EMBEDDED SYSTEMS
TRAFFIC LIGHT

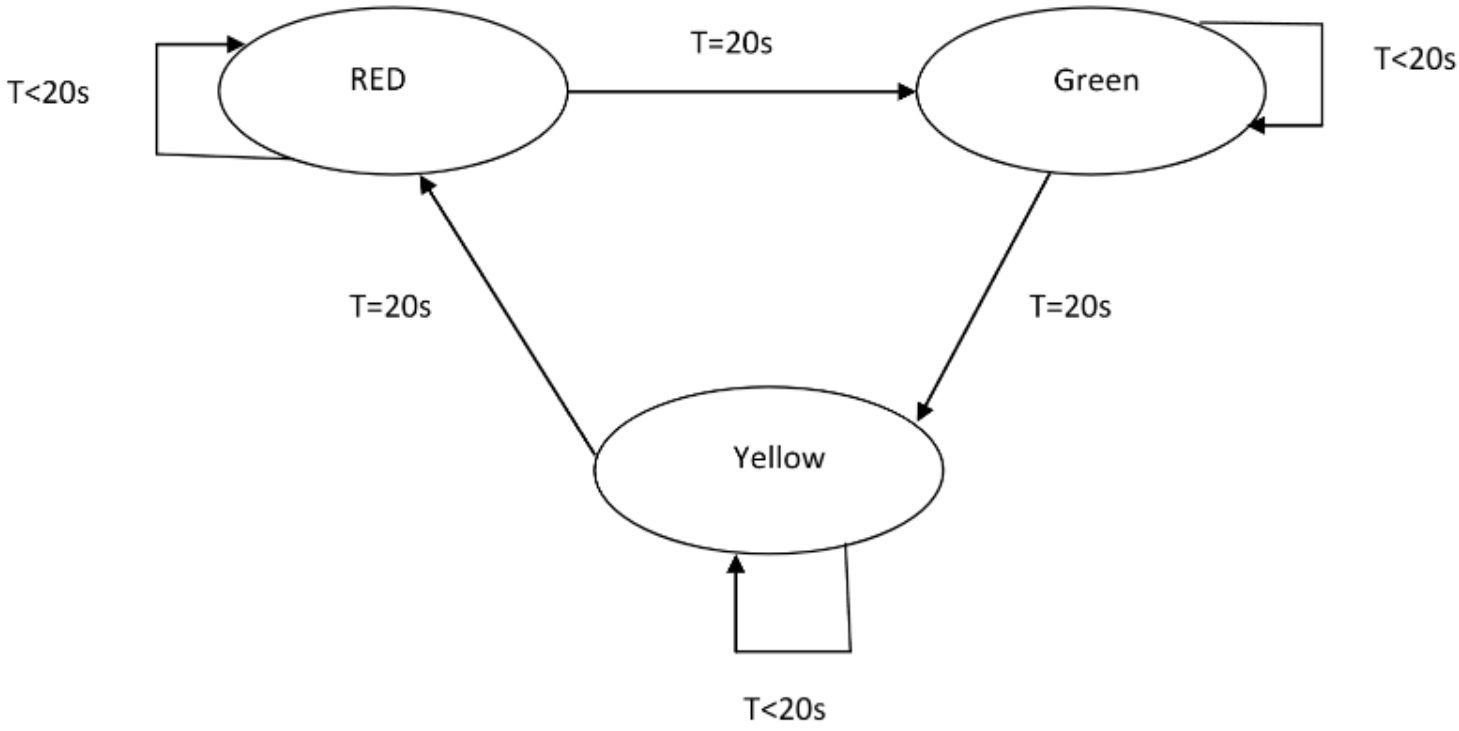
BY

SYED MUHAMMAD ABIS RIZVI

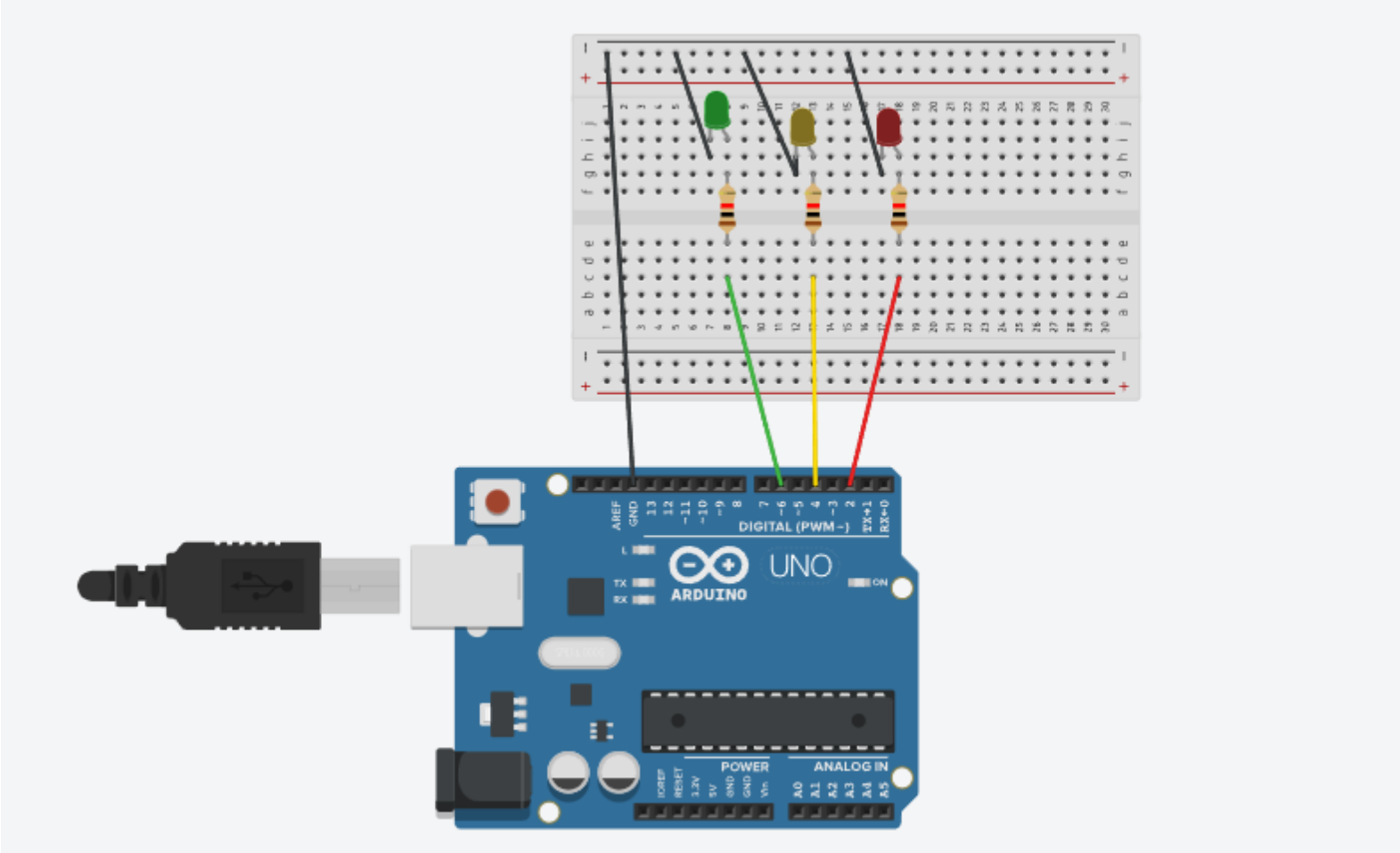
IBRAHIM ABDEMALEK

GITHUB-LINK: https://github.com/ibrahimabdelmalek31/Embedded-Systems-Team-D01

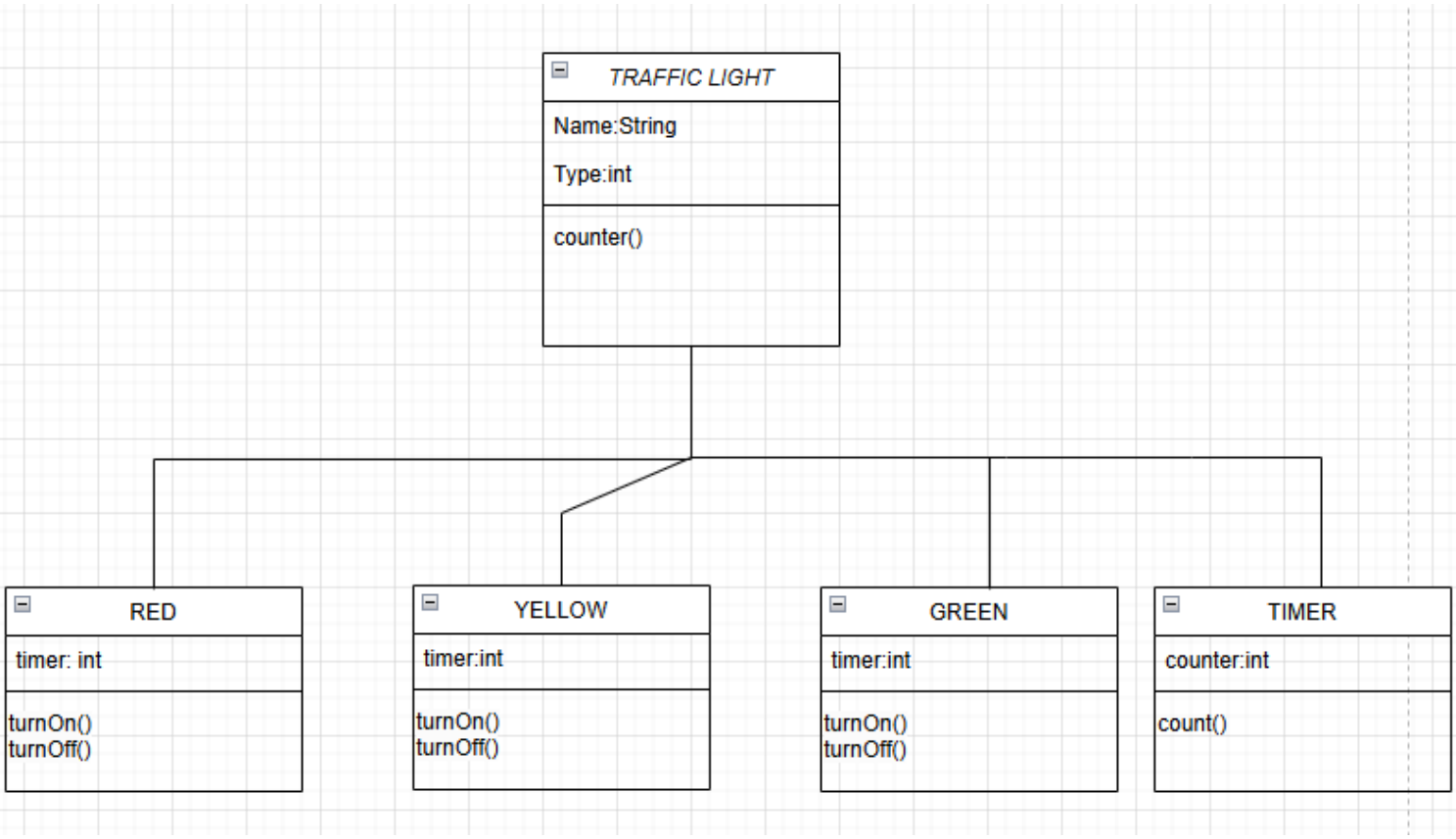
STATE MACHINE DIAGRAM TASK 1



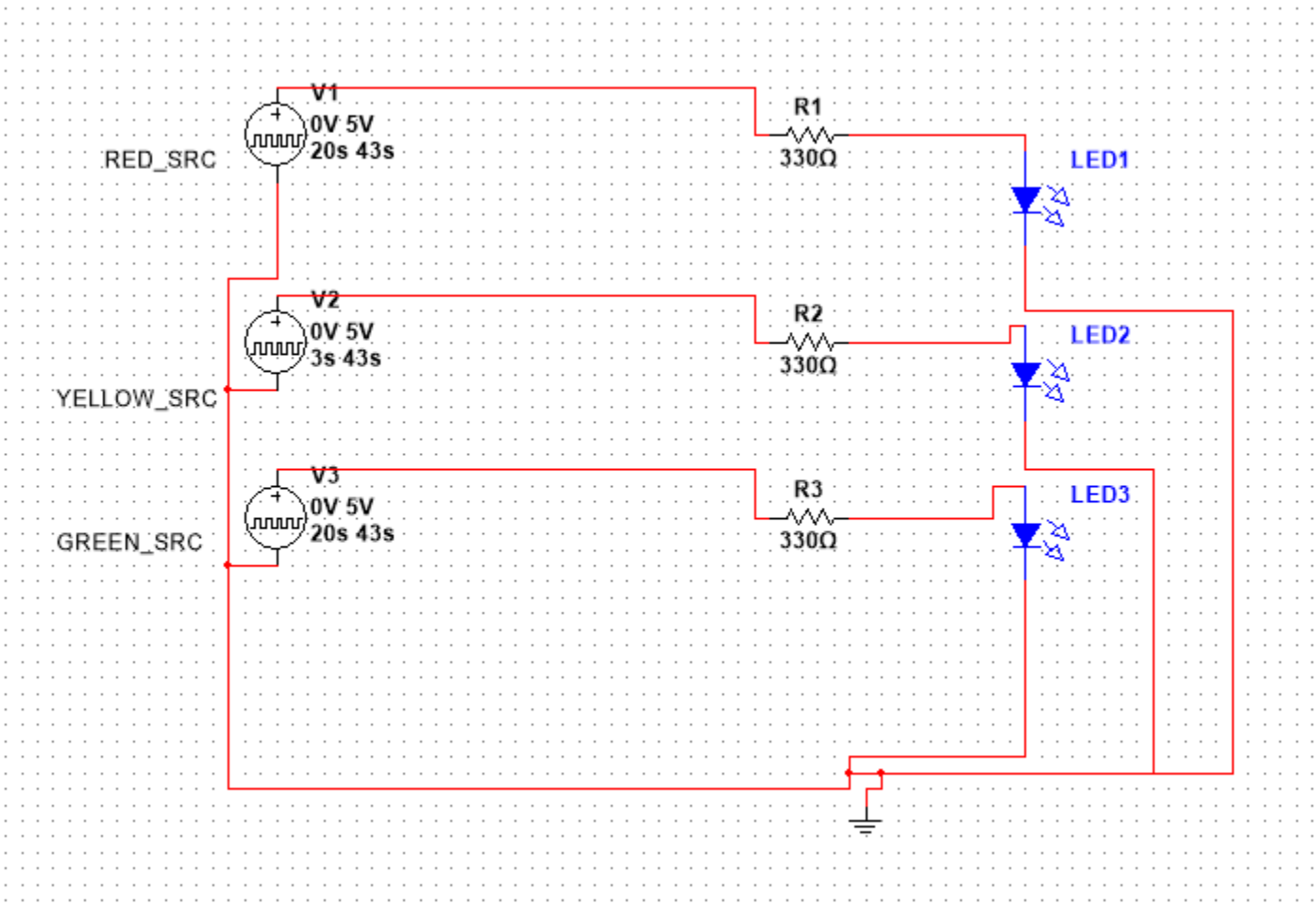
TASK 1



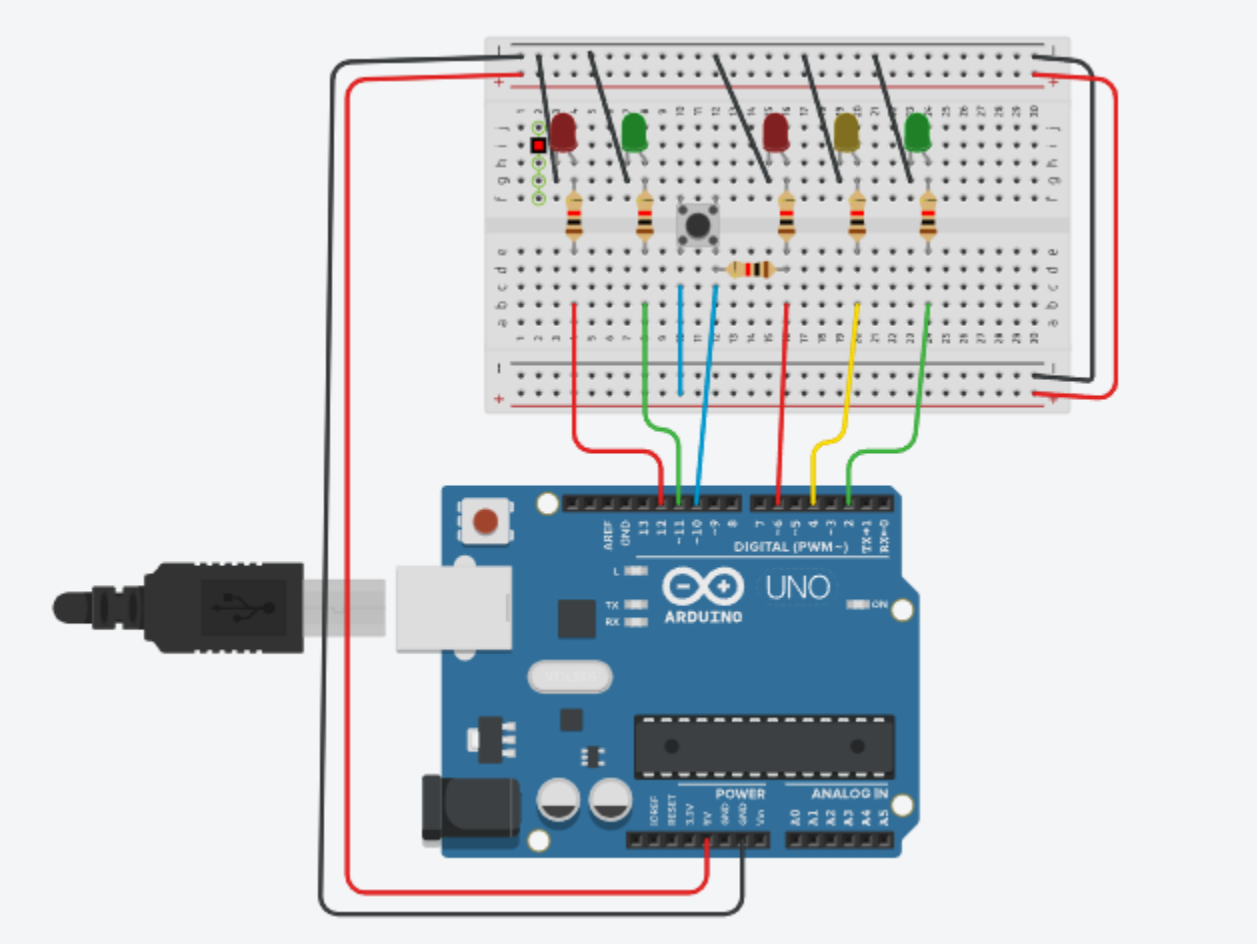
CLASS DIAGRAM TASK 1



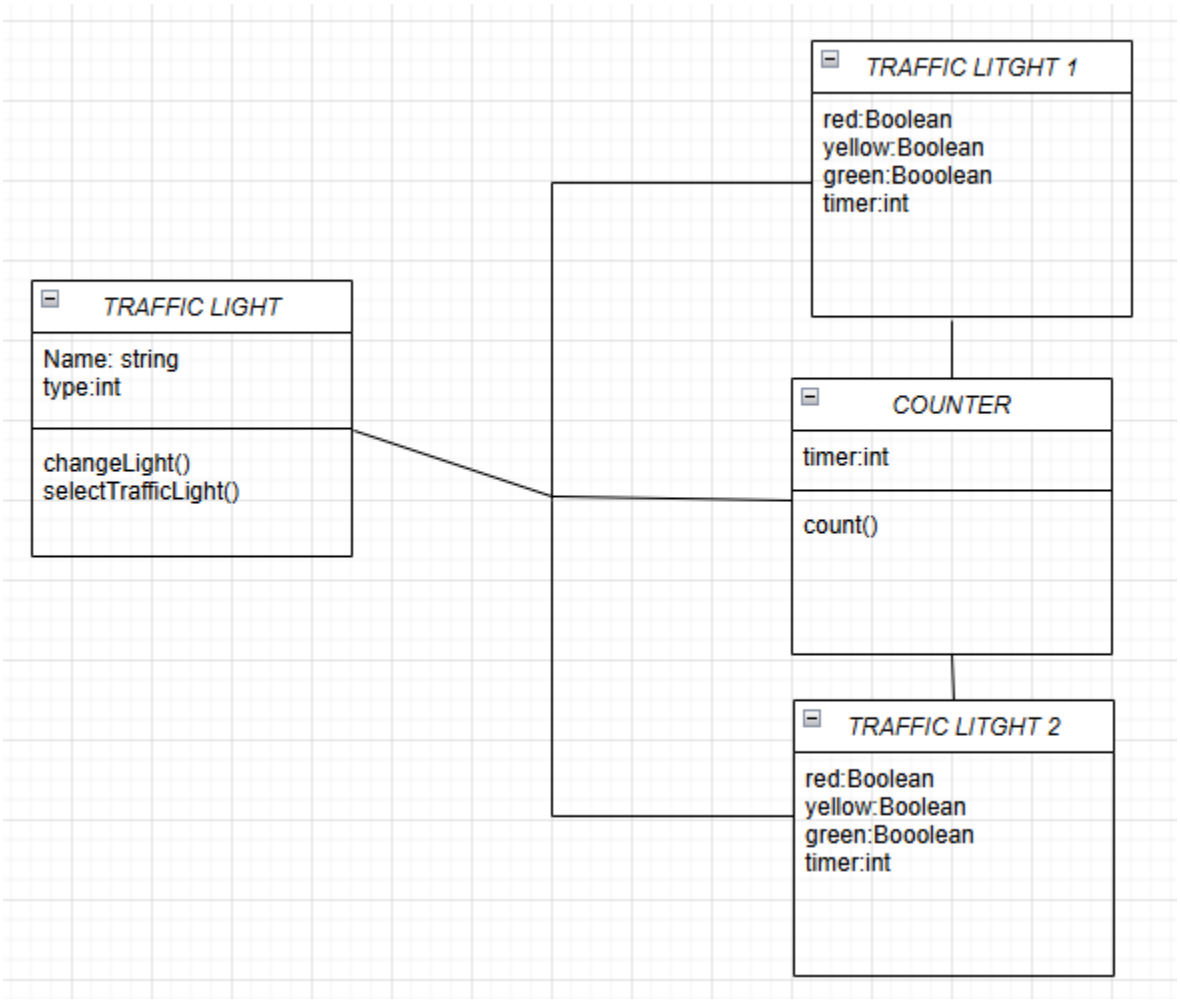
MULTI-SIM IMPLEMENTATION



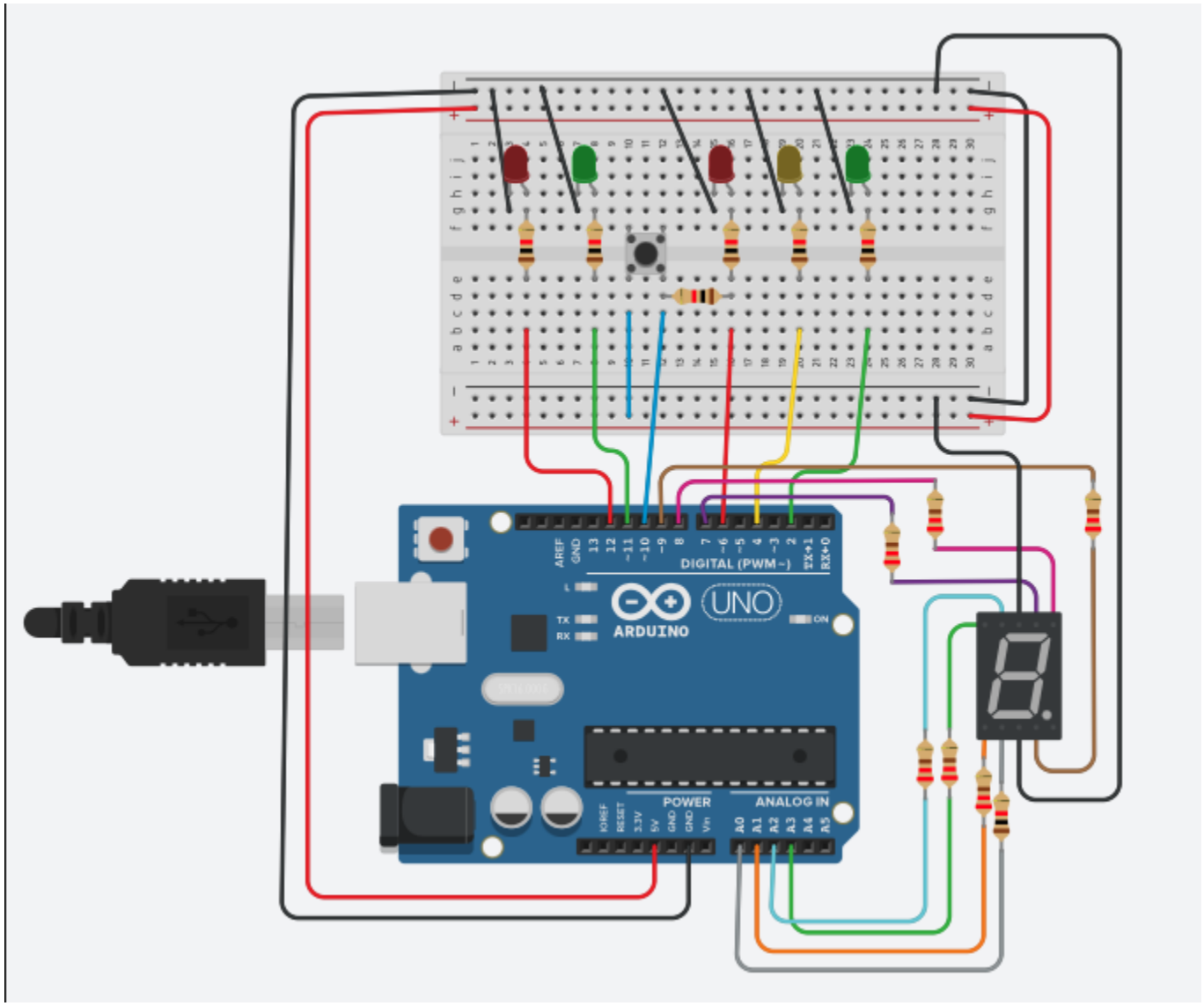
TASK 2



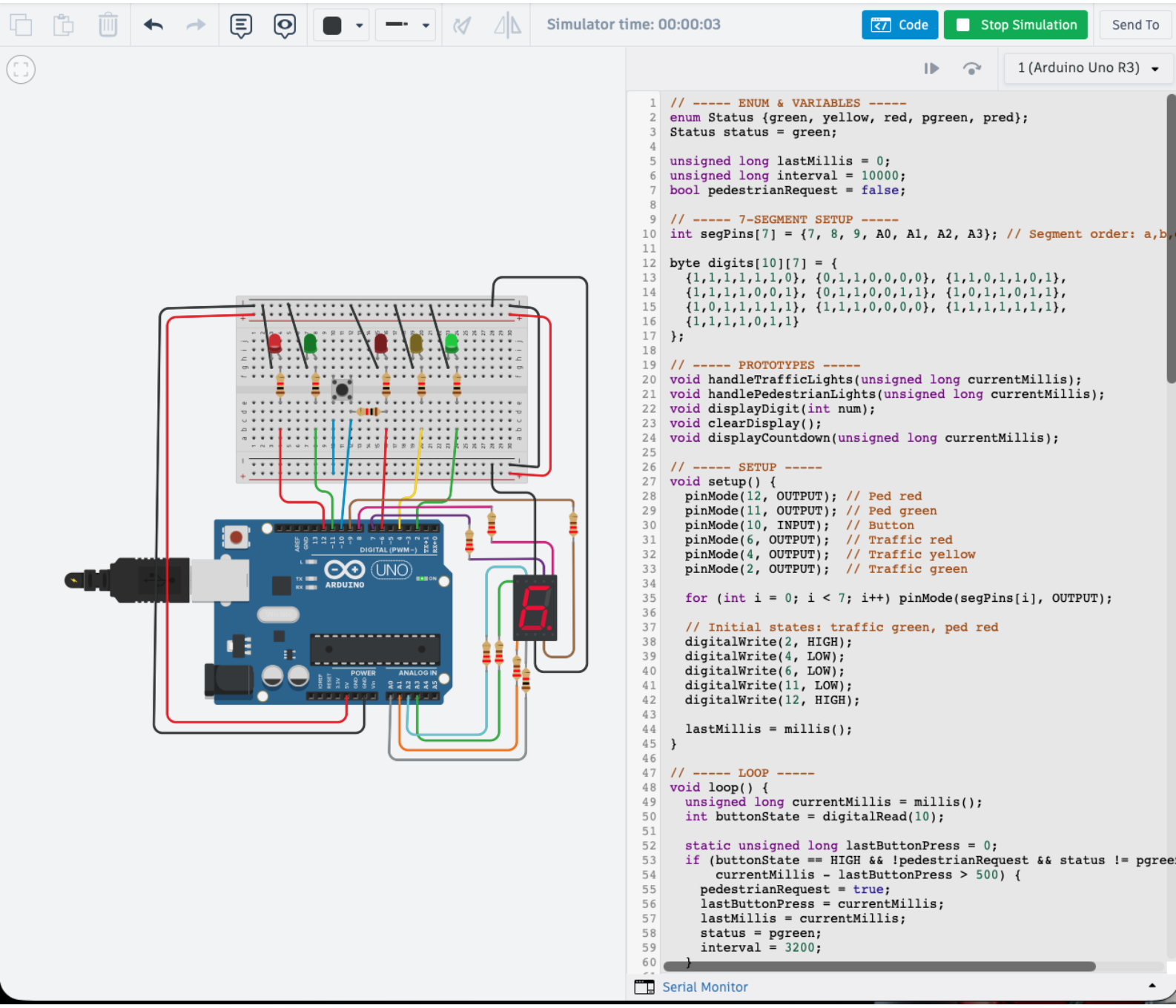
CLASS DIAGRAM TASK 2



TASK 2 WITH SEVEN SEGMENT

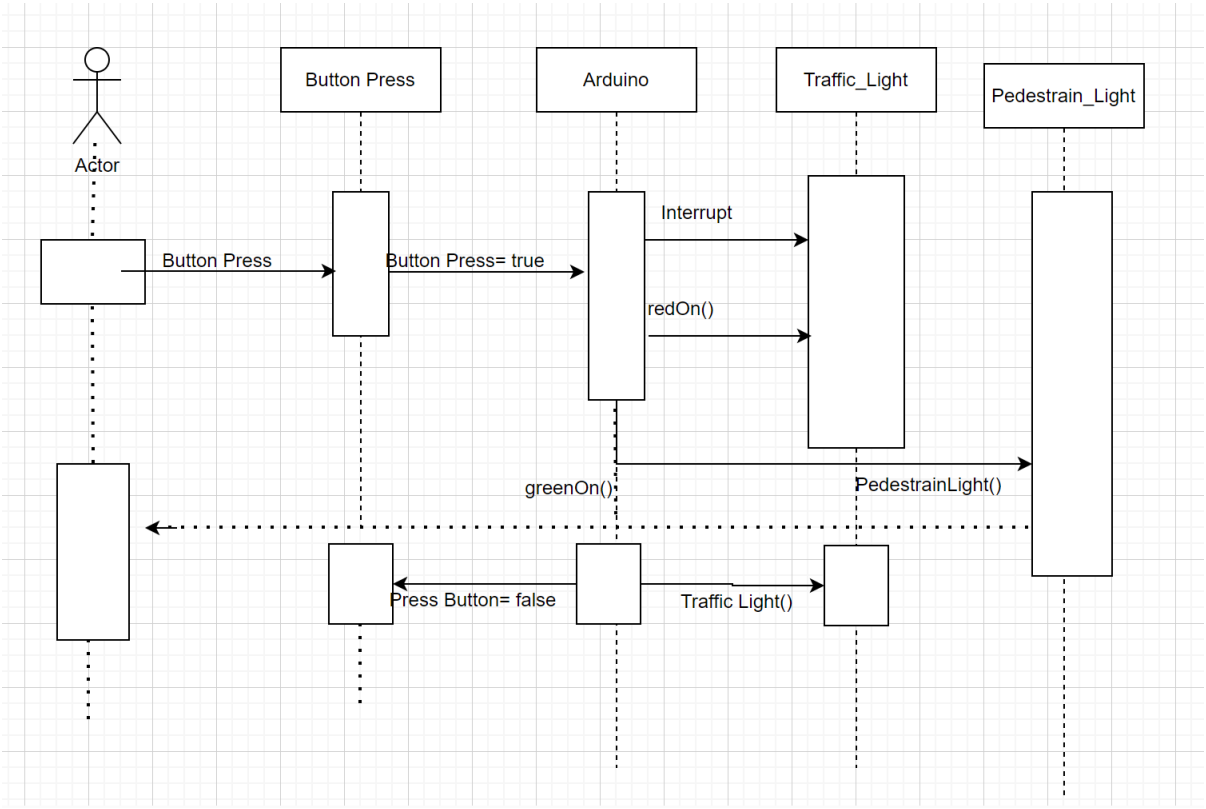


SEGMENT IMPLEMENTATION CODE

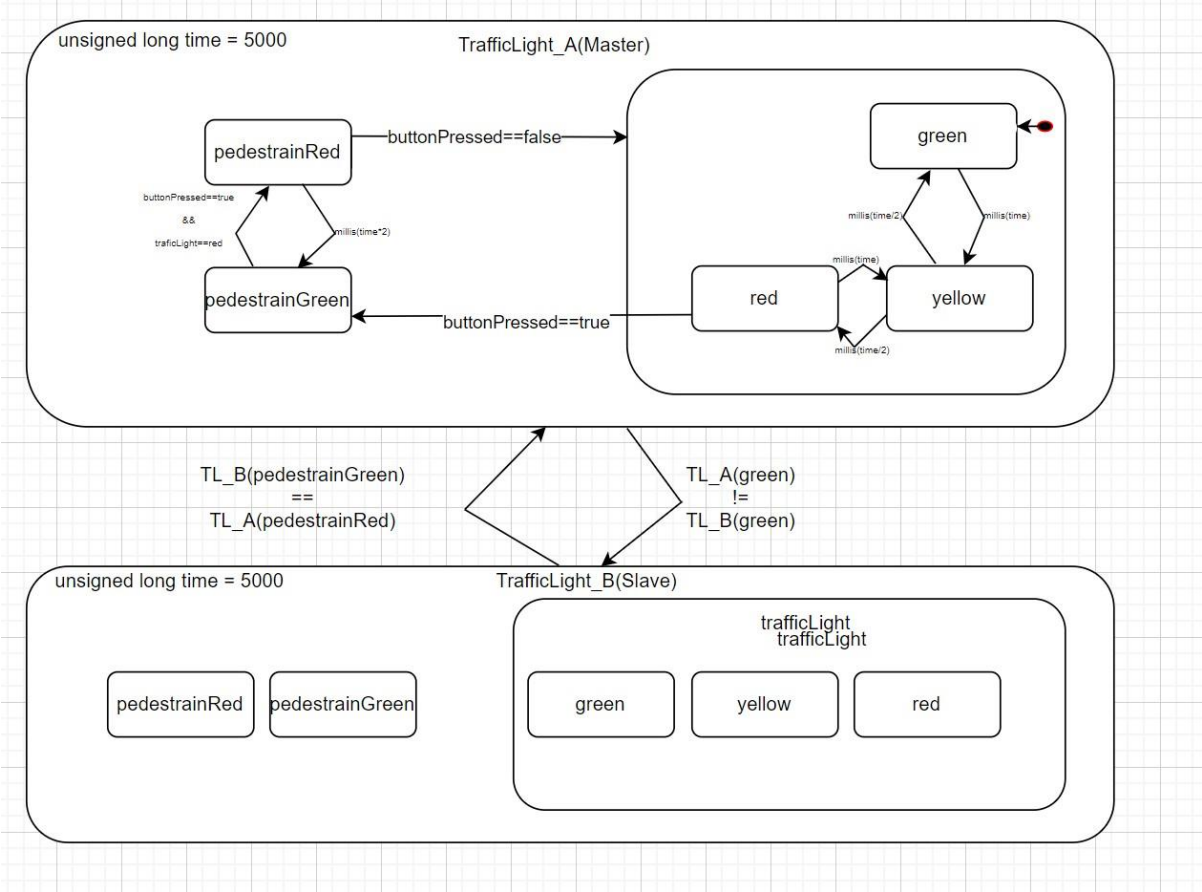


TASK 5

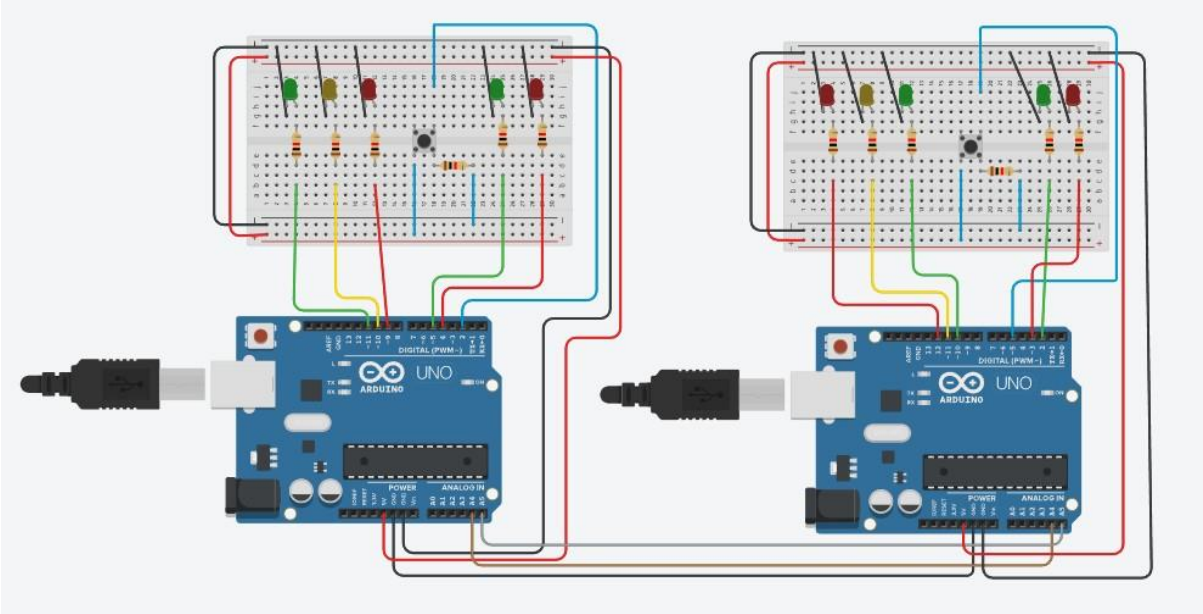
SEQUENCE DIAGRAM



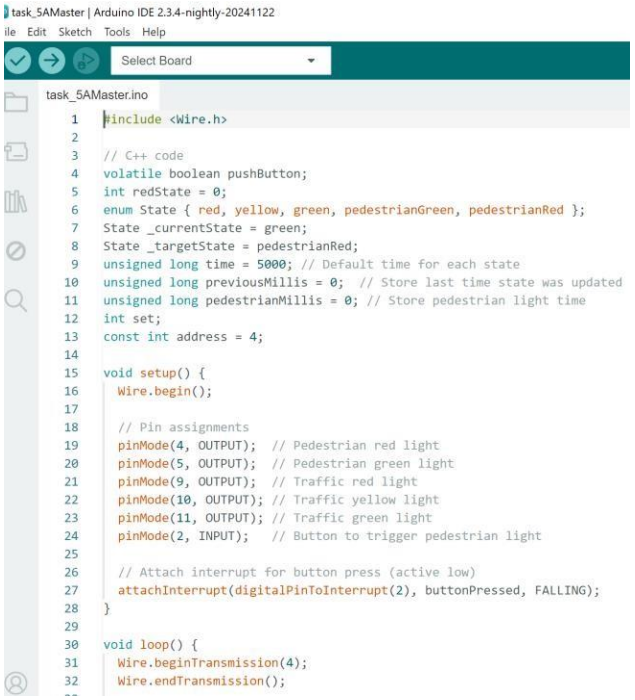
STATE MACHINE DIAGRAM



TRAFFIC LIGHT HAVING TWO MICROCONTROLLERS



MASTER -CODE




```ino
#include <Wire.h>

// C++ code
volatile boolean pushButton;
int redState = 0;
enum State { red, yellow, green, pedestrianGreen, pedestrianRed };
State _currentState = green;
State _targetState = pedestrianRed;
unsigned long time = 5000; // Default time for each state
unsigned long previousMillis = 0;  // Store last time state was updated
unsigned long pedestrianMillis = 0; // Store pedestrian light time
int set;
const int address = 4;

void setup() {
  Wire.begin();

  // Pin assignments
  pinMode(4, OUTPUT);  // Pedestrian red light
  pinMode(5, OUTPUT);  // Pedestrian green light
  pinMode(9, OUTPUT);  // Traffic red light
  pinMode(10, OUTPUT); // Traffic yellow light
  pinMode(11, OUTPUT); // Traffic green light
  pinMode(2, INPUT);   // Button to trigger pedestrian light

  // Attach interrupt for button press (active low)
  attachInterrupt(digitalPinToInterrupt(2), buttonPressed, FALLING);
}

void loop() {
  Wire.beginTransmission(4);
  Wire.endTransmission();

  unsigned long currentMillis = millis();

  if (redState == 1) {
    pedestrian(currentMillis); // Handle pedestrian light state
  } else {
    trafficLight(currentMillis); // Handle traffic light state
  }
}

void trafficLight(unsigned long currentMillis) {
  switch (_currentState) {
    case green:
      digitalWrite(11, HIGH);  // Traffic green light on
      digitalWrite(4, HIGH);   // Pedestrian red light on
      if (currentMillis - previousMillis >= time) {
        digitalWrite(11, LOW);   // Traffic green light off
        previousMillis = currentMillis;
        _currentState = yellow;  // Transition to yellow state
        set = 0;
      }
      break;

    case yellow:
      digitalWrite(10, HIGH);   // Traffic yellow light on
      if (currentMillis - previousMillis >= time / 2) {
        digitalWrite(10, LOW);    // Traffic yellow light off
        previousMillis = currentMillis;
        if (set == 0)
          _currentState = red;    // Transition to red if no pedestrian request
        else
          _currentState = green; // Transition back to green after pedestrian phase
      }
      break;

    case red:
      digitalWrite(9, HIGH);    // Traffic red light on
      if (currentMillis - previousMillis >= time) {
        digitalWrite(9, LOW);     // Traffic red light off
        previousMillis = currentMillis;
        if (redState == 0)        // If no pedestrian request
          _currentState = yellow; // Transition to yellow
        else
          _currentState = pedestrianGreen; // Transition to pedestrian green
        set = 1;
      }
      break;
  }
}

void pedestrian(unsigned long currentMillis) {
  digitalWrite(10, LOW); // Turn off traffic yellow light during pedestrian phase

  switch (_targetState) {
    case pedestrianGreen:
      digitalWrite(9, HIGH);   // Traffic red light on
      digitalWrite(5, HIGH);   // Pedestrian green light on
      if (currentMillis - pedestrianMillis >= time / 5) {
        digitalWrite(5, LOW);    // Pedestrian green light off
        pedestrianMillis = currentMillis;
        redState = 1;
        _targetState = yellow;  // Transition to yellow pedestrian light
      }
      break;

    case yellow:
      digitalWrite(9, HIGH);   // Traffic red light on
      digitalWrite(10, HIGH);  // Traffic yellow light on
      if (currentMillis - pedestrianMillis >= time / 2) {
        digitalWrite(10, LOW);   // Traffic yellow light off
        pedestrianMillis = currentMillis;
        _targetState = pedestrianRed;
      }
      break;

    case pedestrianRed:
      digitalWrite(9, LOW);    // Traffic red light off
      digitalWrite(4, HIGH);   // Pedestrian red light on (pin 4)
      if (currentMillis - pedestrianMillis >= time * 2) {
        digitalWrite(4, LOW);    // Pedestrian red light off
        redState = 0;            // Reset redState after pedestrian phase finishes
        _targetState = _currentState; // Return to the current traffic state
      }
      break;
  }
}

void buttonPressed() {
  // When the button is pressed, change the state for pedestrian light
  if (digitalRead(2) == LOW) {  // Button pressed (active LOW)
    redState = 1;               // Trigger pedestrian phase
    _targetState = pedestrianGreen; // Set to pedestrian green state
    previousMillis = millis();  // Reset the timing
  }
}
```

SLAVE-CODE

task_5BSlave.ino

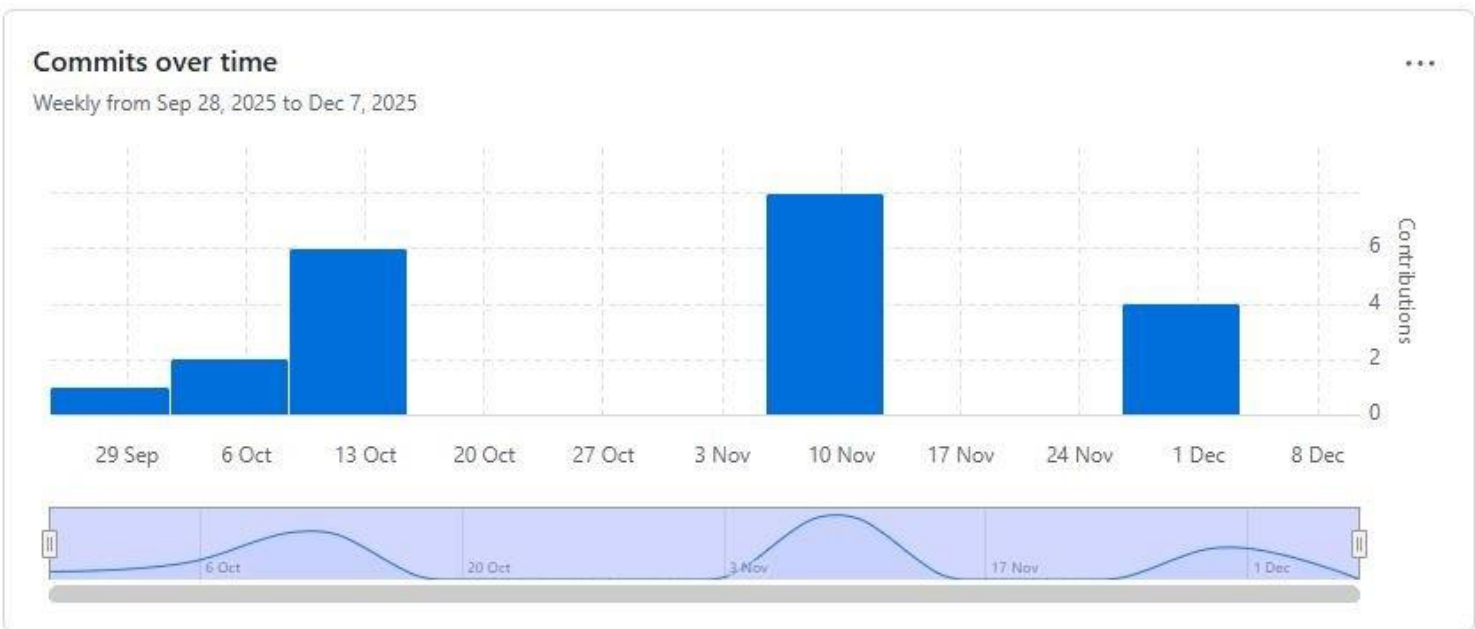
```ino
#include <Wire.h>

// C++ code
volatile boolean pushButton;
int redState = 1;  // Start in red state for traffic lights
enum State {red, yellow, green, pedestrianGreen, pedestrianRed};
State _currentState = green;
State _targetState = pedestrianGreen;
unsigned long time = 5000;
unsigned long previousMillis = 0;  // Used for timing (non-blocking)
unsigned long pedestrianMillis = 0;  // Used for pedestrian light timing
int set;
int event;

void setup() {
  Wire.begin(1);
  Wire.onReceive(receiveEvent);  // Set up Wire communication
  Serial.begin(9600);

  // Pin assignments for the lights
  pinMode(3, OUTPUT);  // Pedestrian red light
  pinMode(2, OUTPUT);  // Pedestrian green light
  pinMode(12, OUTPUT); // Traffic red light
  pinMode(11, OUTPUT); // Traffic yellow light
  pinMode(10, OUTPUT); // Traffic green light
  pinMode(5, INPUT);   // Button to trigger pedestrian light
  pinMode(4, OUTPUT);  // Traffic red light (alternate pin)

  // Attach interrupt for button press
  attachInterrupt(digitalPinToInterrupt(5), buttonPressed, RISING);
}

void loop() {
  unsigned long currentMillis = millis();  // Get current time in milliseconds

  if (redState == 0) {
    pedestrian(currentMillis);  // Handle pedestrian light logic
  }
  if (redState == 1) {
    trafficLight(currentMillis);  // Handle traffic light logic
  }
}

void trafficLight(unsigned long currentMillis) {
  switch (_currentState) {
    case green:
      digitalWrite(10, HIGH);  // Traffic green light on
      digitalWrite(12, HIGH);   // Pedestrian red light on
      if (currentMillis - previousMillis >= time) {
        digitalWrite(10, LOW);  // Traffic green light off
        previousMillis = currentMillis;
        _currentState = yellow;  // Transition to yellow
        set = 0;
      }
      break;

    case yellow:
      digitalWrite(11, HIGH);   // Traffic yellow light on
      if (currentMillis - previousMillis >= time / 2) {
        digitalWrite(11, LOW);   // Traffic yellow light off
        previousMillis = currentMillis;
        if (set == 0) {
          _currentState = red;  // Transition to red
        } else {
          _currentState = green;  // Transition to green
        }
      }
      break;

    case red:
      digitalWrite(12, HIGH);    // Traffic red light on
      if (currentMillis - previousMillis >= time) {
        digitalWrite(12, LOW);    // Traffic red light off
        previousMillis = currentMillis;
        if (redState == 1) {
          _currentState = yellow; // Transition to yellow
        } else {
          _currentState = pedestrianGreen;  // Transition to pedestrian green
        }
        set = 1;
      }
      break;
  }
}

void pedestrian(unsigned long currentMillis) {
  switch (_targetState) {
    case pedestrianGreen:
      digitalWrite(12, HIGH);   // Traffic red light on
      digitalWrite(2, HIGH);    // Pedestrian green light on
      if (currentMillis - pedestrianMillis >= time / 5) {
        digitalWrite(2, LOW);  // Pedestrian green light off
        pedestrianMillis = currentMillis;
        redState = 1;          // Set redState to 1 after pedestrian light
        _targetState = yellow; // Transition to yellow pedestrian light
      }
      break;

    case yellow:
      digitalWrite(12, HIGH);  // Traffic red light on
      digitalWrite(11, HIGH);  // Traffic yellow light on
      if (currentMillis - pedestrianMillis >= time / 2) {
        digitalWrite(11, LOW); // Traffic yellow light off
        pedestrianMillis = currentMillis;
        _targetState = pedestrianRed;
      }
      break;

    case pedestrianRed:
      digitalWrite(12, LOW);   // Traffic red light off
      digitalWrite(3, HIGH);   // Pedestrian red light on
      if (currentMillis - pedestrianMillis >= time * 2) {
        digitalWrite(3, LOW);  // Pedestrian red light off
        redState = 0;            // Reset redState after pedestrian phase
        _targetState = _currentState; // Return to the traffic state
      }
      break;
  }
}

void buttonPressed() {
  // When the button is pressed, set the pedestrian phase
  if (digitalRead(5) == HIGH) {  // Button pressed (active HIGH)
    redState = 0;  // Trigger pedestrian phase
    _targetState = pedestrianGreen;  // Set to pedestrian green state
    previousMillis = millis();  // Reset the timing
  }
}

void receiveEvent(int event) {
  // Handle Wire communication event and set redState
  redState = Wire.read();
}
```

GIT USAGE

main · 1 Branch · 0 Tags · Go to file t · Add file · Code

SyedMuhammadAbisRizvi Add files via upload — 55c74bb · 5 days ago — 22 Commits

Task 2	Add files via upload	2 months ago
Task 3	Add files via upload	last month
Task 5	Add files via upload	2 weeks ago
Task1	made some changes	last month
README.md	Initial commit	2 months ago
Traffic light system test.ms14	Add files via upload	2 months ago
Traffic light.pptx	Add files via upload	5 days ago



THANK YOU.