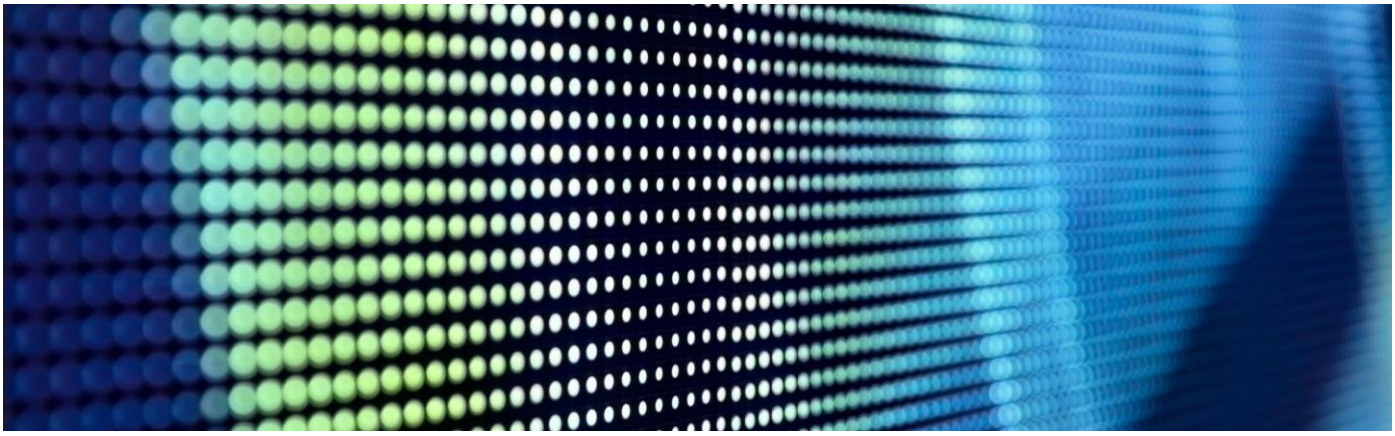


Color-Coded Switch-Controlled LED System

Introduction:

This project involves the development of a dynamic control system where toggle switches with color-coded caps (red, green, blue, yellow) are used to control LEDs of corresponding colors. The system dynamically adapts to changes in the physical arrangement of switches and LEDs, ensuring that the correct LED lights up based on the color of the switch cap, regardless of their rearrangement.

Objective:

- To create a system that lights up the correct LED based on the switch with a matching color-coded cap.
- To demonstrate the adaptability of the system when switch caps or LED placements are interchanged.
- To ensure robust behavior when switches are toggled on and off in any sequence.

Components Used:

- UNO microcontroller
- 4 toggle switches with color-coded caps (red, green, blue, yellow)
- 4 LEDs (red, green, blue, yellow)
- Resistors for LEDs
- Connecting wires
- Breadboard for prototyping

Problem Statement:

Design and implement a system where each switch has a color-coded cap (red, green, blue, yellow) and operates an LED of the corresponding color. The system should dynamically adjust to changes in the placement of the switches or LEDs, ensuring that the correct LED lights up regardless of the rearrangement. The system should handle scenarios where switches are toggled on and off individually and in sequence.

Solution:

The system utilizes an UNO microcontroller programmed to:

1. Read the state of each toggle switch.
2. Dynamically map switches to LEDs based on the activation sequence.
3. Ensure the correct LED lights up based on the switch cap's color, regardless of rearrangement.
4. Handle toggling of switches in real-time, turning LEDs on or off accordingly.
5. Maintain correct mappings after all switches have been toggled once.

Implementation Details:

1. Switch and LED Setup:

   - Switches are connected to the UNO input pins with pull-up resistors to detect toggling.

   - LEDs are connected to UNO output pins via resistors to prevent overcurrent.

2. Logic Description:

   - During the system startup, LEDs follow the state of switches based on their color index.

   - Once all switches are toggled at least once, the system transitions to a dynamic mode where LEDs respond to the sequence in which switches are toggled.

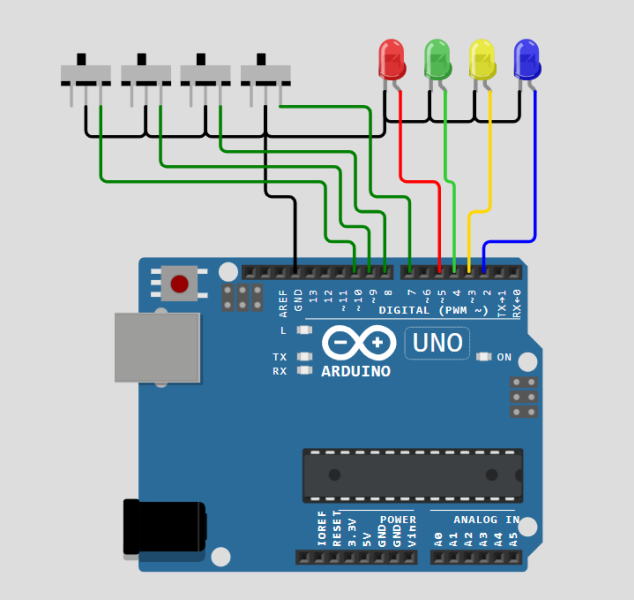
   - When a switch is toggled ON, the next available LED is assigned to it.

   - If a switch is toggled OFF, the corresponding LED is turned off.

3. Dynamic Adaptation:

   - If the physical arrangement of switches or LEDs is changed, the system automatically adapts and ensures correct LED activation based on the color-coded caps.

Circuit Diagram:



Code:

```
// Include necessary libraries
#include <Arduino.h>

// Define pin numbers for switches and LEDs
const int switchPins[] = {10, 9, 8, 7}; // Switches connected to pins 10, 9, 8, 7
const int ledPins[] = {5, 4, 3, 2}; // LEDs connected to pins 5, 4, 3, 2

// Array to track the state of each switch
bool switchState[4] = {false, false, false, false};
// Array to track which LED is currently assigned to a switch
int ledAssignment[4] = {-1, -1, -1, -1};
// Flag to indicate if initial startup behavior is complete
bool startupComplete = false;

void setup() {
    // Configure switch pins as input with pull-up resistors
    for (int i = 0; i < 4; i++) {
        pinMode(switchPins[i], INPUT_PULLUP);
    }

    // Configure LED pins as output
    for (int i = 0; i < 4; i++) {
        pinMode(ledPins[i], OUTPUT);
        digitalWrite(ledPins[i], LOW); // Ensure LEDs are off initially
    }
}

void loop() {
    if (!startupComplete) {
        startupComplete = checkStartup();
        return;
    }

    for (int i = 0; i < 4; i++) {
        // Read the state of each switch (LOW means pressed because of pull-up)
        bool currentState = !digitalRead(switchPins[i]);

        if (currentState && !switchState[i]) {
            // If the switch is toggled ON
            assignLED(i);
            switchState[i] = true;
        } else if (!currentState && switchState[i]) {
            // If the switch is toggled OFF
            unassignLED(i);
            switchState[i] = false;
        }
    }
}



bool checkStartup() {
```

```
    static bool allSwitchesOn[4] = {false, false, false, false};

    for (int i = 0; i < 4; i++) {
        bool currentState = !digitalRead(switchPins[i]);

        if (currentState && !allSwitchesOn[i]) {
            // Turn on LED corresponding to switch index
            digitalWrite(ledPins[i], HIGH);
            allSwitchesOn[i] = true;
        }
    }

    // Check if all switches have been toggled ON at least once
    for (int i = 0; i < 4; i++) {
        if (!allSwitchesOn[i]) {
            return false;
        }
    }

    // Turn off all LEDs after initial startup
    for (int i = 0; i < 4; i++) {
        digitalWrite(ledPins[i], LOW);
    }

    return true;
}

void assignLED(int switchIndex) {
    // Find the next available LED slot
    for (int i = 0; i < 4; i++) {
        if (ledAssignment[i] == -1) {
            // Assign the LED to this switch
            ledAssignment[i] = switchIndex;
            updateLEDs();
            return;
        }
    }
}

void unassignLED(int switchIndex) {
    // Find and unassign the LED corresponding to this switch
    for (int i = 0; i < 4; i++) {
        if (ledAssignment[i] == switchIndex) {
            ledAssignment[i] = -1;
            updateLEDs();
            return;
        }
    }
}



void updateLEDs() {
```

```
    // Turn off all LEDs
    for (int i = 0; i < 4; i++) {
        digitalWrite(ledPins[i], LOW);
    }

    // Turn on LEDs based on the order of assignments
    for (int i = 0; i < 4; i++) {
        if (ledAssignment[i] != -1) {
            digitalWrite(ledPins[i], HIGH);
        }
    }
}
```

Demonstration Plan:

- Start with switches and LEDs in their default arrangement.
- Turn switches on and off in sequence to show correct LED activation.
- Rearrange switch caps and demonstrate that the system still lights the correct LEDs.
- Swap LED positions and verify that the system maintains accurate operation.

Conclusion:

The Color-Coded Switch-Controlled LED System demonstrates the flexibility and reliability of microcontroller-based control logic. By dynamically mapping switches to LEDs, it ensures seamless operation even in scenarios of physical rearrangement, making it a robust solution for color-coded control systems.