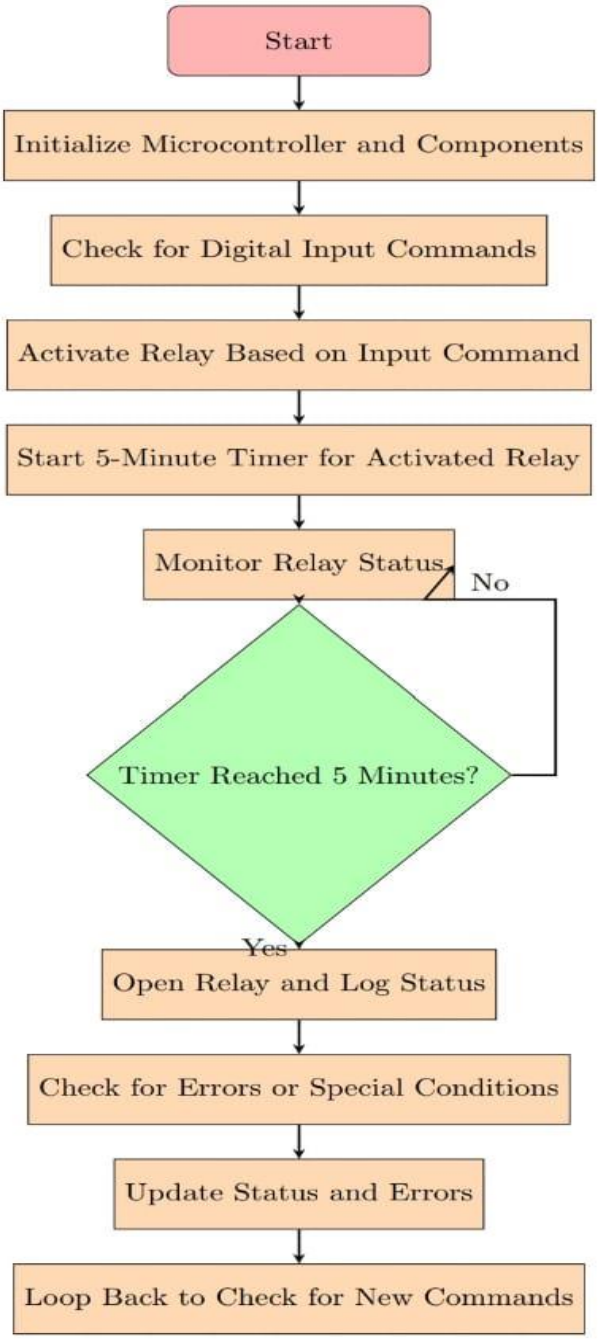
Flow Chart


Start
Initialize Microcontroller and Components
Check for Digital Input Commands
Activate Relay Based on Input Command
Start 5-Minute Timer for Activated Relay
Monitor Relay Status
No
Timer Reached 5 Minutes?
Yes
Open Relay and Log Status
Check for Errors or Special Conditions
Update Status and Errors
Loop Back to Check for New Commands

Flowchart Description:

1. Start

2. Initialize Microcontroller and Components

Setup GPIO pins for relay control and digital inputs

3. Check for Digital Input Commands

Read the digital input states for relay activation

Check which relay input (Button/Command) is pressed

4. Activate Relay Based on Input Command

If command is received, close the corresponding relay

Log the relay state as Closed

5. Start Timer for Active Relay

Start a 5-minute timer for the activated relay

6. Monitor Relay Status Continuously

Check for special conditions (welded relay or constantly open)

7. Check Timer

If the timer reaches 5 minutes

Open the relay

Log relay state as Open

8. Check for Errors

-If a relay fails to open or remains constantly open, log the error condition

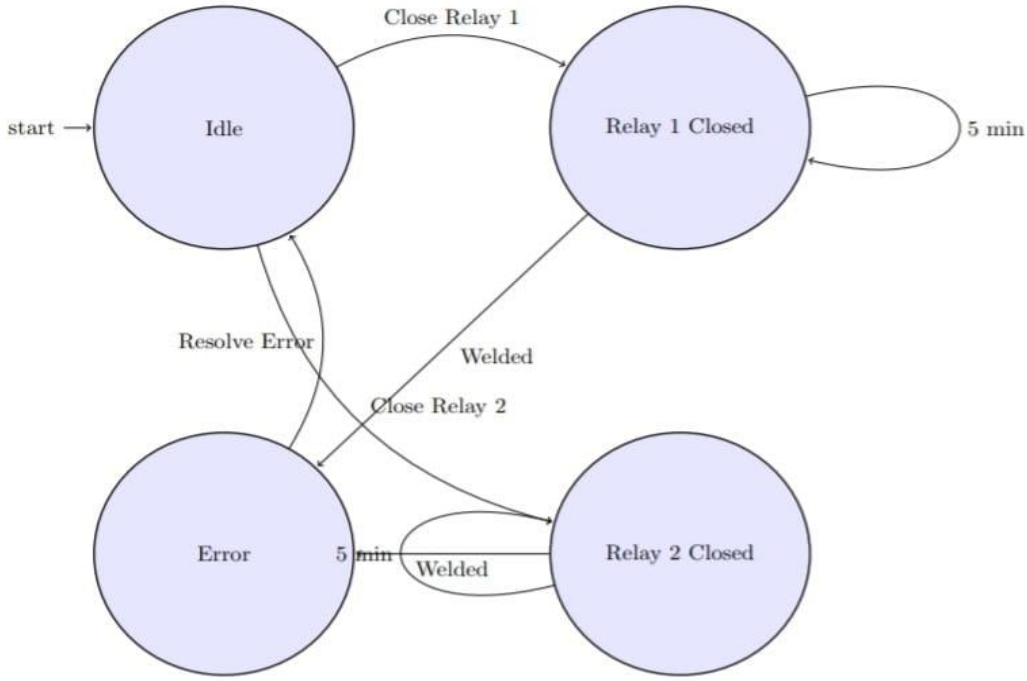
9. Update Relay Status and Errors

Update the relay status to the system or user interface

10. Loop Back to Check for New Commands

End

State Diagram



State Diagram Description:

1. States:

Idle: The initial state when no relay is activated.

Relay1 Closed: State when Relay 1 is closed.

Relay2 Closed: State when Relay 2 is closed.

Error: State indicating a fault, such as a welded relay.

2. Transitions:

From Idle to Relay1 Closed: Triggered by the command to close Relay 1.

From Idle to Relay2 Closed: Triggered by the command to close Relay 2.

From Relay1 Closed to Idle: Occurs after 5 minutes of being closed.

From Relay2 Closed to Idle: Occurs after 5 minutes of being closed.

From Relay1 Closed to Error: Triggered if Relay 1 is detected as welded.

From Relay2 Closed to Error: Triggered if Relay 2 is detected as welded.

From Error to Idle: Transition back to idle state when the error is resolved.

Code Documentation

Designing an automotive motor control system using multiple relays.

The system will control multiple DC motors in an automotive context, using relays to switch the motors on and off based on digital input commands. Each motor will be automatically turned off after a specified duration, with continuous monitoring of each relay's status.

Components Required

1. Microcontroller:

   Arduino Uno.

2. Relay Modules:

   -4-Channel Relay Module (rated for at least 10A at 250V AC or 30V DC).

3. Push Buttons:

   Momentary push button switches (one for each motor).

4. Resistors:

   10kΩ resistors for pull-down configuration on the buttons.

5. LEDs:

   - One for each motor to indicate the status (optional).

6. Diodes:

   1N4001 or similar for flyback protection across the relay coils.

7. Transistors:

   NPN transistors (e.g., 2N2222) for controlling the relays.

8. Power Supply:

   A suitable automotive power supply (typically 12V from the vehicle battery).

9. Fuses:

   Appropriate fuses to protect against overcurrent for each motor.

10. Connector Blocks:

    For reliable connections to the automotive wiring harness.

Component Ratings

- *Relay*: Should be rated for at least 10A/250V AC or 30V DC, depending on the motor's current requirements.

- *DC Motor*: Rated based on the application (e.g., 12V, 5A for a window motor).

4. Wiring Diagram

For 4 motors:

1. Connect the Microcontroller to the Relay Module:

Arduino Pin 2to Relay 1 IN pin

Arduino Pin 3 to Relay 2 IN pin

Arduino Pin 4 to Relay 3 IN pin

Arduino Pin 5 to Relay 4 IN pin

Relay VCC to Microcontroller 5V (ensure it matches relay requirements)

Relay GND to Microcontroller GND

2. Connect the Push Buttons:

One terminal of each push button (for each relay) to an Arduino digital pin(e.g., Pins 6, 7, 8, 9 for Relay 1 to 4).

Connect the other terminal of each button to GND.

Connect a 10kΩ resistor between each button pin and GND (pull-down configuration).

3. Connect the Motors:

Connect one terminal of each DC motor to the NO (Normally Open) terminal of the corresponding relay.

Connect the other terminal of each motor to the COM (Common) terminal of the relay, with the common terminal connected to 12V from the automotive battery.

Ground each motor appropriately to the battery negative.

4. Optional: Connect LEDs for motor status indication:

Connect the anodes of the LEDs to separate digital pins (e.g., Pins 10, 11, 12, 13 for Relay 1 to 4) and the cathodes to GND through a resistor (220Ω).

Code for Arduino

Here's a code to control multiple motors based on button inputs, automatically turning off after a specified time,

```
#define RELAY_COUNT 4          // Number of relays
#define RELAY_PINS {2, 3, 4, 5} // Relay control pins
#define BUTTON_PINS {6, 7, 8, 9} // Button input pins
#define LED_PINS {10, 11, 12, 13} // LED indicator pins
#define MOTOR_RUN_TIME 300000    // Motor run time (5 minutes in milliseconds)

unsigned long relayTimers[RELAY_COUNT]; // Timers for each relay
bool relayStates[RELAY_COUNT] = {false}; // Current state of each relay
bool relayErrors[RELAY_COUNT] = {false}; // Error status for each relay

void setup() {
  // Initialize relay pins
  for (int i = 0; i < RELAY_COUNT; i++) {
    pinMode(RELAY_PINS[i], OUTPUT);
    digitalWrite(RELAY_PINS[i], LOW); // Start with all relays off
    pinMode(LED_PINS[i], OUTPUT);
    digitalWrite(LED_PINS[i], LOW); // Start with LEDs off
    pinMode(BUTTON_PINS[i], INPUT_PULLUP); // Pull-up resistor
  }
  Serial.begin(9600); // For serial monitoring
}

void loop() {
  for (int i = 0; i < RELAY_COUNT; i++) {
    // Check if the button for this relay is pressed
    if (digitalRead(BUTTON_PINS[i]) == LOW) { // Button pressed
      controlRelay(i);
```

```
            delay(300); // Debounce delay
        }

        // Monitor relay state and open it after the specified time
        if (relayStates[i] && (millis() - relayTimers[i] >= MOTOR_RUN_TIME)) {
            openRelay(i);
        }

        // Check for special conditions (welded state)
        if (relayStates[i] && digitalRead(RELAY_PINS[i]) == LOW) {
            relayErrors[i] = true; // Error: welded relay
        } else {
            relayErrors[i] = false; // No error
        }
    }

    // Report status
    reportStatus();
}

void controlRelay(int relayIndex) {
    relayStates[relayIndex] = true; // Set relay state to closed
    digitalWrite(RELAY_PINS[relayIndex], HIGH); // Activate relay
    digitalWrite(LED_PINS[relayIndex], HIGH); // Turn on LED
    relayTimers[relayIndex] = millis(); // Start the timer
}

void openRelay(int relayIndex) {
    relayStates[relayIndex] = false; // Set relay state to open
    digitalWrite(RELAY_PINS[relayIndex], LOW); // Deactivate relay
```

```
    digitalWrite(LED_PINS[relayIndex], LOW); // Turn off LED
}

void reportStatus() {
    for (int i = 0; i < RELAY_COUNT; i++) {
        Serial.print("Relay ");
        Serial.print(i + 1);
        Serial.print(": ");
        Serial.print(relayStates[i] ? "Closed" : "Open");
        Serial.print(", Error: ");
        Serial.print(relayErrors[i] ? "Error" : "No Error");
        Serial.print(", Special Condition: ");
        Serial.println(relayErrors[i] ? "Welded" : "Normal");
    }
    delay(1000); // Update every second
}
```

This above code is to Automotive Motor Control using 4 channel relays for Arduino Uno.Dump the code into arduino IDE and debug it and dump into microcontroller board with specific Port and assembly the components as per design and do test and validation of project

Reusability

1. Safety Components:

Incorporating safety components is crucial in automotive applications due to the harsh environments and potential risks associated with electrical systems. Key components include:

Fuses and Circuit Breakers:These devices protect against overcurrent situations that can cause overheating or damage to the circuit. By specifying appropriate ratings based on the motor's requirements, the system can prevent catastrophic failures.

Diodes (Flyback Diodes): Installing flyback diodes across relay coils protects the microcontroller and other components from voltage spikes generated when the relay is deactivated. This ensures the longevity of the electronic parts.

Transistors: Using transistors to control the relays can further enhance safety by isolating the microcontroller from high current loads. This configuration reduces the risk of damage to sensitive components.

2. Modular Design Approach:

Implementing a modular design for the software enables easier reuse and adaptation of the codebase for different projects or configurations. Key strategies include:

Encapsulated Classes: Create distinct classes for relay control, button handling, and status monitoring. Each class can handle specific functionalities and communicate with others as needed. This separation of concerns makes it easy to update or replace components without affecting the entire system.

Parameterization: Allow configuration settings, such as relay pins, motor run times, and error thresholds, to be defined externally. This flexibility enables users to customize the system for various motor types and applications without modifying the core code.

Reusable Libraries: Leverage or develop libraries for common functionalities, such as button debouncing or relay management. Libraries can be tested independently, ensuring reliability across different projects.

3. Documentation and Testing:

Thorough documentation of both hardware and software components is essential. Clear instructions on setup, usage, and safety considerations will facilitate understanding and future modifications. Additionally, implementing unit tests for the individual modules will help ensure their reliability, making the entire system robust and adaptable.In summary, by integrating safety components and adopting a modular design approach, the automotive

motor control system can achieve high levels of reusability and safety. This ensures that the system not only performs effectively but can also be easily adapted for a variety of applications in the automotive domain.

Hardware Design Document

This document outlines the hardware design for an automotive motor control system utilizing relays to control multiple DC motors based on digital input commands. The design ensures safe operation and easy adaptation for various automotive applications.

Electrical Calculations

Relay Selection

For our application, we need relays that can handle the current requirements of the motors.

Motor Specification:

Voltage: 12V

Current: 5A (typical for automotive window or seat motors)

Using Ohm's Law (V = I * R), the required power rating for the relay can be calculated:

-Power Rating:

\(P = V \times I = 12V \times 5A = 60W \)

To provide a safety margin, the relay should be rated for at least 10A. Thus, a relay with a rating of 10A at 30V DC or 250V AC will suffice.

Resistor Calculation

For pull-down resistors connected to the push buttons:

Resistor Value:

Using a standard 10kΩ resistor is sufficient to ensure reliable low-level logic for button inputs.

3. Bill of Materials (BOM)

Below is the BOM for the automotive motor control system. The costs are indicative and may vary based on supliers.

Bill of Materials (BOM)

Table 1: Bill of Materials for Automotive Motor Control System Using Relay

Item	Quantity	Cost (INR)	Total Cost (INR)
Arduino Uno	1	100	100
4-Channel Relay Module	1	100	100
Push Button Switch	4	100	400
10k Resistors	4	100	400
LEDs	4	100	400
1N4001 Diodes	4	100	400
2N2222 Transistors	4	100	400
12V Power Supply	1	100	100
Automotive Fuses	4	100	400
Connector Blocks	1	100	100
Total Cost			**2,900**