

Her er en C++-pseudokode baseret på det medfølgende flowchart til styring af et solpanelsporingssystem. Koden bruger en servomotor til venstre-højre rotation og en stepmotor til op-ned rotation, idet der antages analoge værdier fra en LDR-sensor for at detektere lysretningen.

| #include <wiringPi.h> #include <iostream>  // Constants for Servo and Stepper Motor Control const int SERVO\_PIN = 1; // GPIO pin for servo (left-right) const int STEPPER\_PIN\_1 = 2; // GPIO pins for stepper (up-down) const int STEPPER\_PIN\_2 = 3; const int STEPPER\_PIN\_3 = 4; const int STEPPER\_PIN\_4 = 5;  // Constants for Light Sensor const int LDR\_PIN = 0; // Analog pin for LDR (assume an ADC module)  // Function to initialize motors void setupMotors() {  wiringPiSetup();  pinMode(SERVO\_PIN, PWM\_OUTPUT);  pinMode(STEPPER\_PIN\_1, OUTPUT);  pinMode(STEPPER\_PIN\_2, OUTPUT);  pinMode(STEPPER\_PIN\_3, OUTPUT);  pinMode(STEPPER\_PIN\_4, OUTPUT); }  // Function to read analog value from LDR int readLDR() {  // Replace with actual ADC reading code  int analogValue = analogRead(LDR\_PIN);  return analogValue; }  // Function to calculate average LDR value int calculateAverage(int readings[], int size) {  int sum = 0;  for (int i = 0; i < size; i++) {  sum += readings[i];  }  return sum / size; }  // Function to control Servo Motor for left-right rotation void rotateServoLeftRight(int direction) {  if (direction > 0) {  std::cout << "Rotating servo right" << std::endl;  pwmWrite(SERVO\_PIN, 1000); // Example PWM value for right  } else if (direction < 0) {  std::cout << "Rotating servo left" << std::endl;  pwmWrite(SERVO\_PIN, 2000); // Example PWM value for left  } else {  std::cout << "Stopping servo" << std::endl;  pwmWrite(SERVO\_PIN, 1500); // Center position (stop)  } }  // Function to control Stepper Motor for up-down rotation void rotateStepperUpDown(int direction) {  if (direction > 0) {  std::cout << "Rotating stepper up" << std::endl;  // Implement step sequence for upward rotation  } else if (direction < 0) {  std::cout << "Rotating stepper down" << std::endl;  // Implement step sequence for downward rotation  } else {  std::cout << "Stopping stepper" << std::endl;  digitalWrite(STEPPER\_PIN\_1, LOW);  digitalWrite(STEPPER\_PIN\_2, LOW);  digitalWrite(STEPPER\_PIN\_3, LOW);  digitalWrite(STEPPER\_PIN\_4, LOW);  } }  int main() {  setupMotors();   int ldrReadings[4]; // Array to store LDR readings (for multiple sensors if needed)   while (true) {  // Step 1: Read analog value from LDR sensor  for (int i = 0; i < 4; i++) {  ldrReadings[i] = readLDR();  }   // Step 2: Calculate average value  int avgLDR = calculateAverage(ldrReadings, 4);   // Step 3: Calculate differences (based on assumed layout of LDR sensors)  int diffVert = ldrReadings[0] - ldrReadings[2];  int diffHoriz = ldrReadings[1] - ldrReadings[3];   // Step 4: Control motors based on difference thresholds  if (abs(diffHoriz) <= 10) {  rotateServoLeftRight(0); // Stop left-right motor  } else if (diffHoriz > 0) {  rotateServoLeftRight(1); // Rotate right  } else {  rotateServoLeftRight(-1); // Rotate left  }   if (abs(diffVert) <= 10) {  rotateStepperUpDown(0); // Stop up-down motor  } else if (diffVert > 0) {  rotateStepperUpDown(1); // Rotate up  } else {  rotateStepperUpDown(-1); // Rotate down  }   delay(1000); // Wait for a second before next read  }   return 0; } |
| --- |

**Opsætning af motorer:** Initialiserer GPIO-ben til servo (PWM) og stepmotor (digital udgang).

**Læs LDR-værdier:** Læser analoge værdier fra LDR-sensorer, simulerer en 4-sensor-opsætning til retningsbestemt sensing.

**Gennemsnitsberegning:** Beregner gennemsnittet af aflæsninger, selvom dette kan være mere nyttigt i en multi-sensor opsætning.

**Beregn forskelle:** Beregner forskelle for vandrette (diffHoriz) og vertikale (diffVert) retninger for at bestemme de nødvendige justeringer.

**Kontrolmotorer:** Bruger tærskler til at bestemme, om servo- og stepmotorerne skal roteres eller stoppes baseret på forskellene.

Denne kode simulerer logikken vist i rutediagrammet ved hjælp af wiringPi til GPIO-styring på Raspberry Pi.