ME102B FALL 2021
The Bike Car

Group No.24 : Ethan Cardenas, Alexander Park, Henry Smith

Dec. 12 2021

1 Project Motivation

Government mandated closures of businesses in response to the recent Covid-19 pandemic outbreak has left gym goers and fitness enthusiasts with limited options when it comes to exercising. Most gyms were either closed or operating under very limited conditions. Even some public parks were closed to the public. This led many seeking ways to exercise from the comfort of their own home. While at-home fitness equipment industry is well established, most products available accomplish little in the way of making their equipment engaging - they simply provide a means to exercise. Our product hopes to address this by having a physical robot that a rider on a stationary bike can control. This adds a level of engagement uncommon with virtual systems, especially if multiple bicycles and robots can come together to create endless possibilities.

2 Design

The system is composed of two main subsystems. The first subsystem is a stationary bike providing direction and angular velocity outputs via rotary incremental encoders. And the second subsystem is a robot that receives the direction and velocity outputs from the bicycle and drives accordingly.

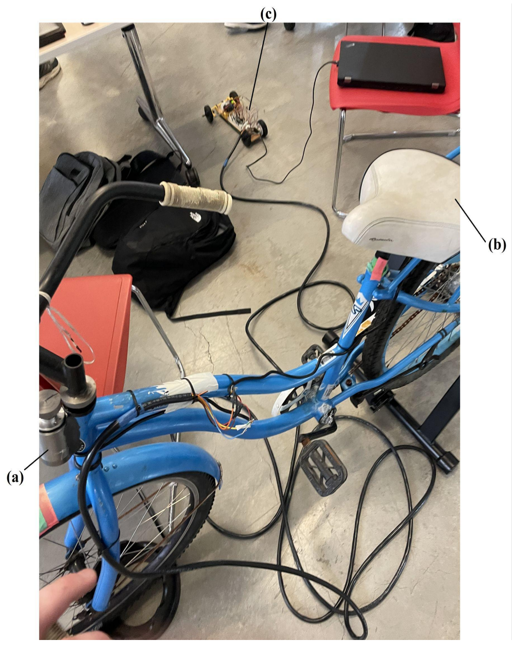
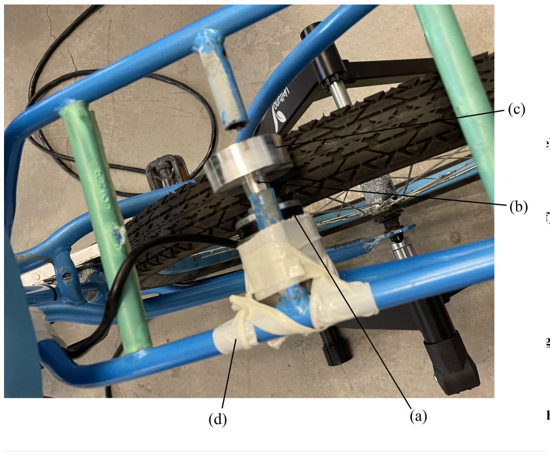


Fig 1. Complete Stationary Bike / Robot system. (a) Direction encoder (b) Angular Velocity Encoder (c) Robot

2.1 Subsystem 1: Stationary Bicycle

The output angular direction and velocity from the bicycle subsystem is obtained via incremental encoders fitted with rolling elements that engage the handle bars for direction and the rear wheel for angular velocity (2). Various improvements could have been made in this subsystem, namely the mounting of the encoders. A better suited mounting mechanism that would allow for adjustments so as to provide proper contact without overloading the bearing inside the rotary encoder could have been created by welding, but without the skills or equipment, metal epoxy was used instead. Further, other simpler methods could have been used to obtain angular velocity from the rear wheel of the bicycle. For example, a hall effect sensor could have been used to replace the rear rotary encoder.



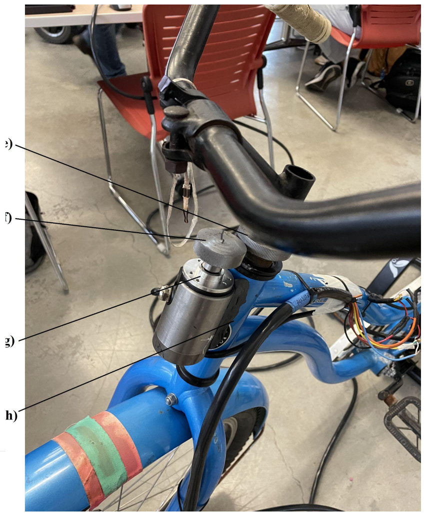
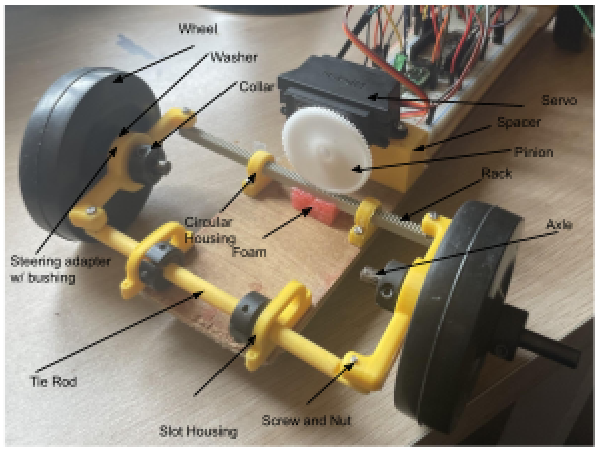


Fig 2. Bicycle Subsystem. Left: (a) Rotary Encoder (b) Set screw (c) Machined Roller (d) Improper mounting using adhesive tape (e) Machined roller for handle bar (f) Machined roller for direction encoder (g) Rotary Encoder (h) Improper mounting with epoxy putty.

2.2 Subsystem 2: Robot / State Machine

The car consists of a rear drive transmission, front steering transmission, and circuit with microcontroller fitted to a sheet of wood. The rear drive uses the micro kit motor, and an input and output gear with a ratio of six were added in order to reduce the torque necessary from the motor. The input gear fits over the motor axle with a force fit, and same with the output over the axle. Custom housings were 3D printed, and easy-install flanged plastic bushings fitted inside. The axle is constrained by collars on the outside of each wheel, and inside of each housing. Washers are also used on the non-flanged side of the bushings.



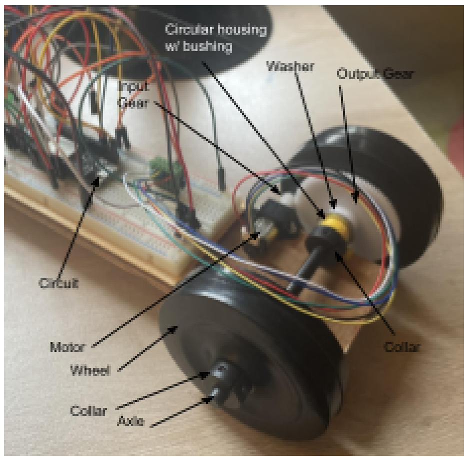


Fig. 3 Components of Robot

The steering works by using a rack and pinion to move a simple linkage. As the servo spins the pinion, the rack moves and pushes/pulls the wheels to the desired angle. The servo is raised to the necessary height with a 3D printed spacer. A piece of foam is placed under the rack to maintain contact with the pinion, and the coefficient of friction between foam and wood, and foam and rack, is low enough to not interfere with movement. The front tie rod is axially constrained with the use of collars, giving the wheels something to pivot against. The housing the tie rod fits through is a slot, allowing it to move perpendicular to the axis as necessary when turning. The yellow housings attaching the wheels, rack, and tie rod are fitted with the same plastic bushings as earlier, and a washer is again used on the non-flanged side. The wheels are fitted on the outside, and collars on the inside and outside again used to constrain the transmission.

Gear ratios needed to be used to ensure the small microkit motor could power the car. Choosing an input gear with 16 teeth and output with 96 gives a gear ratio of 6, or 6 times the torque for a ⅙ reduction in speed. The motor used has a max torque of 1.3 kg*cm, or 12.8 N*cm, so the output gear will put out 76.8 N*cm of torque. The car weighs about 5lbs, or 22 N, and the wheels are 3" in diameter, or a radius of 3.8cm. This means it would require 83.6 N*cm to spin the whole car around the axle, let alone just roll the wheel. The microkit motor has more than enough torque with this added gear ratio to achieve that.

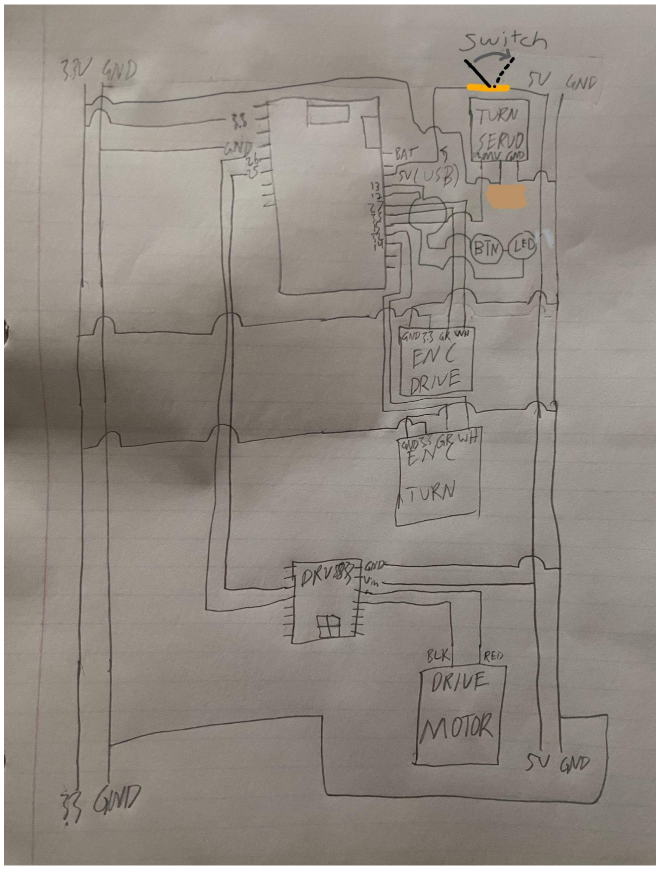
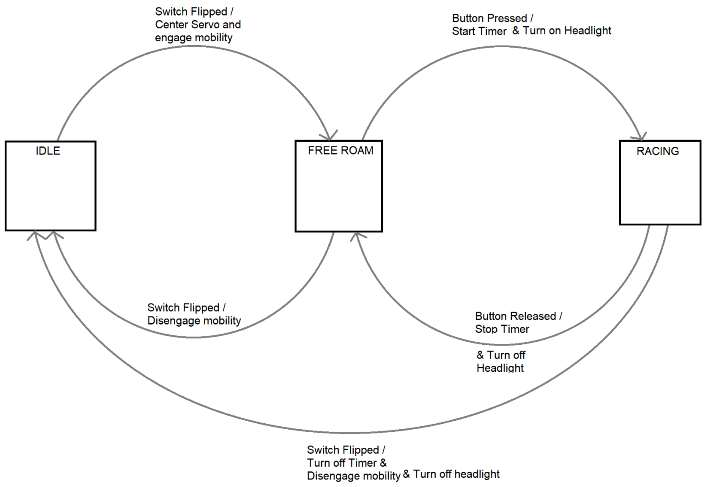


Fig 4. Wiring Diagram



Switch Flipped / Center Servo and engage mobility
Button Pressed / Start Timer & Turn on Headlight
IDLE
FREE ROAM
RACING
Switch Flipped / Disengage mobility
Button Released / Stop Timer & Turn off Headlight
Switch Flipped / Turn off Timer & Disengage mobility & Turn off headlight

Fig 5. State machine Diagram

3 Project Reflection

An advice we wish we had received before venturing into this project is that time would be very limited and to keep the scope of the project small. Our initial goal was to have two complete systems with wireless communication between them. Issues that arose due to wireless communication included the risk of loss of control due to latency or EM fluctuations. Due to funding and time constraints we decided to keep wired connections for simplicity in addition to having just one complete system. Despite this, we were successful in keeping costs low by reusing given parts, printing at Jacobs, and finding materials around the house. Treating this project as more of a proof-of-concept project rather than a polished product may be helpful.

Appendix

A.1 Bill of Material

Robot Subsystem:

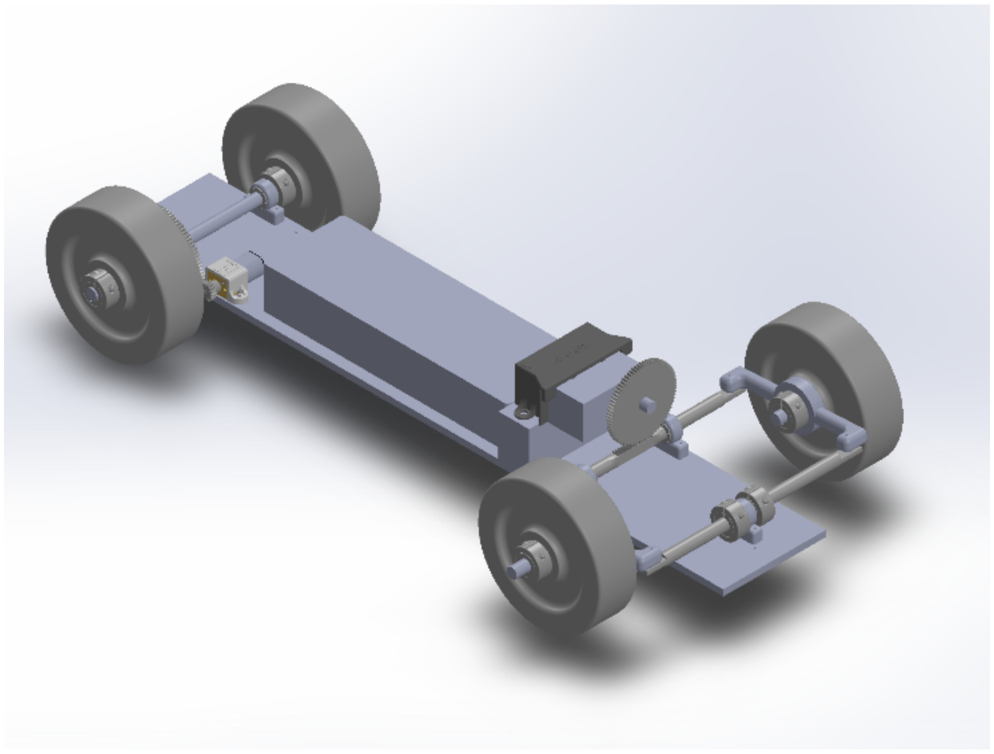
Item No.	Part Number	Source	Description	Qty.
1	n/a	n/a	Wooden Robot Base; Breadboard	1
2	n/a	Jacobs Hall	3D Printed Circular Bearing Housing	4
3	n/a	Jacobs Hall	Servo Spacer	1
4	n/a	Polulu.com	Servo Mounting Bracket	1
5	n/a	Jacobs Hall	Steering Adapter	2
6	#2215	Polulu.com	DC Gearmotor (MicroKit)	1
7	2662N312	McMaster-Carr	20 Deg. Pressure Ang. Plastic Gear	1
8	2662N321	McMaster-Carr	20 Deg. Pressure Ang. Plastic Gear	1
9	2662N55	McMaster-Carr	20 Deg. Pressure Ang. Plastic Gear Rack	1
10	6435K12	McMaster-Carr	Clamping Shaft Collar	10
11	1281N12	McMaster-Carr	Flanged Sleeve Bearing	4
12	5227T232	McMaster-Carr	¼” 12L14 Rod	2
13	n/a	Jacob Hall	3D Printed Slot Bearing Housing	2
14	n/a	Garage/home	#56 Machine screws	16

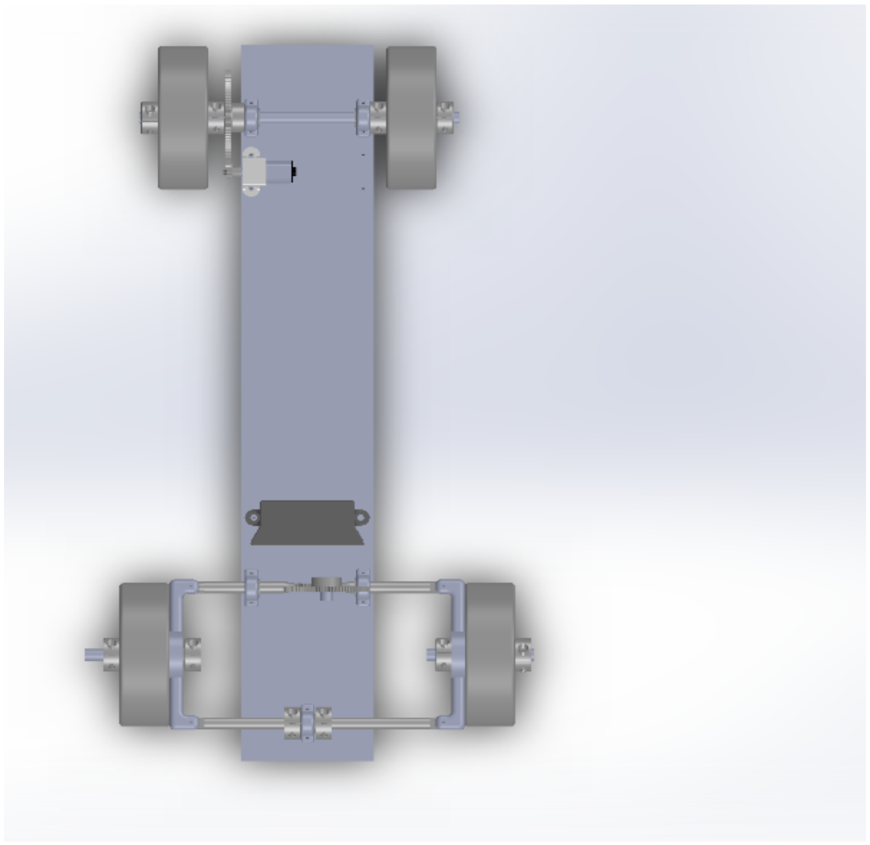
15	n/a	Garage/home	#56 Machine nuts	16
16	n/a	Garage/home	¼" Washers	4
17	n/a	MicroKit	Foam block	1

Bicycle Subsystem:

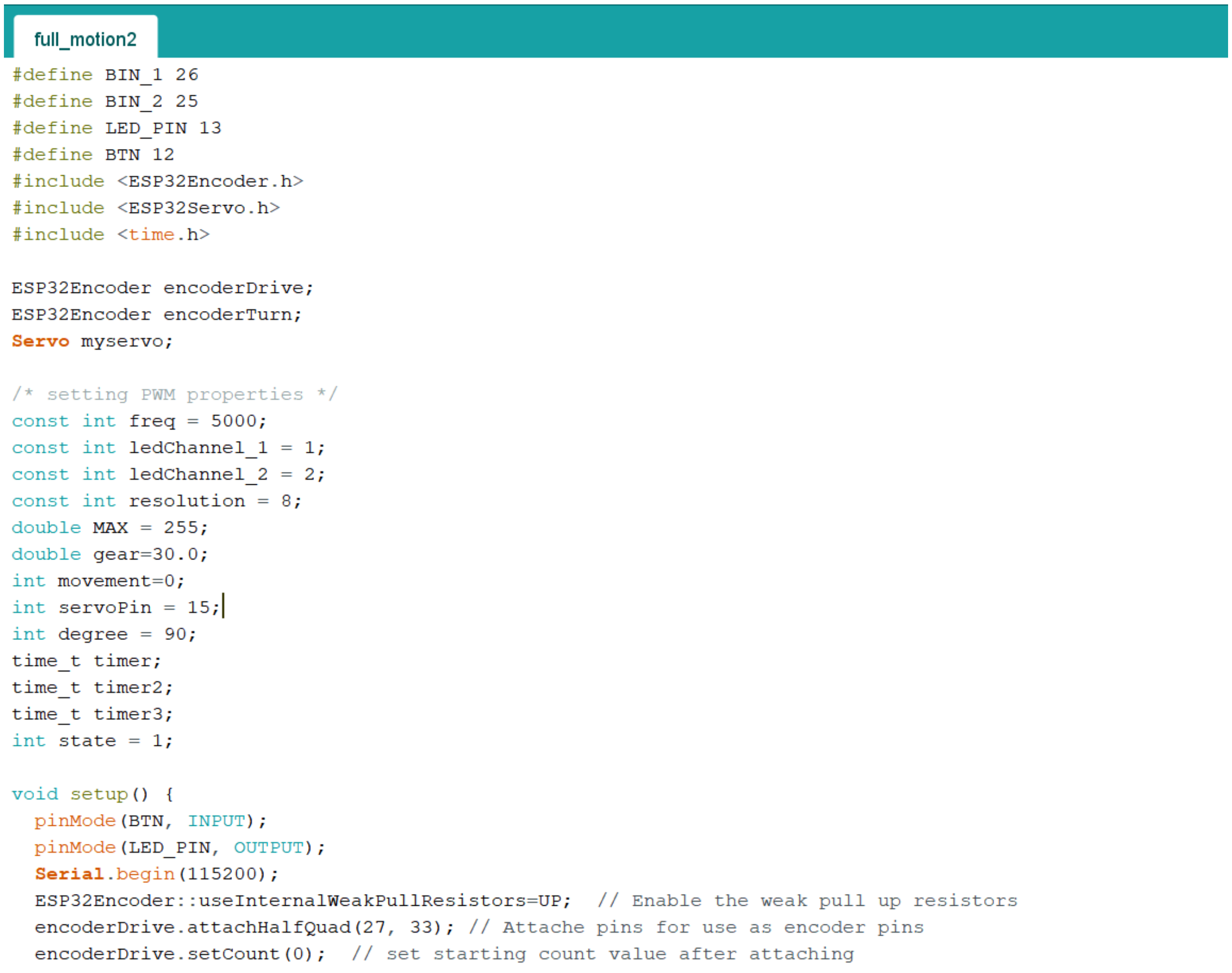
Item No.	Part Number	Source	Description	Qty.
1	Taiss/AB 2	Amazon	100Pulse/Rev Rotary Encoder	2
2	91390A087	McMaster-Carr	M3 Set Screw	2
3	n/a	Downtown Berkeley	Abandoned Bicycle	1
4	n/a	ME Shop	Custom machined encoder mounts	2
5	n/a	ME Shop	Custom machined roller for rear wheel	1
6	n/a	ME Shop	Custom machined roller for handle bar	1
7	n/a	ME Shop	Custom machined roller for direction encoder	1

A.2 CAD





A.3 State Machine Code



full_motion2

```
#define BIN_1 26
#define BIN_2 25
#define LED_PIN 13
#define BTN 12
#include <ESP32Encoder.h>
#include <ESP32Servo.h>
#include <time.h>

ESP32Encoder encoderDrive;
ESP32Encoder encoderTurn;
Servo myservo;

/* setting PWM properties */
const int freq = 5000;
const int ledChannel_1 = 1;
const int ledChannel_2 = 2;
const int resolution = 8;
double MAX = 255;
double gear=30.0;
int movement=0;
int servoPin = 15;
int degree = 90;
time_t timer;
time_t timer2;
time_t timer3;
int state = 1;

void setup() {
  pinMode(BTN, INPUT);
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(115200);
  ESP32Encoder::useInternalWeakPullResistors=UP;  // Enable the weak pull up resistors
  encoderDrive.attachHalfQuad(27, 33); // Attache pins for use as encoder pins
  encoderDrive.setCount(0);  // set starting count value after attaching
```

```
void setup() {
  pinMode(BTN, INPUT);
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(115200);
  ESP32Encoder::useInternalWeakPullResistors=UP;  // Enable the weak pull up resistors
  encoderDrive.attachHalfQuad(27, 33); // Attache pins for use as encoder pins
  encoderDrive.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);
  ESP32PWM::allocateTimer(0);
  ESP32PWM::allocateTimer(1);
  ESP32PWM::allocateTimer(2);
  ESP32PWM::allocateTimer(3);
  myservo.setPeriodHertz(50);
  myservo.attach(servoPin, 750, 2400);
  encoderTurn.attachHalfQuad(32, 14); // Attache pins for use as encoder pins

}
```

```
  /* attach the channel to the GPIO to be controlled */
  ledcAttachPin(BIN_1, ledChannel_1);
  ledcAttachPin(BIN_2, ledChannel_2);
  ESP32PWM::allocateTimer(0);
  ESP32PWM::allocateTimer(1);
  ESP32PWM::allocateTimer(2);
  ESP32PWM::allocateTimer(3);
  myservo.setPeriodHertz(50);
  myservo.attach(servoPin, 750, 2400);
  encoderTurn.attachHalfQuad(32, 14); // Attache pins for use as encoder pins

}
```

```
void loop() {
  switch (state) {
    case 0:
      waitForSwitch();
    break;
    case 1:
      mobility();
      break;
    case 2:
      mobility();
      racing();
      break;

  }

}
```

```
      break;
    case 2:
      mobility();
      racing();
      break;

  }

}

void mobility() {
  movement = (int32_t)encoderDrive.getCount();
  encoderDrive.setCount(0);

  if (movement > 0)
  {
    ledcWrite(ledChannel_1, LOW);
    ledcWrite(ledChannel_2, (min((movement/gear)*MAX, MAX)));
  }
  else if (movement < 0)
  {
    ledcWrite(ledChannel_2, LOW);
    //ledcWrite(ledChannel_1, MAX);
    ledcWrite(ledChannel_1, (min((abs(movement)/gear)*MAX, MAX)));
    //delay(1000);
  }
  else
  {
    ledcWrite(ledChannel_1, LOW);
    ledcWrite(ledChannel_2, LOW);
  }
  degree = (-(int32_t)encoderTurn.getCount()/1000.0*90+90);
  myservo.write(degree);

  if (digitalRead(LED_PIN)==LOW) {
    state = 2;
  }
}
```

```
void racing() {
  if (digitalRead(LED_PIN)==LOW)
  {
    state = 1;
    time(&timer);
    time(&timer2);
  }
  else
  {
    time(&timer2);
  }
  Serial.println("Timer = "+String(-difftime(timer,timer2))+" seconds");
}

void waitForSwitch() {
  //switch toggling is done manually with the switch to toggle motor power as shown in STD.
}
```