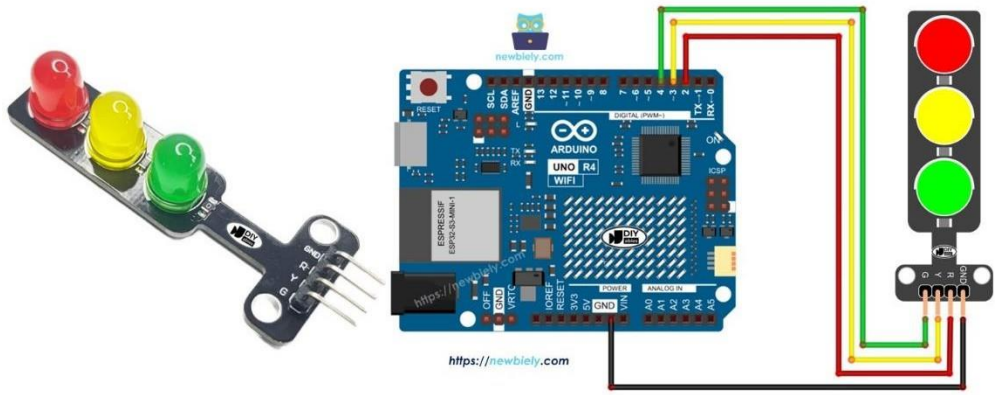
Ex. No: 5	Simulate a Traffic Light Controller using any IoT IDE
16-08-2024	



Abstract:

This report focuses on the implementation of a traffic light controller system using an IoT IDE. The system is designed to manage traffic at an intersection with three colored lights: Red, Yellow, and Green. Each light is active for a defined period, and the system transitions between the lights following the standard rules of traffic control. The project is implemented using the Arduino IDE and simulated using appropriate IoT hardware components.

Aim:

To design and implement a Traffic Light Controller (TLC) system using an Arduino board and LEDs, simulating real-world traffic light sequences in an IoT-based environment.

Introduction:

Traffic lights are essential for controlling vehicular movement at intersections. A Traffic Light Controller (TLC) regulates the sequence of the lights and ensures that vehicles move efficiently and safely. The aim of this project is to simulate a TLC using the Arduino IDE with an embedded system that mimics real-world traffic light sequences. The system can be extended to handle real-world traffic conditions by connecting with additional sensors.

Objectives:

1. To develop an efficient Traffic Light Controller system.
2. To simulate traffic light sequences using an IoT framework.
3. To understand the interaction between IoT hardware and the IDE for real-time control.

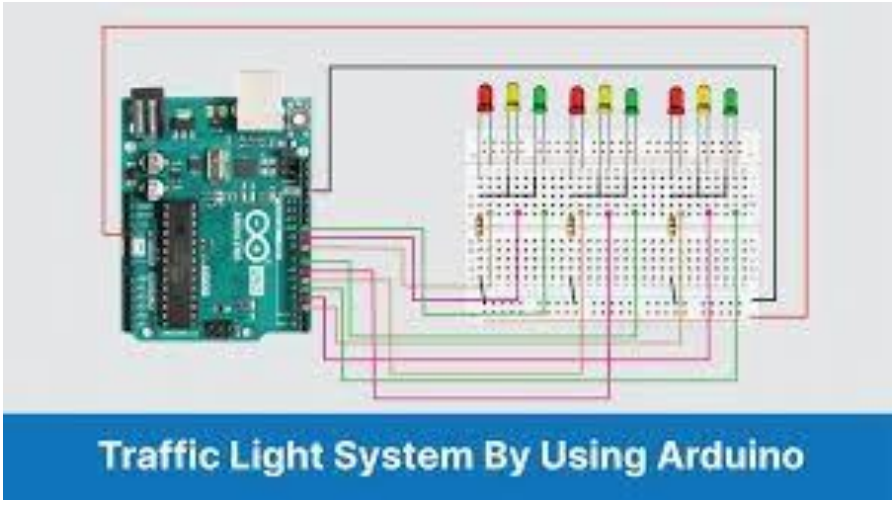
- Hardware Components:

- Arduino Uno board
- Three LEDs (Red, Yellow, Green)
- Resistors (220 ohms)
- Jumper Wires
- Breadboard

- Software Components:

- Arduino IDE for programming and simulation.
- Serial Monitor for debugging and real-time feedback.

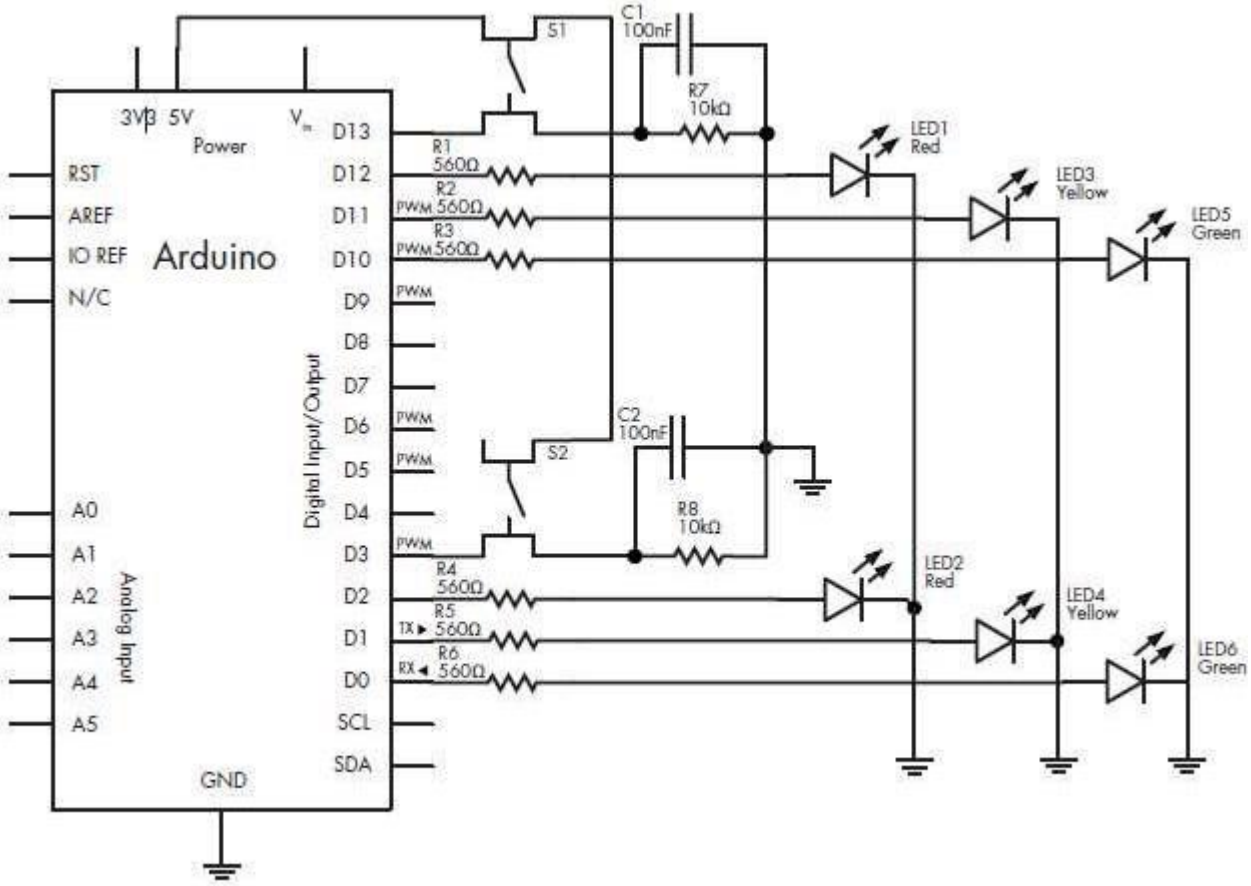
System Design and Architecture



The system uses three LEDs to represent the traffic lights. The Red LED indicates stopping traffic, the Yellow LED signals slowing down, and the Green LED allows vehicles to proceed. These LEDs are connected to the Arduino board, and each light is programmed to stay on for a defined period using delay functions. The transition from Red to Green follows the standard cycle, where Red -> Green -> Yellow -> Red. The system repeats the cycle endlessly.

Block Diagram

The traffic light controller is set up as shown in the diagram below, with each LED connected to a specific digital pin on the Arduino board.



Algorithm

1. Start.
2. Initialize pins for Red, Yellow, and Green LEDs.
3. Turn ON the Red LED.
   - Delay for 5 seconds.
4. Turn OFF the Red LED.
5. Turn ON the Green LED.
   - Delay for 7 seconds.

6. Turn OFF the Green LED.
7. Turn ON the Yellow LED.
   - Delay for 3 seconds.
8. Turn OFF the Yellow LED.
9. Repeat steps 3-8 in an infinite loop.

Code Implementation:

```
// Define the pins for each color
const int redPin = 2;
const int yellowPin = 3;
const int greenPin = 4;

// Define the timing intervals (in milliseconds)
const unsigned long redDuration = 5000;
const unsigned long yellowDuration = 2000;
const unsigned long greenDuration = 5000;
const unsigned long yellowBlinkInterval = 500; // Blinking interval for yellow

// State variables
unsigned long previousMillis = 0;
int currentState = 0;       // 0: Red, 1: Yellow, 2: Green
bool yellowBlinkState = LOW; // Yellow light blink state

void initializePins() {
  pinMode(redPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
}

void updateTrafficLight() {
  unsigned long currentMillis = millis();

  // Control the traffic light based on the current state
  switch (currentState) {
    case 0: // Red Light
      digitalWrite(redPin, HIGH);
      digitalWrite(yellowPin, LOW);
      digitalWrite(greenPin, LOW);

      if (currentMillis - previousMillis >= redDuration) {
        previousMillis = currentMillis;
        currentState = 1; // Move to yellow
      }
      break;
```

```
    case 1: // Blinking Yellow Light
      digitalWrite(redPin, LOW);
      digitalWrite(greenPin, LOW);

      if (currentMillis - previousMillis >= yellowBlinkInterval) {
        yellowBlinkState = !yellowBlinkState; // Toggle yellow light
        digitalWrite(yellowPin, yellowBlinkState);
        previousMillis = currentMillis;
      }

      // Move to next state after full yellow duration
      if (currentMillis - previousMillis >= yellowDuration) {
        currentState = 2; // Move to green
        previousMillis = currentMillis;
      }
      break;

    case 2: // Green Light
      digitalWrite(redPin, LOW);
      digitalWrite(yellowPin, LOW);
      digitalWrite(greenPin, HIGH);

      if (currentMillis - previousMillis >= greenDuration) {
        previousMillis = currentMillis;
        currentState = 0; // Reset to red
      }
      break;
  }
}

void setup() {
  initializePins();
  previousMillis = millis();
}

void loop() {
  updateTrafficLight();
}
```

Working Principle

The above code uses the `pinMode()` function to define the pins for the LEDs. The `digitalWrite()` function is used to control whether an LED is ON or OFF. The `delay()` function is utilized to simulate the duration for which a particular traffic light remains active.

- Red Light: Stays ON for 5 seconds to indicate the "STOP" signal.

- Green Light: Stays ON for 7 seconds to indicate the "GO" signal.

- Yellow Light: Stays ON for 3 seconds to indicate the "SLOW DOWN" signal.

The entire sequence repeats continuously, as is typical for a traffic light system.

Challenges Faced

1. Synchronization of Light Timings: Initially, setting accurate delays between transitions required fine-tuning.

2. LED Power Supply: Managing power to each LED to ensure even brightness across the setup was a challenge due to voltage differences.

Conclusion

The Traffic Light Controller was successfully simulated using the Arduino IDE. The project demonstrated how IoT devices and embedded systems can be utilized to manage real-world problems like traffic control. Future expansions of this project could involve adding real-time sensor input to adjust light timings dynamically, based on traffic density.

References

1. Arduino Documentation:
https://www.arduino.cc/en/Guide

2. Traffic Light Control Systems: A Practical Guide to Design and Simulation

Result:

The simulation was carried out using the Arduino IDE. The serial monitor displayed real-time status updates of the traffic lights, and the LEDs transitioned according to the programmed sequence. The expected results, i.e., timed light transitions, were successfully achieved.