**SOLAR PANEL COVERING SYSTEM IN EV VEHICLES**

**ABSTRACT**

This project proposes the design and implementation of a solar panel cover control system for solar EV (Electric Vehicle) cars, aimed at optimizing energy harvesting efficiency and protecting the solar panels during non-charging periods. The system utilizes a motor-driven mechanism controlled wirelessly via Wi-Fi transmission and reception, enabling seamless operation and integration with existing vehicle control systems. The Solar EV Car Solar Panel Cover Control System comprises several key components, including a motorized cover mechanism, Wi-Fi transmitter and receiver modules, microcontroller unit, and power supply system. The motorized cover mechanism is responsible for deploying and retracting the protective sheet over the solar panels, safeguarding them from environmental factors such as debris, rain, or sunlight when not in use. The Wi-Fi transmitter and receiver modules facilitate wireless communication between the control unit and the motorized mechanism, allowing for remote operation and monitoring via a smart phone app or a central vehicle control interface. This wireless functionality enhances user convenience and enables automatic or manual control of the solar panel cover system. The microcontroller unit serves as the central processing unit, receiving commands from the user interface or vehicle control system and translating them into motor control signals. It also manages safety features, such as position sensors to prevent overextension or damage to the cover mechanism.

**Components:**

* **Motorized Cover Mechanism**
* **Wi-Fi Transmitter and Receiver Modules**
* **Microcontroller Unit**
* **Position Sensors**

**Code:**

**#include <ESP8266WiFi.h>**

**#include <WiFiClient.h>**

**#include <ESP8266WebServer.h>**

**// Wi-Fi network credentials**

**const char\* ssid = "YOUR\_WIFI\_SSID";**

**const char\* password = "YOUR\_WIFI\_PASSWORD";**

**// Create instance of ESP8266WebServer**

**ESP8266WebServer server(80);**

**// Define motor control pins**

**const int motorPin1 = D1; // Motor control pin 1**

**const int motorPin2 = D2; // Motor control pin 2**

**const int enablePin = D3; // Motor enable pin**

**// Variables to store motor direction**

**bool motorForward = false;**

**bool motorBackward = false;**

**void setup() {**

**// Initialize serial communication**

**Serial.begin(9600);**

**// Initialize motor control pins as output**

**pinMode(motorPin1, OUTPUT);**

**pinMode(motorPin2, OUTPUT);**

**pinMode(enablePin, OUTPUT);**

**// Disable motor initially**

**digitalWrite(enablePin, LOW);**

**// Connect to Wi-Fi network**

**WiFi.begin(ssid, password);**

**Serial.println("Connecting to Wi-Fi...");**

**while (WiFi.status() != WL\_CONNECTED) {**

**delay(1000);**

**Serial.print(".");**

**}**

**Serial.println("\nConnected to Wi-Fi");**

**// Set up HTTP routes**

**server.on("/", handleRoot);**

**server.on("/cover", handleCover);**

**server.begin();**

**Serial.println("HTTP server started");**

**}**

**void loop() {**

**server.handleClient(); // Handle incoming HTTP requests**

**// Add any additional code here if needed**

**}**

**// Handler for the root URL**

**void handleRoot() {**

**server.send(200, "text/plain", "Solar EV Car Solar Panel Cover Control System");**

**}**

**// Handler for controlling the cover mechanism**

**void handleCover() {**

**// Check if the HTTP GET request has a parameter named "action"**

**if (server.hasArg("action")) {**

**String action = server.arg("action");**

**if (action == "deploy") {**

**// Deploy cover**

**deployCover();**

**server.send(200, "text/plain", "Cover deployed");**

**} else if (action == "retract") {**

**// Retract cover**

**retractCover();**

**server.send(200, "text/plain", "Cover retracted");**

**} else {**

**// Invalid action**

**server.send(400, "text/plain", "Invalid action");**

**}**

**} else {**

**// No action parameter provided**

**server.send(400, "text/plain", "Action parameter missing");**

**}**

**}**

**// Function to deploy the cover**

**void deployCover() {**

**digitalWrite(enablePin, HIGH); // Enable motor**

**digitalWrite(motorPin1, HIGH); // Set motor direction**

**digitalWrite(motorPin2, LOW);**

**delay(1000); // Adjust delay as needed**

**digitalWrite(enablePin, LOW); // Disable motor**

**}**

**// Function to retract the cover**

**void retractCover() {**

**digitalWrite(enablePin, HIGH); // Enable motor**

**digitalWrite(motorPin1, LOW); // Set motor direction**

**digitalWrite(motorPin2, HIGH);**

**delay(1000); // Adjust delay as needed**

**digitalWrite(enablePin, LOW); // Disable motor**

**}**