Team Name/Number: **Team 4**
Team Member Names (Last, First):
Gorji, Aveed
Bartz, Daniel
Zhu, Steven
Date of submission:
12/10/24

Moving Steven Air

Introduction:

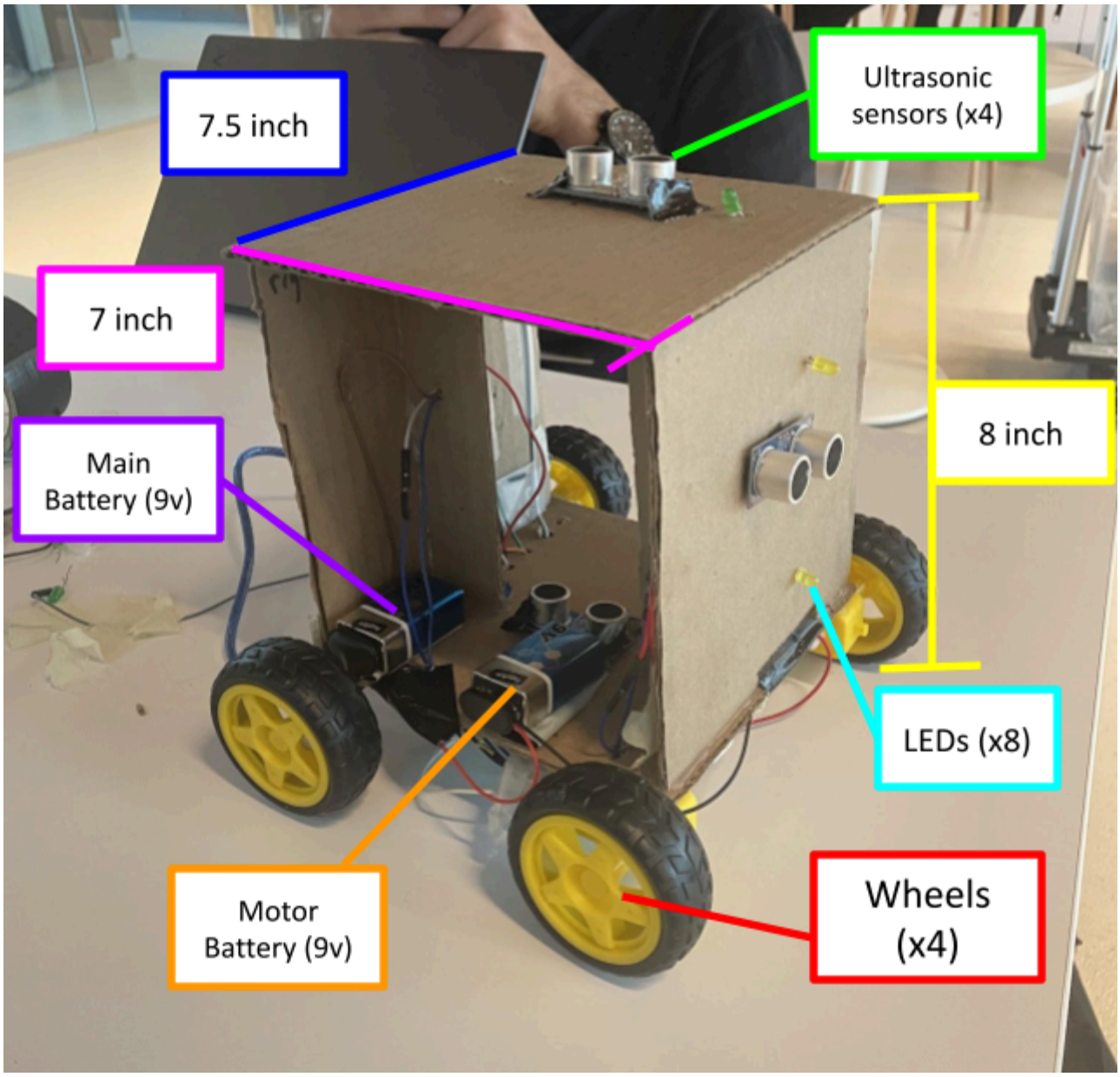
Simon Air® is a popular memory game enjoyed by millions of children worldwide. As the game progresses, it's supposed to get more challenging allowing for intense situations. Despite its enjoyment, the game is far too easy for the majority of children. Children need a game that offers them a real challenge. Therefore, improvements could be made to Simon Air® to make it even more intense and interesting to challenge-seeking individuals. Our solution: put the game on wheels. Putting the Simon Air® on wheels allows for even more intense situations as the game moves away from the player as they wave their hand in front of the motion sensor; in addition to memorizing the color patterns, this adds another level of difficulty. The Moving Steven Air is not only difficult to master, but the most advanced iteration of the Simon Air® ever created.

Project Criteria:

The most important criterion our robot needed to fulfill was movement. To do this, we placed our robot on wheels, which allowed us to implement higher difficulty into our Moving Steven Air. We purchased four wheels and two DC motors and attached them to the bottom of the robot, allowing it to move from side to side whenever one of the ultrasonic sensors is stimulated. In addition, the second most important criterion our robot needed to fulfill was to be intractable. Ultrasonic sensors allowed us to achieve interaction between the player and our robot. The ultrasonic sensors detect hand movement, signaling the player made an input to the game.

Final design details:

Photo of Final Design:



Electronic schematic:

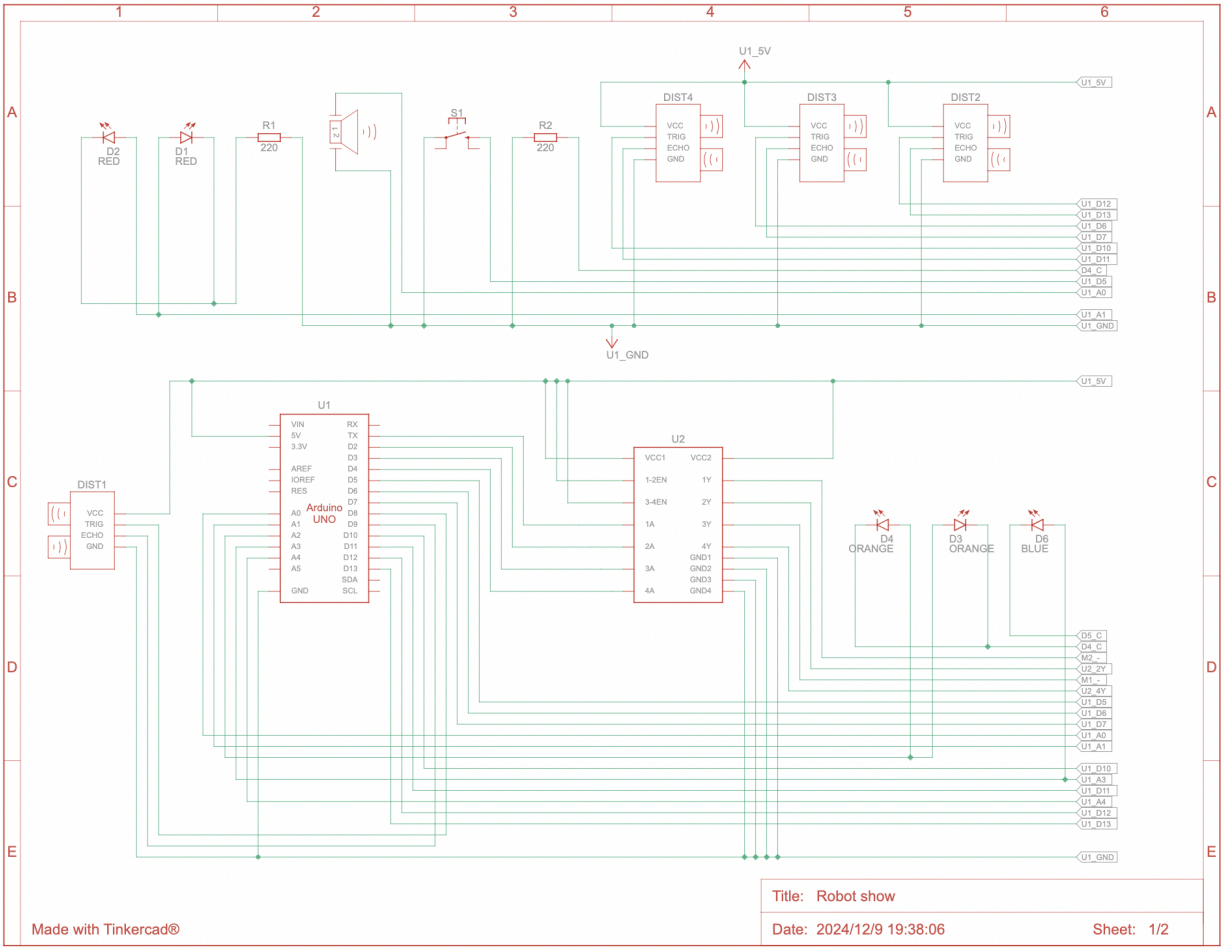
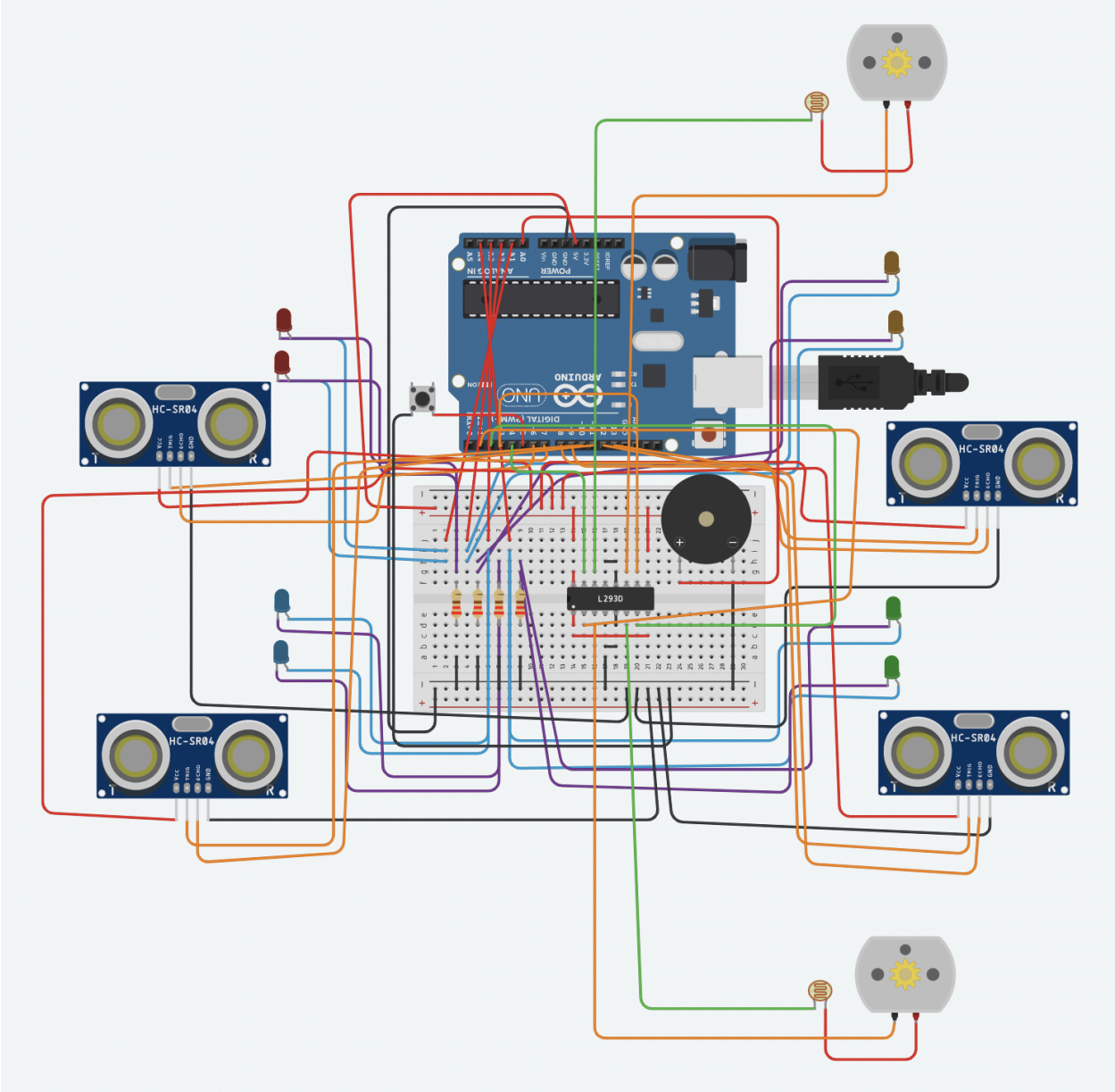
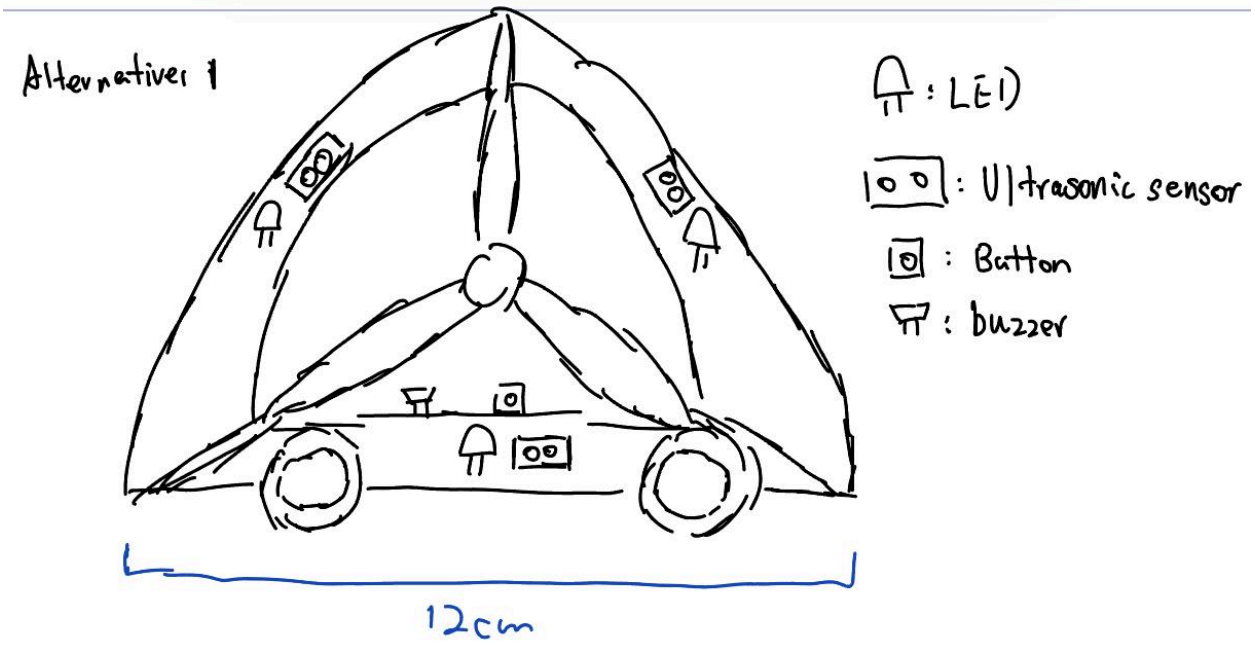


Figure A1: Electronic schematic diagram for Moving Steven Air

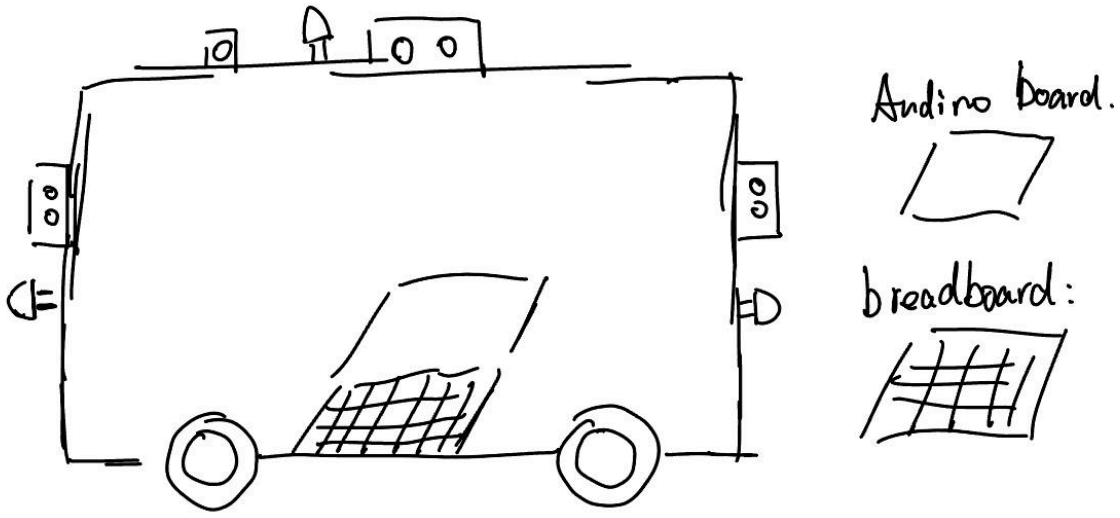
TinkerCAD fritzing diagram:


HC-SR04
HC-SR04
HC-SR04
HC-SR04
L293D
UNO
ARDUINO

Design alternatives:



Alternative 2:



Analysis:

The design Alternative 1 is the first prototype that we wanted to make. We drew this one because it looks like the real Simon Air® design most. However, since all three ultrasonic sensors are on the same plane, it is highly possible to find three ultrasonic sensors affect each other since our project is moving during the game. Design alternative 2 is similar to our final project on appearance. But it only has three LEDs and three sensors for users to memorize. It is too simple for a memorization game. Therefore, we chose our final design which has four LEDs and has them on the four sides of the box so that the sensors won't affect each other.

Analysis Matrix:

Criteria rating	**Cost**	**Viability**	**Impact**	**Team Preference**
Alternative 1	4	4	5	0
Alternative 2	5	7	6	0
Alternative 3 (Final Design)	7	9	8	10

Appendix:

Table A1: Bill of Materials for Moving Steven Air

Item	Quantity	Description	Cost per item
Microcontroller	1	Arduino Uno	N/A - ENGR 1 loan
Computer cable	1	USB	N/A - ENGR 1 loan
Battery snap	1	Arduino barrel to 9V connector	N/A - ENGR 1 loan
Breadboard	1	Small	N/A - ENGR 1 loan
Wheels	4	Role on the ground	$6.50
DC motor	2	drive the wheels forward/back	N/A - ENGR 1 loan
Ultrasonic sensors	4	sense distance to object	N/A - ENGR 1 loan
Red LED	2	indicate red side	N/A - ENGR 1 loan
Blue LED	2	indicate blue side	N/A - ENGR 1 loan
Yellow LED	2	indicate yellow side	N/A - ENGR 1 loan
Green LED	2	indicate green side	N/A - ENGR 1 loan
Resistors	14	proper input voltage	N/A - ENGR 1 loan
Buzzer	1	indicate detected user input	N/A - ENGR 1 loan
Button	1	initiate game	N/A - ENGR 1 loan
Wires	20	connect all parts	N/A - ENGR 1 loan
Board	1	contain all parts	N/A - ENGR 1 loan
			Sum of all parts - Total Cost: $6.50

Table A2: Team member project time log

Team Member	**Time spent on Moving Steven Air**	
	In-class (any lab periods attended)	Out-of-class (non-lab periods)
Aveed	6 hrs	6 hrs
Steven	6 hrs	6 hrs
Daniel	6 hrs	5.5 hrs

Commented computer code for Moving Steven Air:

```
// Define pins for Ultrasonic sensors, LEDs, Button, and Buzzer

const int trigPins[] = {6, 7, 8, 9}; // bottom left top right

const int echoPins[] = {10, 11, 12, 13}; // bottom left top
right

const int ledPins[] = {A1, A2, A3, A4}; // bottom/blue
yellow/left, green/top, red/right

const int buttonPin = 5;

const int buzzerPin = A0;

// Motor direction control pins

const int motorA = 2;  // IN1 - Motor A (Direction control)

const int motorB = 3;  // IN3 - Motor B (Direction control)

const int enablePin = A5;  // ENA and ENB - Speed control



const int numSensors = 4;

const int detectionThreshold = 5; // Distance threshold in cm

const int gameTimeout = 10000;      // 5 seconds timeout for
user input



bool gameRunning = false;

int gameState = 0; // 0 = Stopped, 1 = LED Sequence, 2 = User
Touch, 3 = Game Over

int ledSequence[5]; // Store the random LED sequence

int userSequence[5]; // Store the user's input sequence
```

```
int sequenceLength = 3; // Length of the random sequence (starts
with 3)

int currentStep = 0; // Current step in the user's input

unsigned long lastInteractionTime = 0; // Track time since last
user input



void setup() {

  Serial.begin(9600);



  // Set motor direction pins as outputs

  pinMode(motorA, OUTPUT);

  pinMode(motorB, OUTPUT);



  // Set the enable pin for speed control

  pinMode(enablePin, OUTPUT);



  // Set motor speed to maximum (255 for full speed)

  analogWrite(enablePin, 255);  // Full speed



  // Set pin modes

  for (int i = 0; i < numSensors; i++) {

    pinMode(trigPins[i], OUTPUT);

    pinMode(echoPins[i], INPUT);
```

```
    pinMode(ledPins[i], OUTPUT);
  }
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(buzzerPin, OUTPUT);

  // Ensure LEDs are off
  for (int i = 0; i < numSensors; i++) {
    digitalWrite(ledPins[i], LOW);
  }

  // Ensure buzzer is off
  noTone(buzzerPin);
}

void loop() {
  static bool lastButtonState = HIGH;
  bool currentButtonState = digitalRead(buttonPin);

  // Button press to start/stop game
  if (lastButtonState == HIGH && currentButtonState == LOW) {
    gameRunning = !gameRunning;
```

```
    if (gameRunning) {
      startGame();
    } else {
      endGame();
    }
  }
  lastButtonState = currentButtonState;


  // Game logic
  if (gameRunning) {
    switch (gameState) {
      case 1: // LED Sequence Phase
        playLedSequence();
        break;


      case 2: // User Touch Phase
        checkUserInput();
        break;


      case 3: // Game Over
        handleGameOver();
        break;
```

```
    }
  }
}


void startGame() {
  Serial.println("Game Started!");
  tone(buzzerPin, 1000, 1000); // Play tone at 1000 Hz for 1
second


  // Generate random LED sequence (ensure no duplicate LEDs)
  for (int i = 0; i < sequenceLength; i++) {
    int randomLED;
    bool unique = false;
    while (!unique) {
      randomLED = random(0, numSensors);
      unique = true;
      // Ensure no duplicates
      for (int j = 0; j < i; j++) {
        if (ledSequence[j] == randomLED) {
          unique = false;
          break;
        }
      }
```

```
    }
    ledSequence[i] = randomLED;
  }

  gameState = 1; // Move to LED Sequence Phase
}

void playLedSequence() {
  Serial.println("Playing LED Sequence...");
  for (int i = 0; i < sequenceLength; i++) {
    int led = ledSequence[i];
    digitalWrite(ledPins[led], HIGH);
    tone(buzzerPin, 440, 500); // Play tone at 440 Hz for 500 ms
    delay(500);
    digitalWrite(ledPins[led], LOW);
    delay(500);
  }

  // Signal end of sequence
  tone(buzzerPin, 800, 1000); // Long beep at 800 Hz for 1
second
  delay(1000);
```

```
  gameState = 2; // Move to User Touch Phase
  currentStep = 0;
  lastInteractionTime = millis(); // Start timeout timer
}

void checkUserInput() {
  bool userTouched = false;
  for (int i = 0; i < numSensors; i++) {
    int distance = getDistance(trigPins[i], echoPins[i]);
    if (distance > 0 && distance < detectionThreshold) {
      userTouched = true;

      // Light up LED for feedback
      digitalWrite(ledPins[i], HIGH);
      tone(buzzerPin, 500, 500); // Play tone at 500 Hz for 500
ms
      delay(1000); // Hold for 1 second
      digitalWrite(ledPins[i], LOW);

      // Check if the touched LED matches the sequence
      if (i == ledSequence[currentStep]) {
        currentStep++;
        Serial.print("Correct! Step: ");
```

```
        Serial.println(currentStep);


        if (currentStep == sequenceLength) {
          // User completed the sequence
          indicateCorrect();


          if (sequenceLength == 3) {
            // First round complete, increase sequence length
for second round
            sequenceLength = 5;
            startGame(); // Restart game with longer sequence
          } else {
            // Game fully completed
            Serial.println("You completed all rounds!");
            tone(buzzerPin, 1500, 2000); // Victory tone
            delay(2000);
            gameRunning = false; // End game
          }
          return;
        }
      } else {
        // Incorrect input
        handleGameOver();
```

```
        return;
      }

      lastInteractionTime = millis(); // Reset timeout timer
    }
  }

  if (!userTouched && millis() - lastInteractionTime > gameTimeout) {
    // Timeout: No user input
    handleGameOver();
  }
}

void indicateCorrect() {
  Serial.println("Round Passed!");
  for (int i = 0; i < 3; i++) {
    tone(buzzerPin, 1000 + (i * 200), 200); // Increasing pitch tones
    delay(300);
  }
}
```

```
void handleGameOver() {
  Serial.println("Game Over!");
  // All LEDs light up as error signal
  for (int i = 0; i < numSensors; i++) {
    digitalWrite(ledPins[i], HIGH);
  }
  tone(buzzerPin, 200, 500); // Error beep at 200 Hz for 500 ms
  delay(500);
  tone(buzzerPin, 300, 500); // Error beep at 300 Hz for 500 ms
  delay(500);

  delay(5000); // Keep LEDs on and buzzer for 5 seconds

  for (int i = 0; i < numSensors; i++) {
    digitalWrite(ledPins[i], LOW); // Turn off LEDs
  }

  gameRunning = false; // End game
  gameState = 0; // Reset state
}

void endGame() {
```

```
  Serial.println("Game Stopped!");
  for (int i = 0; i < numSensors; i++) {
    digitalWrite(ledPins[i], LOW);
  }
  noTone(buzzerPin);
  gameRunning = false;
  gameState = 0;
}


int getDistance(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);


  long duration = pulseIn(echoPin, HIGH);
  int distance = duration * 0.034 / 2; // Convert to cm
  Serial.println(distance);
  return distance;
}
```

```
void moveForward() {
  digitalWrite(motorA, HIGH);  // Motor A forward
  digitalWrite(motorB, HIGH);  // Motor B forward
}

void moveBackward() {
  Serial.println("Moving Backward");
  digitalWrite(motorA, LOW);  // Motor A backward
  digitalWrite(motorB, LOW);  // Motor B backward
}
```

Conclusion:

The Moving Steven Air takes the beloved classic memory game to a whole new level of excitement and difficulty. By introducing mobility, players are challenged not only to memorize color patterns but also to keep up with the game as it moves. This innovative twist transforms the experience into a dynamic and thrilling test of coordination and focus, appealing to challenge-seeking individuals of all ages. The Moving Steven Air is the perfect evolution of the original, offering a more intense and engaging experience that's sure to captivate players and push their skills to the limit.