Mec 101 - Fall 2024 Robot Design Final Project Report

Team 9 - Cat Demands

Yufan Pan, Aaron Lin, Ethan Luong, Matthew Rogowski

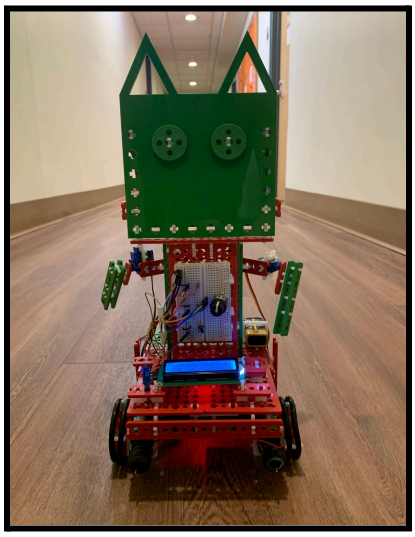


Figure #1.
Front of the Cat
(arms connected and moved by servos)

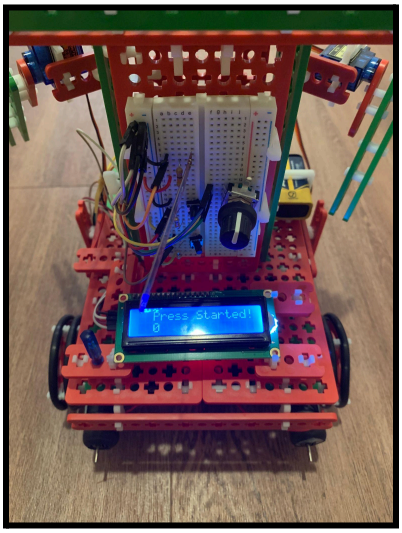


Figure #2.
LCD Display
&
Interactive Buttons

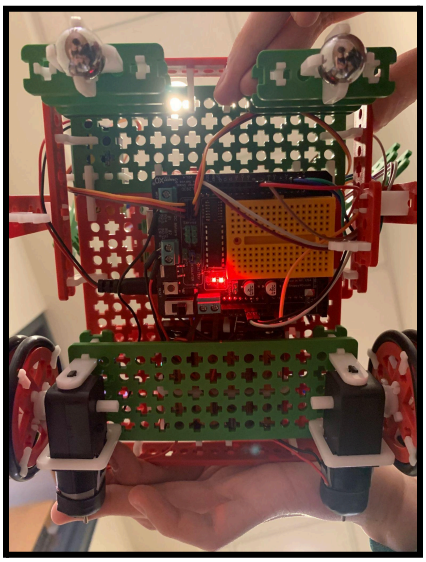


Figure #3.
Underside of the Cat
(houses the Arduino, battery pack, both DC motors)

Figure #4.
Side View of the Cat

Introduction

Our robot is an interactive cat-bot to mimic the game “Bop It”. More specifically, our robot gives the user a series of random commands allowing them to interact with the bot differently. These commands are displayed on the LCD. Each time the user completes the correct action, the robot will provide a new command for the user to complete. This will repeat until the user performs the wrong action or fails to perform the correct action in three seconds. The LCD will also keep track of the user’s score based on the number of correct actions that they’ve performed.

The three main actions the user will perform are to bend, twist, or push. Our robot will detect these actions through the flex sensor, rotary encoder, and button, respectively. The user will interact directly with these sensors (ie bending the flex sensor, rotating the rotary encoder, and pushing the button). These sensors are attached to the front of the robot. We decided to make this robot for children to help them understand basic motor functions and simple actions. We tried appealing to this younger demographic by attaching a cat head to the main body.

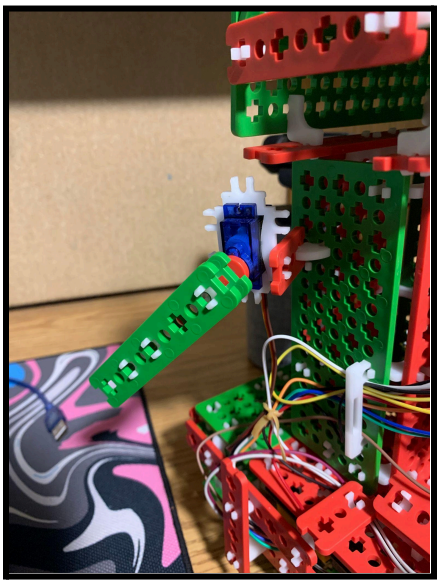
The cat-bot also contains two movable arms that are each controlled by a servo motor. The arms are programmed to rotate 140 degrees every time the user performs the correct action. In addition, after the game is over, the robot will perform a little dance with its arms. We also attached four wheels to the robot to stabilize the robot and allow it to move around once the game is completed as well. The main function of the wheels is to allow the robot to spin around in two circles once the game is completed.

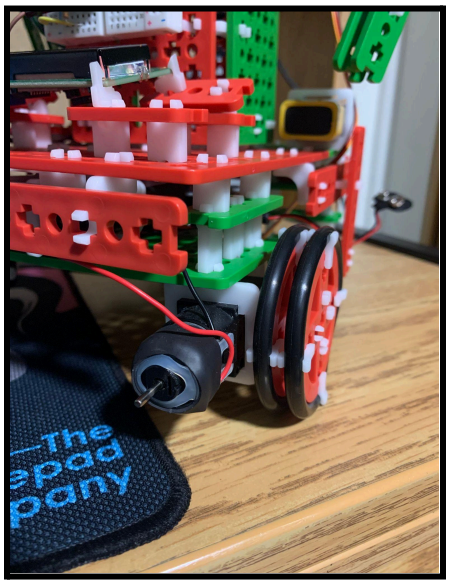
Mechanism Design

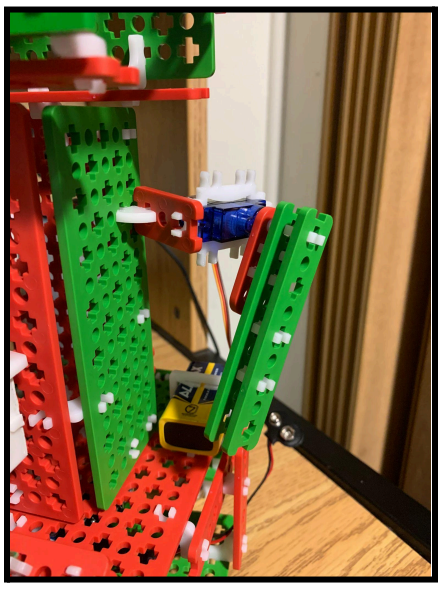
The two arms (connected to the either side of the body), are in place and controlled by two servo motors. The servos are programmed to move 140 degrees, once the action is completed. The arms which are connected to the servos are placed at different 0 point positions, in order to give a more spontaneous motion for the cat. They are also connected and oriented in a way in which they don't get stuck on any wires or hit the head of the cat, further getting stuck. The movement action is completed when either the push it, twist it, or bend it actions are completed, to give the player a sense of accomplishment, and to keep children engaged.

The arms have another movement for when the game is over, compared to when an action is completed. The angle is kept at 140 degrees, but this time is repeated nine times with a delay to create a sense of the game being over. This movement is faster and more repetitive, yet accomplished by the same servos at the same angle.

Along with the arms flying around when the game is over, we have two DC motors connected to the bottom of our robot. To enhance the features of when the game ends, we have the DC motors rotate in opposite directions for 10 seconds. This is after the game over motion of the arms is completed. Both motoros are supported by a beam which acts as an axel, with two wheels on each one for support, as the robot turned out to be quite heavy.







Custom Part Design

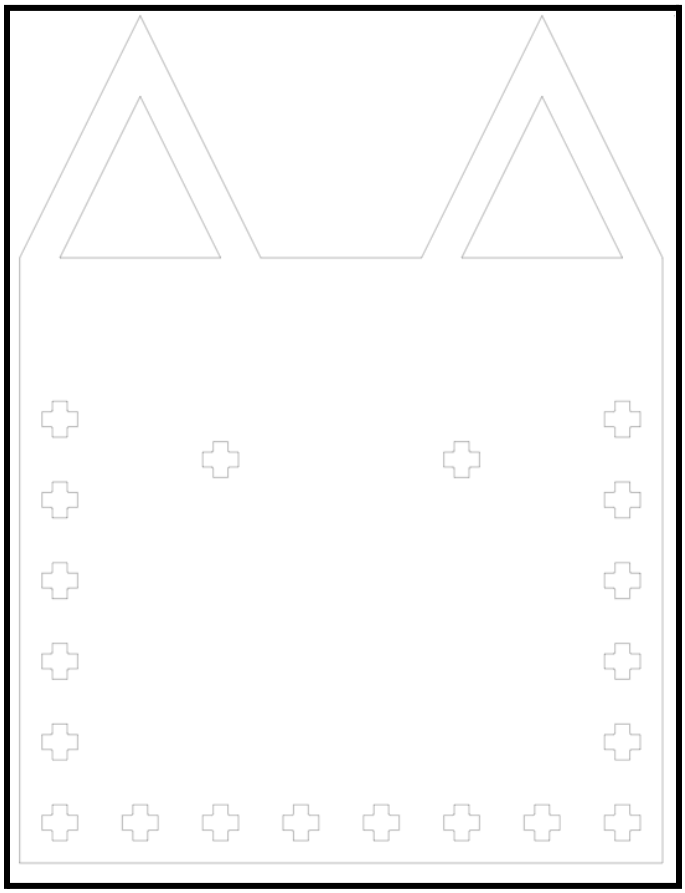


Figure #5: Custom part was created for the face of the robot. Used for aesthetic purposes and creation of the head. The “x” holes along the side were placed, so that surrounding parts could be connected to create the full head. Followed by holes in the center for a connection for the eyes, to display more cat like features.

—----------

YouTube Link: https://youtube.com/shorts/vw4mJVFfnbA

—----------

Bill of Materials

Quantity	Source	Description
7	SnappyXO Kit	3X Plate
4	SnappyXO Kit	4X Plate
4	SnappyXO Kit	8X Beam

10	SnappyXO Kit	3X Beam
11	SnappyXO Kit	2X Beam
2	SnappyXO Kit	6X Beam
4	SnappyXO Kit	5X Beam
2	SnappyXO Kit	Half Servo Horns
2	SnappyXO Kit	Breadboard Mount
2	SnappyXO Kit	Servo Mount
4	SnappyXO Kit	Wheel
4	SnappyXO Kit	Tire
16	SnappyXO Kit	Wheel Clips
2	SnappyXO Kit	Wheel Flange
2	SnappyXO Kit	Motor Mount
1	SnappyXO Kit	6V Battery Holder
1	SnappyXO Kit	9V Battery Holder
8	SnappyXO Kit	60 Degree Clips
2	SnappyXO Kit	Sliding C Clips
15	SnappyXO Kit	L H-Clips
16	SnappyXO Kit	M H-Clips
8	SnappyXO Kit	S H-Clips
3	SnappyXO Kit	Arduino Board Clips
24	SnappyXO Kit	L-Clip
4	SnappyXO Kit	C-Clip
1	Mechatronics Kit	Arduino Uno
2	Mechatronics Kit	330 Ohm Resistors
1	https://www.amazon.com/dp/B0185FIOPE?ref=ppx_yo2ov	47k Ohm Resistors

	_dt_b_fed_asin_title	
5	https://www.amazon.com/dp/B07GD2PGY4?ref=ppx_yo2ov_dt_b_fed_asin_title&th=1	11.8 inch Male to Female,
9	https://www.amazon.com/dp/B07GD2PGY4?ref=ppx_yo2ov_dt_b_fed_asin_title&th=1	11.8 inch Male to Male
1	https://www.adafruit.com/product/377	Rotary encoder
1	https://www.adafruit.com/product/1070	Flex Sensor
1	https://www.amazon.com/SunFounder-Serial-Module-Display-Arduino/dp/B019K5X53O/ref=sr_1_7?crid=3G3TUQ1SUXDJT&dib=eyJ2IjoiMSJ9.0Rf-4IW_epgEqOAsjb_IO-76xVM9dgw78hzgIEM3F9gkQErJo-t1zBCrOHE4Sqg1MWYR0kL5knfaqbT90jfqycTVxIevDnDubXQZwMCTLXTWdkZdGy4geIDyMg92d--ZWyghVcvFCZTy1li3fn5j-aiXIroxnlqgairrCsROEi1IJr7JrSqV-Yj0wLsskEedwPjyEMURWglpFfAWmjZivfCTkGzDw69xOqpej8TP9Cc.9d8iQG1wmL7UZ7IW6TQOKta0ujLmdCtnEK0oW_XGOgA&dib_tag=se&keywords=i2c+lcd+display&qid=1733783212&sprefix=i2c+lcd%2Caps%2C88&sr=8-7	I2C LCD Display
2	Mechatronics Kit	Servo Motor
2	Mechatronics Kit	DC Gearhead Motor
2	Mechatronics Kit	Push Button
1	Mechatronics Kit	Barrel Jack

1	Mechatronics Kit	I- Type
1	Mechatronics Kit	9V Battery
4	Mechatronics Kit	1.5V Battery

Circuit

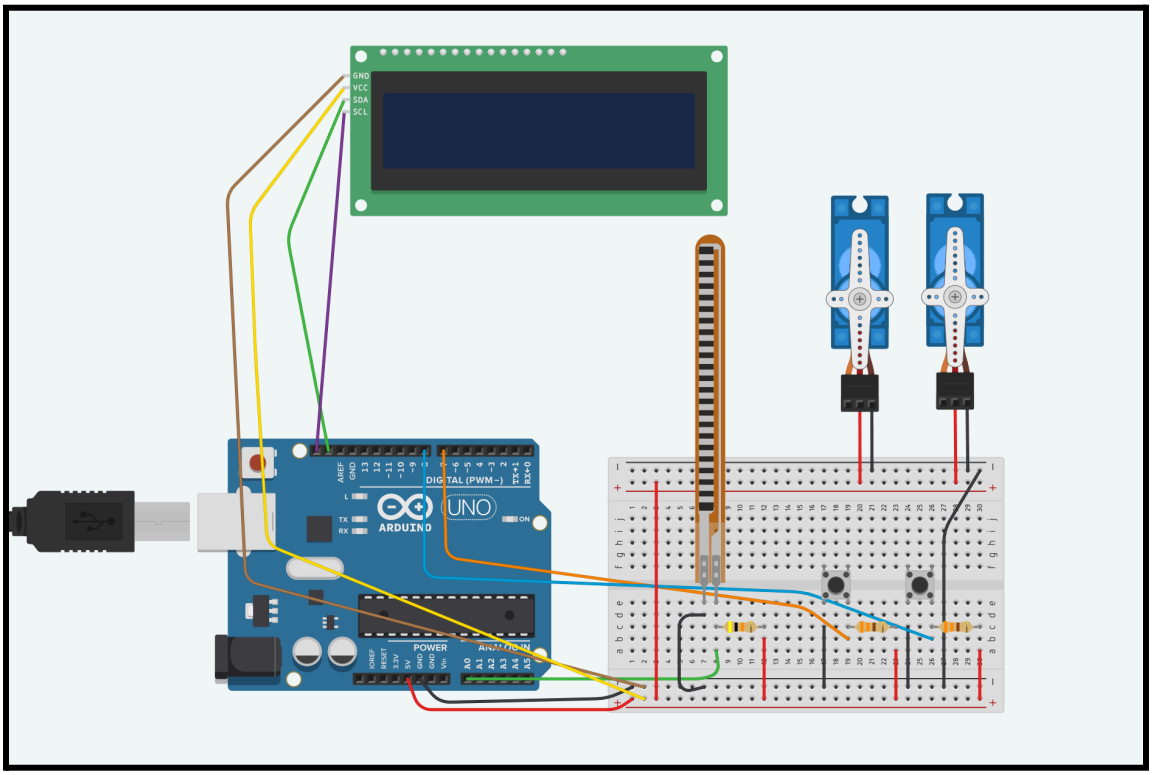


Figure #6: This figure shows the Tinkercad circuit behind our cat-bot. It should be noted that the rotary encoder isn't included in this picture because it isn't an available sensor on the Tinkercad website itself.

Link - https://www.tinkercad.com/things/6lQ8UaNs7pS-demo/editel?returnTo=https%3A%2F%2Fwww.tinkercad.com%2Fdashboard%2Fdesigns%2Fcircuits&sharecode=Di8t-sTQQWnHOGdPlCatoxV3OCDUPoWrv5SrfPbcypg

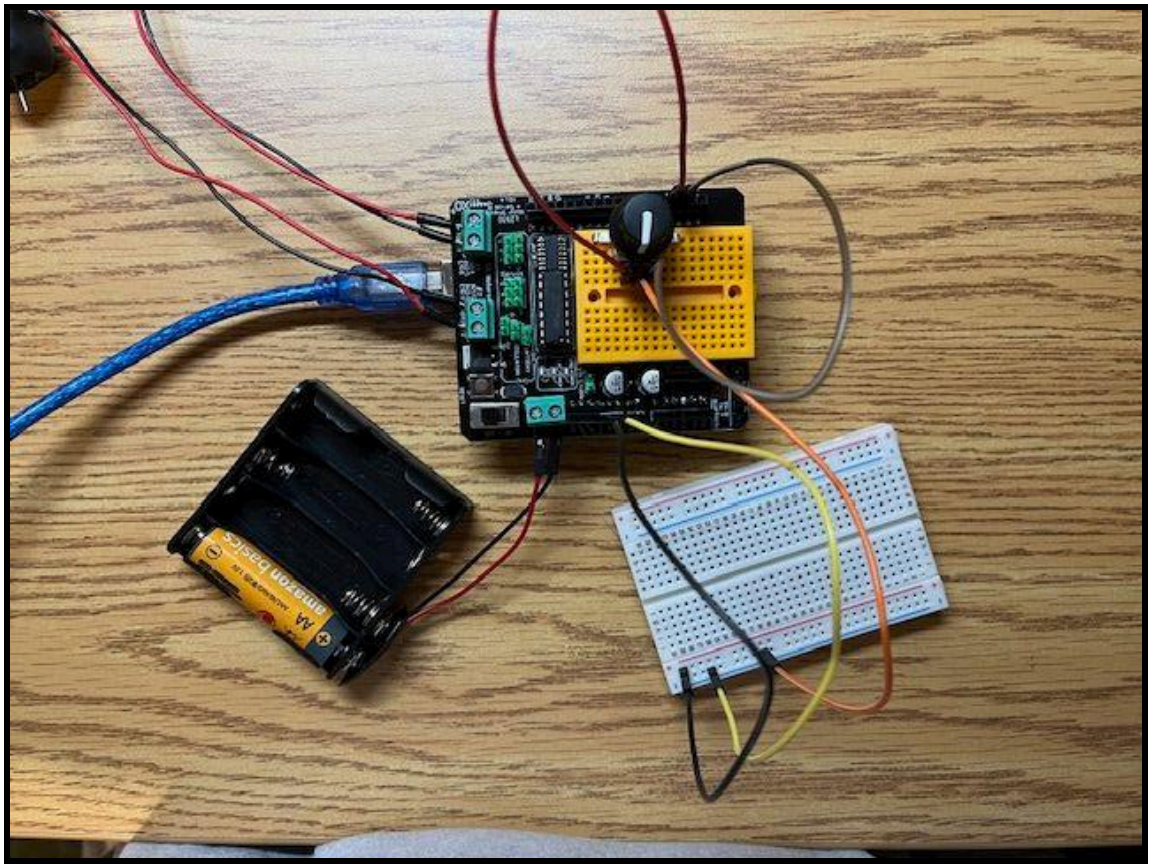


Figure #7: Physical Circuit for Rotary encoder and DC gearhead motor

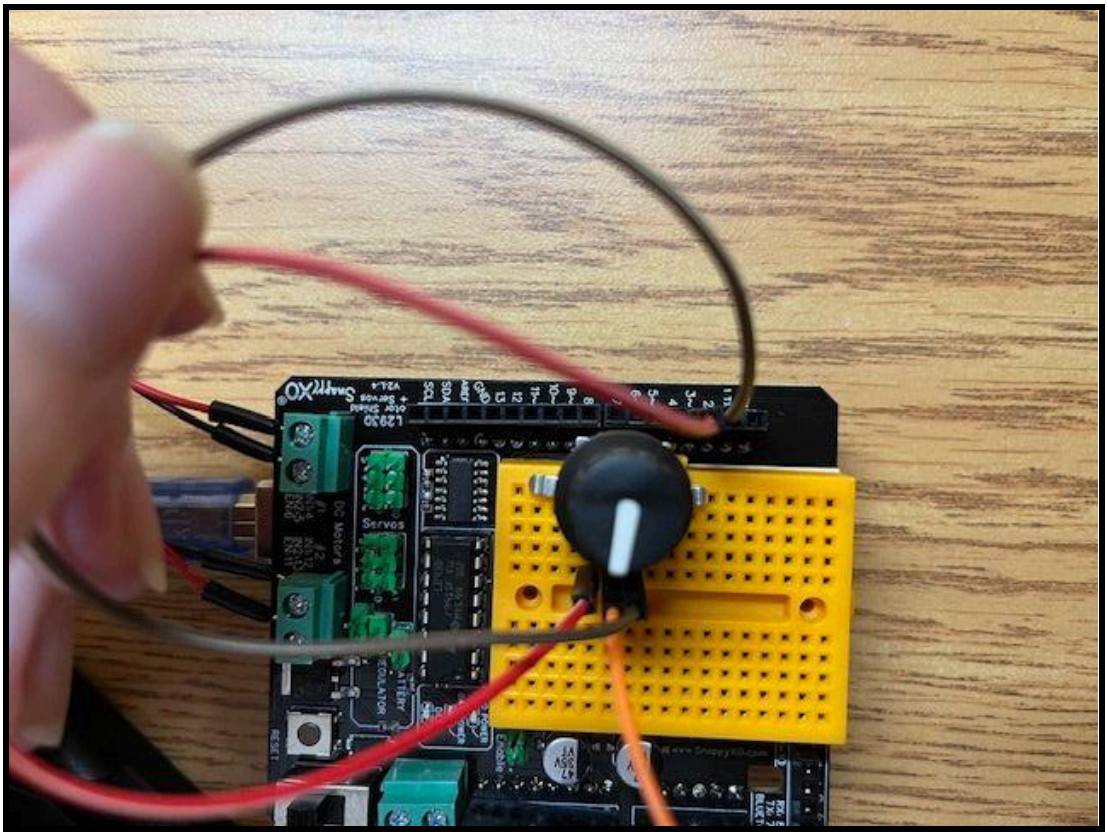


Figure #8: Close up image

Arduino Code

```
#include <LiquidCrystal_I2C.h>
#include <Servo.h>
#define CLK 3
#define DT 2

LiquidCrystal_I2C lcd(0x27,16,2);

int flexPin = A0;
int flexState;
int flexThreshold = 600;

int clkState;
int clkLast;
int dtState;
volatile int rotationDirection = 0;

int pushPin = 8;
int degree = 0;
```

```
Servo servoleft;
Servo servoright;

unsigned long gameTime = 3000;
unsigned long StartTime;

int actionCompleted = 0;
bool gameEnd = false;
bool gameStart = false;
int startButtonPin = 7;
int score = 0;

int motorLeft_InputTwo = 4;
int motorLeft_InputOne = 5;
int motorLeft_Enable = 6;
int motorRight_Enable = 11;
int motorRight_InputOne = 12;
int motorRight_InputTwo = 13;

void setup() {
  Serial.begin(9600);

  lcd.init();
  lcd.clear();
  lcd.backlight();

  randomSeed(analogRead(0));
  pinMode(flexPin, INPUT);
  pinMode(pushPin, INPUT_PULLUP);
  pinMode(startButtonPin, INPUT_PULLUP);

  servoleft.attach(9);
  servoright.attach(10);

  pinMode(CLK,INPUT_PULLUP);
     pinMode(DT,INPUT_PULLUP);

  pinMode(motorLeft_InputOne, OUTPUT);
  pinMode(motorLeft_InputTwo, OUTPUT);
  pinMode(motorLeft_Enable, OUTPUT);
  pinMode(motorRight_InputOne, OUTPUT);
  pinMode(motorRight_InputTwo, OUTPUT);
  pinMode(motorRight_Enable, OUTPUT);

  clkLast = digitalRead(CLK);
```

```
  attachInterrupt(digitalPinToInterrupt(CLK), handleEncoder, CHANGE);
}

void loop() {
  Serial.println("Press Start");

  lcd.setCursor(2, 0);
  lcd.print("Press Start");

  if (!gameStart) {
    if (digitalRead(startButtonPin) == LOW) {
      gameStart = true;
    }
  } else if (gameEnd) {
    resetGame();
    Serial.println("Game Over");
    arm();
    turnRight();
    lcd.setCursor(2, 0);
    lcd.print("Game Over");
    delay(2000);
  } else {
    int action = random(0, 3);
    if (!checkInput(action)) {
      gameEnd = true;
    }
  }
}

void resetGame() {
  gameEnd = false;
  gameStart = false;
  score = 0;
  actionCompleted = 0;
  flexState = 0;
}

bool checkInput(int action) {
  if (action == 0) { // Bend It
    Serial.println("Bend It");
    lcd.setCursor(2, 0);
    lcd.print("Bend It");
    StartTime = millis();
    while (millis() - StartTime <= gameTime) {
      flexState = analogRead(flexPin);
```

```
    if (flexState > flexThreshold) {
      score++;
      Serial.println("Action completed!");
      lcd.setCursor(2, 0);
      lcd.print("Action Done!");
      armOnce(); // Call the servo movement function
      delay(1000);
      return true;
    }
  }
  Serial.println("Action failed!");
  Serial.println(score);
  lcd.setCursor(2, 0);
  lcd.print("Action failed");
  lcd.setCursor(2, 1);
  lcd.print(score);
  delay(1000);
  return false;

} else if (action == 1) { // Push It
  Serial.println("Push It");
  lcd.setCursor(2, 0);
  lcd.print("Push It");
  StartTime = millis();
  while (millis() - StartTime <= gameTime) {
    if (digitalRead(pushPin) == LOW) {
      score++;
      Serial.println("Action completed!");
      lcd.setCursor(2, 0);
      lcd.print("Action Done!");
      armOnce(); // Call the servo movement function
      delay(1000);
      return true;
    }
  }
  Serial.println("Action failed!");
  Serial.println(score);
  lcd.setCursor(2, 0);
  lcd.print("Action failed!");
  lcd.setCursor(2, 1);
  lcd.print(score);
  delay(1000);
  return false;

} else if (action == 2) { // Rotate It
```

```
    Serial.println("Rotate it!");
    lcd.setCursor(2, 0);
    lcd.print("Rotate it!");
    StartTime = millis();
    rotationDirection = 0;
    while (millis() - StartTime <= gameTime) {
      if (rotationDirection != 0) {
        score++;
        rotationDirection = 0;
        Serial.println("Action completed!");
        lcd.setCursor(2, 0);
        lcd.print("Action Done!");
        armOnce(); // Call the servo movement function
        delay(1000);
        return true;
      }
    }
    Serial.println("Action failed!");
    Serial.println(score);
    lcd.setCursor(2, 0);
    lcd.print("Action failed!");
    lcd.setCursor(2, 1);
    lcd.print(score);
    delay(1000);
    return false;
  }
  return false;
}

void handleEncoder() {
  clkState = digitalRead(CLK); //reads current CLK state
  if (clkState != clkLast && clkState == HIGH) { // If last and
current state of CLK are different, then pulse occurred
    if (digitalRead(DT) != clkState) {
      rotationDirection = -1; // If the DT state is different than
the CLK state then
    } else {
      rotationDirection = 1;
    }
  }
  clkLast = clkState;
}

void arm() {
  Serial.println("Rapid servo movement initiated...");
```

```
  // Perform rapid back-and-forth movement
  for (int i = 0; i < 10; i++) {  // Adjust the number of repetitions
as needed
    servoleft.write(140);
    servoright.write(140);
    delay(200);  // 200 ms delay; adjust for desired speed
    servoleft.write(0);
    servoright.write(0);
    delay(200);  // 200 ms delay; adjust for desired speed
  }

  Serial.println("Rapid servo movement completed.");
}

void armOnce () {
  Serial.println("Activating servos...");
    servoleft.write(140);
    servoright.write(140);
    delay(1250);
    servoleft.write(0);
    servoright.write(0);
  Serial.println("Servos completed movement.");
}


void turnRight() {
  Serial.println("Turning Right...");

  // Set the left motor to forward
  digitalWrite(motorLeft_InputOne, LOW);
  digitalWrite(motorLeft_InputTwo, HIGH);

  // Set the right motor to backward
  digitalWrite(motorRight_InputOne, HIGH);
  digitalWrite(motorRight_InputTwo, LOW);

  // Enable motors
  analogWrite(motorLeft_Enable, 255);
  analogWrite(motorRight_Enable, 255);

  // Fixed delay for the turn
  delay(5000); // Adjust the value as needed for the desired turn
duration
```

```
  // Stop motors
  stopMotors();
  Serial.println("Turn completed.");
}

void stopMotors() {
  Serial.println("Stopping Motors...");
  digitalWrite(motorLeft_InputOne, LOW);
  digitalWrite(motorLeft_InputTwo, LOW);
  digitalWrite(motorRight_InputOne, LOW);
  digitalWrite(motorRight_InputTwo, LOW);
  analogWrite(motorLeft_Enable, 0);
  analogWrite(motorRight_Enable, 0);
}
```