

Project Name: Automatic Sun tracking solar Panel

Project Description:

This project aims to develop a dual-axis solar tracking system that maximizes solar panel efficiency by aligning it toward the Sun based on pre-calculated astronomical data, rather than using physical light sensors like LDRs.

The system uses a microcontroller (Arduino/ESP32) connected to a Real-Time Clock (RTC) module to get the current date and time. Based on the user's fixed geographical location (latitude and longitude), the microcontroller calculates the Sun's azimuth and elevation angles using trigonometric and solar position formulas.

These angles are then translated into commands for two stepper motors that adjust the panel's horizontal and vertical angles accordingly. This approach ensures accurate sun tracking regardless of weather conditions or obstacles, unlike LDR-based systems which can be misled by clouds, shadows, or reflections.

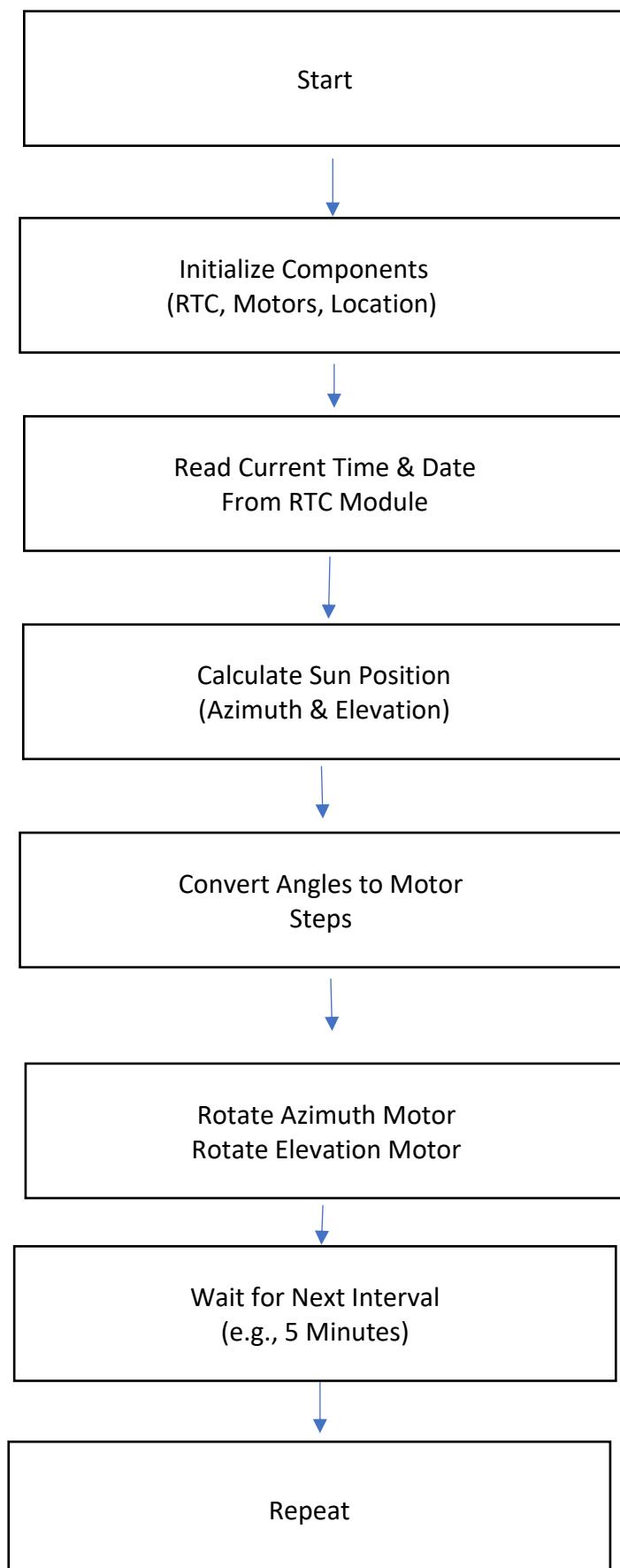
Features:

- No light sensors (LDR) needed
- Uses Solar Position Algorithm (SPA) based on time and location
- Accurate dual-axis control using stepper motors
- RTC module ensures real-time tracking
- Optional GPS support for dynamic positioning
- Energy-efficient and weather-resistant

Algorithm

1. Get current date and time from RTC
2. Use latitude and longitude (hardcoded)
3. Calculate Sun's azimuth and elevation angles
4. Convert those angles to motor positions
5. Move motors to track the Sun
6. Repeat periodically (e.g., every 5 minutes)

Flow Chart:

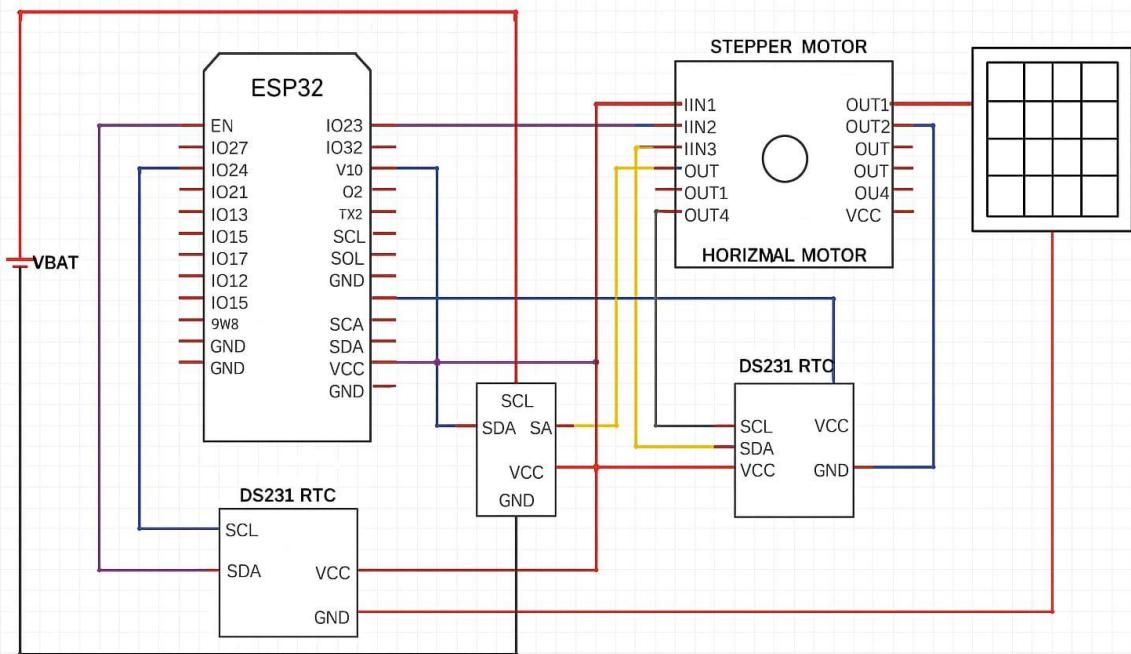


Components Required:

Component
Arduino Uno
RTC Module
Stepper
Motor Driver (L298N or ULN2003)
Solar Panel
GPS Module (optional)
Power Source
Jumper wires, breadboard, frame, etc.

Approx Expenditure:

Component	Quantity	Approx Price (BDT)
Arduino Uno	1	2100
DS3231 RTC Module	1	360
28BYJ-48 Stepper Motors + ULN2003 Driver	1 set	400
Small Solar Panel (10W–20W)	2	200
Jumper Wires, Breadboard	1 set	300
Power Adapter / USB Bank	1	400
Mounting Parts (brackets, screws, plastic/wood base)	1 set	500



Sun-Tracking Solar Panel

```
#include <iostream>
#include <cmath>
#include <cstdlib>
#include <ctime>
#include <thread>
#include <chrono>

using namespace std;

// Mocking LDR readings using random values for simulation

int analogRead(int pin) {
    return rand() % 1024; // Simulate analog input (0-1023)
}

// Simulate servo position (virtual)

class Servo {
private:
    int pos;

public:
    Servo() : pos(90) {}

    void attach(int pin) {
        cout << "Servo attached to pin " << pin << endl;
    }

    void write(int position) {
```

```
pos = position;

cout << "Servo moved to position: " << pos << "°" << endl;

}

int getPosition() const {

    return pos;

}

};

int main() {

    srand(static_cast<unsigned int>(time(0))); // Seed random

    const int ldrLeft = 0;

    const int ldrRight = 1;

    int pos = 90;

    Servo trackerServo;

    trackerServo.attach(9);

    trackerServo.write(pos);

    while (true) {

        int leftVal = analogRead(ldrLeft);

        int rightVal = analogRead(ldrRight);
```

```
int diff = leftVal - rightVal;

cout << "Left: " << leftVal << " | Right: " << rightVal
     << " | Pos: " << pos << endl;

if (abs(diff) > 30) {
    if (diff > 0 && pos < 180) pos++;
    else if (diff < 0 && pos > 0) pos--;
    trackerServo.write(pos);
}

this_thread::sleep_for(chrono::milliseconds(500));      // Delay
simulation

}

return 0;
}
```