**SMART WINDOWS MOTION CONTROL**

**Abstract:**

To control the motor using a receiver and transmitter setup with Arduino, we'll use two Arduino boards—one for the transmitter and one for the receiver. The transmitter Arduino will send control signals wirelessly to the receiver Arduino, which will then control the motor based on those signals. We'll use RF (Radio Frequency) modules for wireless communication between the two Arduinos.

**USED CODE**

**#include <WiFi.h>**

**const char\* ssid = "YOUR\_WIFI\_SSID"; // Enter your WiFi SSID**

**const char\* password = "YOUR\_WIFI\_PASSWORD"; // Enter your WiFi Password**

**WiFiServer server(80);**

**void setup() {**

**Serial.begin(115200);**

**delay(10);**

**// Connect to WiFi network**

**Serial.println();**

**Serial.println();**

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

**Serial.println(ssid);**

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

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

**delay(500);**

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

**}**

**Serial.println("");**

**Serial.println("WiFi connected.");**

**// Start the server**

**server.begin();**

**Serial.println("Server started");**

**}**

**void loop() {**

**WiFiClient client = server.available(); // Listen for incoming clients**

**if (client) { // If a new client connects,**

**Serial.println("New Client."); // print a message out in the serial port**

**String currentLine = ""; // Make a String to hold incoming data from the client**

**while (client.connected()) { // Loop while the client's connected**

**if (client.available()) { // If there's bytes to read from the client,**

**char c = client.read(); // read a byte, then**

**Serial.write(c); // print it out the serial monitor**

**if (c == '\n') { // if the byte is a newline character**

**// if the current line is blank, you got two newline characters in a row**

**// that's the end of the client HTTP request, so send a response**

**if (currentLine.length() == 0) {**

**// HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)**

**// and a content-type so the client knows what's coming, then a blank line**

**client.println("HTTP/1.1 200 OK");**

**client.println("Content-type:text/html");**

**client.println();**

**// the content of the HTTP response follows the header:**

**client.println("<html><body><h1>Hello from ESP32!</h1></body></html>");**

**// The HTTP response ends with another blank line**

**client.println();**

**// Break out of the while loop**

**break;**

**} else { // if you got a newline, then clear currentLine**

**currentLine = "";**

**}**

**} else if (c != '\r') { // if you got anything else but a carriage return character,**

**currentLine += c; // add it to the end of the currentLine**

**}**

**// Check if the client's request is to open or close the window**

**if (currentLine.endsWith("open")) {**

**// Code to open the window**

**Serial.println("Opening the window...");**

**} else if (currentLine.endsWith("close")) {**

**// Code to close the window**

**Serial.println("Closing the window...");**

**}**

**}**

**}**

**// Clear the header variable**

**currentLine = "";**

**// Close the connection**

**client.stop();**

**Serial.println("Client disconnected.");**

**}**

**}**

**EXPLANATION**

This code creates a server on a NodeMCU ESP32, connecting it to a WiFi network. It listens for incoming HTTP requests and responds with a simple HTML page. When a request is received, it checks if it's for opening or closing the window. If it's an open request, it prints a message indicating the window is opening. If it's a close request, it prints a message indicating the window is closing. The server runs continuously, waiting for client connections. It handles each client request individually and closes the connection after responding

**Note:**

Replace **"YOUR\_WIFI\_SSID"** and **"YOUR\_WIFI\_PASSWORD"** with your actual WiFi credentials. This code listens for HTTP requests and responds with a simple HTML page. You can send commands to open or close the window by making a GET request to the ESP32's IP address with **/open** or **/close** appended to the URL, respectively. For example, **http://192.168.1.100/open** would open the window.