

EMBEDDED SYSTEMS
TRAFFIC LIGHT

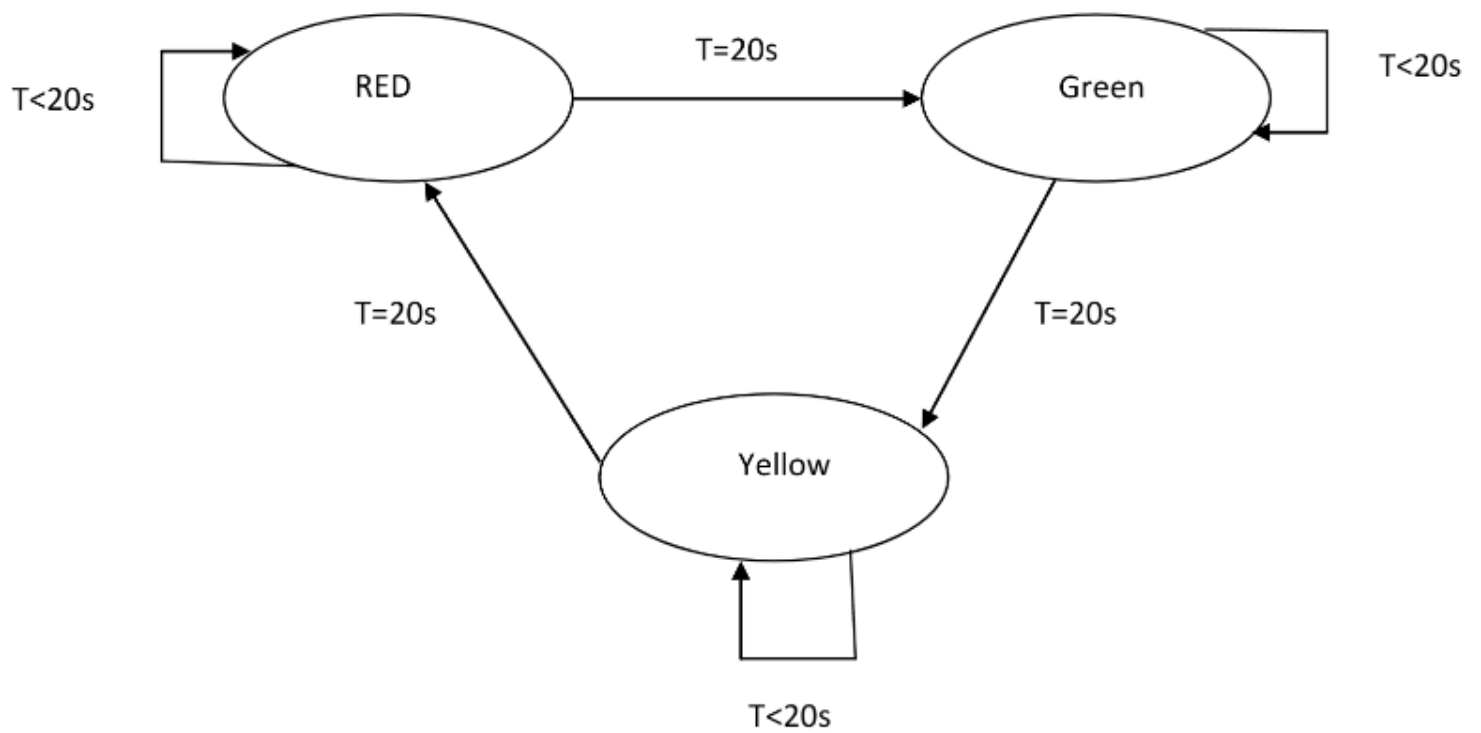
BY

SYED MUHAMMAD ABIS RIZVI

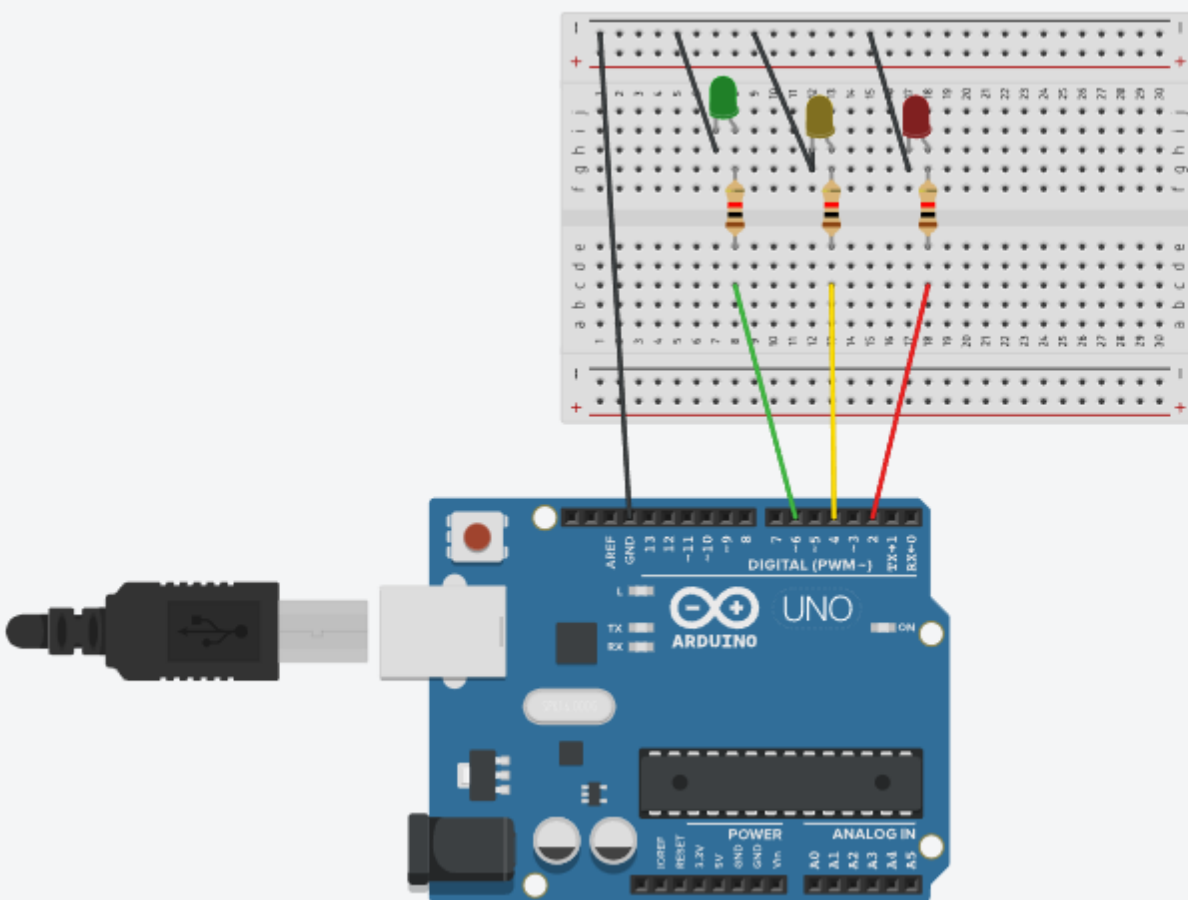
IBRAHIM ABDEMALEK

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

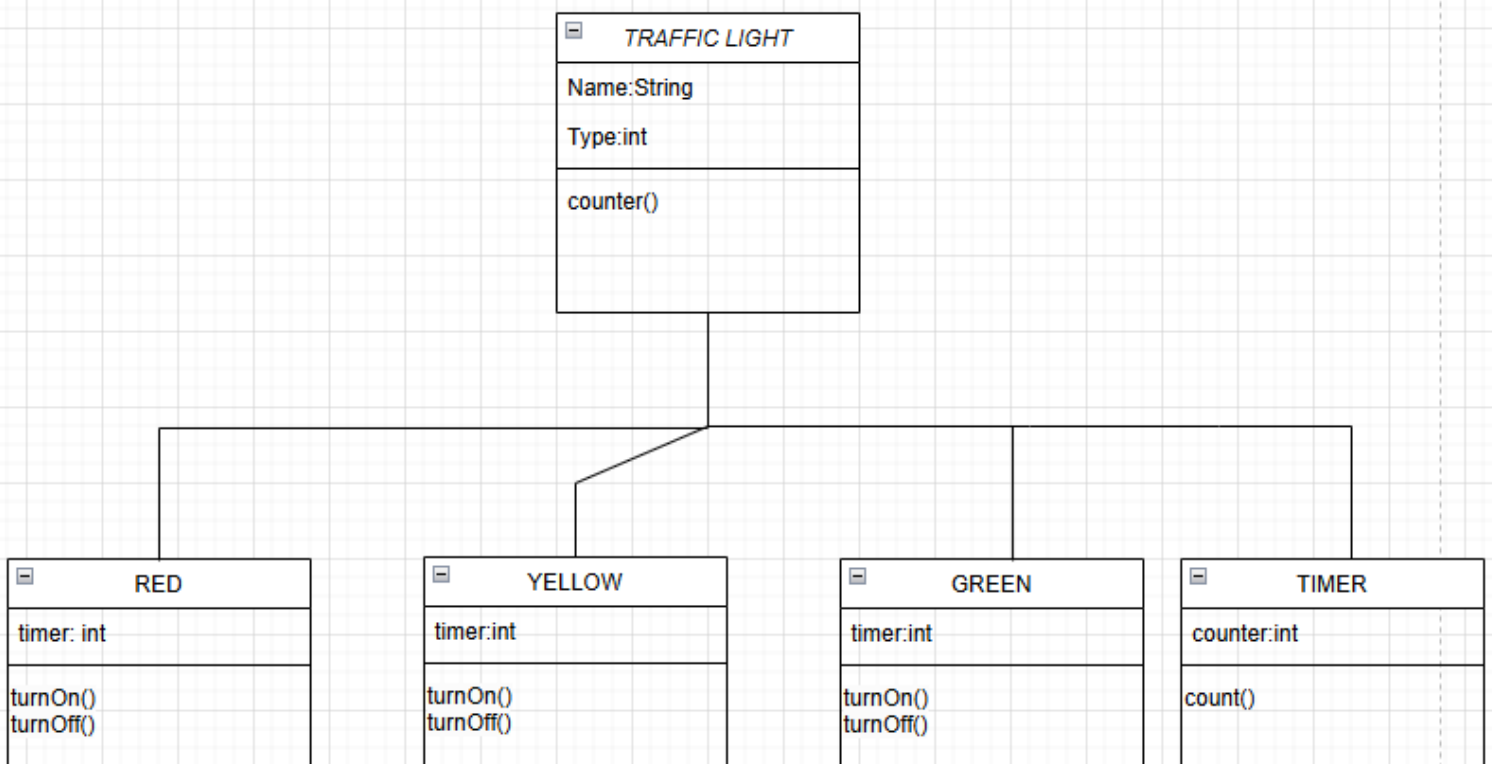
STATE MACHINE DIAGRAM TASK 1



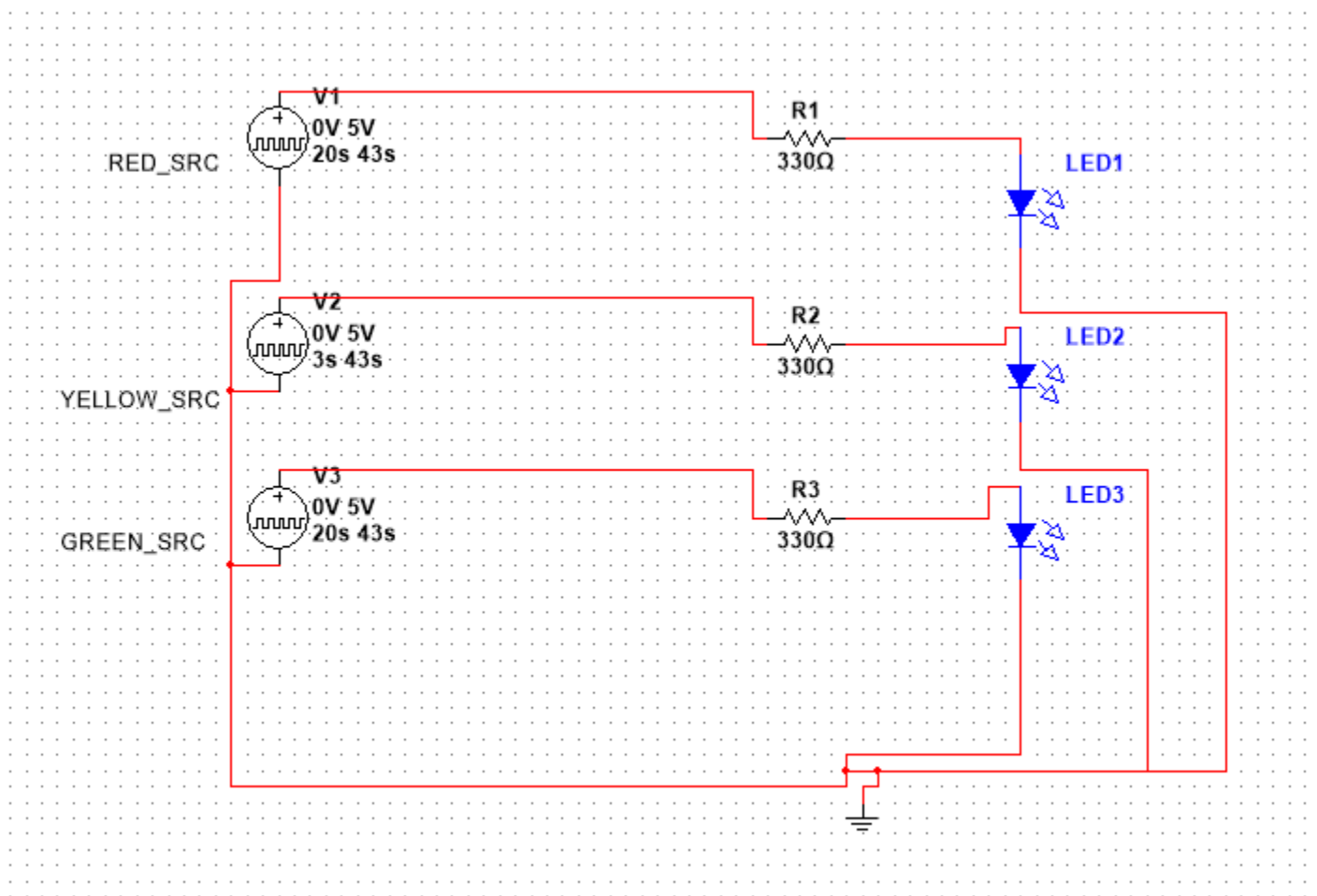
TASK 1



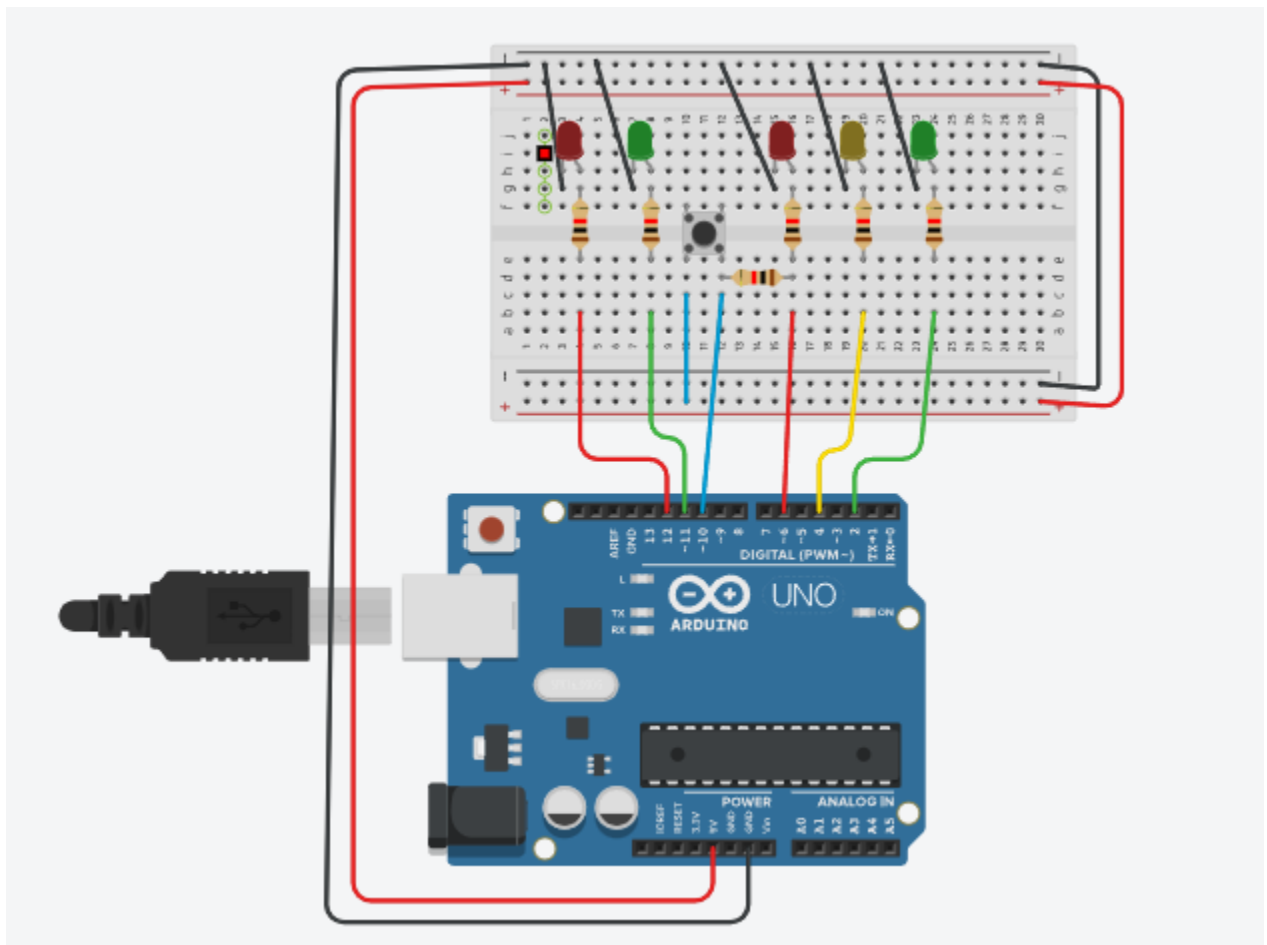
CLASS DIAGRAM TASK 1



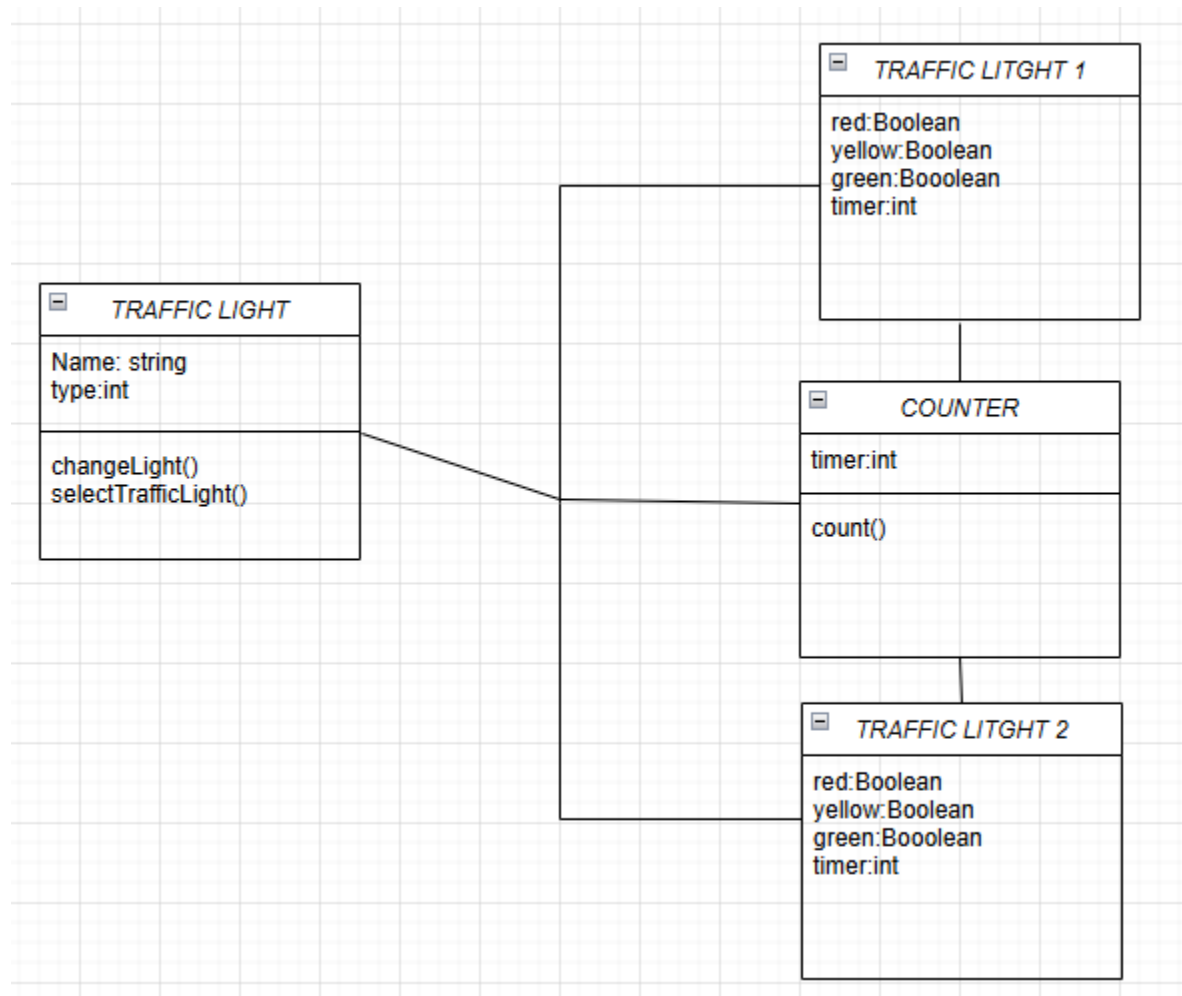
MULTI-SIM IMPLEMENTATION



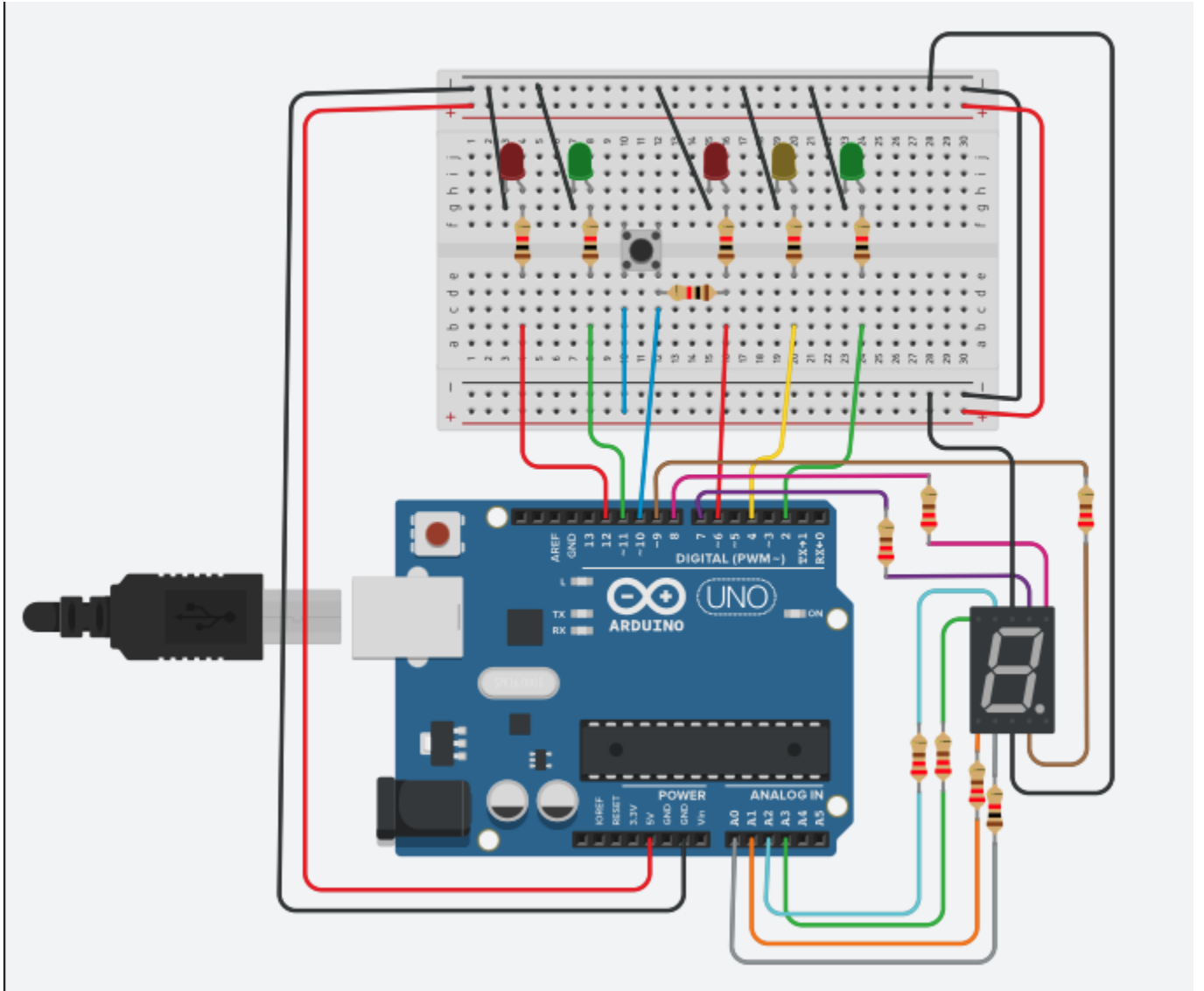
TASK 2



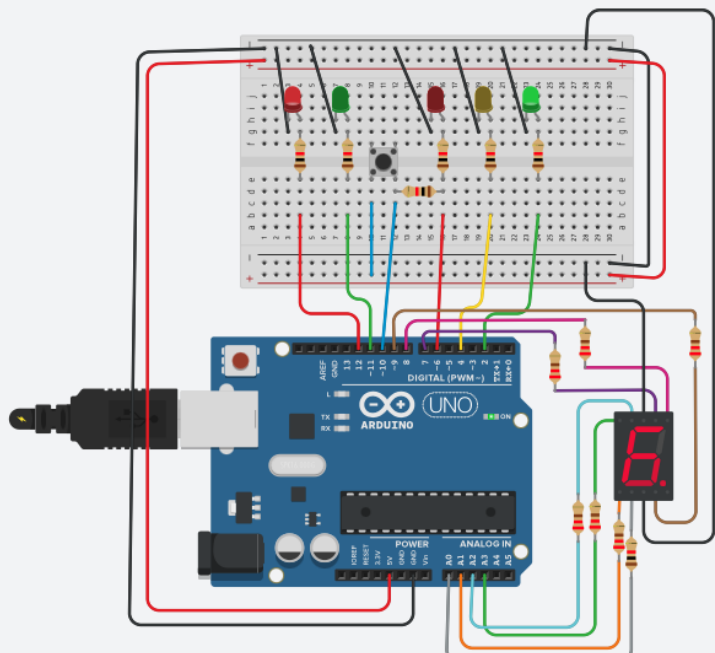
CLASS DIAGRAM TASK 2



TASK 2 WITH SEVEN SEGMENT



SEGMENT IMPLEMENTATION CODE



Simulator time: 00:00:03

Code Stop Simulation Send To

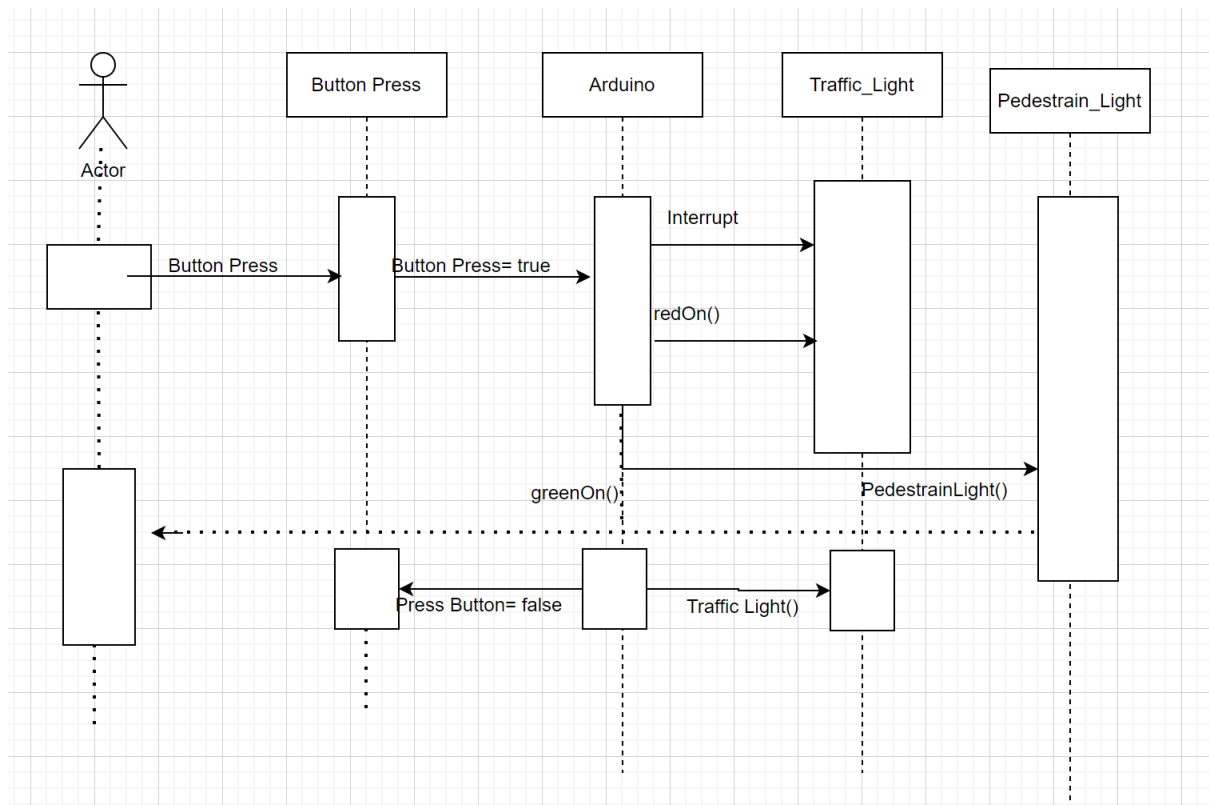
1 (Arduino Uno R3)

```
1 // ----- ENUM & VARIABLES -----
2 enum Status {green, yellow, red, pgreen, pred};
3 Status status = green;
4
5 unsigned long lastMillis = 0;
6 unsigned long interval = 10000;
7 bool pedestrianRequest = false;
8
9 // ----- 7-SEGMENT SETUP -----
10 int segPins[7] = {7, 8, 9, A0, A1, A2, A3}; // Segment order: a,b,c,d,e,f,g
11
12 byte digits[10][7] = {
13   {1,1,1,1,1,0}, {0,1,0,0,0,0}, {1,1,0,1,1,0,1},
14   {1,1,1,1,0,0,1}, {0,1,1,0,0,1,1}, {1,0,1,1,0,1,1},
15   {1,0,1,1,1,1,1}, {1,1,1,0,0,0,0}, {1,1,1,1,1,1,1},
16   {1,1,1,1,0,1,1}
17 };
18
19 // ----- PROTOTYPES -----
20 void handleTrafficLights(unsigned long currentMillis);
21 void handlePedestrianLights(unsigned long currentMillis);
22 void displayDigit(int num);
23 void clearDisplay();
24 void displayCountdown(unsigned long currentMillis);
25
26 // ----- SETUP -----
27 void setup() {
28   pinMode(12, OUTPUT); // Ped red
29   pinMode(11, OUTPUT); // Ped green
30   pinMode(10, INPUT); // Button
31   pinMode(6, OUTPUT); // Traffic red
32   pinMode(4, OUTPUT); // Traffic yellow
33   pinMode(2, OUTPUT); // Traffic green
34
35   for (int i = 0; i < 7; i++) pinMode(segPins[i], OUTPUT);
36
37   // Initial states: traffic green, ped red
38   digitalWrite(2, HIGH);
39   digitalWrite(4, LOW);
40   digitalWrite(6, LOW);
41   digitalWrite(11, LOW);
42   digitalWrite(12, HIGH);
43
44   lastMillis = millis();
45 }
46
47 // ----- LOOP -----
48 void loop() {
49   unsigned long currentMillis = millis();
50   int buttonState = digitalRead(10);
51
52   static unsigned long lastButtonPress = 0;
53   if (buttonState == HIGH && !pedestrianRequest && status != pgreen &&
54       currentMillis - lastButtonPress > 500) {
55     pedestrianRequest = true;
56     lastButtonPress = currentMillis;
57     lastMillis = currentMillis;
58     status = pgreen;
59     interval = 3200;
60   }
61 }
```

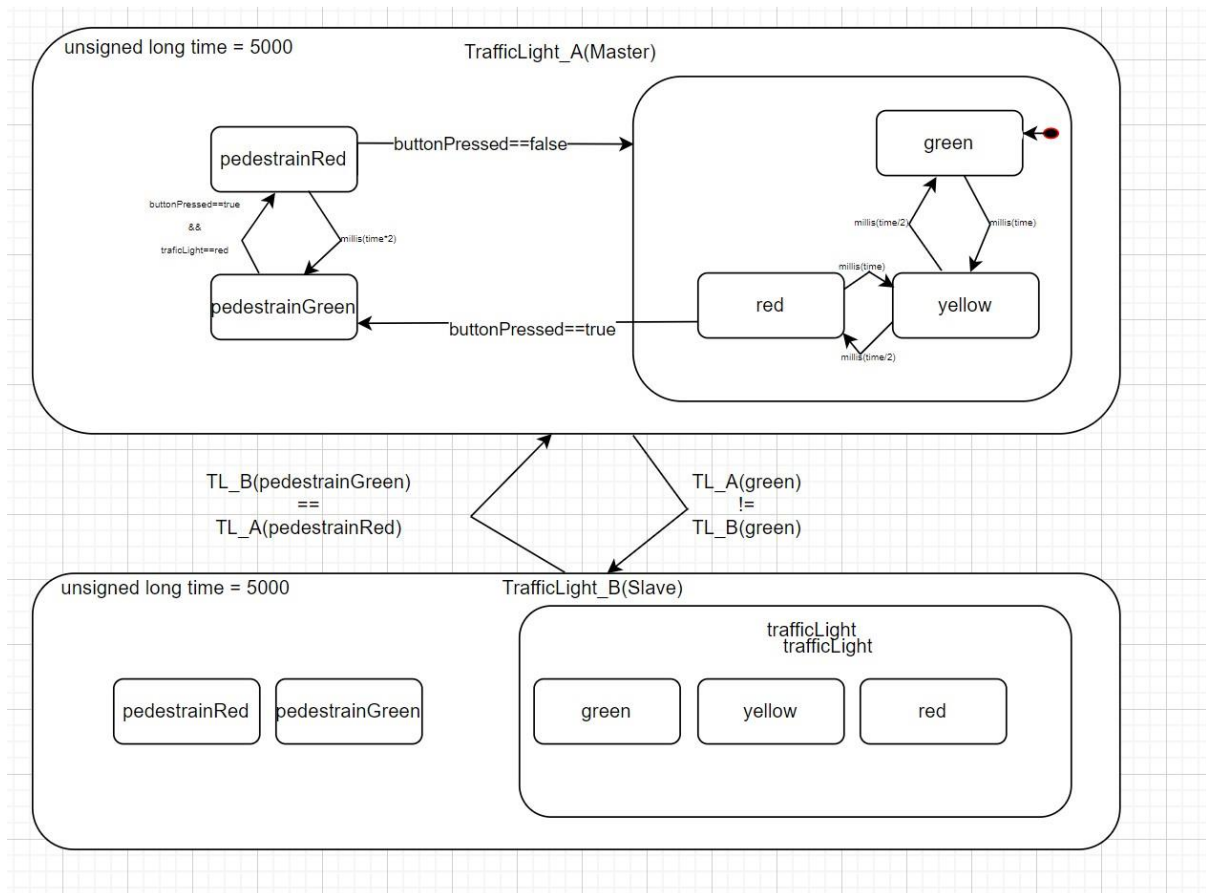
Serial Monitor

TASK 5

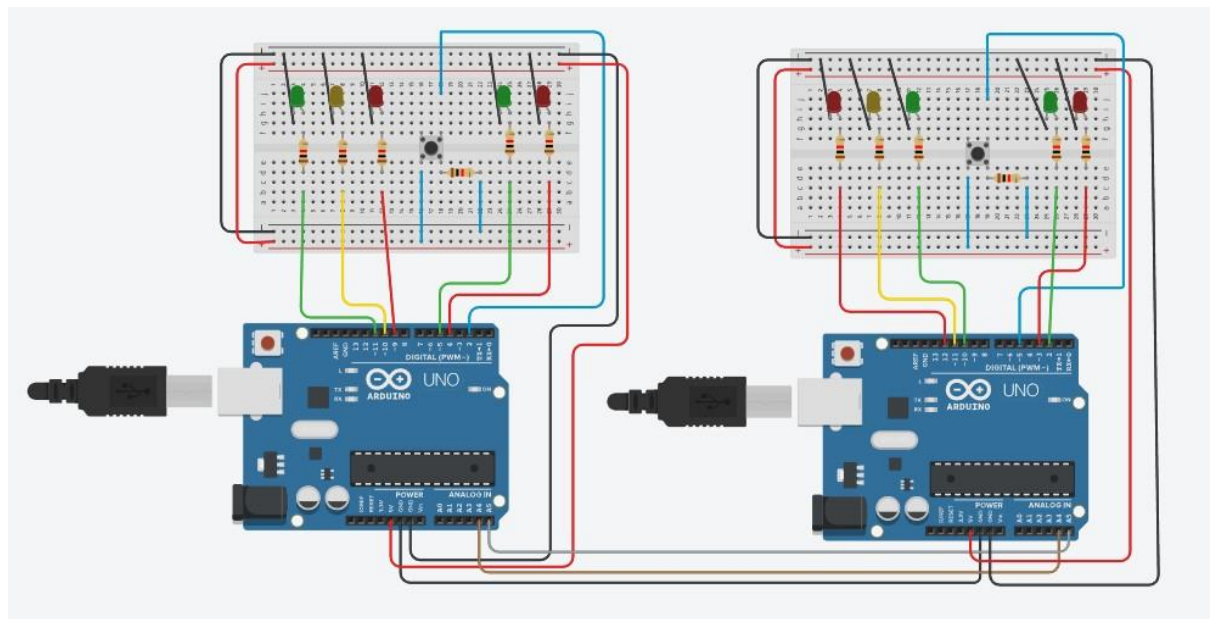
SEQUENCE DIAGRAM



STATE MACHINE DIAGRAM



TRAFFIC LIGHT HAVING TWO MICROCONTROLLERS



MASTER -CODE

```
task_5AMaster | Arduino IDE 2.3.4-nightly-20241122
File Edit Sketch Tools Help

task_5AMaster.ino
1 #include <Wire.h>
2
3 // C++ code
4 volatile boolean pushButton;
5 int redState = 0;
6 enum State { red, yellow, green, pedestrianGreen, pedestrianRed };
7 State _currentState = green;
8 State _targetState = pedestrianRed;
9 unsigned long time = 5000; // Default time for each state
10 unsigned long previousMillis = 0; // Store last time state was updated
11 unsigned long pedestrianMillis = 0; // Store pedestrian light time
12 int set;
13 const int address = 4;
14
15 void setup() {
16   Wire.begin();
17
18   // Pin assignments
19   pinMode(4, OUTPUT); // Pedestrian red light
20   pinMode(5, OUTPUT); // Pedestrian green light
21   pinMode(9, OUTPUT); // Traffic red light
22   pinMode(10, OUTPUT); // Traffic yellow light
23   pinMode(11, OUTPUT); // Traffic green light
24   pinMode(2, INPUT); // Button to trigger pedestrian light
25
26   // Attach interrupt for button press (active low)
27   attachInterrupt(digitalPinToInterrupt(2), buttonPressed, FALLING);
28 }
29
30 void loop() {
31   Wire.beginTransmission(4);
32   Wire.endTransmission();
33
34   _currentState = red; // Transition to red if no pedestrian request
35   else
36     _currentState = green; // Transition back to green after pedestrian phase
37   }
38   break;
39
40   case red:
41     digitalWrite(9, HIGH); // Traffic red light on
42     if (currentMillis - previousMillis >= time) {
43       digitalWrite(9, LOW); // Traffic red light off
44       previousMillis = currentMillis;
45       if (redState == 0) // If no pedestrian request
46         _currentState = yellow; // Transition to yellow
47       else
48         _currentState = pedestrianGreen; // Transition to pedestrian green
49       set = 1;
50     }
51     break;
52   }
53 }
54
55 void pedestrian(unsigned long currentMillis) {
56   digitalWrite(10, LOW); // Turn off traffic yellow light during pedestrian phase
57
58   switch (_targetState) {
59     case pedestrianGreen:
60       digitalWrite(9, HIGH); // Traffic red light on
61       digitalWrite(5, HIGH); // Pedestrian green light on
62       if (currentMillis - pedestrianMillis >= time / 5) {
63         digitalWrite(5, LOW); // Pedestrian green light off
64         pedestrianMillis = currentMillis;
65         redState = 1;
66         _targetState = yellow; // Transition to yellow pedestrian light
67       }
68     }
69   }
70 }
71
72 void trafficLight(unsigned long currentMillis) {
73   switch (_currentState) {
74     case green:
75       digitalWrite(11, HIGH); // Traffic green light on
76       digitalWrite(4, HIGH); // Pedestrian red light on
77       if (currentMillis - previousMillis >= time) {
78         digitalWrite(11, LOW); // Traffic green light off
79         previousMillis = currentMillis;
80         _currentState = yellow; // Transition to yellow state
81         set = 0;
82       }
83       break;
84     case yellow:
85       digitalWrite(10, HIGH); // Traffic yellow light on
86       if (currentMillis - previousMillis >= time / 2) {
87         digitalWrite(10, LOW); // Traffic yellow light off
88         previousMillis = currentMillis;
89         if (set == 0)
90           _currentState = red; // Transition to red if no pedestrian request
91       }
92     }
93   }
94
95   _targetState = yellow; // Transition to yellow pedestrian light
96   }
97   break;
98
99   case yellow:
100     digitalWrite(9, HIGH); // Traffic red light on
101     digitalWrite(10, HIGH); // Traffic yellow light on
102     if (currentMillis - pedestrianMillis >= time / 2) {
103       digitalWrite(10, LOW); // Traffic yellow light off
104       pedestrianMillis = currentMillis;
105       _targetState = pedestrianRed;
106     }
107     break;
108   case pedestrianRed:
109     digitalWrite(9, LOW); // Traffic red light off
110     digitalWrite(4, HIGH); // Pedestrian red light on (pin 4)
111     if (currentMillis - pedestrianMillis >= time * 2) {
112       digitalWrite(4, LOW); // Pedestrian red light off
113       redState = 0; // Reset redState after pedestrian phase finishes
114       _targetState = _currentState; // Return to the current traffic state
115     }
116     break;
117   }
118 }
119
120 void buttonPressed() {
121   // When the button is pressed, change the state for pedestrian light
122   if (digitalRead(2) == LOW) { // Button pressed (active LOW)
123     redState = 1; // Trigger pedestrian phase
124     _targetState = pedestrianGreen; // Set to pedestrian green state
125     previousMillis = millis(); // Reset the timing
126   }
127 }
128 }
```

SLAVE-CODE

task_5BSlave.ino

```
1 #include <Wire.h>
2
3 // C++ code
4 volatile boolean pushButton;
5 int redState = 1; // Start in red state for traffic lights
6 enum State {red, yellow, green, pedestrianGreen, pedestrianRed};
7 State _currentState = green;
8 State _targetState = pedestrianGreen;
9 unsigned long time = 5000;
10 unsigned long previousMillis = 0; // Used for timing (non-blocking)
11 unsigned long pedestrianMillis = 0; // Used for pedestrian light timing
12 int set;
13 int event;
14
15 void setup() {
16   Wire.begin(1);
17   Wire.onReceive(receiveEvent); // Set up Wire communication
18   Serial.begin(9600);
19
20   // Pin assignments for the lights
21   pinMode(3, OUTPUT); // Pedestrian red light
22   pinMode(2, OUTPUT); // Pedestrian green light
23   pinMode(12, OUTPUT); // Traffic red light
24   pinMode(11, OUTPUT); // Traffic yellow light
25   pinMode(10, OUTPUT); // Traffic green light
26   pinMode(5, INPUT); // Button to trigger pedestrian light
27   pinMode(4, OUTPUT); // Traffic red light (alternate pin)
28
29   // Attach interrupt for button press
30   attachInterrupt(digitalPinToInterrupt(5), buttonPressed, RISING);
31 }
32
33 void loop() {
34   unsigned long currentMillis = millis(); // Get current time in milliseconds
35
36   if (redState == 0) {
37     pedestrian(currentMillis); // Handle pedestrian light logic
38
39     digitalWrite(12, LOW); // Traffic red light off
40     previousMillis = currentMillis;
41     if (redState == 1) {
42       _currentState = yellow; // Transition to yellow
43     } else {
44       _currentState = pedestrianGreen; // Transition to pedestrian green
45     }
46     set = 1;
47   }
48   break;
49 }
50
51 void pedestrian(unsigned long currentMillis) {
52   switch (_targetState) {
53     case pedestrianGreen:
54       digitalWrite(12, HIGH); // Traffic red light on
55       digitalWrite(2, HIGH); // Pedestrian green light on
56       if (currentMillis - pedestrianMillis >= time / 5) {
57         digitalWrite(2, LOW); // Pedestrian green light off
58         pedestrianMillis = currentMillis;
59         redState = 1; // Set redState to 1 after pedestrian light
60         _targetState = yellow; // Transition to yellow pedestrian light
61       }
62       break;
63     case yellow:
64       digitalWrite(12, HIGH); // Traffic red light on
65       digitalWrite(11, HIGH); // Traffic yellow light on
66       if (currentMillis - pedestrianMillis >= time / 2) {
67         digitalWrite(11, LOW); // Traffic yellow light off
68         pedestrianMillis = currentMillis;
69         _targetState = pedestrianRed;
70       }
71       break;
72     case pedestrianRed:
73       digitalWrite(12, LOW); // Traffic red light off
74       digitalWrite(3, HIGH); // Pedestrian red light on
75       if (currentMillis - pedestrianMillis >= time * 2) {
76         digitalWrite(3, LOW); // Pedestrian red light off
77         redState = 0; // Reset redState after pedestrian phase
78         _targetState = _currentState; // Return to the traffic state
79       }
80       break;
81   }
82 }
83
84 void buttonPressed() {
85   // When the button is pressed, set the pedestrian phase
86   if (digitalRead(5) == HIGH) { // Button pressed (active HIGH)
87     redState = 0; // Trigger pedestrian phase
88     _targetState = pedestrianGreen; // Set to pedestrian green state
89     previousMillis = millis(); // Reset the timing
90   }
91 }
92
93 void receiveEvent(int event) {
94   // Handle Wire communication event and set redState
95   redState = Wire.read();
96 }
```


GIT USAGE









main

1 Branch 0 Tags

Go to file

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

Commits over time

Weekly from Sep 28, 2025 to Dec 7, 2025



 SyedMuhammadAbisRizvi #1

18 commits 736 ++ 181 --



 ibrahimabdelmalek31 #2

3 commits 180 ++ 0 --



THANK YOU.