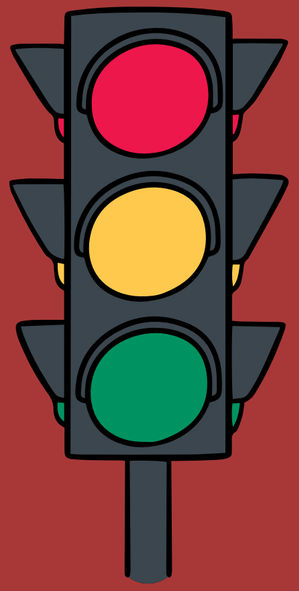
Traffic Light System

Documentation of Task 1-5



Traffic_Light

Public

Pin | Unwatch 1 | Fork 0 | Star 0

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

moaz-chowdhury first commit	2bbd325 · 1 hour ago	1 Commit
Task1	first commit	1 hour ago
Task2	first commit	1 hour ago
Task3	first commit	1 hour ago
Task4	first commit	1 hour ago
Task5	first commit	1 hour ago

README

Add a README

Help people interested in this repository understand your project by adding a README.

Add a README

About

No description, website, or topics provided.

- Activity
- 0 stars
- 1 watching
- 0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Languages

C++ 77.7% Mermaid 22.3%

Suggested workflows

Files

main

Go to file t

- Task1
  - Code
  - .$Task 1.drawio.bkp
  - Task_01.drawio
  - TrafficLight_ClassDiagram_01-2...
  - TrafficLight_ClassDiagram_01.m...
  - state_machine_01.png
- Task2
  - Code
  - StateMachine_02.drawio
  - TrafficLight_ClassDiagram.png
  - TrafficLight_ClassDiagram_02.m...
  - TrafficLight_SequenceDiagram....
  - TrafficLight_SequenceDiagram.p...
  - state_machine_diagram.png
- Task3
  - Task1_implementation.pdf

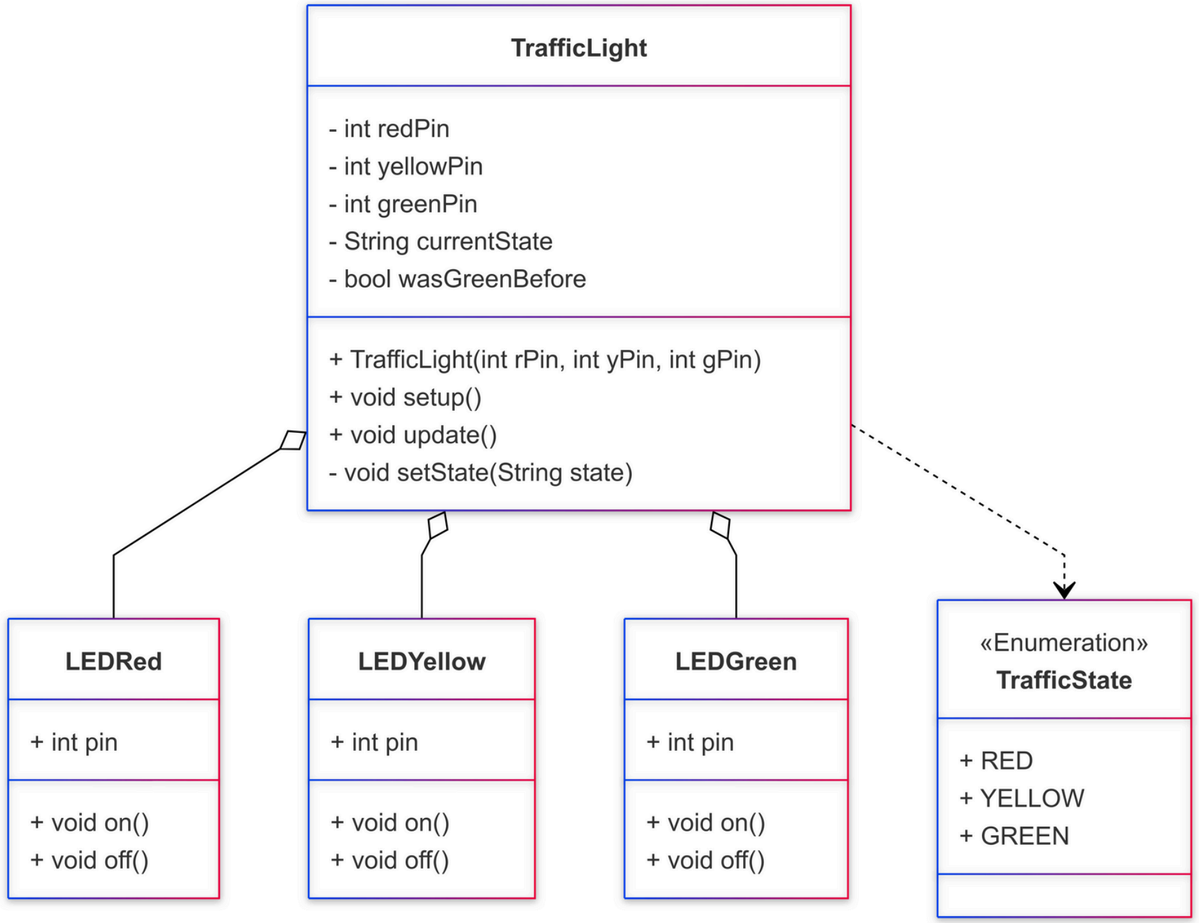
Traffic_Light / Task1 /

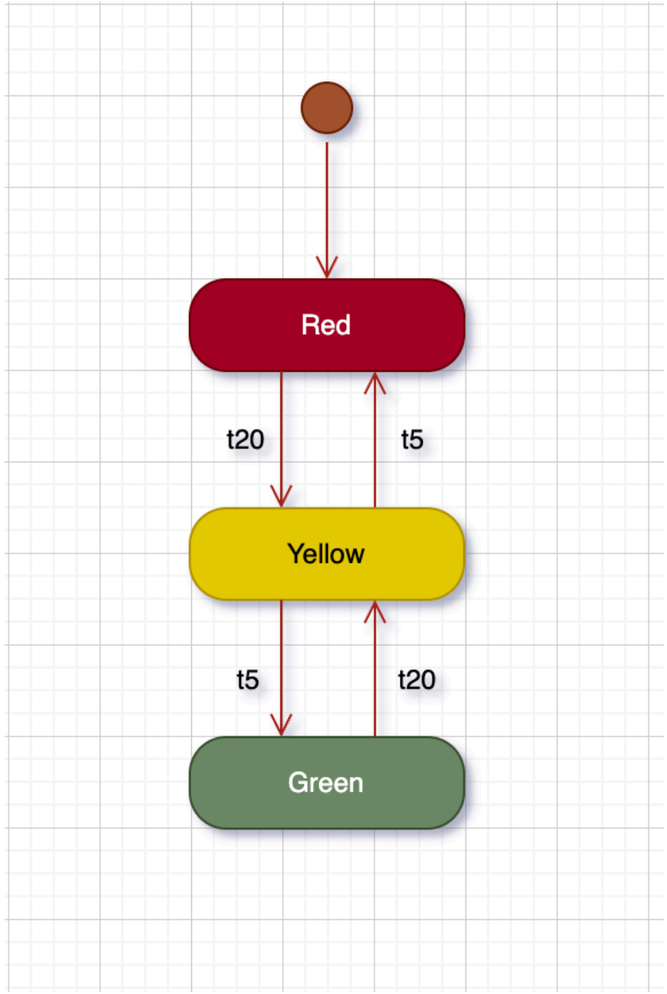
Add file

moaz-chowdhury first commit 2bbd325 · 1 hour ago History

Name	Last commit message	Last commit date
..		
Code	first commit	1 hour ago
.$Task 1.drawio.bkp	first commit	1 hour ago
Task_01.drawio	first commit	1 hour ago
TrafficLight_ClassDiagram_01-2024-11-26-154715.png	first commit	1 hour ago
TrafficLight_ClassDiagram_01.mmd	first commit	1 hour ago
state_machine_01.png	first commit	1 hour ago

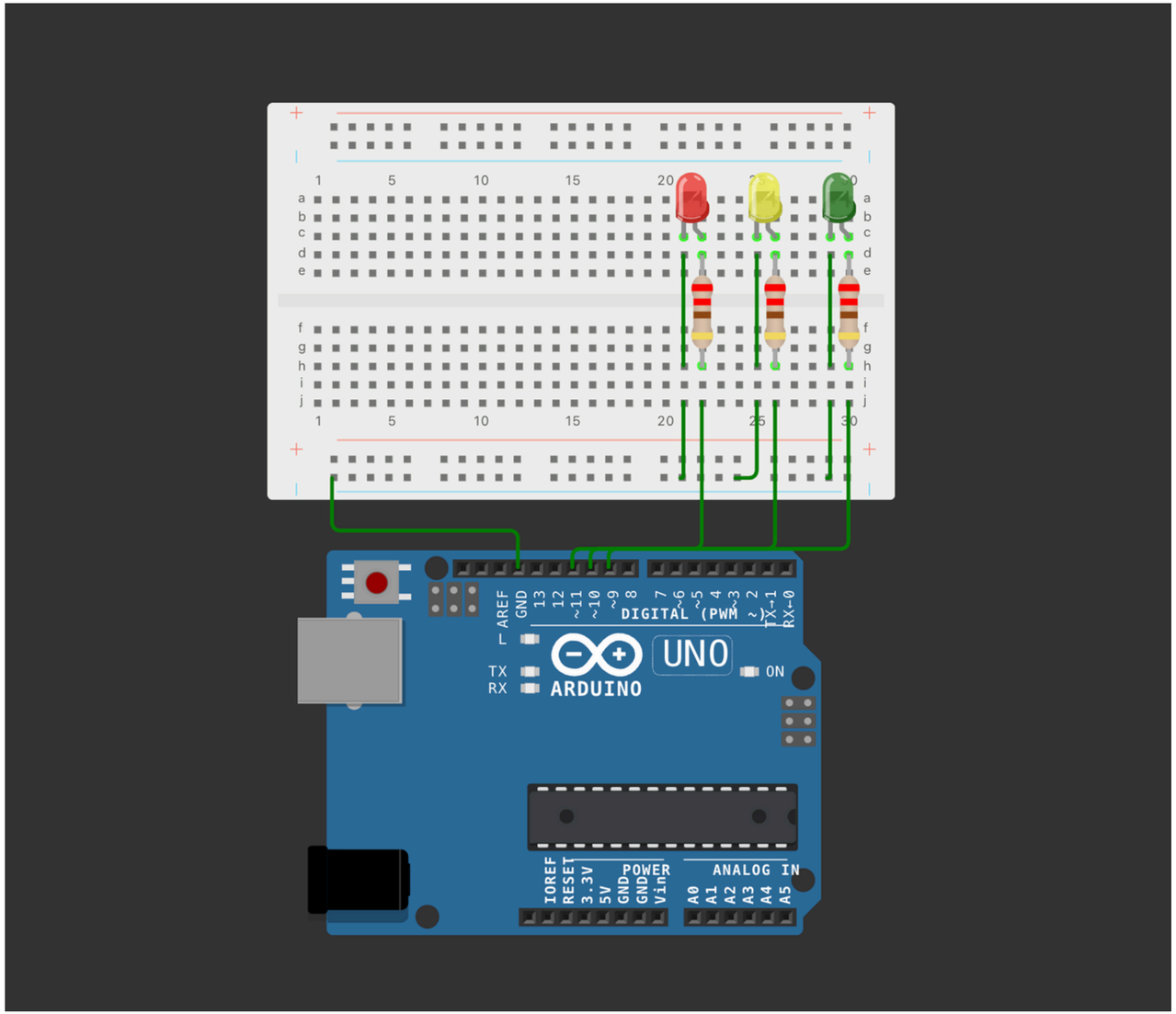
Task-01


TrafficLight
- int redPin
- int yellowPin
- int greenPin
- String currentState
- bool wasGreenBefore
+ TrafficLight(int rPin, int yPin, int gPin)
+ void setup()
+ void update()
- void setState(String state)
LEDRed
+ int pin
+ void on()
+ void off()
LEDYellow
+ int pin
+ void on()
+ void off()
LEDGreen
+ int pin
+ void on()
+ void off()
«Enumeration»
TrafficState
+ RED
+ YELLOW
+ GREEN

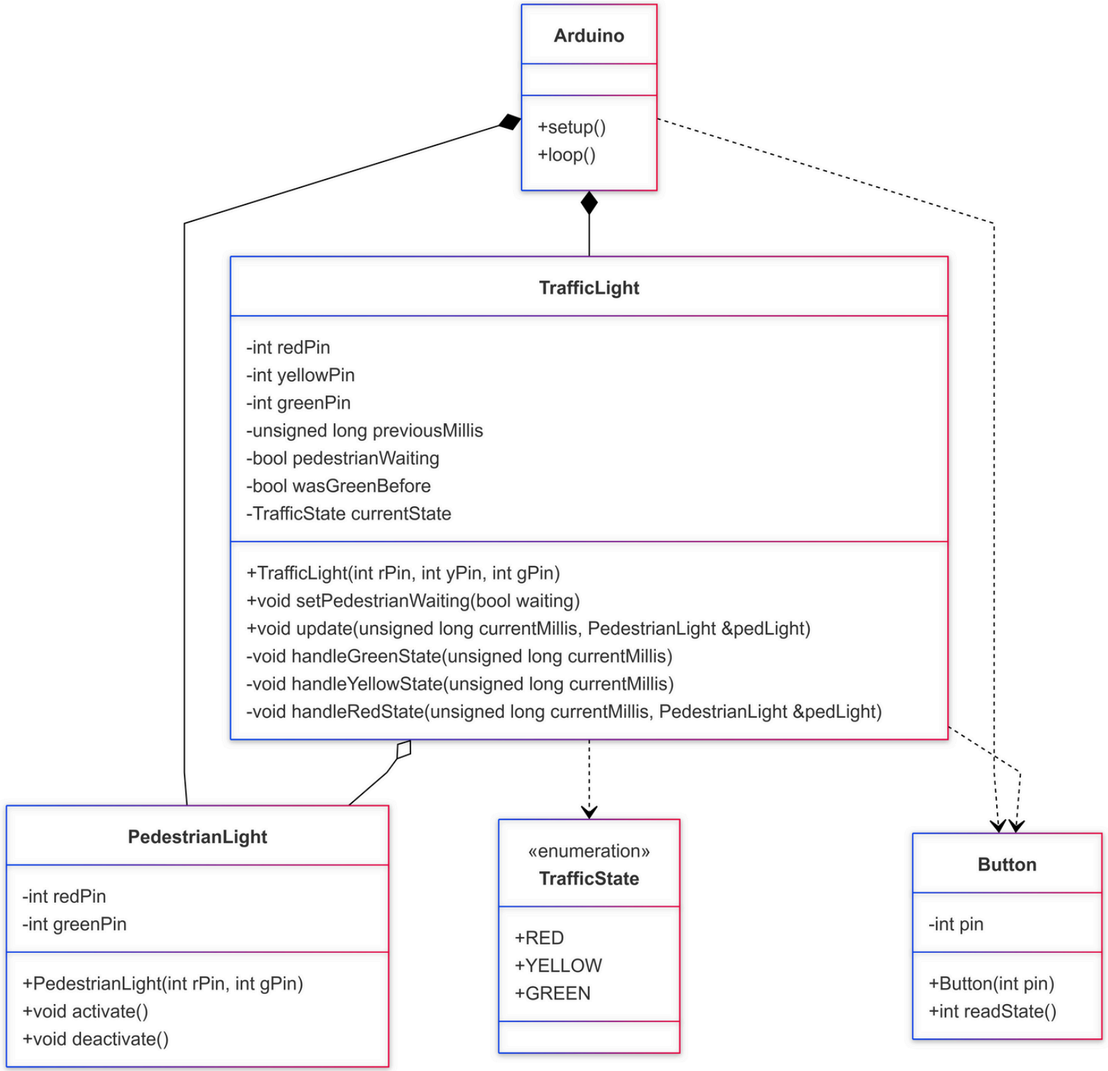

Red
t20
t5
Yellow
t5
t20
Green

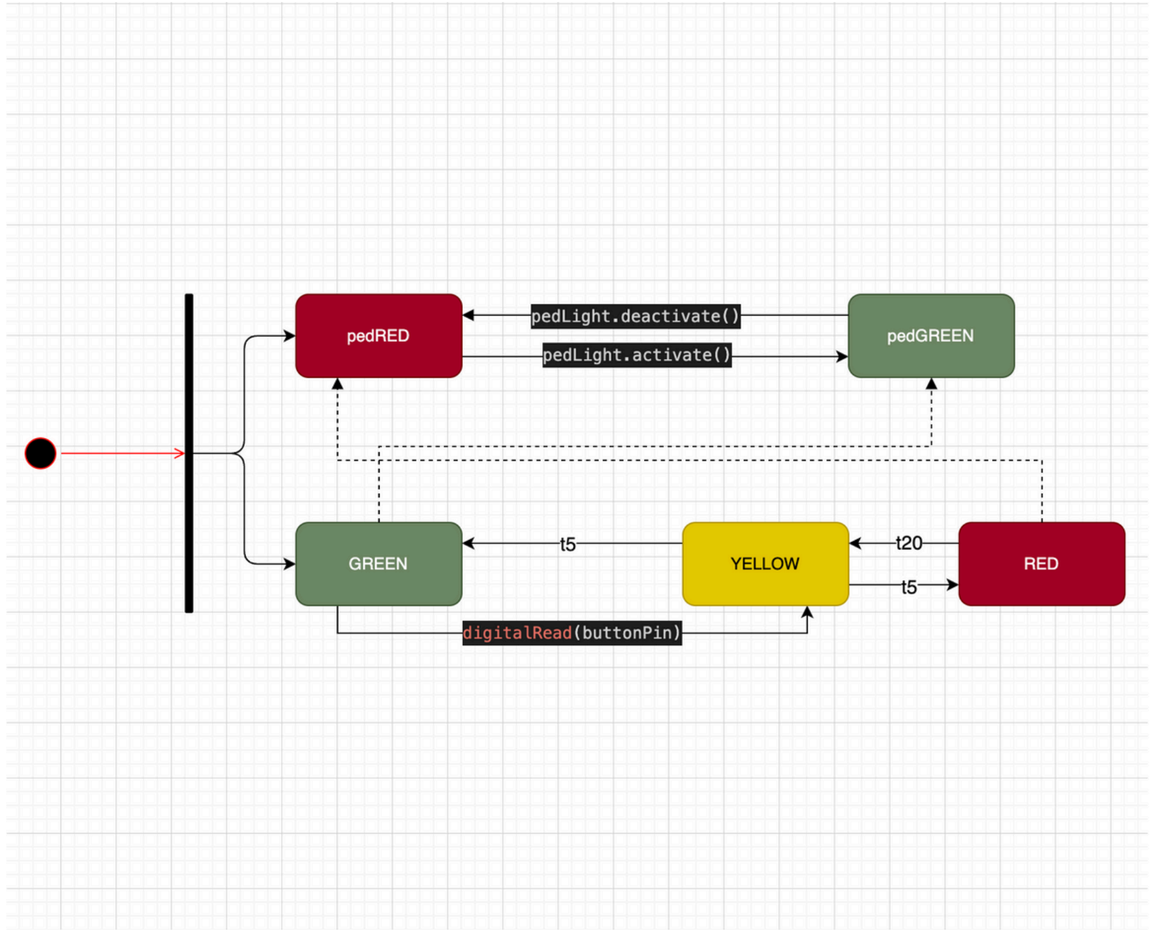
sketch.ino diagram.json Library Manager

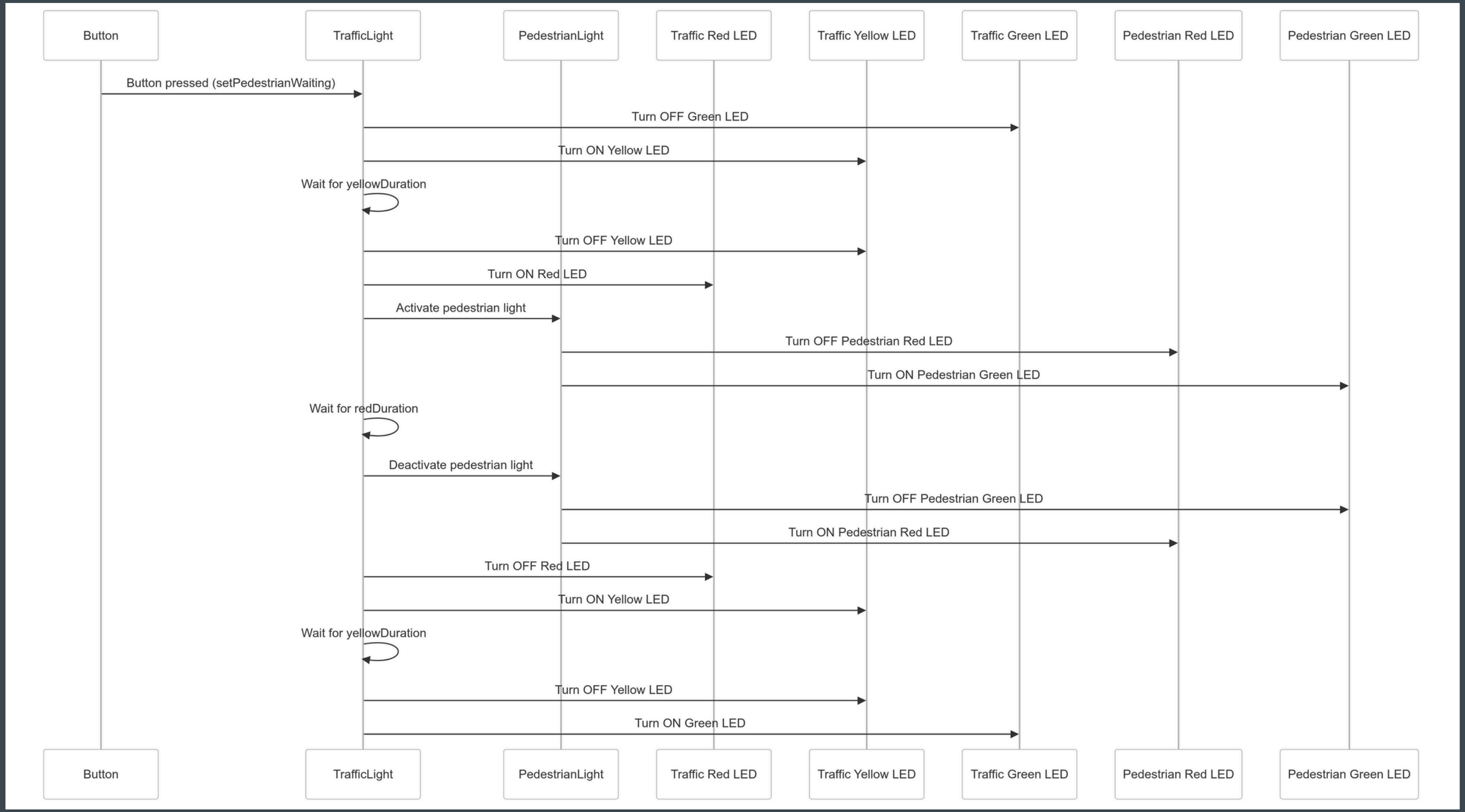
```
const int redPin = 9;
const int yellowPin = 10;
const int greenPin = 11;

enum TrafficState { RED, YELLOW, GREEN };

class TrafficLight {
private:
    int redPin;
    int yellowPin;
    int greenPin;
    TrafficState currentState;
    bool wasGreenBefore;

public:
    TrafficLight(int rPin, int yPin, int gPin)
        : redPin(rPin),
        yellowPin(yPin),
        greenPin(gPin),
        currentState(RED),
        wasGreenBefore(false) {}

    void setup() {
        pinMode(redPin, OUTPUT);
        pinMode(yellowPin, OUTPUT);
        pinMode(greenPin, OUTPUT);
        setState(RED);  // Start with the Red light on
    }

    void update() {
        switch (currentState) {
            case RED:
                setState(RED);
                delay(20000); // Stay Red for 20 seconds
                setState(YELLOW);
                wasGreenBefore = false; // Reset flag
                break;

            case YELLOW:
                setState(YELLOW);
                delay(5000); // Stay Yellow for 5 seconds
                if (wasGreenBefore) {
                    setState(RED);  // Go back to Red after Green
                } else {
                    setState(GREEN); // Go to Green after Red
                    wasGreenBefore = true; // Set flag for next Yell
                }
                break;

            case GREEN:
                setState(GREEN);
                delay(20000); // Stay Green for 20 seconds
                setState(YELLOW); // Transition to Yellow
                break;
        }
    }

private:
    void setState(TrafficState state) {
        // Turn off all lights initially
        digitalWrite(redPin, LOW);
        digitalWrite(yellowPin, LOW);
        digitalWrite(greenPin, LOW);

        // Turn on the appropriate light
        currentState = state;
        switch (state) {
            case RED:
                digitalWrite(redPin, HIGH);
                break;
            case YELLOW:
                digitalWrite(yellowPin, HIGH);
                break;
            case GREEN:
                digitalWrite(greenPin, HIGH);
                break;
        }
    }
};

// Instantiate the TrafficLight object
TrafficLight trafficLight(redPin, yellowPin, greenPin);

void setup() {
    trafficLight.setup();
}

void loop() {
    trafficLight.update();
}
```


1
5
10
15
20
25
30
a
b
c
d
e
f
g
h
i
j
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ARDUINO
UNO
ON
IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN

Task-02


Arduino
+setup()
+loop()
TrafficLight
-int redPin
-int yellowPin
-int greenPin
-unsigned long previousMillis
-bool pedestrianWaiting
-bool wasGreenBefore
-TrafficState currentState
+TrafficLight(int rPin, int yPin, int gPin)
+void setPedestrianWaiting(bool waiting)
+void update(unsigned long currentMillis, PedestrianLight &pedLight)
-void handleGreenState(unsigned long currentMillis)
-void handleYellowState(unsigned long currentMillis)
-void handleRedState(unsigned long currentMillis, PedestrianLight &pedLight)
PedestrianLight
-int redPin
-int greenPin
+PedestrianLight(int rPin, int gPin)
+void activate()
+void deactivate()
«enumeration»
TrafficState
+RED
+YELLOW
+GREEN
Button
-int pin
+Button(int pin)
+int readState()


pedRED
pedLight.deactivate()
pedLight.activate()
pedGREEN
GREEN
t5
YELLOW
t20
t5
RED
digitalRead(buttonPin)


Button
TrafficLight
PedestrianLight
Traffic Red LED
Traffic Yellow LED
Traffic Green LED
Pedestrian Red LED
Pedestrian Green LED
Button pressed (setPedestrianWaiting)
Turn OFF Green LED
Turn ON Yellow LED
Wait for yellowDuration
Turn OFF Yellow LED
Turn ON Red LED
Activate pedestrian light
Turn OFF Pedestrian Red LED
Turn ON Pedestrian Green LED
Wait for redDuration
Deactivate pedestrian light
Turn OFF Pedestrian Green LED
Turn ON Pedestrian Red LED
Turn OFF Red LED
Turn ON Yellow LED
Wait for yellowDuration
Turn OFF Yellow LED
Turn ON Green LED
Button
TrafficLight
PedestrianLight
Traffic Red LED
Traffic Yellow LED
Traffic Green LED
Pedestrian Red LED
Pedestrian Green LED

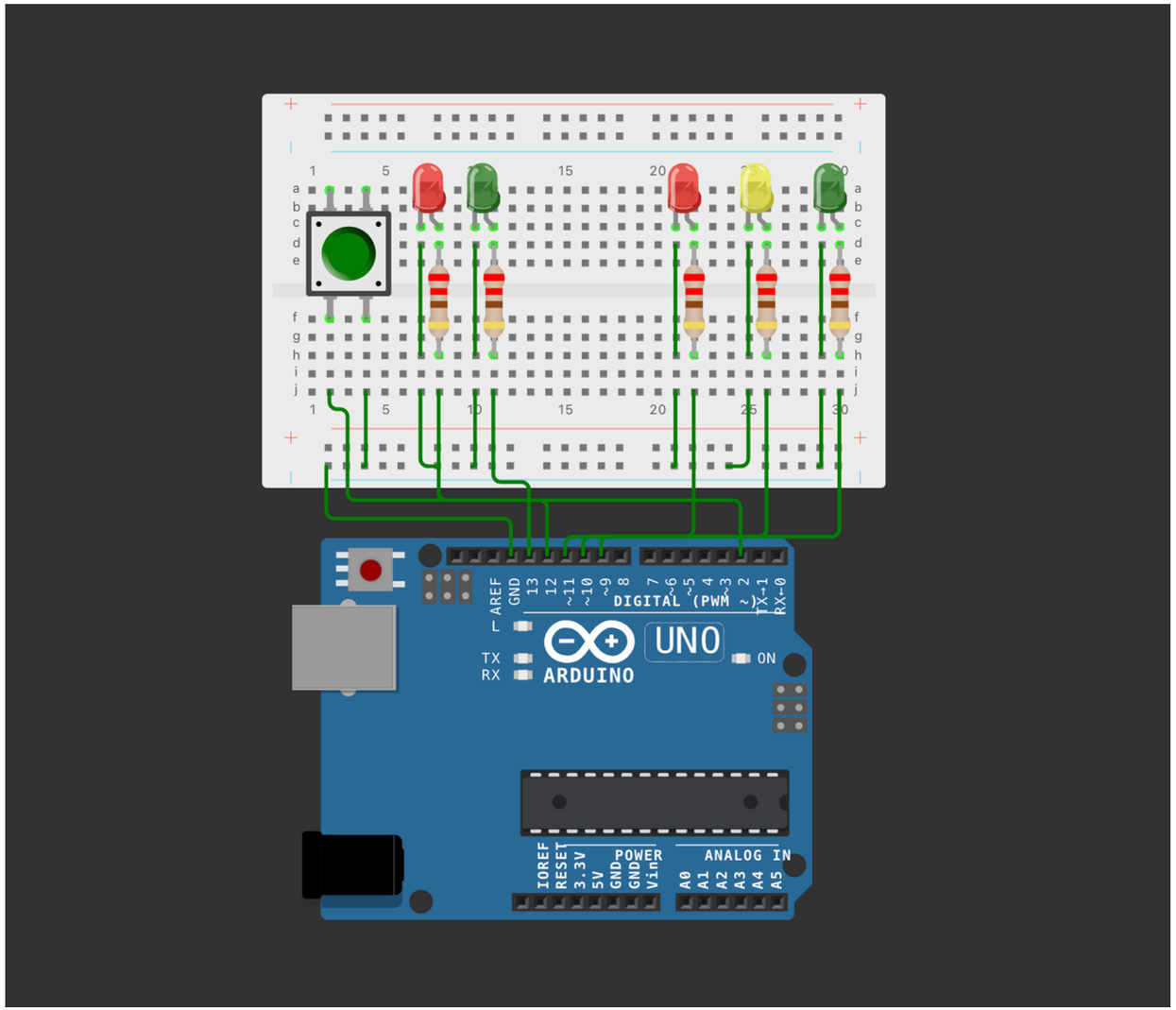
```cpp
#include <Arduino.h>

// Variables for the traffic light
const int trafficRedPin = 9;
const int trafficYellowPin = 10;
const int trafficGreenPin = 11;

// Variables for the pedestrian light
const int pedRedPin = 12;
const int pedGreenPin = 13;

// Button pin for pedestrian crossing
const int buttonPin = 2;

// Timing durations
const unsigned long greenDuration = 5000;
const unsigned long yellowDuration = 5000;
const unsigned long redDuration = 20000;

// Debounce timing
const unsigned long debounceDelay = 50;

// Declaration of PedestrianLight class
class PedestrianLight;

// TrafficLight class definition
class TrafficLight {
private:
    int redPin, yellowPin, greenPin; // Traffic light pins
    unsigned long previousMillis; // State and timing variables
    bool pedestrianWaiting;
    bool wasGreenBefore; // Flag to track if the previous state was GREEN
    enum TrafficState { RED, YELLOW, GREEN };
    TrafficState currentState;
```

```cpp
public:
    // Constructor
    TrafficLight(int rPin, int yPin, int gPin)
        : redPin(rPin), yellowPin(yPin), greenPin(gPin),
          previousMillis(0), pedestrianWaiting(false),
          wasGreenBefore(true), currentState(GREEN) {
        // Initialize pin modes
        pinMode(redPin, OUTPUT);
        pinMode(yellowPin, OUTPUT);
        pinMode(greenPin, OUTPUT);
        digitalWrite(greenPin, HIGH); // Start with the green light on
    }

    // Method to set pedestrian waiting flag
    void setPedestrianWaiting(bool waiting) {
        pedestrianWaiting = waiting;
    }

    // Method to update traffic light states
    void update(unsigned long currentMillis, PedestrianLight &pedLight);

private:
    // Method to handle the GREEN state
    void handleGreenState(unsigned long currentMillis) {
        digitalWrite(greenPin, HIGH);
        digitalWrite(redPin, LOW);
        if (currentMillis - previousMillis >= greenDuration) {
            previousMillis = currentMillis;
            digitalWrite(greenPin, LOW);
            if (pedestrianWaiting) {
                currentState = YELLOW;
                wasGreenBefore = true;
            } else {
                previousMillis = millis();
                digitalWrite(greenPin, HIGH);
            }
```

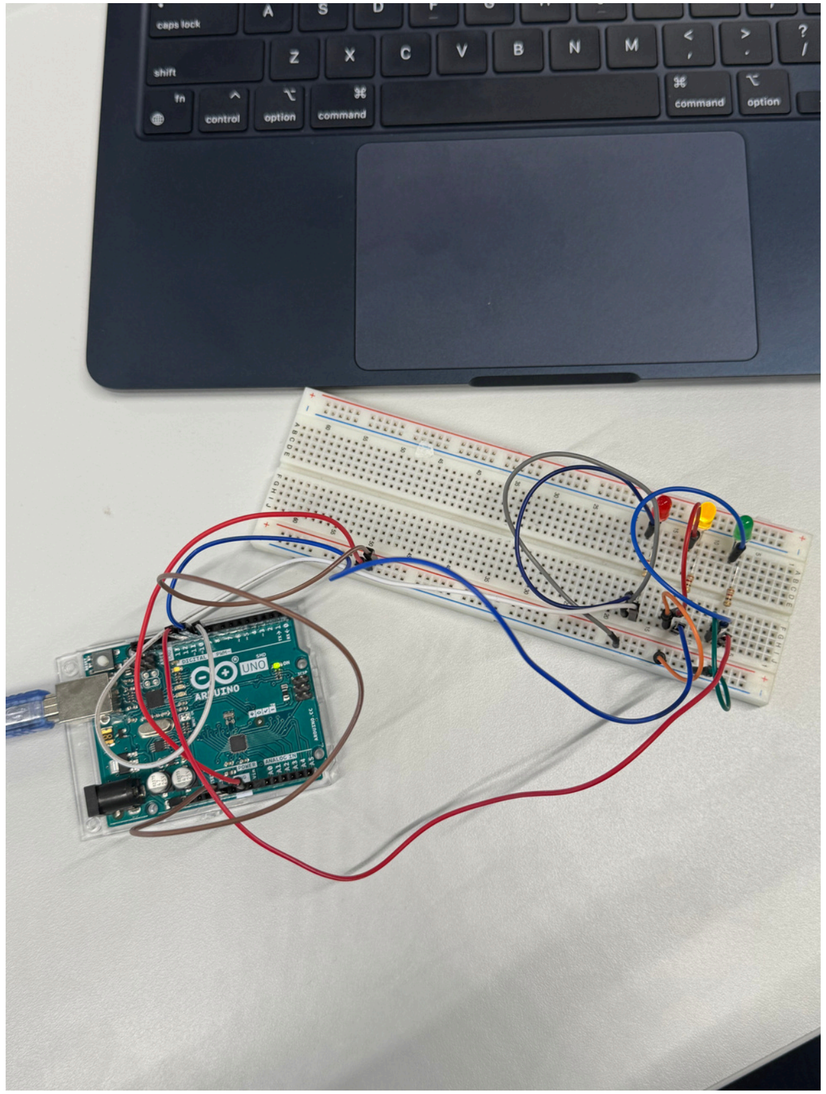
```cpp
    // Method to handle the YELLOW state
    void handleYellowState(unsigned long currentMillis) {
        digitalWrite(yellowPin, HIGH);
        if (currentMillis - previousMillis >= yellowDuration) {
            previousMillis = currentMillis;
            digitalWrite(yellowPin, LOW);
            currentState = wasGreenBefore ? RED : GREEN;
        }
    }

    // Method to handle the RED state
    void handleRedState(unsigned long currentMillis, PedestrianLight &pedLight);
};

// PedestrianLight class definition
class PedestrianLight {
private:
    int redPin, greenPin;

public:
    // Constructor
    PedestrianLight(int rPin, int gPin)
        : redPin(rPin), greenPin(gPin) {
        // Initialize pin modes
        pinMode(redPin, OUTPUT);
        pinMode(greenPin, OUTPUT);
        digitalWrite(redPin, HIGH); // Start with the red light on
    }

    // Method to activate the green pedestrian light
    void activate() {
        digitalWrite(redPin, LOW);
        digitalWrite(greenPin, HIGH);
    }
```

```
    // Method to deactivate the green pedestrian light
    void deactivate() {
        digitalWrite(greenPin, LOW);
        digitalWrite(redPin, HIGH);
    }
};

// Method implementation for TrafficLight class that requires PedestrianLight
void TrafficLight::update(unsigned long currentMillis, PedestrianLight &pedLight) {
    switch (currentState) {
        case GREEN:
            handleGreenState(currentMillis);
            break;
        case YELLOW:
            handleYellowState(currentMillis);
            break;
        case RED:
            handleRedState(currentMillis, pedLight);
            break;
    }
}

void TrafficLight::handleRedState(unsigned long currentMillis, PedestrianLight &pedLight) {
    digitalWrite(redPin, HIGH);
    pedLight.activate(); // Activate the pedestrian light
    if (currentMillis - previousMillis >= redDuration) {
        previousMillis = currentMillis;
        digitalWrite(redPin, LOW);
        pedLight.deactivate(); // Deactivate the pedestrian light
        pedestrianWaiting = false;
        currentState = YELLOW;
        wasGreenBefore = false;
    }
}
```

```
// Main function and global variables
TrafficLight trafficTrafficLight(trafficRedPin, trafficYellowPin, trafficGreenPin);
PedestrianLight pedestrianLight(pedRedPin, pedGreenPin);
unsigned long lastDebounceTime = 0;

void setup() {
    pinMode(buttonPin, INPUT_PULLUP);
}

void loop() {
    unsigned long currentMillis = millis();
    trafficTrafficLight.update(currentMillis, pedestrianLight);

    int buttonState = digitalRead(buttonPin);
    if (buttonState == LOW && (currentMillis - lastDebounceTime) > debounceDelay) {
        trafficTrafficLight.setPedestrianWaiting(true);
        lastDebounceTime = currentMillis;
    }
}
```

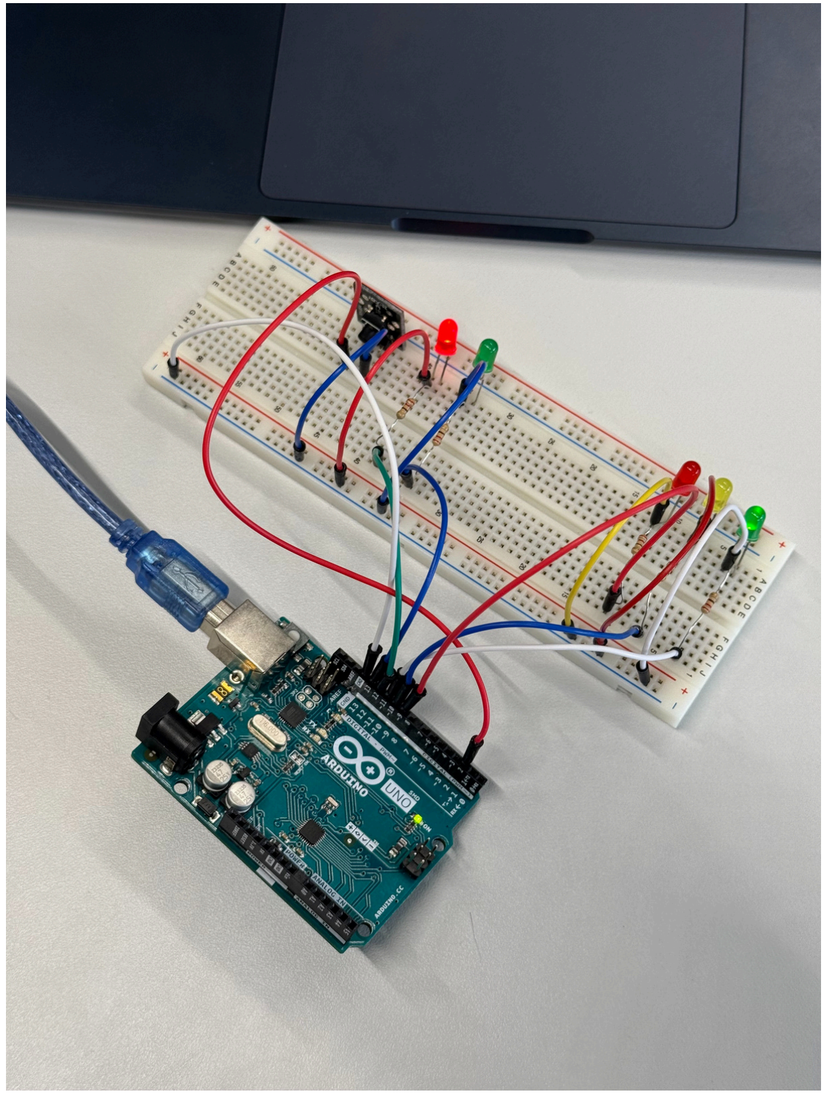

UNO
ARDUINO
DIGITAL (PWM ~)
POWER
ANALOG IN

Task-03



Implementation of Task 1

Task-04



Implementation of Task 2

Task-05

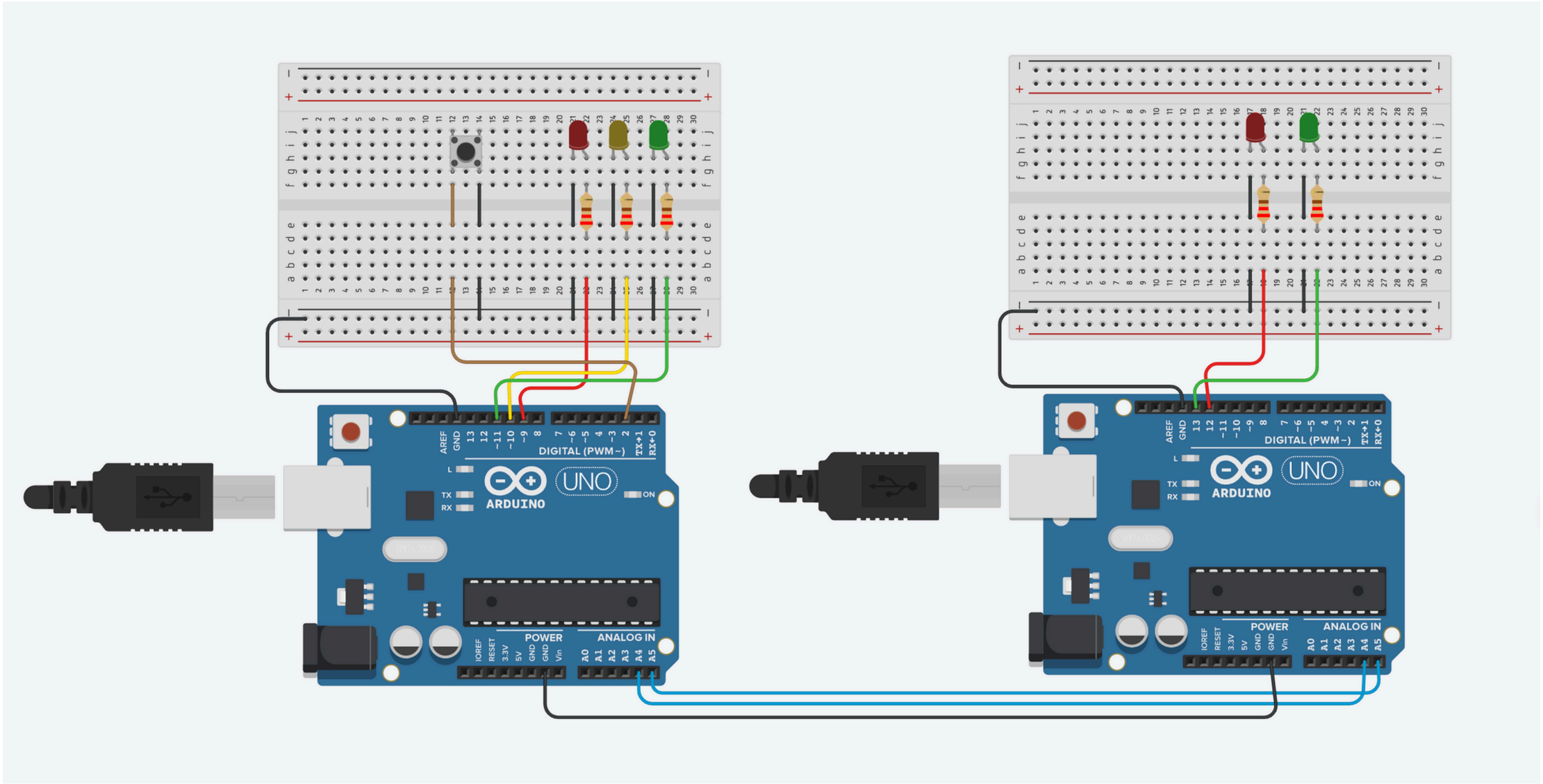
Inter-Integrated Circuit connection between
two Arduino boards

Code for Master Arduino:

```
#include <Wire.h>

// Variables for the traffic light
const int trafficRedPin = 9;
const int trafficYellowPin = 10;
const int trafficGreenPin = 11;

// Button pin for pedestrian crossing
const int buttonPin = 2;

// Timing durations
const unsigned long greenDuration = 5000;
const unsigned long yellowDuration = 5000;
const unsigned long redDuration = 20000;

// Debounce timing
const unsigned long debounceDelay = 50;

// TrafficLight class definition
class TrafficLight {
private:
    int redPin, yellowPin, greenPin;
    unsigned long previousMillis;
    bool pedestrianWaiting;
    bool wasGreenBefore;
    enum TrafficState { RED, YELLOW, GREEN };
    TrafficState currentState;

public:
    TrafficLight(int rPin, int yPin, int gPin)
        : redPin(rPin), yellowPin(yPin), greenPin(gPin),
          previousMillis(0), pedestrianWaiting(false),
          wasGreenBefore(true), currentState(GREEN) {
        pinMode(redPin, OUTPUT);
        pinMode(yellowPin, OUTPUT);
        pinMode(greenPin, OUTPUT);
        digitalWrite(greenPin, HIGH);
    }

    void setPedestrianWaiting(bool waiting) {
        pedestrianWaiting = waiting;
    }

    void update(unsigned long currentMillis) {
        switch (currentState) {
            case GREEN:
                handleGreenState(currentMillis);
                break;
            case YELLOW:
                handleYellowState(currentMillis);
                break;
            case RED:
                handleRedState(currentMillis);
                break;
        }
    }

private:
    void handleGreenState(unsigned long currentMillis) {
        digitalWrite(greenPin, HIGH);
        digitalWrite(redPin, LOW);
        if (currentMillis - previousMillis >= greenDuration) {
            previousMillis = currentMillis;
            digitalWrite(greenPin, LOW);
            if (pedestrianWaiting) {
                currentState = YELLOW;
                wasGreenBefore = true;
            } else {
                previousMillis = millis();
                digitalWrite(greenPin, HIGH);
            }
        }
    }

    void handleYellowState(unsigned long currentMillis) {
        digitalWrite(yellowPin, HIGH);
        if (currentMillis - previousMillis >= yellowDuration) {
            previousMillis = currentMillis;
            digitalWrite(yellowPin, LOW);
            currentState = wasGreenBefore ? RED : GREEN;
        }
    }

    void handleRedState(unsigned long currentMillis) {
        digitalWrite(redPin, HIGH);
        Wire.beginTransmission(8); // Address of the slave
        Wire.write("activate"); // Command to activate the pedestrian light
        Wire.endTransmission();
        if (currentMillis - previousMillis >= redDuration) {
            previousMillis = currentMillis;
            digitalWrite(redPin, LOW);
            Wire.beginTransmission(8);
            Wire.write("deactivate"); // Command to deactivate the pedestrian light
            Wire.endTransmission();
            pedestrianWaiting = false;
            currentState = YELLOW;
            wasGreenBefore = false;
        }
    }
};

// Main function and global variables
TrafficLight trafficTrafficLight(trafficRedPin, trafficYellowPin, trafficGreenPin);
unsigned long lastDebounceTime = 0;

void setup() {
    Wire.begin(); // Start I2C communication
    pinMode(buttonPin, INPUT_PULLUP);
}

void loop() {
    unsigned long currentMillis = millis();
    trafficTrafficLight.update(currentMillis);

    int buttonState = digitalRead(buttonPin);
    if (buttonState == LOW && (currentMillis - lastDebounceTime) > debounceDelay) {
        trafficTrafficLight.setPedestrianWaiting(true);
        lastDebounceTime = currentMillis;
    }
}
```

Code for Slave Arduino

```
#include <Wire.h>

// Variables for the pedestrian light
const int pedRedPin = 12;
const int pedGreenPin = 13;

void setup() {
    Wire.begin(8); // Join I2C bus with address #8
    Wire.onReceive(receiveEvent); // Register event handler
    pinMode(pedRedPin, OUTPUT);
    pinMode(pedGreenPin, OUTPUT);
    digitalWrite(pedRedPin, HIGH); // Start with the red light on
}

void loop() {

}

void receiveEvent(int howMany) {
    String command = "";
    while (Wire.available()) {
        char c = Wire.read();
        command += c;
    }

    if (command == "activate") {
        digitalWrite(pedRedPin, LOW);
        digitalWrite(pedGreenPin, HIGH);
    } else if (command == "deactivate") {
        digitalWrite(pedGreenPin, LOW);
        digitalWrite(pedRedPin, HIGH);
    }
}

```


DIGITAL (PWM~)
POWER
ANALOG IN
UNO
ARDUINO