Microcontrollers

Milestone 1

Muhammad Ali

Github

Task 1 & 2

Github Link: **https://github.com/MAliSohail/Traffic-Pedestrian-Lights-Simulation**

Arduino Code:

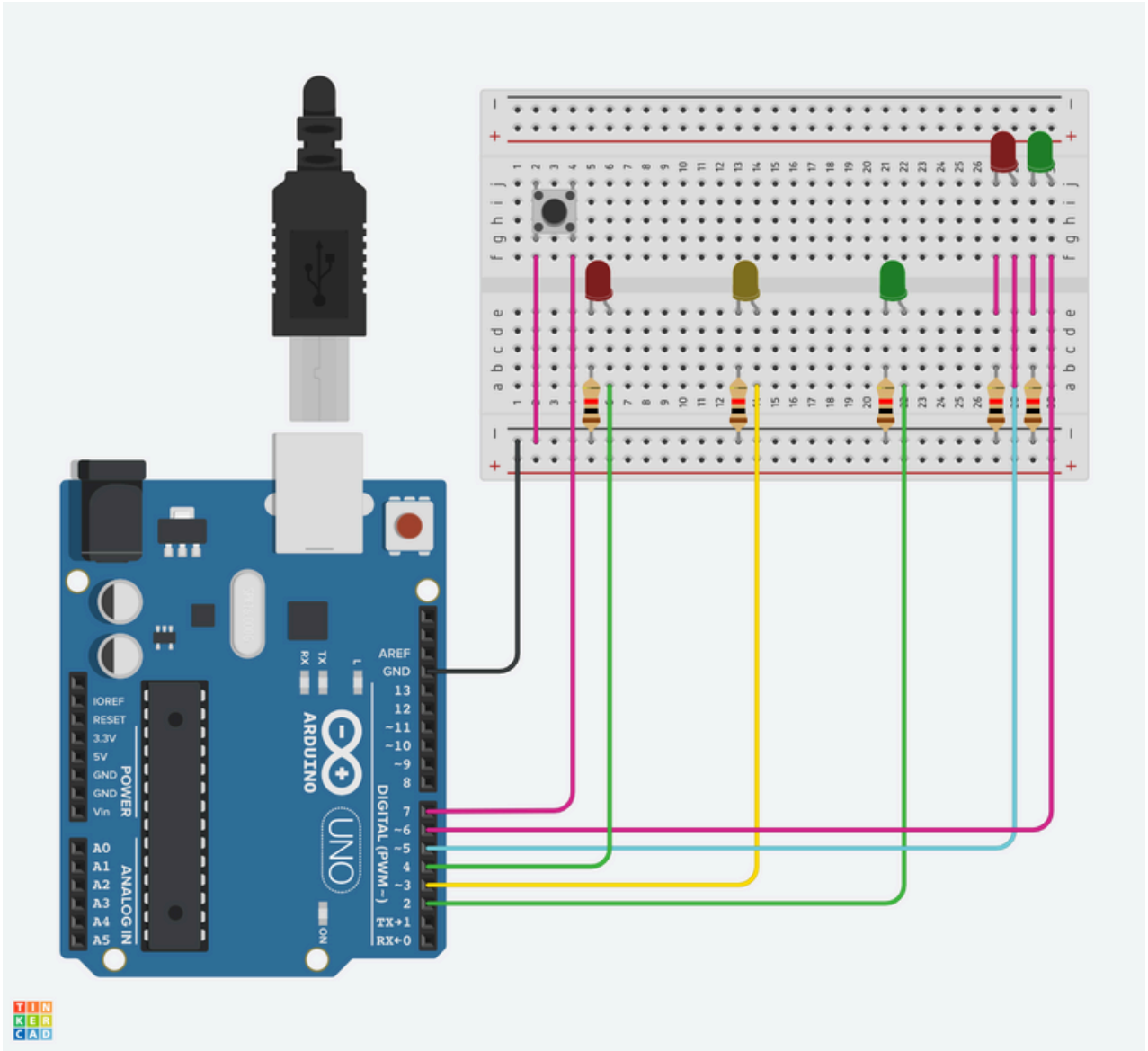
traffic_light_simulation1.ino

```
const int greenLight = 2;
const int yellowLight = 3;
const int redLight = 4;
const int pedRed = 5;
const int pedGreen = 6;
const int button = 7;

enum TrafficState { GREEN, WAIT, YELLOW, RED };
TrafficState currentState = GREEN;

unsigned long stateTimer = 0;
volatile bool buttonPressed = false;

void setup() {
  pinMode(greenLight, OUTPUT);
  pinMode(yellowLight, OUTPUT);
  pinMode(redLight, OUTPUT);
  pinMode(pedRed, OUTPUT);
  pinMode(pedGreen, OUTPUT);
  pinMode(button, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(button), buttonISR, FALLING);

  digitalWrite(greenLight, HIGH);
  digitalWrite(pedRed, HIGH);
  stateTimer = millis();
}

void loop() {
  if (buttonPressed) {
    buttonPressed = false;
    if (currentState == GREEN) {
      stateTimer = millis();
      currentState = WAIT;
    }
  }

  switch (currentState) {
    case GREEN:
      if (millis() - stateTimer >= 10000) {
        stateTimer = millis();
        currentState = WAIT;
      }
      break;

    case WAIT:
```

traffic_light_simulation1.ino

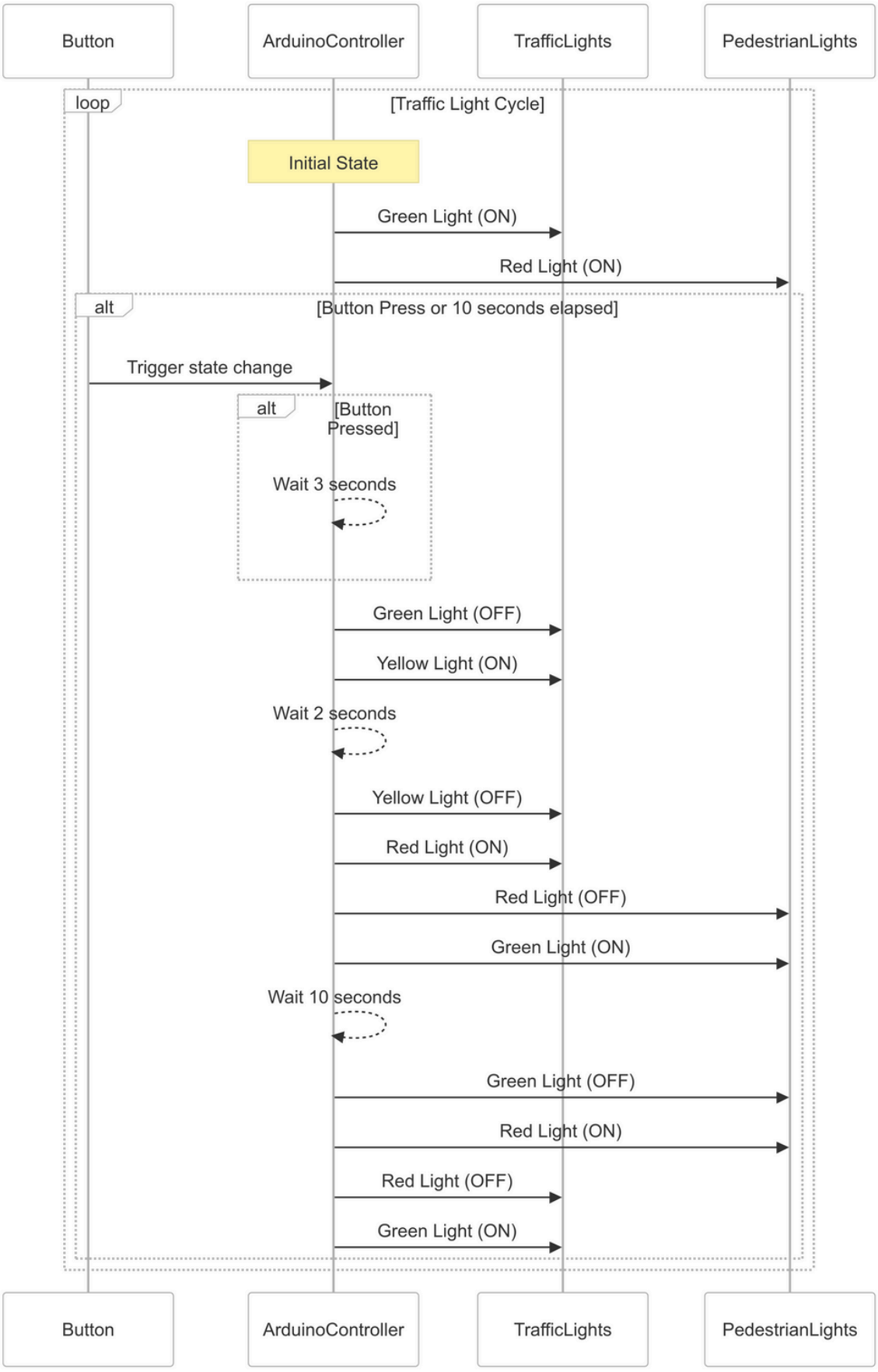
```
      if (millis() - stateTimer >= 10000) {
        stateTimer = millis();
        currentState = WAIT;
      }
      break;

    case WAIT:
      if (millis() - stateTimer >= 3000) {
        stateTimer = millis();
        currentState = YELLOW;
      }
      break;

    case YELLOW:
      if (millis() - stateTimer >= 2000) {
        stateTimer = millis();
        currentState = RED;
      }
      digitalWrite(greenLight, LOW);
      digitalWrite(yellowLight, HIGH);
      break;

    case RED:
      if (millis() - stateTimer >= 10000) {
        stateTimer = millis();
        currentState = GREEN;
      }
      digitalWrite(yellowLight, LOW);
      digitalWrite(redLight, HIGH);
      digitalWrite(pedRed, LOW);
      digitalWrite(pedGreen, HIGH);

      if (currentState == GREEN) {
        digitalWrite(pedGreen, LOW);
        digitalWrite(pedRed, HIGH);
        digitalWrite(redLight, LOW);
        digitalWrite(greenLight, HIGH);
      }
      break;
  }
}

void buttonISR() {
  buttonPressed = true;
}

```

Circuit Design

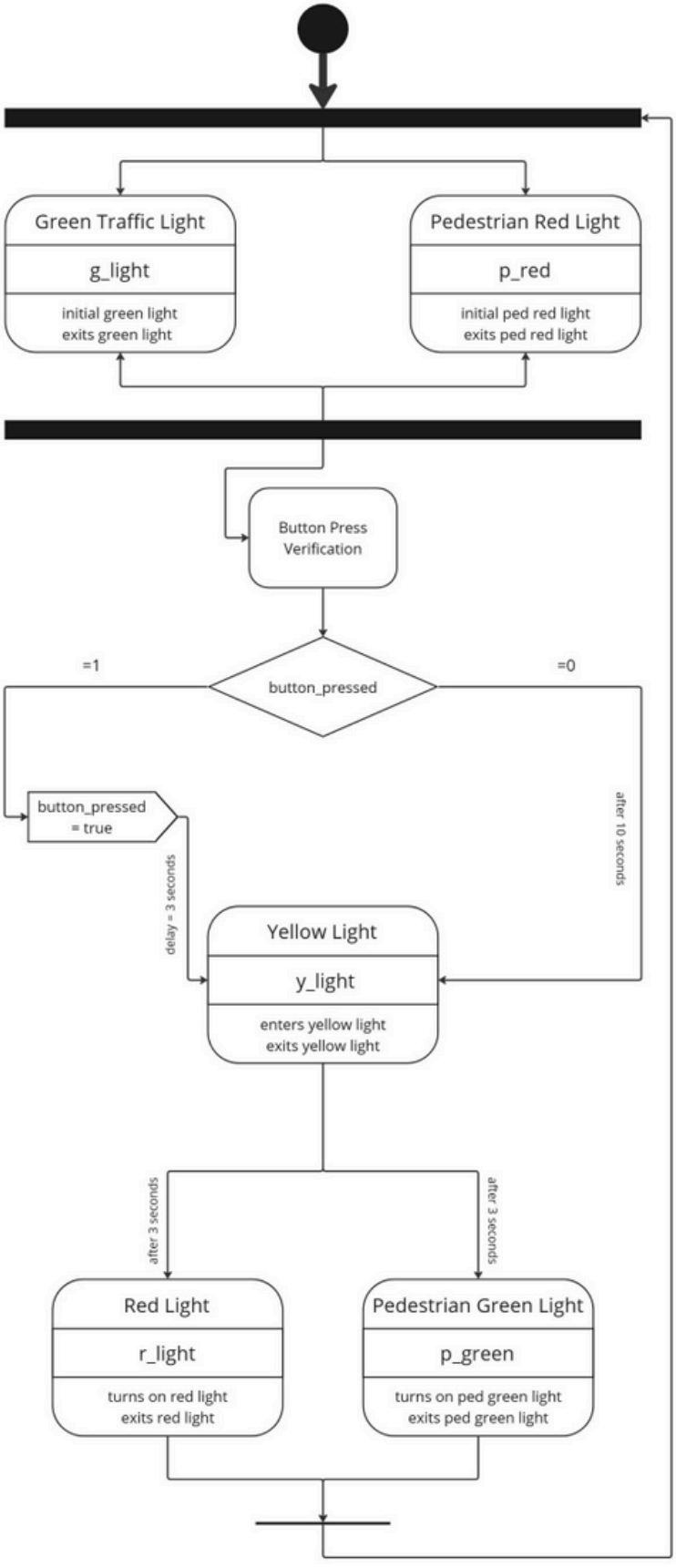


Class Diagram

Sequence Diagram



State Machine Diagram



Task 4

Polling-Based Implementation

How It Works

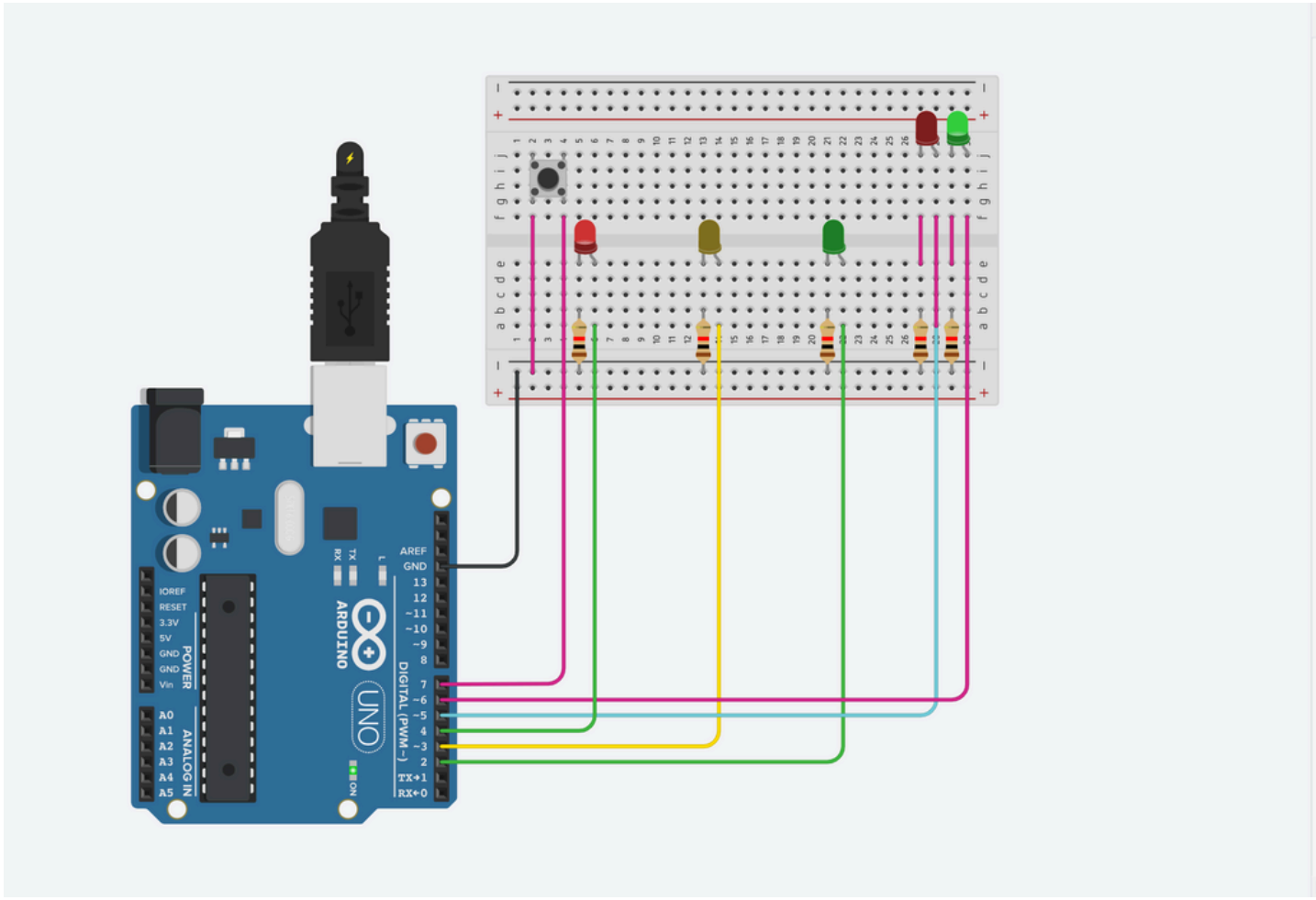
The loop() function continuously checks the state of the button using digitalRead(). If the button is pressed, the state is updated.

Characteristics

- Simple: Easy to implement and understand.
- Inefficient: Continuously checking the button consumes CPU resources, even when no button is pressed.
- Slower Response: Delays in detecting button presses can occur if the loop contains time-consuming tasks.

Use Case

Suitable for simple systems where the program doesn't have to handle multiple tasks or real-time constraints.



1 (Arduino Uno R3)

```
const int greenLight = 2;
const int yellowLight = 3;
const int redLight = 4;
const int pedRed = 5;
const int pedGreen = 6;
const int button = 7;

enum TrafficState { GREEN, WAIT, YELLOW, RED };
TrafficState currentState = GREEN;

unsigned long stateTimer = 0;
bool buttonPressed = false;

void setup() {
  pinMode(greenLight, OUTPUT);
  pinMode(yellowLight, OUTPUT);
  pinMode(redLight, OUTPUT);
  pinMode(pedRed, OUTPUT);
  pinMode(pedGreen, OUTPUT);
  pinMode(button, INPUT_PULLUP);

  digitalWrite(greenLight, HIGH);
  digitalWrite(pedRed, HIGH);
  stateTimer = millis();
}

void loop() {
  if (digitalRead(button) == LOW) {
    buttonPressed = true;
  }

  switch (currentState) {
    case GREEN:
      if (buttonPressed || millis() - stateTimer >= 10000) {
        buttonPressed = false;
        stateTimer = millis();
        currentState = WAIT;
      }
      break;

    case WAIT:
      if (millis() - stateTimer >= 3000) {
        stateTimer = millis();
        currentState = YELLOW;
      }
      break;

    case YELLOW:
      if (millis() - stateTimer >= 2000) {
        stateTimer = millis();
        currentState = RED;
      }
      digitalWrite(greenLight, LOW);
      digitalWrite(yellowLight, HIGH);
      break;

    case RED:
      if (millis() - stateTimer >= 10000) {
        stateTimer = millis();
        currentState = GREEN;
      }
      digitalWrite(yellowLight, LOW);
      digitalWrite(redLight, HIGH);
      digitalWrite(pedRed, LOW);
      digitalWrite(pedGreen, HIGH);

      if (currentState == GREEN) {
        digitalWrite(pedGreen, LOW);
```

Serial Monitor

Interrupt-Based Implementation

How It Works

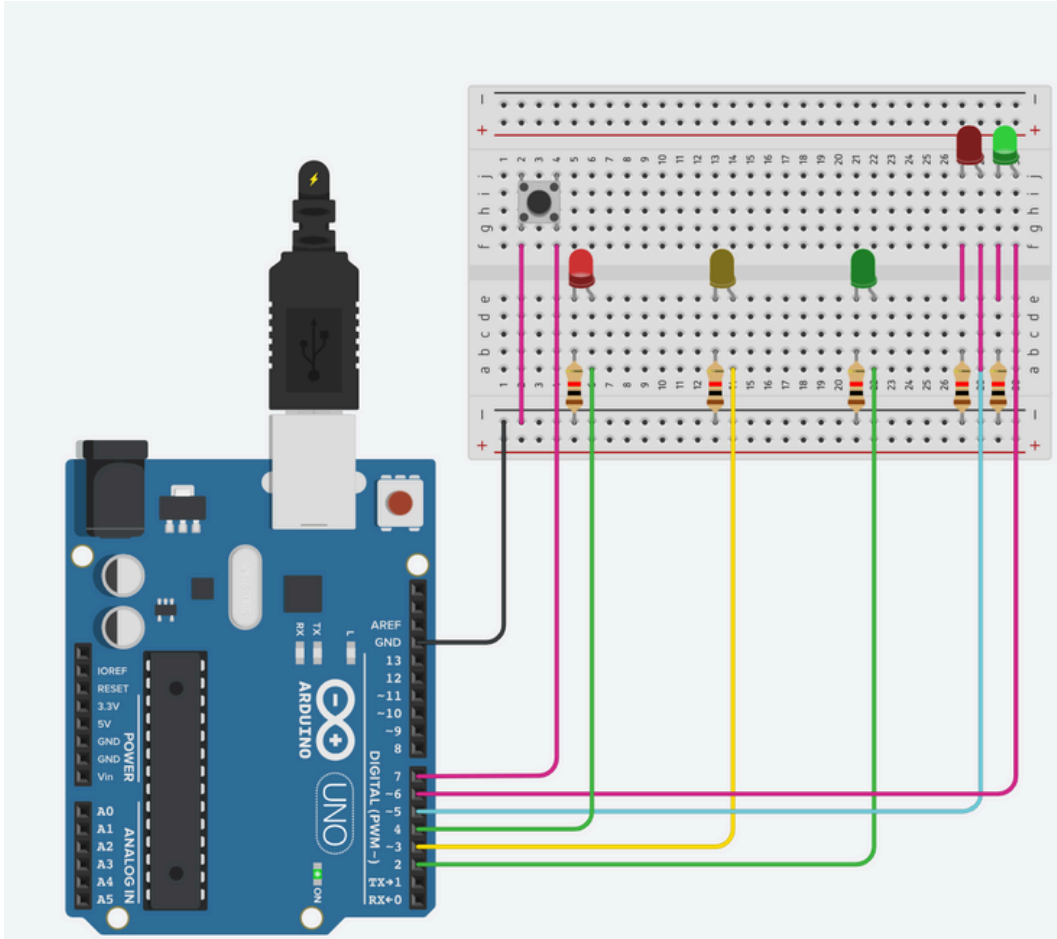
- The attachInterrupt() function links the button pin to an Interrupt Service Routine (ISR).
- The ISR is triggered automatically when the button is pressed (FALLING edge)

Characteristics

- Efficient: The CPU only reacts when the button is pressed, freeing resources for other tasks.
- Fast Response: Button presses are detected immediately, regardless of what the loop is doing.
- More Complex: Requires managing volatile variables and keeping the ISR short.

Use Case

Ideal for systems that need real-time response or must perform multiple tasks concurrently..



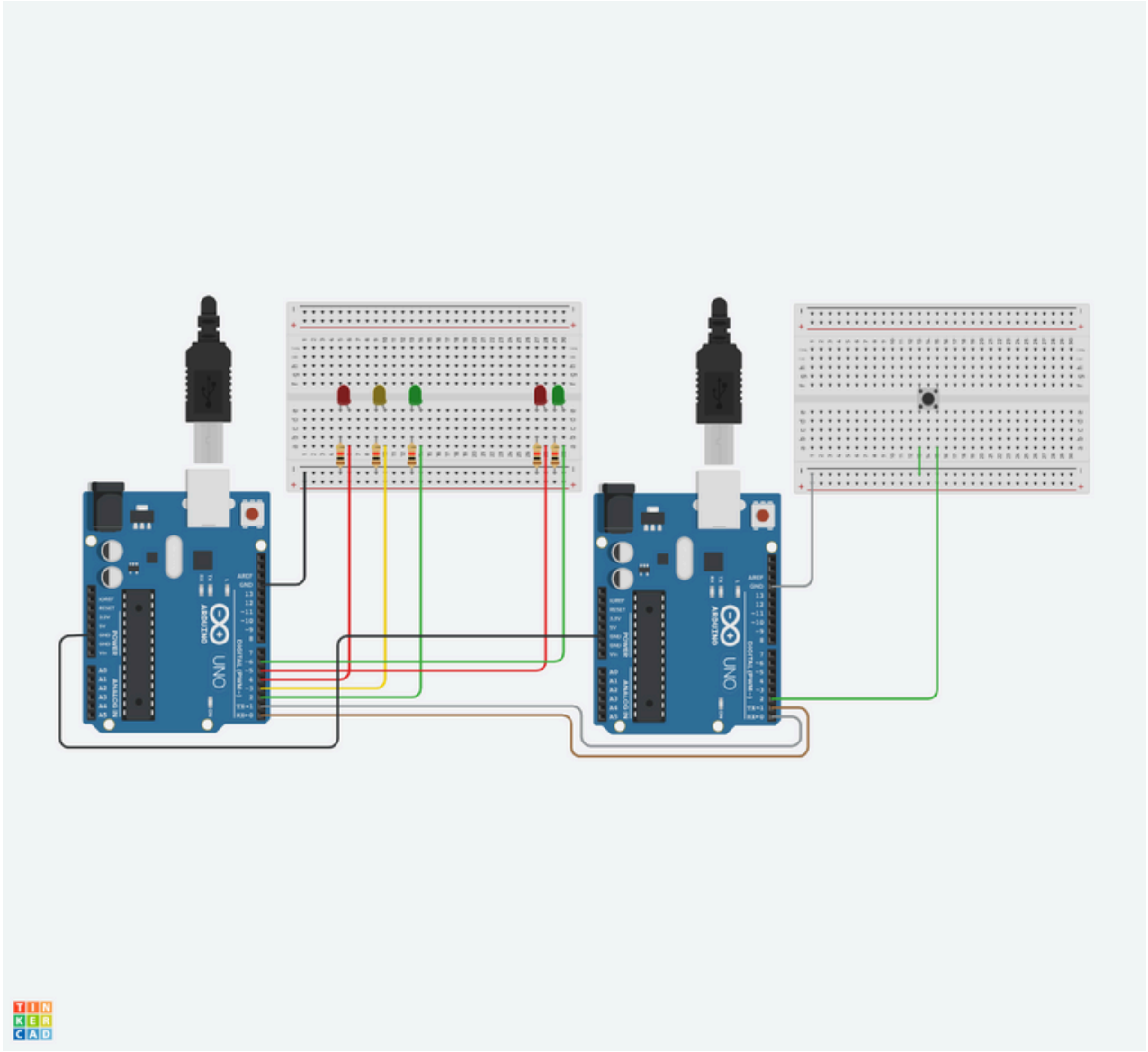
1 (Arduino Uno R3)

```ino
const int greenLight = 2;
const int yellowLight = 3;
const int redLight = 4;
const int pedRed = 5;
const int pedGreen = 6;
const int button = 7;

enum TrafficState { GREEN, WAIT, YELLOW, RED };
TrafficState currentState = GREEN;

unsigned long stateTimer = 0;
volatile bool buttonPressed = false;

void setup() {
  pinMode(greenLight, OUTPUT);
  pinMode(yellowLight, OUTPUT);
  pinMode(redLight, OUTPUT);
  pinMode(pedRed, OUTPUT);
  pinMode(pedGreen, OUTPUT);
  pinMode(button, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(button), buttonISR, FALL

  digitalWrite(greenLight, HIGH);
  digitalWrite(pedRed, HIGH);
  stateTimer = millis();
}

void loop() {
  if (buttonPressed) {
    buttonPressed = false;
    if (currentState == GREEN) {
      stateTimer = millis();
      currentState = WAIT;
    }
  }

  switch (currentState) {
    case GREEN:
      if (millis() - stateTimer >= 10000) {
        stateTimer = millis();
        currentState = WAIT;
      }
      break;

    case WAIT:
      if (millis() - stateTimer >= 3000) {
        stateTimer = millis();
        currentState = YELLOW;
      }
      break;

    case YELLOW:
      if (millis() - stateTimer >= 2000) {
        stateTimer = millis();
        currentState = RED;
      }
      digitalWrite(greenLight, LOW);
      digitalWrite(yellowLight, HIGH);
      break;

    case RED:
      if (millis() - stateTimer >= 10000) {
        stateTimer = millis();
        currentState = GREEN;
      }
      digitalWrite(yellowLight, LOW);
```

Serial Monitor

Task 5

Circuit Design

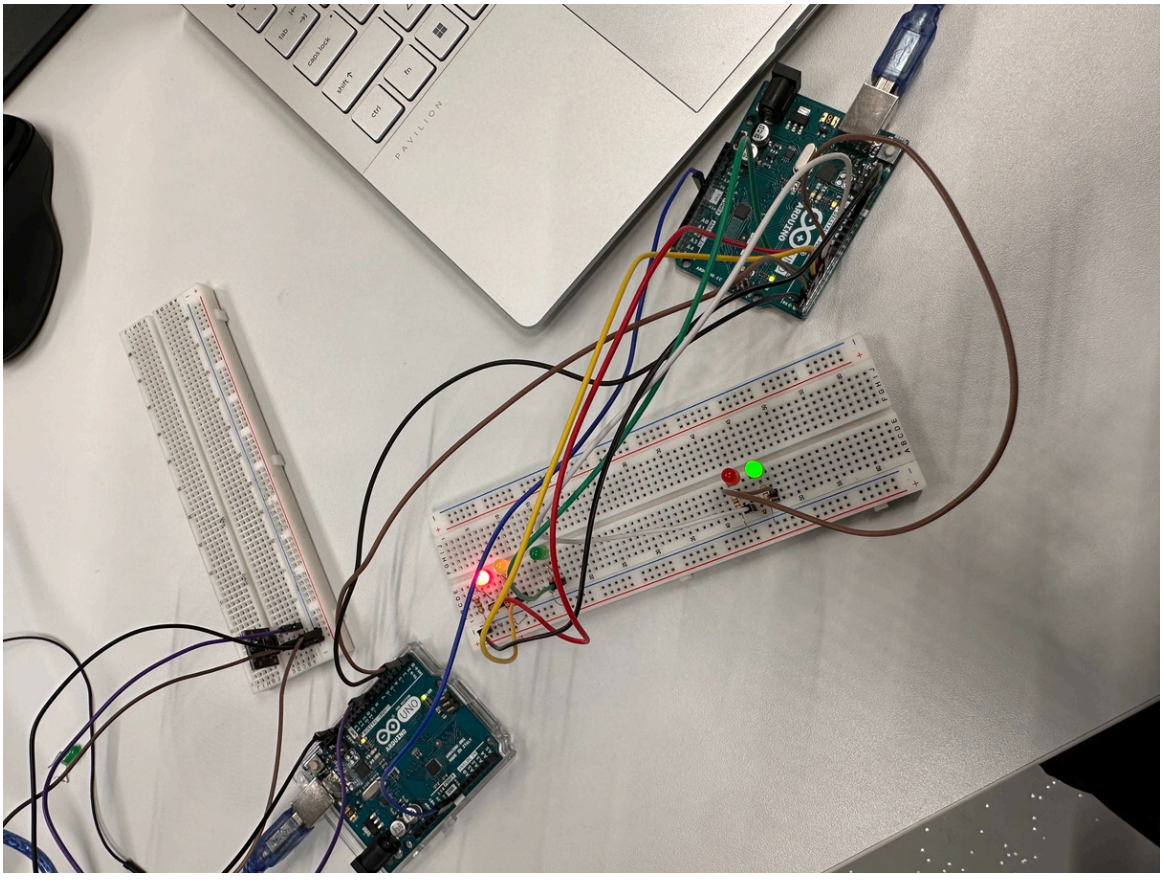


Class Diagram

Traffic Light Routine
-- g_light: int -- y_light: int -- r_light: int -- p_red: int -- p_green: int -- g_delay: unsigned long -- y_duration: unsigned long -- button_pressed: bool -- current_state: enum -- last_state_change: unsigned long
+ setup(): void + loop(): void + transition_yellow(): void + transtion_red(): void + ped_signal(): void

Pedestrian Button Control
-- button_pin: int -- button_state: bool
+ setup(): void + read_button(): bool + send_signal(): void

Physical Demonstration



Master Arduino Code

```
const int button_pin = 2;
enum button_state { IDLE, PRESSED };
button_state currentState = IDLE;

void setup() {
  pinMode(button_pin, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {
  switch (currentState) {
    case IDLE:
      if (digitalRead(button_pin) == LOW) {
        delay(50);  // Debounce delay
        if (digitalRead(button_pin) == LOW) {
          Serial.println('1');
          currentState = PRESSED;
        }
      }
      break;

    case PRESSED:
      if (digitalRead(button_pin) == HIGH) {
        currentState = IDLE;
      }
      break;
  }
}
```

Slave Arduino Code

```
const int g_light = 2;
const int y_light = 3;
const int r_light = 4;
const int p_red = 5;
const int p_green = 6;

// Variables
const unsigned long y_duration = 2000;
const unsigned long g_delay = 3000;
bool button_pressed = false;
enum TrafficState { GREEN, YELLOW, RED };
TrafficState current_state = GREEN;
unsigned long last_state_change = 0;

void setup() {
  pinMode(g_light, OUTPUT);
  pinMode(y_light, OUTPUT);
  pinMode(r_light, OUTPUT);
  pinMode(p_red, OUTPUT);
  pinMode(p_green, OUTPUT);

  digitalWrite(g_light, HIGH);
  digitalWrite(p_red, HIGH);

  Serial.begin(9600);
}

void loop() {
  if (Serial.available() > 0) {
    char receivedChar = Serial.read();
    if (receivedChar == '1') {
      button_pressed = true;
    }
```

```
  }

  unsigned long currentMillis = millis();

  switch (current_state) {
    case GREEN:
      if (button_pressed) {
        if (currentMillis - last_state_change >= g_delay) {
          button_pressed = false;
          current_state = YELLOW;
          last_state_change = currentMillis;
        }
      } else if (currentMillis - last_state_change >= 10000) {
        current_state = YELLOW;
        last_state_change = currentMillis;
      }
      break;

    case YELLOW:
      if (currentMillis - last_state_change >= y_duration) {
        current_state = RED;
        last_state_change = currentMillis;

        digitalWrite(g_light, LOW);
        digitalWrite(y_light, LOW);
        digitalWrite(r_light, HIGH);
        digitalWrite(p_red, LOW);
        digitalWrite(p_green, HIGH);
      }
      break;

```

```
      case RED:
        if (currentMillis - last_state_change >= 10000) {
          current_state = GREEN;
          last_state_change = currentMillis;

          digitalWrite(p_green, LOW);
          digitalWrite(p_red, HIGH);

          digitalWrite(r_light, LOW);
          digitalWrite(g_light, HIGH);
        }
        break;
    }

    if (current_state == GREEN) {
      digitalWrite(g_light, HIGH);
      digitalWrite(y_light, LOW);
      digitalWrite(r_light, LOW);
    } else if (current_state == YELLOW) {
      digitalWrite(g_light, LOW);
      digitalWrite(y_light, HIGH);
      digitalWrite(r_light, LOW);
    }
  }
```