Smart Traffic Light Controller

Name: Adarsh Hinsodiya R

Roll Number: 106124008

Date: 29-08-2025

1. Requirements

Hardware Components:

- **Microcontroller:** Arduino UNO R3
- **Actuators (Lights):**
    - 4 x Red LEDs
    - 4 x Yellow LEDs
    - 4 x Green LEDs
- **Sensors:**
    - 4 x IR Proximity Sensors (for traffic density detection)
    - 1 x Sound Sensor Module (for emergency vehicle siren detection)
- **Resistors:** 12 x 220Ω (for current limiting the LEDs)
- **Prototyping:**
    - 1 x Large Breadboard
    - Jumper Wires (Assorted)
- **Power:** 9V Battery or USB Cable for Arduino

Software:

- **Simulation & Design:** Autodesk Tinkercad
- **Programming:** Arduino IDE (Integrated Development Environment)

2. Abstract

Traditional traffic light systems operate on fixed, predetermined timers, which often leads to inefficient traffic flow, unnecessary vehicle idling, and increased congestion, especially during fluctuating traffic conditions. This project presents the design and implementation of a **Smart Traffic Light Controller** that adapts to real-time traffic conditions. The system utilizes an Arduino UNO microcontroller as its central processing unit. IR sensors are deployed on each of the four roads of an intersection to measure traffic density, allowing for dynamic adjustment of the green light duration to

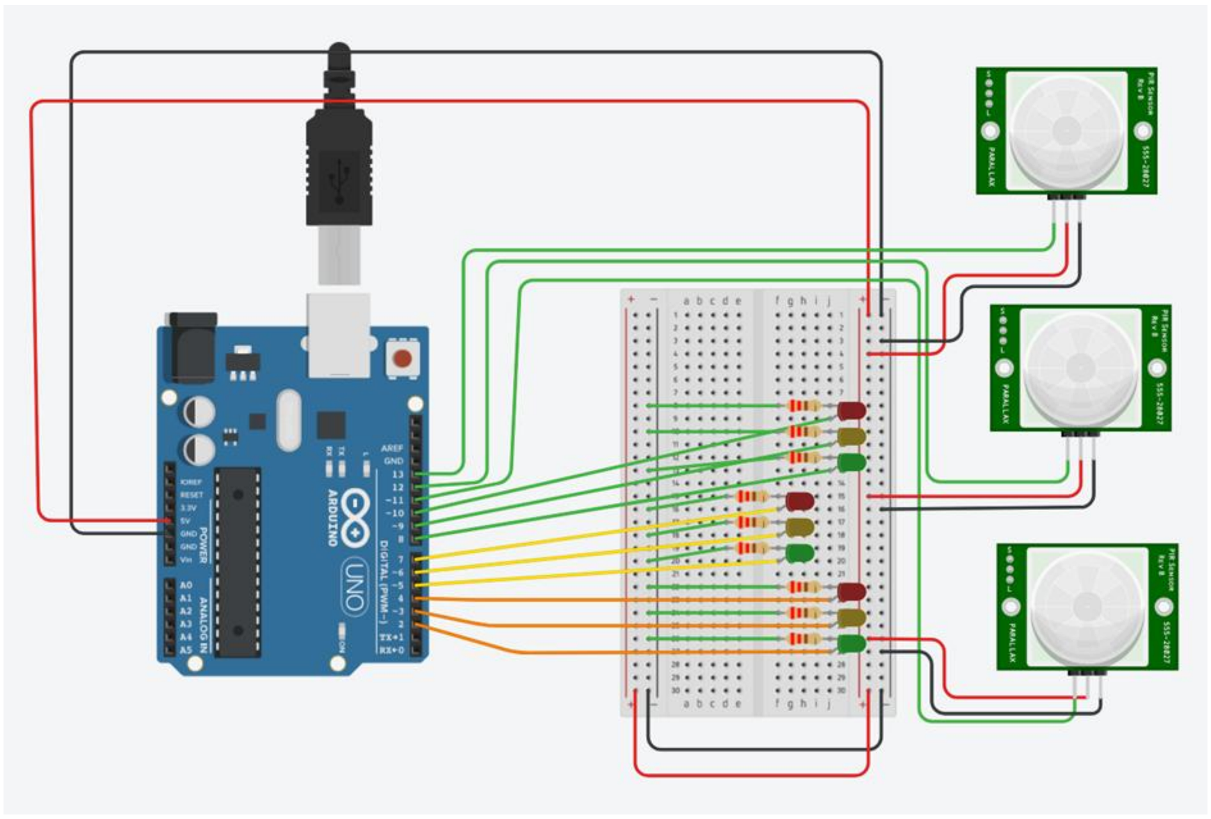
prioritize lanes with heavier traffic. Furthermore, the system incorporates a sound sensor to detect the sirens of emergency vehicles. Upon detection, it initiates an emergency preemption protocol, clearing a path for the vehicle to pass through the intersection without delay. This intelligent, sensor-based approach significantly improves traffic management efficiency, reduces waiting times, and enhances the responsiveness of emergency services.

3. Circuit Diagram

The circuit is designed for a four-way intersection. Each of the four lanes is equipped with a set of Red, Yellow, and Green LEDs and an IR sensor to detect vehicles. A single sound sensor is placed to monitor for emergency sirens.

Connections to Arduino UNO:

- **LEDs:** Each of the 12 LEDs (4 Red, 4 Yellow, 4 Green) is connected to a separate digital pin on the Arduino (e.g., pins 2 through 13). The anode (longer leg) of each LED is connected in series with a 220Ω resistor, which is then connected to the Arduino pin. The cathode (shorter leg) is connected to the GND rail.
- **IR Sensors:** The VCC and GND pins of the four IR sensors are connected to the 5V and GND rails of the breadboard, respectively. The OUT pin of each sensor is connected to a digital input pin on the Arduino (e.g., A0, A1, A2, A3) to read the presence of a vehicle.
- **Sound Sensor:** The VCC and GND pins are connected to 5V and GND. The OUT pin is connected to a digital pin on the Arduino (e.g., pin 1) to detect a high signal when a loud sound is registered.



4. Arduino Code

The following code implements the logic for dynamic signal timing based on traffic density and includes the emergency vehicle preemption feature.

```
// Smart Traffic Light Controller Code


// --- Pin Definitions for LEDs (Intersection 1 & 2) ---

// Lane 1 (North)

const int R1 = 13;

const int Y1 = 12;

const int G1 = 11;

// Lane 2 (South)

const int R2 = 10;

const int Y2 = 9;

const int G2 = 8;

// Lane 3 (East)
```

```
const int R3 = 7;
const int Y3 = 6;
const int G3 = 5;
// Lane 4 (West)
const int R4 = 4;
const int Y4 = 3;
const int G4 = 2;

// --- Pin Definitions for Sensors ---
const int SENSOR_1 = A0; // IR Sensor for Lane 1
const int SENSOR_2 = A1; // IR Sensor for Lane 2
const int SENSOR_3 = A2; // IR Sensor for Lane 3
const int SENSOR_4 = A3; // IR Sensor for Lane 4
const int SOUND_SENSOR = 1; // Sound Sensor for Emergency Vehicle

// --- Timing Variables ---
int greenLightDuration = 5000; // Default green light time (5 seconds)
const int yellowLightDuration = 2000; // Yellow light time (2 seconds)
const int densityThreshold = 0; // IR sensor threshold (0 for detected, 1 for not detected)

void setup() {
 // Initialize all LED pins as OUTPUT
 for (int i = 2; i <= 13; i++) {
  pinMode(i, OUTPUT);
 }

 // Initialize all sensor pins as INPUT
 pinMode(SENSOR_1, INPUT);
```

```
  pinMode(SENSOR_2, INPUT);
  pinMode(SENSOR_3, INPUT);
  pinMode(SENSOR_4, INPUT);
  pinMode(SOUND_SENSOR, INPUT);

  // Start with all lights RED
  allLightsRed();
}

void loop() {
  // 1. Check for emergency vehicles first
  if (digitalRead(SOUND_SENSOR) == HIGH) {
    emergencyMode();
  } else {
    // 2. Run normal traffic cycle if no emergency
    runTrafficCycle();
  }
}

void runTrafficCycle() {
  // --- Check Traffic Density ---
  // Read sensor values (LOW or 0 means a vehicle is detected)
  bool density1 = digitalRead(SENSOR_1) == densityThreshold;
  bool density2 = digitalRead(SENSOR_2) == densityThreshold;
  bool density3 = digitalRead(SENSOR_3) == densityThreshold;
  bool density4 = digitalRead(SENSOR_4) == densityThreshold;

  // --- Dynamic Green Light Duration ---
  // Give longer green time to denser lanes
  int greenDuration1_2 = greenLightDuration;
  int greenDuration3_4 = greenLightDuration;

  if (density1 || density2) {
    greenDuration1_2 = 8000; // Increase to 8 seconds if traffic on N-S
  }
  if (density3 || density4) {
    greenDuration3_4 = 8000; // Increase to 8 seconds if traffic on E-W
  }

  // --- Traffic Light Sequence ---
  // Cycle 1: North-South Green, East-West Red
  allLightsRed();
  delay(1000);
  digitalWrite(G1, HIGH);
  digitalWrite(G2, HIGH);
  delay(greenDuration1_2);
  digitalWrite(G1, LOW);
  digitalWrite(G2, LOW);
  digitalWrite(Y1, HIGH);
  digitalWrite(Y2, HIGH);
  delay(yellowLightDuration);
  digitalWrite(Y1, LOW);
  digitalWrite(Y2, LOW);

  // Cycle 2: East-West Green, North-South Red
  allLightsRed();
```

```
  delay(1000);
  digitalWrite(G3, HIGH);
  digitalWrite(G4, HIGH);
  delay(greenDuration3_4);
  digitalWrite(G3, LOW);
  digitalWrite(G4, LOW);
  digitalWrite(Y3, HIGH);
  digitalWrite(Y4, HIGH);
  delay(yellowLightDuration);
  digitalWrite(Y3, LOW);
  digitalWrite(Y4, LOW);
}

// Function to handle emergency vehicle detection
void emergencyMode() {
  // Turn all lights red immediately to stop traffic
  allLightsRed();

  // For this demo, we will assume the emergency vehicle is on Lane 1 (North)
  // and clear its path. A more advanced system could use multiple sound sensors
  // or RF communication to determine direction.
  delay(1000); // Wait for traffic to stop
  digitalWrite(G1, HIGH); // Green for Lane 1

  // Keep this path open until the sound is no longer detected
  while (digitalRead(SOUND_SENSOR) == HIGH) {
    delay(100); // Check every 100ms
  }
  // Once siren passes, return to normal operation
  digitalWrite(G1, LOW);
  allLightsRed();
  delay(2000); // A brief pause before resuming normal cycle
}

// Utility function to set all lights to RED
void allLightsRed() {
  digitalWrite(G1, LOW); digitalWrite(Y1, LOW); digitalWrite(R1, HIGH);
  digitalWrite(G2, LOW); digitalWrite(Y2, LOW); digitalWrite(R2, HIGH);
  digitalWrite(G3, LOW); digitalWrite(Y3, LOW); digitalWrite(R3, HIGH);
  digitalWrite(G4, LOW); digitalWrite(Y4, LOW); digitalWrite(R4, HIGH);
}
```

5. Description of the Project

This project moves beyond the conventional fixed-timer traffic light systems by introducing two key intelligent features: dynamic signal timing and emergency vehicle preemption.

Dynamic Signal Timing: The core of the system's efficiency lies in its ability to adapt to traffic flow. An IR sensor is placed on each of the four lanes approaching the intersection. These sensors detect the presence of vehicles waiting at the red light. The Arduino continuously polls these sensors. Before initiating a green light cycle for a particular axis (e.g., North-South), it checks if vehicles are detected on that axis. If significant traffic is detected, the default green light duration is extended from 5 seconds to 8 seconds. This ensures that congested lanes are cleared more effectively, reducing overall wait times and improving traffic throughput.

Emergency Vehicle Preemption: Public safety is a critical aspect of urban management. This system includes a sound sensor that listens for the high-decibel signature of a siren. When a siren is detected, the Arduino immediately triggers an emergencyMode() function. This function overrides the current traffic light state and sets all signals to red, safely halting all traffic at the intersection. After a brief pause, it provides a green light to a predetermined lane (in this demo, the North-bound lane) to allow the emergency vehicle to pass unimpeded. The system remains in this state until the sound sensor no longer detects the siren, after which it safely transitions back to its normal operational cycle. This feature can drastically reduce the response time for emergency services.

6. Future Work

This project serves as a robust foundation for a more advanced and integrated traffic management system. Future enhancements could include:

- **Computer Vision:** Replacing IR sensors with a camera and using a more powerful processor (like a Raspberry Pi) to run object detection algorithms. This would allow for more accurate vehicle counting, classification (cars, trucks, buses), and pedestrian detection.
- **Vehicle-to-Infrastructure (V2I) Communication:** Implementing a network of smart intersections that communicate with each other. This would allow the system to anticipate traffic flow and make more informed decisions, preventing the "green wave" problem where one intersection clears traffic only for it to get stuck at the next one.
- **Real-Time Clock (RTC) Integration:** Adding an RTC module to enable different traffic control profiles based on the time of day or day of the week (e.g., rush hour, nighttime, weekend).
- **Data Logging and Analysis:** Storing traffic data over time to analyze patterns and further optimize the signal timing algorithms, potentially using machine learning to predict traffic flow.
- **Solar Power Integration:** Making the system self-sustainable by powering it with solar panels and a rechargeable battery pack, reducing its environmental footprint.