ME102B - Mechatronics Design

Final Project Report - Horse Racer

Alex Castillo, Luis Flores, Jesse Medina, Eymon Wong

1 Opportunity

The opportunity stated in P2 was to create a game to help college students de-stress. After going through the process of creating our carnival game, we've updated the opportunity this game presents. Our carnival game is an affordable and somewhat portable source of fun for people of all ages that can be customized to provide different game modes.

2 Strategy

Initially, we planned on using a launcher mechanism as a way to score points for our game. After speaking to the teaching team, we decided to drop the launching mechanism and make the actuation of our racing horses as clean as possible and that's what we ended up achieving. We then decided to have the players throw balls at a target with varying point targets. The objective of the game stayed the same in that the more accurate a player is in a shorter amount of time will determine the winner. Instead of using a force sensor to determine when a target is hit, we instead attached an ultrasonic sensor behind each target and had it read how far away it detected a ball. Different distances meant different point distributions for each player which translated into different amounts of steps a stepper motor would move. The stepper motor was also a change we implemented and used it as the actual horse racing actuation. The DC brushed motor was used to raise flags at the end of the race to indicate which of the two players had won. In the end, we used two Arduinos, one for each player because we ran into an issue with blocking code. We wanted to only use one controller but in the end used an Arduino for each player because it proved difficult in the rest of the allotted time to fix the blocking code issue. The force sensor (FSR) was used to communicate to the other Arduino when one player had won the race and reset the game. Another change made was in regards to the analog sensor and how we utilized it. We planned to use a potentiometer to vary the speed at which the game could be played but ultimately used a sliding potentiometer to turn the game on and off.

3 Integrated Physical Device

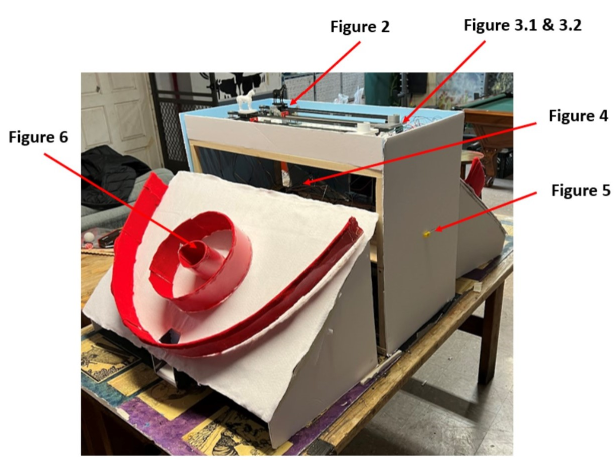
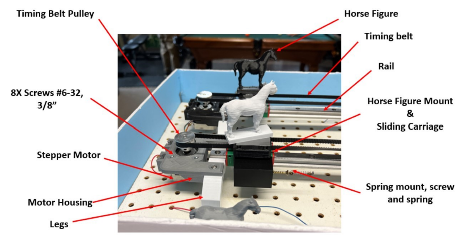
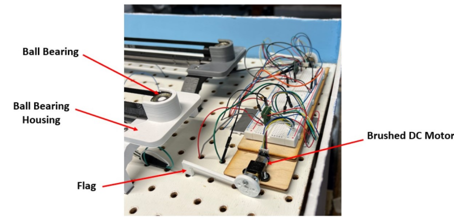


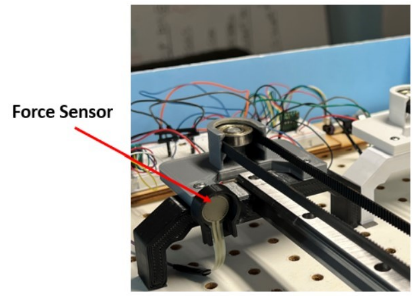
Figure 1: Image of fully integrated system



(a) One end of the railing system

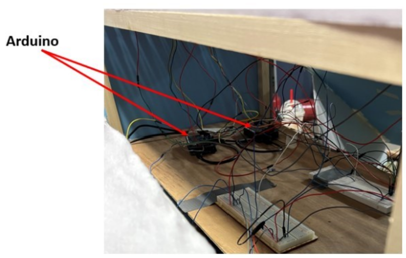


(b) Opposite end of railing system

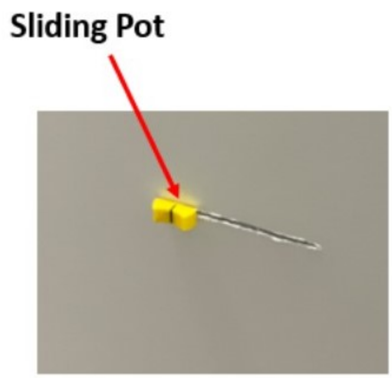


(c) Illustrates FSR position

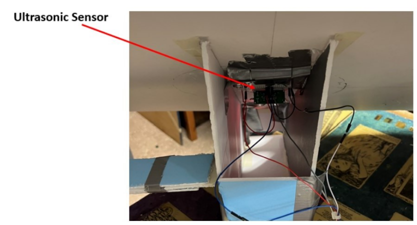
Figure 2: Railing system



(a) Arduino and wiring housing



(b) Sliding Potentiometer

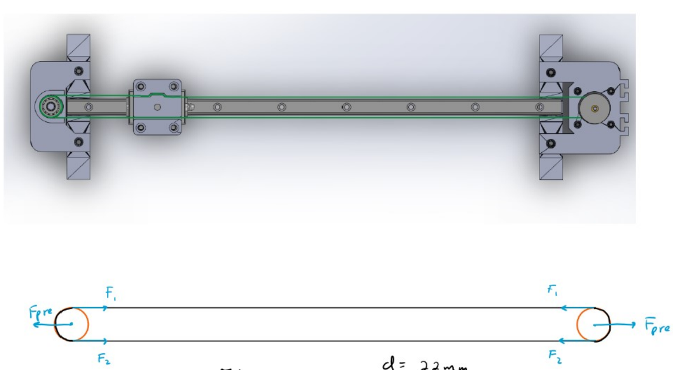


(c) Positioning of the Ultrasonic sensor

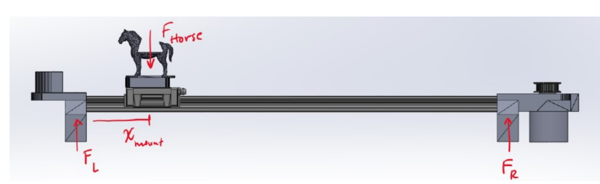
Figure 3: Housing for Arduino and Other Sensors

4 Function-Critical Decisions

The functional-critical decisions began with choosing the timing pulley and belt. The GT2 timing pulley included a 6 mm belt with 36 teeth and a 5 mm bore. The timing belt included 582 teeth on a 2mm tooth pitch which works with the pulley since it also has a 2 mm pitch.



(a) Force diagram for pulley system



(b) Force diagram of the racing horse on the system

Figure 4: Forces acting in the system

First, we look at the pulley system on both ends to determine the pretension.

$$F_1 = F_i + \tau/d, F_2 = F_i - \tau/d \tag{1}$$

$$F_i = \frac{F_{pre}}{2} = \tau/d \rightarrow F_{pre} = 2\frac{\tau}{d} = 2\frac{0.013kgm}{0.022m} = 1.18N \tag{2}$$

For the mounted portion sliding along the rail, we have the following calculations:

$$\sum M_L = 0 = F_{Horse}(x_{mount}) - F_R(0.5m) \tag{3}$$

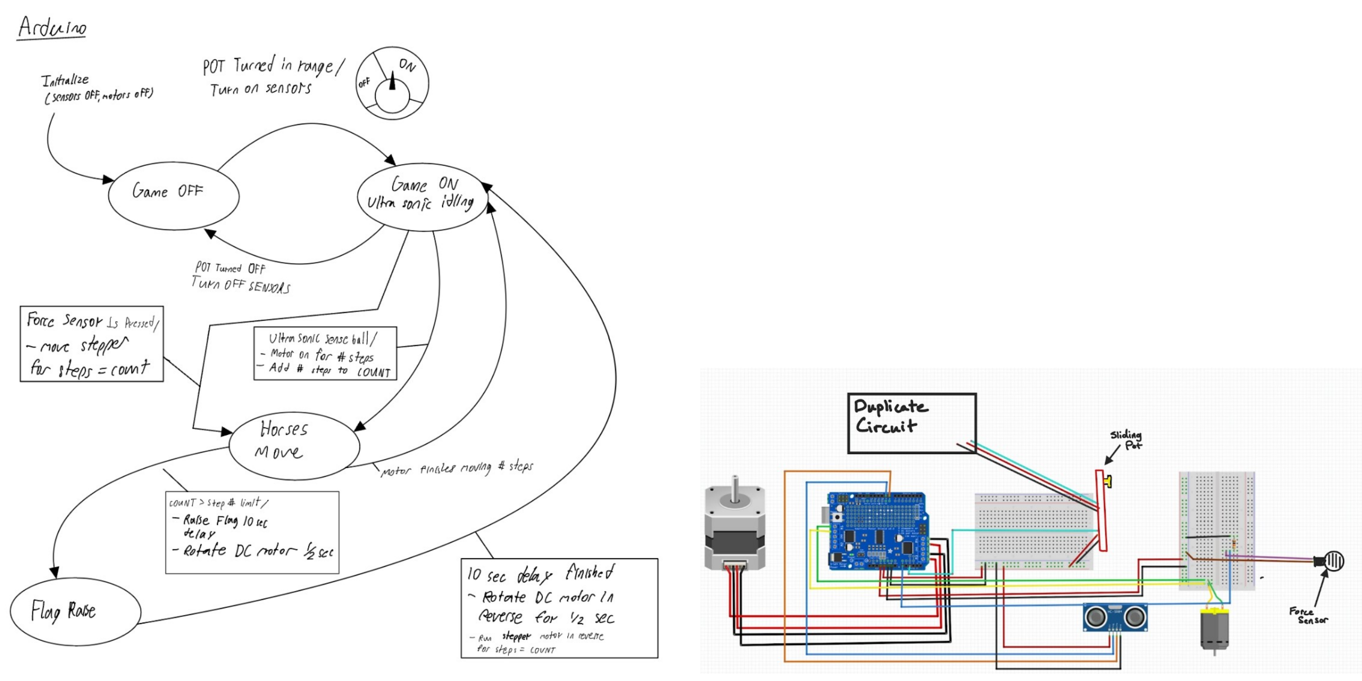
$$F_R = \frac{F_{Horse}x_{mount}}{0.5m} = 2(0.051kg)(9.81m/s^2)x_{mount} = 1.00x_{mount}N \tag{4}$$

$$\sum F_y = 0 = F_L + F_R - F_{Horse} \tag{5}$$

$$F_L = F_{Horse} - F_R = F_{Horse} - 2F_{Horse}x_{mount} = F_{Horse}(1 - 2x_{mount}) = (0.051kg)(9.81m/s^2)(1 - 2x_{mount}) \tag{6}$$

$$F_L = 0.500(1 - 2x_{mount})N \tag{7}$$

5 Circuit Diagram & State Transition Diagram



(a) State Diagram

(b) Circuit Diagram

Figure 5: State and Circuit Diagram (NOTE: Create two of the same circuit diagram shown above, and connect both onto the sliding potentiometer)

6 Reflection

Going through the motion of creating a project of this scope is bound to teach us a few things about teamwork. There were a few things we found that worked and few that didn't work so much while collaborating over the span of the semester. Our communication was key throughout the semester with two scheduled meetings a week where we could go over the aspects of the project. We communicated well throughout the semester and all kept on the same page. Our communication allowed us to turn in our deliverables P1-4 on time and allowed us to brainstorm ahead of time. While the work we produced was guided by the deliverable due dates, we felt that we needed to start working on the hardware way ahead of time so that we could discover the obstacles that would come with integrating the software. We wish we would've known to choose the ESP32 instead of the Arduino ahead of time so that we could have only used one microcontroller.

7 Appendix

A	B	C	D	E	F	G	H	I
#	Part Name	Part Number	Amount	Cost	Total Cost	Link	Purchaser	Vendor
1	Timing Belt GT2 Profile	1184	2	$9.95	$19.90	https://www.adafruit.com/product/118		Adafruit
2	DC brushed motor		2	$0	$0	Lab Kit		
3	Bread boards	N/A	2	$13.00	$26.00	https://www.amazon	Eymon	Amazon
4	#6–32 3/4in hardware screws		8	$1.38	$11.04	https://www.lowes.com/pd/Hillman-6-3		Lowes
5	#6–32 3/8in hardware screws		6	$1.38	$8.28	https://www.lowes.com/pd/Hillman-6-3		Lowes
6	Elmer's® Black Core Foam Board, 20" x 30"		4	$8	$32	https://www.michaels.com/elmers-bla		Michaels
7	Sliding Potentiometer		2	$0	$0	Kit		
8	Radial Ball Bearing 608ZZ	1178	1	$6.95	$6.95	https://www.adafruit.com/product/117		Adafruit
9	15mm Diameter Linear Bearing Pillow	1860	2	$21.95	$43.90	https://www.adafruit.com/product/186		Adafruit
10	Linear Bearing Supported Slide Rail -	1861	2	$29.95	$59.90	https://www.adafruit.com/product/186		Adafruit
11	Aluminum GT2 Timing Pulley - 6mm E	1253	2	$11.95	$23.90	https://www.adafruit.com/product/125		Adafruit
12	Stepper motor - NEMA-17 size - 200 s	324	2	$14.00	$28.00	https://www.adafruit.com/product/324		Adafruit
13	Cardboard	N/A	2	$0	$0	Previously Owned	Alex	
14	Wood Plank	N/A	5	$4.80	$24.00		Luis	Home Depot
15	36x36 Basic Felt	N/A	4	$4.49	$17.96		Luis	Michaels
16	Arduino	N/A	2	$0	$0	Arduino Kits	Luis	
17	Motor Shield for Arduino	1438	2	$19.95	$39.90	https://www.adafruit.	Eymon	Adafruit
18	Force Sensor	N/A	2	$0	$0	Previoulsy Owned	Luis	
19	Ultrasonic Sensor	N/A	2	$0	$0	Lab Kit		
20	3D-printed parts	N/A		$0	$0			Citris Invention Lab
				Overall Cost:	$341.73			

Figure 6: Bill of Materials

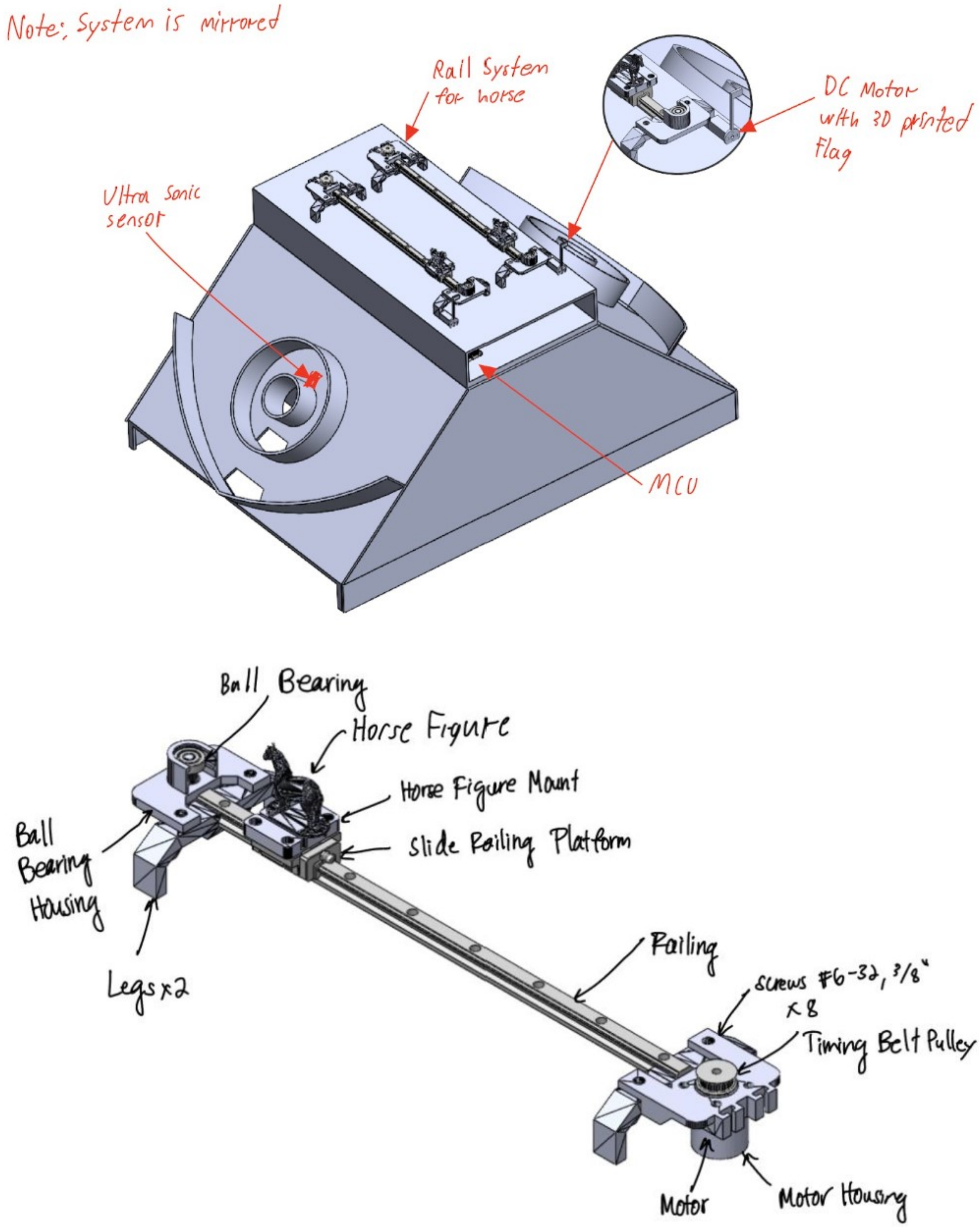


Figure 7: CAD assembly of entire setup

```
/*
Final code for project, utilizes ultra-sonics, potentiometer, motors,
and force sensors

*/

#include <Adafruit_MotorShield.h>
#include <Arduino.h>
// Create the motor shield object with the default I2C address
Adafruit_MotorShield AFMS = Adafruit_MotorShield();
Adafruit_StepperMotor *Motor1 = AFMS.getStepper(200, 2);
Adafruit_DCMotor *myMotor = AFMS.getMotor(1);

// Settup variables:

int pressurePin = A0;//Force sensor
int force;

int POT = A2;
int potReading;

//Ultra-Sonic Sensors
int trigPin = 9;    // TRIG pin
int echoPin = 8;    // ECHO pin
int pastDist;
float duration_us, distance_cm;

//Potentiometer


// Game Variables:

int state = 1;

int goal = 1950;
int progress = 0;
int onePoint = 400;
int twoPoint = 200;
int threePoint = 100;
int distance1start = 1;
int distance1end = 10;
int distance2start = 15;
int distance2end = 26;
int distance3start = 10000; //not in use right now
int distance3end = 10001;

//Game Settup
void setup() {
  Serial.begin(9600);           // set up Serial library at 9600 bps
  while (!Serial);
  Serial.println("Stepper test!");

```

Figure 8: Code block 1

```

  // configure the trigger pin to output mode
  pinMode(trigPin, OUTPUT);
  // configure the echo pin to input mode
  pinMode(echoPin, INPUT);

  if (!AFMS.begin()) {         // create with the default frequency 1.6KHz
  // if (!AFMS.begin(1000)) {  // OR with a different frequency, say 1KHz
    Serial.println("Could not find Motor Shield. Check wiring.");
    while (1);
  }
  Serial.println("Motor Shield found.");


  Motor1->setSpeed(20);
}

void loop() {
//while
  // generate 10-microsecond pulse to TRIG pin
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
   // measure duration of pulse from ECHO pin
  duration_us = pulseIn(echoPin, HIGH);

  // calculate the distance
  distance_cm = 0.017 * duration_us;

  switch (state) {
    case 1:
    Serial.println("Waiting for game to start...");
      if (StartGame() == true){
        state = 2;
        }


    break;

    case 2:
     Serial.print("Score: ");
     Serial.println(progress);

     //Represents force sensor reading if other player wins
     force = analogRead(pressurePin);
     Serial.print("Force: ");
     Serial.println(force);

     //Pot reading
     potReading = analogRead(POT);
     Serial.print("Pot reading: ");
     Serial.println(potReading);
```

Figure 9: Code block 2

```
    Serial.print("Pot reading: ");
    Serial.println(potReading);

    // Scoring systems
    if (force>50){
     state = 7;
     }
    else if (progress >= goal){
     state = 6;
    }
    else if  (potReading < 200){
     EndDurringGame();
     reset_motor();
     progress = 0;
     state = 1;
     }
   else{
     if ((distance_cm > distance1start) && (distance_cm <= distance1end)){
     Serial.println(distance_cm);
     state = 3;
     }

     if ((distance_cm > distance2start) && (distance_cm <= distance2end)){
     Serial.println(distance_cm);
     state = 4;
     }

     if ((distance_cm > distance3start) && (distance_cm <= distance3end)){
     Serial.println(distance_cm);
     state = 5;
     }
   }


   break;

   case 3:

     Serial.println("Motor run for One Point!");

     run_motor_score1();
     state = 2;

   break;

   case 4:

     Serial.println("Motor run for Two Points!");

     run_motor_score2();
     state = 2;

```

Figure 10: Code block 3

```
    break;

    case 5:

      Serial.println("Motor run for Three Points!");

      run_motor_score3();
      state = 2;

    break;

    case 6:

      Serial.println("Winner");
      Serial.println("Resetting...");

      raise_flag();

      reset_motor();
      progress = 0;
      state = 1;
      delay(5000);

    break;

    case 7:

      Serial.println("Loser");
      Serial.println("Resetting...");

      reset_motor();
      progress = 0;
      state = 1;
      delay(5000);

    break;

  }
  }

bool StartGame(){
  potReading = analogRead(POT);
    if(potReading > 200){
      Serial.print("Starting Game with Pot reading: ");
      Serial.println(potReading);
      return true;
    }
    else{
      return false;
    }
}

```

Figure 11: Code block 4

```
bool EndDurringGame(){
  potReading = analogRead(POT);

  if (potReading < 200){
    Serial.print("Current Pot Value: ");
    Serial.println(potReading);
    Serial.println("Ending the game");
    return true;
  }
  else {
  return false;
  }
}
void run_motor_score1(){
  if (progress + onePoint >= goal){
    int leftover;
    leftover = goal - progress;
    Motor1->step(leftover, BACKWARD, INTERLEAVE);
    progress = progress + leftover;
  }
  else{
    Motor1->step(onePoint, BACKWARD, INTERLEAVE);
    progress = progress + onePoint;
  }
}

void run_motor_score2(){
  if (progress + twoPoint >= goal){
    int leftover = goal - progress;
    Motor1->step(leftover, BACKWARD, INTERLEAVE);
    progress = progress + leftover;
  }
  else{
    Motor1->step(twoPoint, BACKWARD, INTERLEAVE);
    progress = progress + twoPoint;
  }
}

//not in use
void run_motor_score3(){
   if (progress + threePoint >= goal){
    int leftover = goal - progress;
    Motor1->step(leftover, BACKWARD, INTERLEAVE);
    progress = progress + leftover;
  }
  else{
    Motor1->step(threePoint, BACKWARD, INTERLEAVE);
    progress = progress + threePoint;
  }
}
```

Figure 12: Code block 5

```
}

void reset_motor(){
  Serial.println("Resetting the motor");
  Motor1->step(progress, FORWARD, INTERLEAVE);
}

// Code for flag raising
void raise_flag(){
  Serial.println("Raising the flag");
   uint8_t i;
   int maxSpeed = 40;

  Serial.println("Raising Flag");

  myMotor->run(FORWARD);
  for (i=0; i<maxSpeed; i++) {
    myMotor->setSpeed(i);
    delay(10);
  }
  for (i=maxSpeed; i!=0; i--) {
    myMotor->setSpeed(i);
    delay(10);
  }

  delay(5000);

  Serial.println("Putting Flag down");

  myMotor->run(BACKWARD);
  for (i=0; i<maxSpeed; i++) {
    myMotor->setSpeed(i);
    delay(10);
  }
  for (i=maxSpeed; i!=0; i--) {
    myMotor->setSpeed(i);
    delay(10);
  }

  myMotor->run(RELEASE);
  delay(1000);
}
```

Figure 13: Code block 6