travel mode, while mimicking the real world interaction users will have with the bot. The second button is in order to reset the bot to its idle state.

Function-Critical Calculations

To select the DC motor, we calculated the weight of the robot:

- Mass of the main wooden body= area * thickness * density of plywood= 3.32kg
- Mass of motor= mass of DC motors + mass of servo motors= 0.104*2+0.07*2= 0.348kg
- Mass of shafts= number of axes * base area * length * density steel= 0.43kg
- Mass of battery= 2.75kg
- Mass of food= we assume the weight of the food is 1kg
- Weight of the robot= total mass*9.8= 7.85*9.8=76.91N
- Weight with 10% tolerance=76.91*1.1= 84.601N

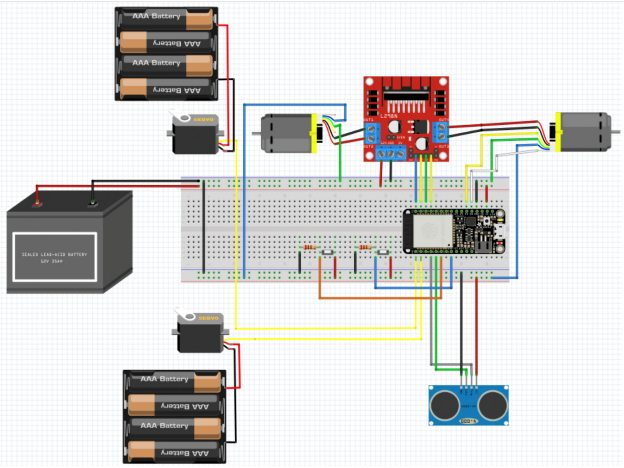
After we got the weight, we calculated the friction and the torque required by the robot. The wheels we used has a radius of 0.04m, so the torque is:

2T= f*r= μ*weight*r= 0.6*84.61*0.04= 2.03 N*m= 20.7 kg*cm

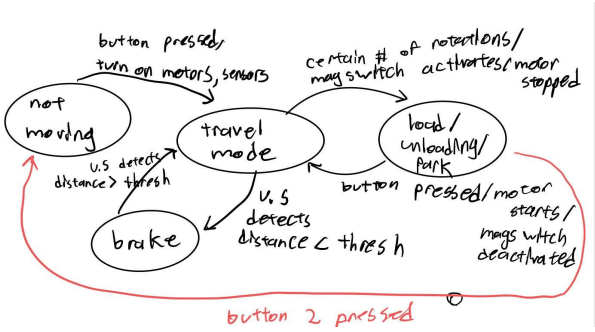
Then we calculated the velocity of the robot. The radius of wheels we used is 0.04m, and we want the motion speed is no less than 0.2m/s, so according to the conversion formula of angular velocity and linear velocity: RPM= 60v/(2πr), angular velocity of motor is about 50rpm.

According to the requirements of speed and torque, we finally chosed two 12 V, 130 RPM, 22 kg · cm DC motors.

Circuit Diagram

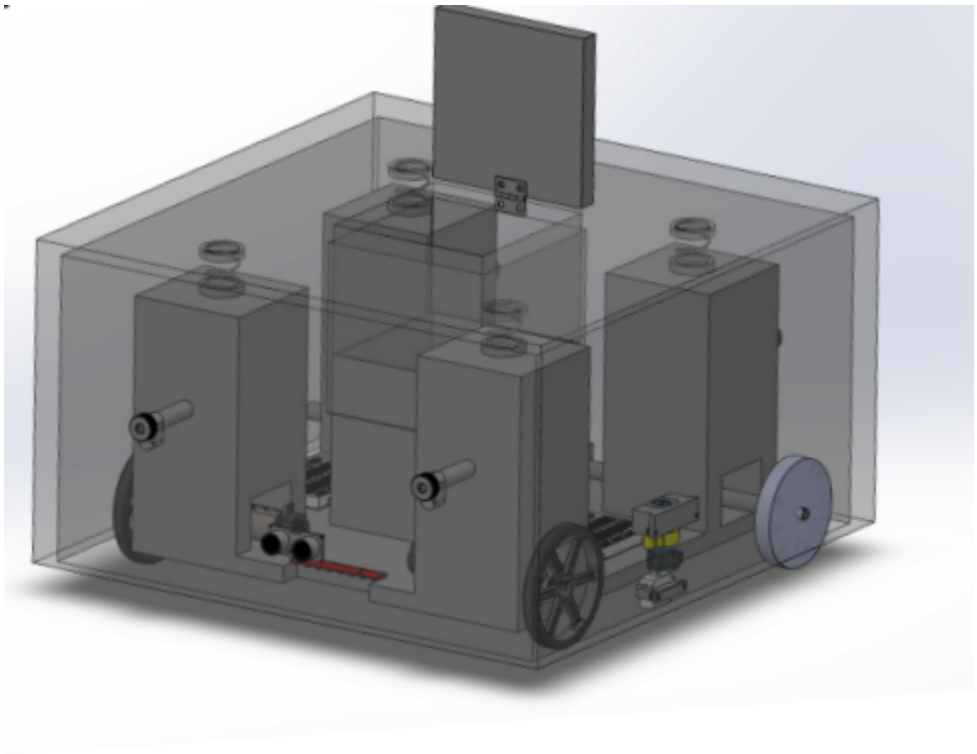


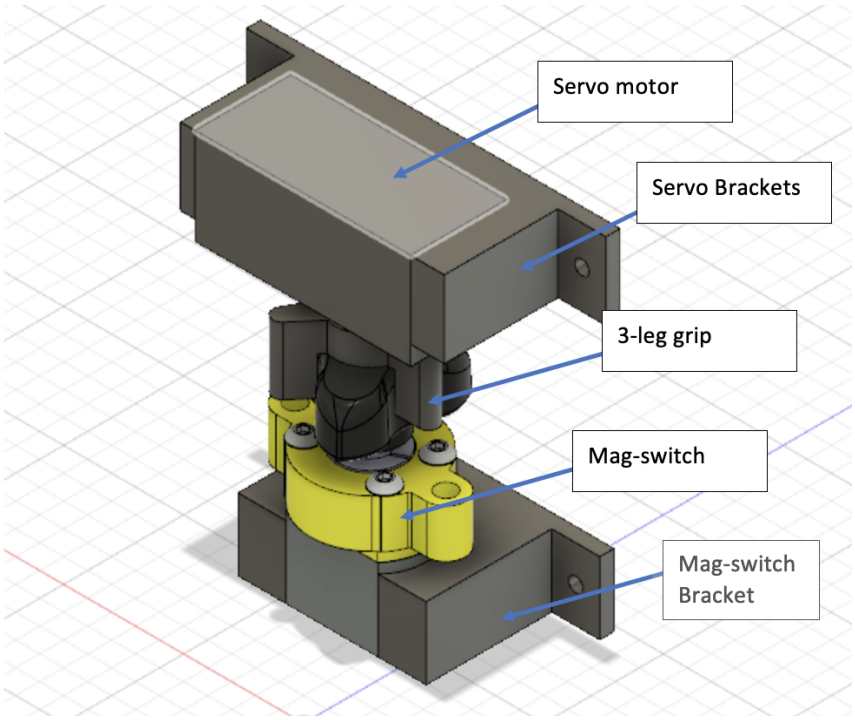
State Transition Diagram

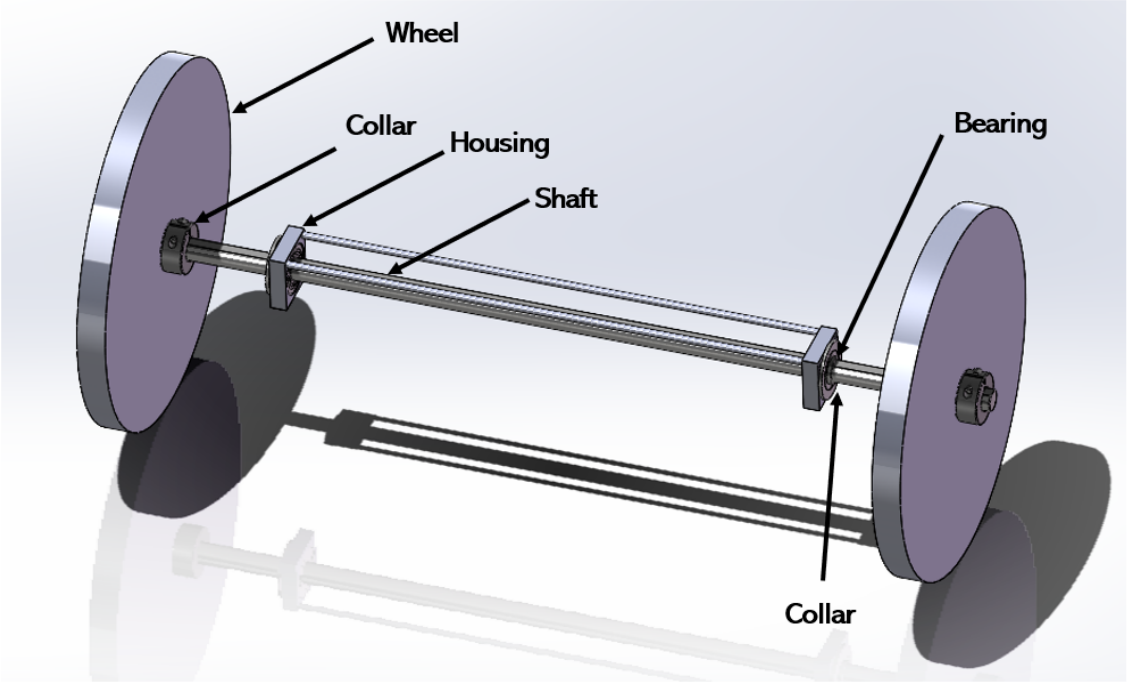


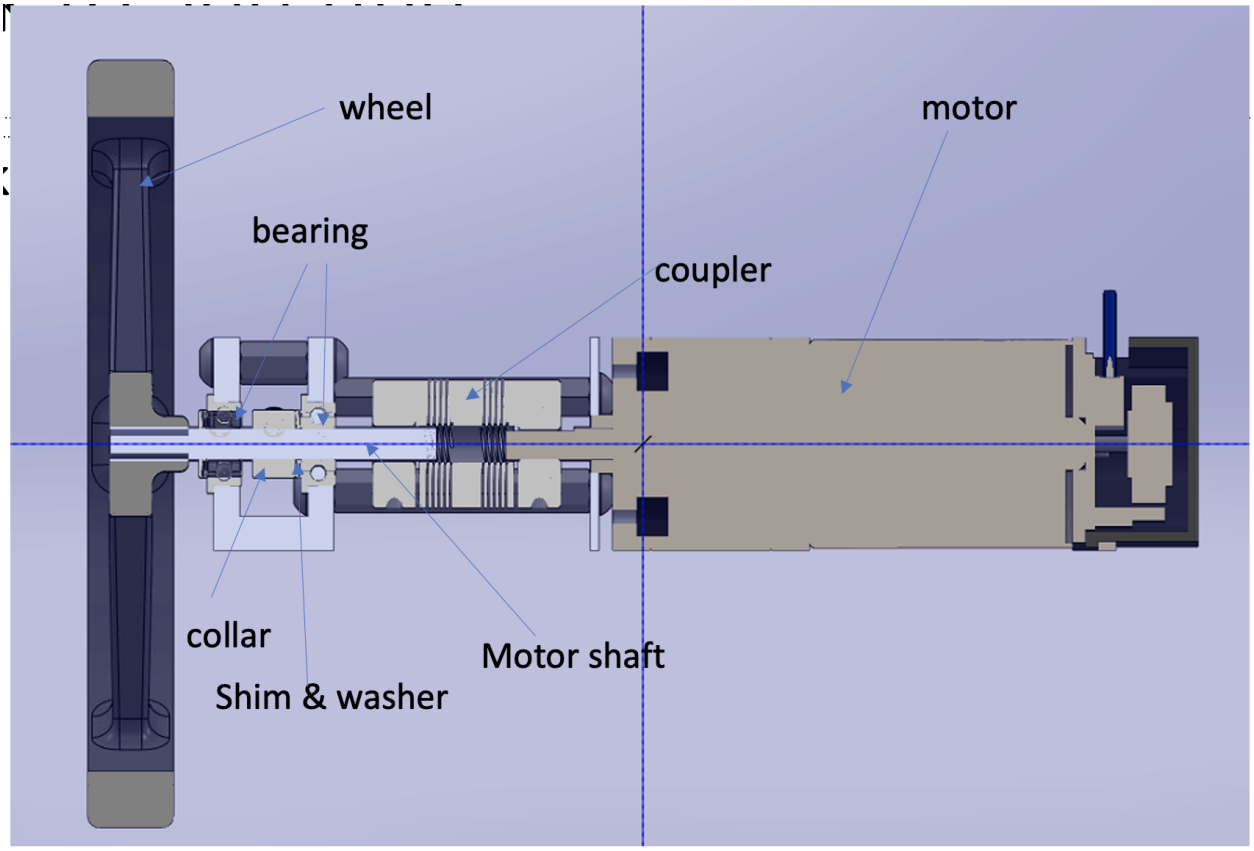
Appendix

CAD






Wheel
Collar
Housing
Shaft
Bearing
Collar


wheel
motor
bearing
coupler
collar
Motor shaft
Shim & washer

Bill of Materials

Part Name	Notes	Quantity	Cost
Servo motors	From Hesse Hall	2	0
L298N motor driver	From Hesse	1	0
Springs pack	Amazon	1	8.81
Mag switch 60lbf	Amazon	2	48
Gear motor 75:1 with encoder	Pololu	2	119
2-Wheels pack	Pololu	1	11.6
Ultrasonic Sensor	From lab kit	1	0
Breadboards and wires	From lab kit	1	0
Mix set of screws and so on	From Hesse	NAN	0
2-pack 4mm coupling	Amazon	1	8.39
Shaft collars	Ace Hardware	4	16.72
Steel Plate	Ace Hardware	1	34.45
Plywood plate	Ace Hardware	2	57
Wood screws	Ace Hardware	1	1.59
L-shaped angle	Ace Hardware	8	8.3
Flanged Bearings pack	Amazon	1	12.99
Flanged Bearings pack	Amazon	1	12.64
12V battery	Amazon	1	23.04
7.4V battery	Hesse	1	0

Stand offs kit	Amazon	1	13.2
Collars pack	Amazon	1	8.2
Collar pack	Amazon	1	7.85
1/4 stainless steel	Ace Hardware	1	20
3D printed parts	Jacobs	14	0
Button pack	Amazon	1	7.7
Wooden rod	Ace Hardware	1	10
		Total	430

Code

(I've uploaded the code to the folder. Could you guys also double check if something is missing?)

state_clean

```
#include <Arduino.h>
#include <ESP32Encoder.h>
#include <Servo.h>

//Setup

//motors
int motor1Pin1 = 17;
int motor1Pin2 = 21;
int enable1Pin = 4;
int motor2Pin1 = 16;
int motor2Pin2 = 19;
int enable2Pin = 5;
int motor = false;

const int freq = 30000;
const int pwmChannel = 0;
const int resolution = 8;
int dutyCycle = 250;

//ultrasonic sensors
const int trigPin = 15;
const int echoPin = 32;
#define SOUND_SPEED 0.034
#define CM_TO_INCH 0.393701

long duration;
float distanceCm;
float distanceInch;
const int Threshold = 15;
int distance;

//servo motors
int val;
Servo myservo1;
Servo myservo2;
int pos;
```

```
//Encoders
ESP32Encoder encoder;
int count;
int v = 0;
int no_counts = 10000;

//Buttons
#define BTN 13
#define BTN2 27
volatile bool buttonIsPressed = false;
volatile bool button2IsPressed = false;

volatile bool idle = true;
volatile bool idle2 = true;
hw_timer_t * timer = NULL;
hw_timer_t * timer1 = NULL;

int state = 1;

//Initialization ------------------------------------

//Debounce
void IRAM_ATTR timer_isr() {
  idle = true;
}

void IRAM_ATTR timer_isr1() {
  idle2 = true;
}

void TimerInterruptInit() {
  timer = timerBegin(0, 80, true);
  timerAttachInterrupt(timer, timer_isr, true);
  timerAlarmWrite(timer, 500000, true);
  timerAlarmEnable(timer);

  timer1 = timerBegin(1, 80, true);
  timerAttachInterrupt(timer1, timer_isr1, true);
  timerAlarmWrite(timer1, 500000, true);
  timerAlarmEnable(timer1); // Enables timer
}


void IRAM_ATTR isr1() {
  button2IsPressed = true;
  idle2 = false;
  timerStop(timer1);
  timerRestart(timer1);
  timerStart(timer1);
}


void IRAM_ATTR isr() {
  buttonIsPressed = true;
  idle = false;
  timerStop(timer);
  timerRestart(timer);
  timerStart(timer);
}
```

```
void setup() {
  // ultrasonic sensors
  Serial.begin(115200);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  //button
  pinMode(BTN, INPUT);
  attachInterrupt(BTN, isr, CHANGE);
  pinMode(BTN2, INPUT);
  attachInterrupt(BTN2, isr1, CHANGE);

  //Encoder
  ESP32Encoder::useInternalWeakPullResistors=UP;
  encoder.attachHalfQuad(26, 25);
  encoder.setCount(0);
  TimerInterruptInit();
  timerAlarmWrite(timer, 500000, true);
  timerAlarmEnable(timer);

  //motors
  pinMode(motor1Pin1, OUTPUT);
  pinMode(motor1Pin2, OUTPUT);
  pinMode(enable1Pin, OUTPUT);
  pinMode(motor2Pin1, OUTPUT);
  pinMode(motor2Pin2, OUTPUT);
  pinMode(enable2Pin, OUTPUT);

  ledcSetup(pwmChannel, freq, resolution);
  ledcAttachPin(enable1Pin, pwmChannel);
  ledcSetup(pwmChannel, freq, resolution);
  ledcAttachPin(enable2Pin, pwmChannel);

  //servo motors
  myservo1.attach(22);
  myservo2.attach(23);
}
```

```
void loop() {

  //ultrasonic sensors
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distanceCm = duration * SOUND_SPEED/2;
  distanceInch = distanceCm * CM_TO_INCH;
  Serial.print("Distance (inch): ");
  Serial.println(distanceInch);

  //Encoder
  count = abs((int32_t)encoder.getCount());
  Serial.println("Encoder count = "+String((int32_t)encoder.getCount()));
```

```
//motors
if (motor) {
  dutyCycle = 180;
  digitalWrite(motor1Pin1, LOW);
  digitalWrite(motor1Pin2, HIGH);

  digitalWrite(motor2Pin1, LOW);
  digitalWrite(motor2Pin2, HIGH);

  ledcWrite(pwmChannel, dutyCycle);
  Serial.println("motor");
   } else {
    dutyCycle = 0;
  digitalWrite(motor1Pin1, LOW);
  digitalWrite(motor1Pin2, LOW);

  digitalWrite(motor2Pin1, LOW);
  digitalWrite(motor2Pin2, LOW);

  ledcWrite(pwmChannel, dutyCycle);
};


// Serial Print for debugging
//Serial.println(String(state));
//Serial.println(String(motor));
//delay(500);

// State Machine

switch (state) {
  case 1: //idle state
    if (CheckForButtonPress()) {
      magswitch_off();
      motor_on();
      ButtonResponse();
      state = 2;
    };
    break;

  case 2: //travel mode
    if (!CheckForThresh()) {
      motor_off();
      state = 3;
    };
    if (CheckForRotations()) {
      state = 4;
    };
    break;

  case 3: //brake (obstacle)
    if (CheckForThresh()) {
      magswitch_off();
      motor_on();
      state = 2;
    };
    break;
```

```
    case 4: // loading/unloading/park
      if (CheckForRotations()) {
        motor_off();
        magswitch_on();
        //make sure encoder count resets
      } ;

      if (CheckForButtonPress()) {
        magswitch_off();
        ButtonResponse();
        delay(3000);
        encoder.setCount(0); //reset encoder value
        motor_on();
        state = 2;

     } ;
      if (CheckForButton2()) {
        motor_off(); //every motor on travels 1000 rotations
        magswitch_off();
        Button2Response();
        state = 1;
      };
      break;
  }
}


// functions to be called

// button 1
bool CheckForButtonPress() {
  if (buttonIsPressed == true && idle == true) {
    return true;
  } else {
    return false;
  }
}

// button 2
bool CheckForButton2() {
  if (button2IsPressed == true && idle2 == true) {
    return true;
  } else {
    return false;
  }
}

// checking obstacles
bool CheckForThresh() {
  if (distanceInch >= Threshold){
    return true;
  }
  else {
    return false;
  }
}

// motors on
void motor_on() {
  //digitalWrite(LED_PIN, HIGH);//replace led code with motor code
  Serial.println("motor is turned on");
  motor = true;
}

// motos off
void motor_off() {
  //digitalWrite(LED_PIN, LOW);
  //replace led code with motor code
  //ledcWrite(ledChannel_1, LOW);
  //ledcWrite(ledChannel_2, LOW);
  motor = false;
  Serial.println("motor is turned off");
}
```

```
// magswitch on
void magswitch_on() {
  //replace code with servo code
  Serial.println("servo is turned on");
  pos = 130;
  Serial.println(pos);
  myservo1.write(pos);
  myservo2.write(pos);
}

//magswitch off
void magswitch_off() {
  //replace code with servo code
  Serial.println("servo is turned off");
  pos = 0;
  Serial.println(pos);
  myservo1.write(pos);
  myservo2.write(pos);
}

//Button 1 Response
void ButtonResponse() {
  buttonIsPressed = false;
  Serial.println("Pressed!");
}

//Button 2 Response
void Button2Response() {
  button2IsPressed = false;
  Serial.println("Pressed!");
}
```

```
// Checking for distance or location
bool CheckForRotations() {
  if (count >= no_counts) {
    return true;
  } else {
    return false;
  }
}
```