```
#include <AccelStepper.h>
#include <EEPROM.h>

// Motor Pins
const int stepPin = D6;
const int dirPin = D5;

// Control Pins
const int switchPin = D7;
int switchVal;

// Sensor Pins
const int aseatPin = D0;
const int dseatPin = D8;
const int moisturePin = D1;

// Movement Settings
const int TOTAL_MOVE_STEPS = 20;
const unsigned long CYCLE_DELAY = 10000;    // 10 seconds between movements
const unsigned long MOVE_DURATION = 10000;  // 10 seconds to complete movement
const unsigned long PAUSE_TIMEOUT = 10000;  // 10 seconds pause between movements

// Speed Settings
const float TOP_SPEED = 3.0;
const float SPEED_RAMP = 1.0;

// Safety Thresholds
const int SEAT_PRESSURE_LEVEL = 1000;
const int SEAT_WET_THRESHOLD = 2000;
const int SEAT_DRY_THRESHOLD = 3000;  // updated dry threshold

// Motor setup
AccelStepper motor(1, stepPin, dirPin);

// Movement tracking
unsigned long lastCycleTime = 0;
unsigned long moveStartTime = 0;
unsigned long pauseStartTime = 0;
bool isMoving = false;
bool isPaused = false;
bool isMoistureLock = false;
bool firstCycle = true;
int lastCompletedPosition = 0;

// Function declarations
long checkSeat(int samples = 100);
long checkMoisture(int samples = 100);
void stopMovement();
void startMovement();
void resumeMovement();
```

```
void returnToStartPosition();

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

  pinMode(stepPin, OUTPUT);
  pinMode(dirPin, OUTPUT);
  motor.setMaxSpeed(TOP_SPEED);
  motor.setAcceleration(SPEED_RAMP);

  pinMode(switchPin, INPUT_PULLUP);
  pinMode(dseatPin, OUTPUT);
  pinMode(moisturePin, INPUT);
  Serial.println("Morning Glory Blooms");
}

long checkSeat(int samples) {
  long total = 0;
  for (int i = 0; i < samples; i++) {
    digitalWrite(dseatPin, HIGH);
    int high_read = analogRead(aseatPin);
    delayMicroseconds(100);
    digitalWrite(dseatPin, LOW);
    int low_read = analogRead(aseatPin);
    total += (high_read - low_read);
    delay(2);
  }
  return total / samples;
}

long checkMoisture(int samples) {
  long total = 0;
  for (int i = 0; i < samples; i++) {
    total += analogRead(moisturePin);
    delay(2);
  }
  return total / samples;
}

void stopMovement() {
  lastCompletedPosition = motor.currentPosition();
  motor.stop();
  isMoving = false;
  isPaused = true;
  pauseStartTime = millis();
  Serial.print("Movement Paused at Position: ");
  Serial.println(lastCompletedPosition);
}

void startMovement() {
```

```
  motor.setCurrentPosition(0);
  lastCompletedPosition = 0;
  motor.moveTo(TOTAL_MOVE_STEPS);
  moveStartTime = millis();
  isMoving = true;
  isPaused = false;
  Serial.println("Movement Started from 0");
}

void resumeMovement() {
  motor.setCurrentPosition(lastCompletedPosition);
  motor.moveTo(TOTAL_MOVE_STEPS);
  moveStartTime = millis();
  isMoving = true;
  isPaused = false;
  Serial.print("Resuming Movement from Position: ");
  Serial.println(lastCompletedPosition);
}

void returnToStartPosition() {
  Serial.println("ITS RAINING!!!! PROTECT THE CHAIR!!!!!!!!!!");
  motor.setCurrentPosition(motor.currentPosition());
  motor.moveTo(0);
  while (motor.currentPosition() > 0) {
    motor.run();
    Serial.print("Returning... Position: ");
    Serial.println(motor.currentPosition());
    delay(5);  // Small delay for readability
  }
  motor.setCurrentPosition(0);      // Ensure internal position is set to 0
  lastCompletedPosition = 0;        // ☑ Update the logical position tracker
  Serial.println("CHAIR FULLY CLOSED. WAITING FOR RAIN SEAT TO DRY/ RAIN TO
STOP...");
}

void loop() {
  int currentSwitchState = digitalRead(switchPin);
  if (currentSwitchState == HIGH) {
    if (isMoving) {
      stopMovement();
    }
    Serial.println("Morning Glory DIES");
    firstCycle = true;
    return;
  }

  // RAIN PROTECTION
  long moistureCheck = checkMoisture();
  if (moistureCheck < SEAT_WET_THRESHOLD) {
    if (!isMoistureLock) {
```

```
      stopMovement();
      returnToStartPosition();
      isMoistureLock = true;
    }
    return;
  } else if (isMoistureLock && moistureCheck > SEAT_DRY_THRESHOLD) {
    Serial.println("SEAT IS NOW DRY!!! BLOSSOM AGAIN");
    isMoistureLock = false;
  }

  // Seat pressure check
  long seatCheck = checkSeat();
  if (seatCheck > SEAT_PRESSURE_LEVEL) {
    if (!isPaused) {
      stopMovement();
    }
    return;
  }

  // Handle pause timeout
  unsigned long currentTime = millis();
  if (isPaused) {
    if (currentTime - pauseStartTime >= PAUSE_TIMEOUT) {
      resumeMovement();  // resume from last position
    } else {
      return;
    }
  }

  // Start new movement cycle
  if (!isMoving) {
    if (firstCycle || (currentTime - lastCycleTime >= CYCLE_DELAY)) {
      startMovement();
      firstCycle = false;
    }
  } else {
    motor.run();
    Serial.print("Current Motor Position: ");
    Serial.println(motor.currentPosition());

    // Movement finished or timed out
    if (motor.currentPosition() >= TOTAL_MOVE_STEPS ||
        (currentTime - moveStartTime >= MOVE_DURATION)) {
      motor.stop();
      isMoving = false;
      lastCycleTime = currentTime;
      Serial.println("Movement cycle completed. Waiting before next cycle...");
    }
  }
}
```