MILESTONE 1

microcontroller

documentation

Presented to

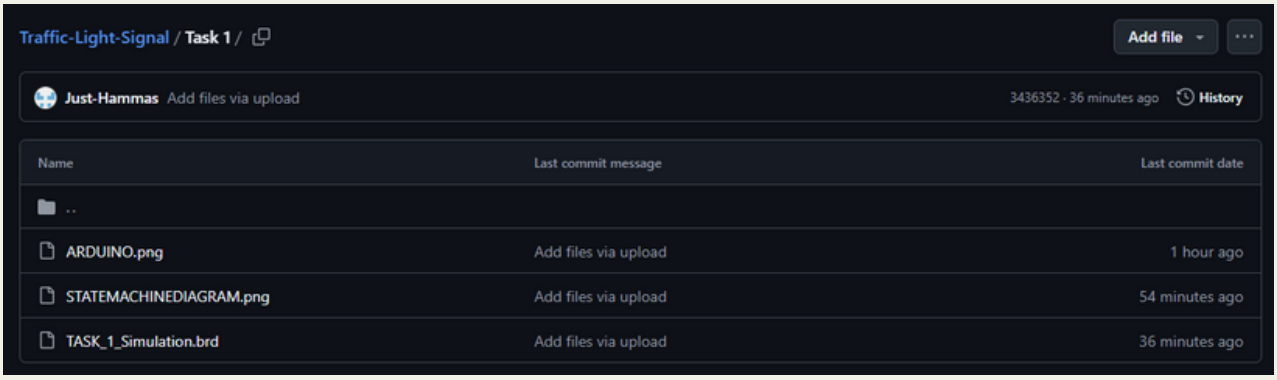
Prof. Pasendideh Faezeh

Presented by

Muhammad.Hamas@stud.hshl.de

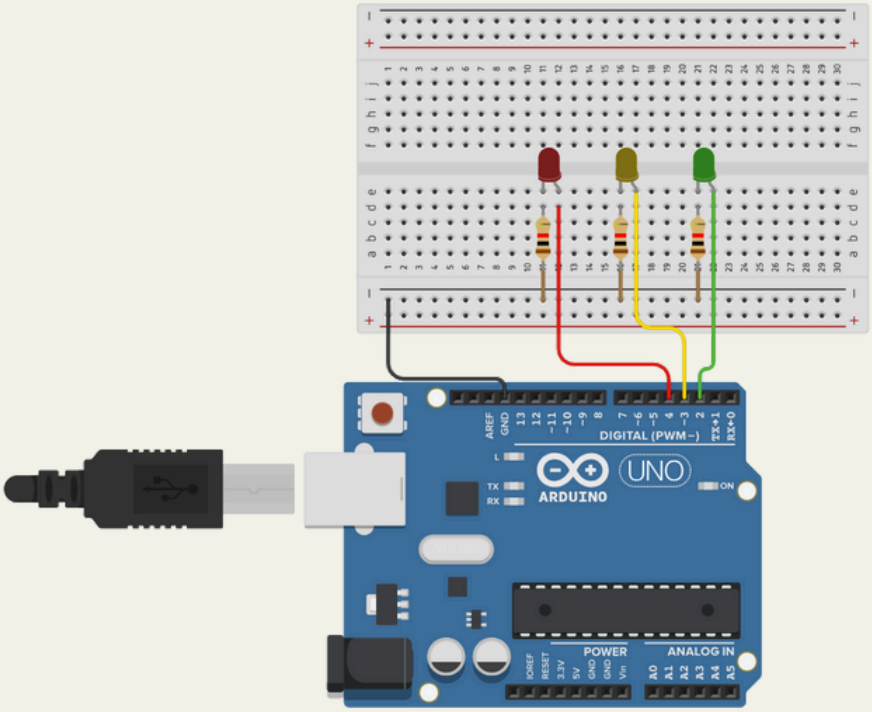
table of contents

github

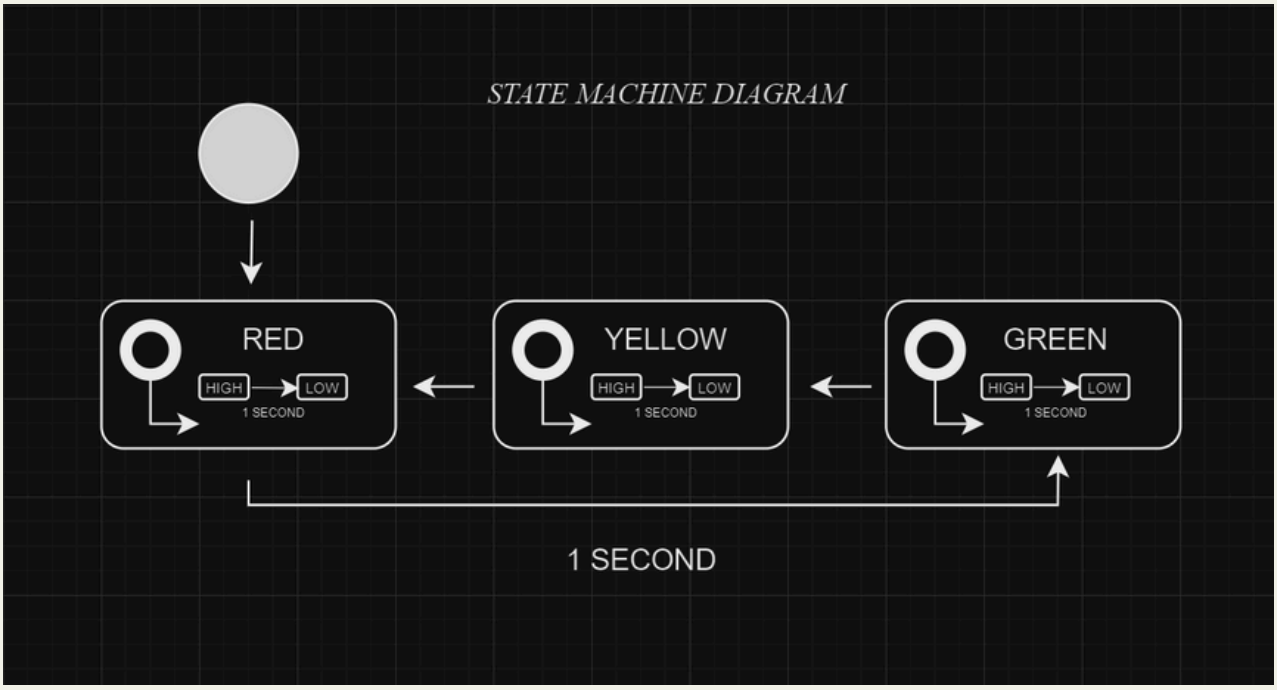


https://github.com/Just-Hammas/Traffic-Light-Signal

simulation



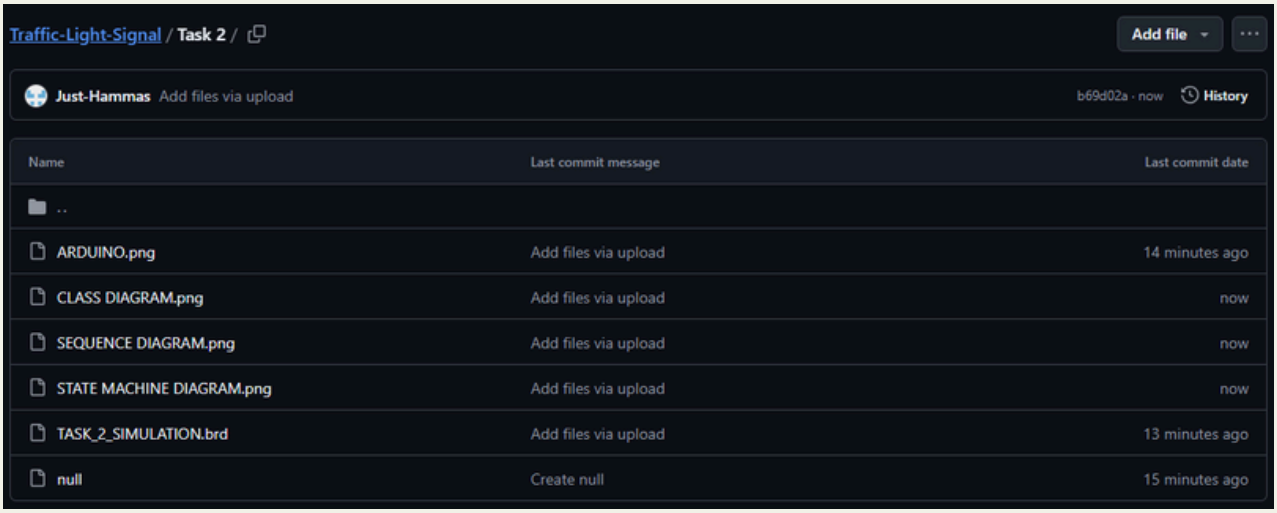
state machine diagram



code

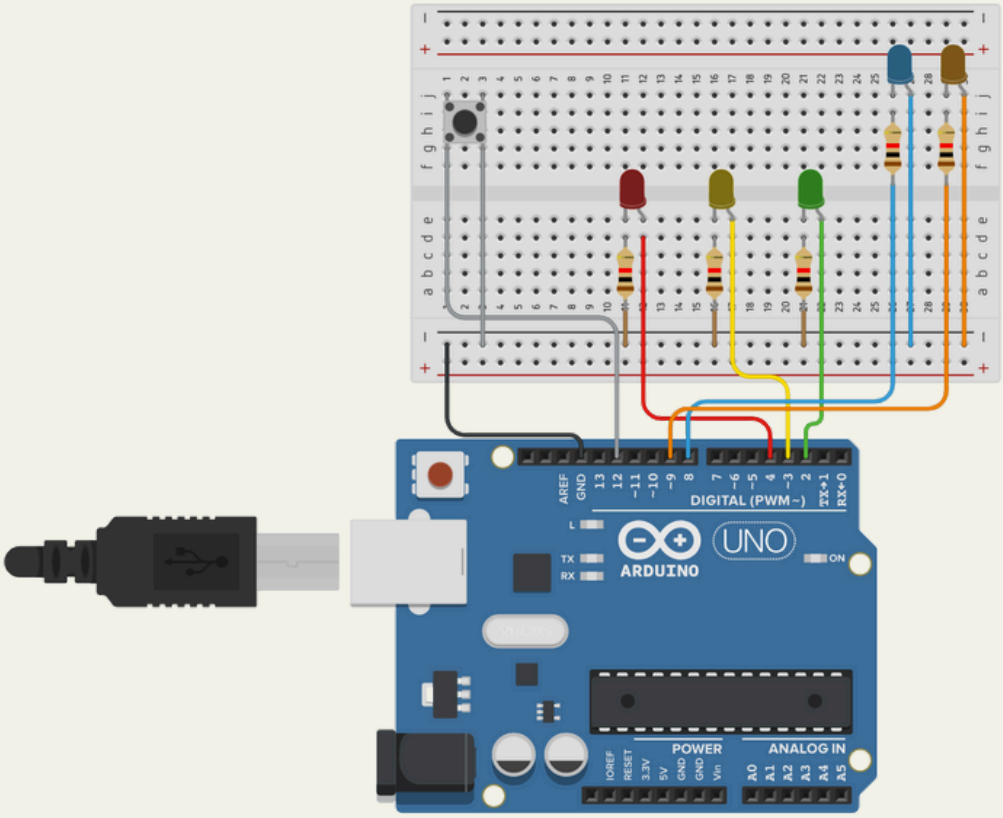
```
void setup() {
  pinMode(2, OUTPUT); // Green LED
  pinMode(3, OUTPUT); // Yellow LED
  pinMode(4, OUTPUT); // Red LED
}

void loop() {
  // Red LED on
  digitalWrite(4, HIGH);
  digitalWrite(3, LOW);
  digitalWrite(2, LOW);
  delay(5000); // 5 seconds delay

  // Yellow LED on
  digitalWrite(4, LOW);
  digitalWrite(3, HIGH);
  digitalWrite(2, LOW);
  delay(1000); // 1 second delay

  // Green LED on
  digitalWrite(4, LOW);
  digitalWrite(3, LOW);
  digitalWrite(2, HIGH);
  delay(5000); // 5 seconds delay
}
```

github



https://github.com/Just-Hammas/Traffic-Light-Signal

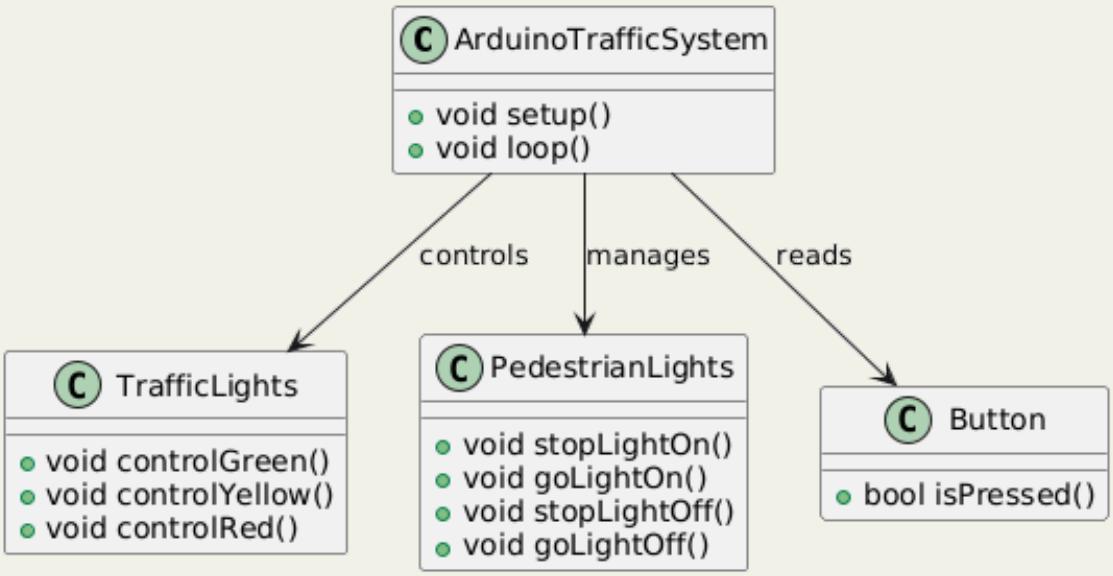
simulation



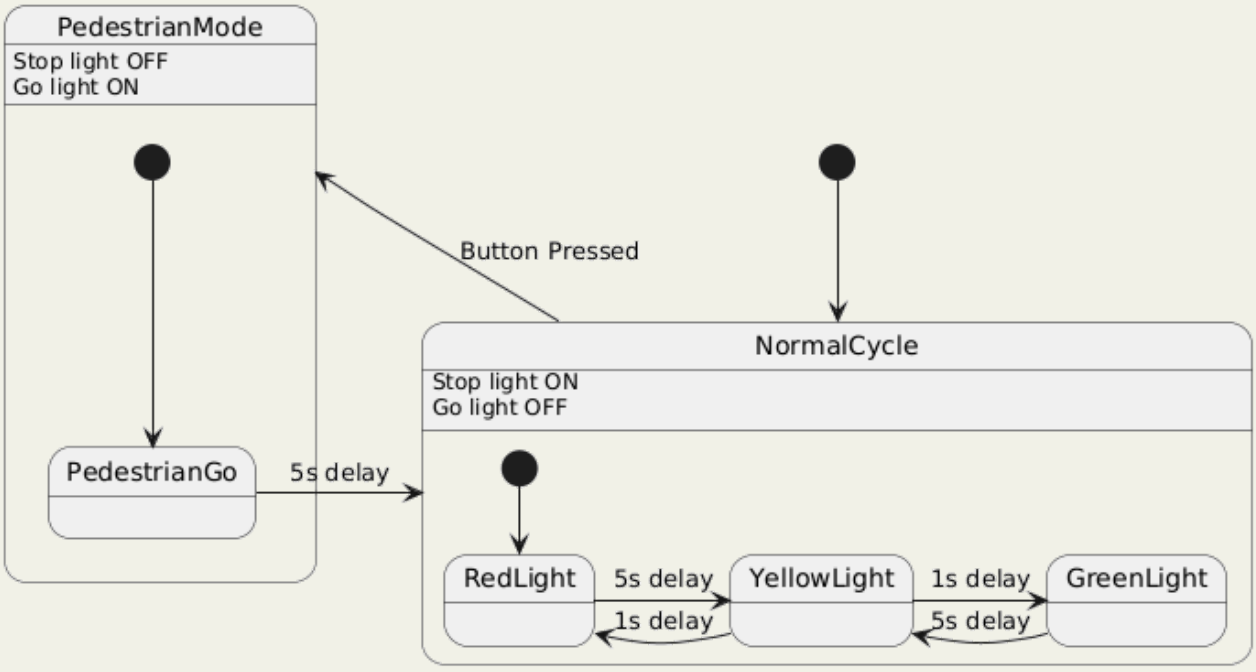
code

```
void setup() {
  // Traffic lights
  pinMode(2, OUTPUT); // Green LED
  pinMode(3, OUTPUT); // Yellow LED
  pinMode(4, OUTPUT); // Red LED

  // Pedestrian lights
  pinMode(8, OUTPUT);      // Go LED
  pinMode(9, OUTPUT);      // Stop LED
  pinMode(13, INPUT_PULLUP); // Button with pull-up resistor
}

void loop() {
  if (digitalRead(13) == LOW) { // Button pressed
    // Pedestrian crossing sequence
    digitalWrite(9, LOW);  // Turn off Stop light
    digitalWrite(8, HIGH); // Turn on Go light
    digitalWrite(4, HIGH); // Turn on Red traffic light
    digitalWrite(3, LOW);
    digitalWrite(2, LOW);
    delay(5000);           // Pedestrian Go time
    digitalWrite(8, LOW);  // Turn off Go light
    digitalWrite(9, HIGH); // Turn Stop light back on
  } else {
    // Normal traffic light cycle
    digitalWrite(9, HIGH); // Ensure Stop light is on
    digitalWrite(4, HIGH); // Red traffic light
    digitalWrite(3, LOW);
    digitalWrite(2, LOW);
    delay(5000);

    digitalWrite(4, LOW);  // Yellow traffic light
    digitalWrite(3, HIGH);
    digitalWrite(2, LOW);
    delay(1000);

    digitalWrite(4, LOW);  // Green traffic light
    digitalWrite(3, LOW);
    digitalWrite(2, HIGH);
    delay(5000);

    digitalWrite(4, LOW);  // Yellow light for reset
    digitalWrite(3, HIGH);
    digitalWrite(2, LOW);
    delay(1000);
  }
}
```

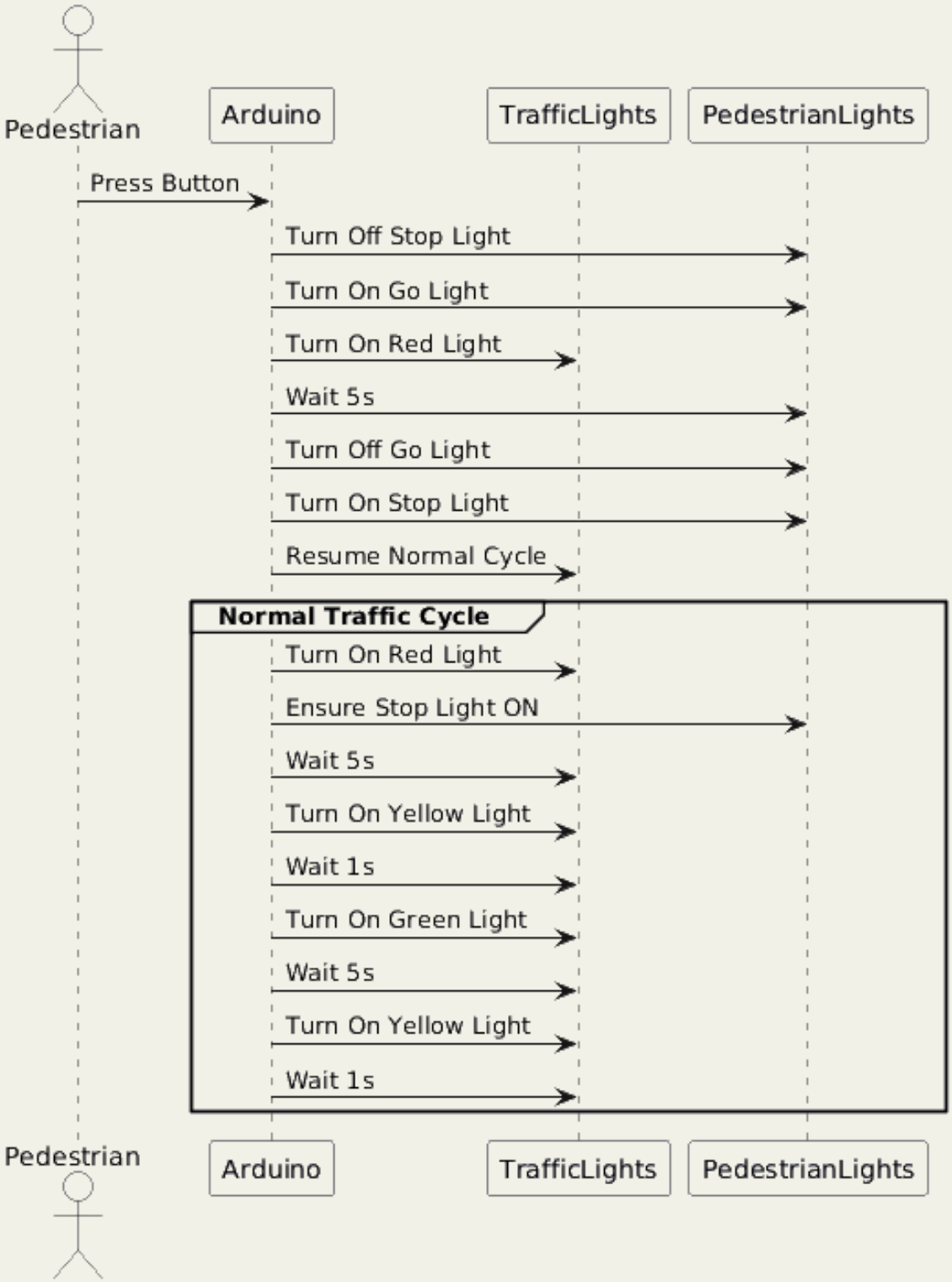
class diagram



state machine diagram



sequence diagram

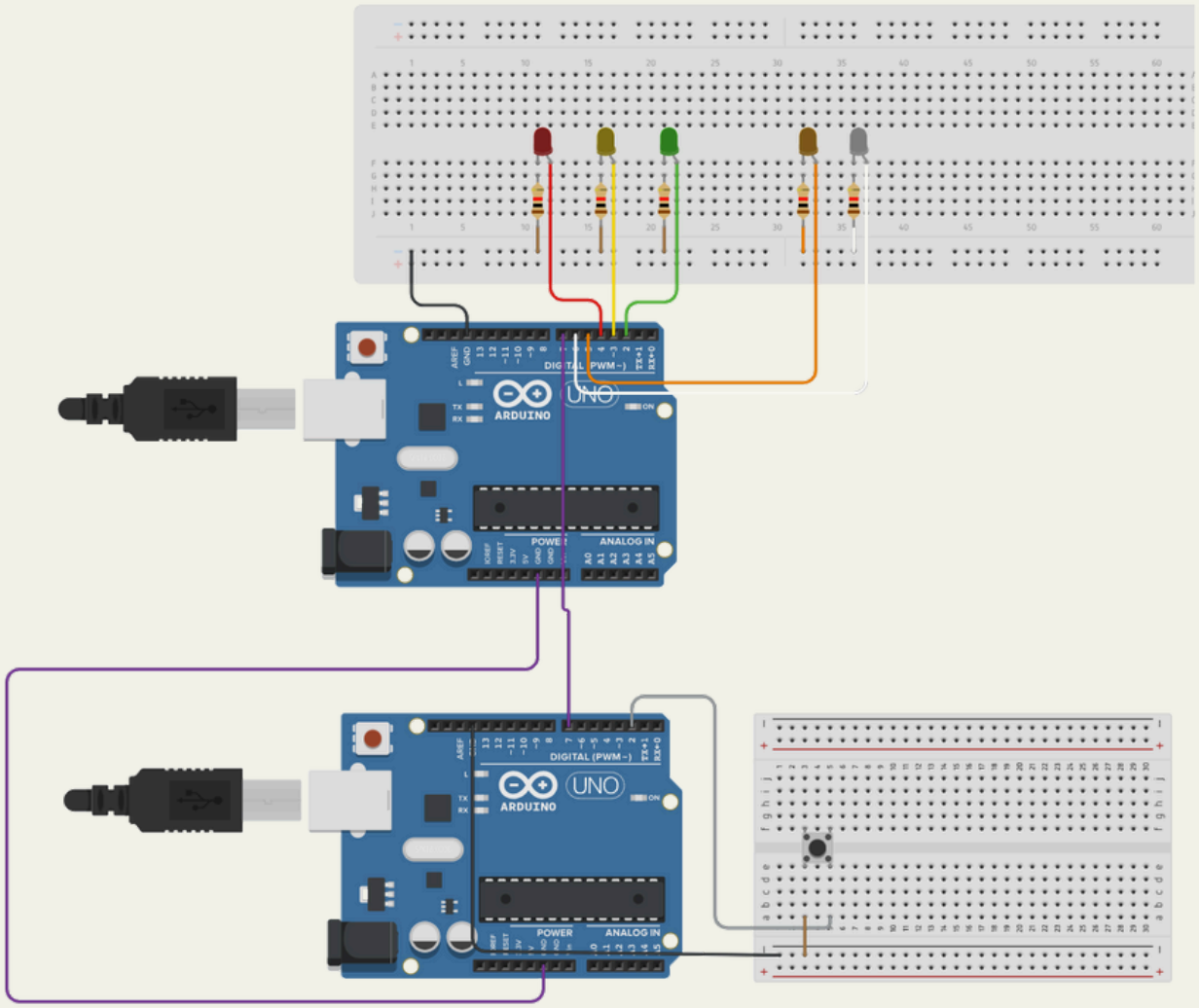


interrupt-based code

```
volatile bool buttonPressed = false;

void setup() {
  // Traffic lights
  pinMode(2, OUTPUT); // Green LED
  pinMode(3, OUTPUT); // Yellow LED
  pinMode(4, OUTPUT); // Red LED

  // Pedestrian lights
  pinMode(8, OUTPUT);      // Go LED
  pinMode(9, OUTPUT);      // Stop LED
  pinMode(13, INPUT_PULLUP); // Button with pull-up resistor

  // Attach interrupt for button press
  attachInterrupt(digitalPinToInterrupt(13), handleButtonPress, FALLING);
}

void loop() {
  if (buttonPressed) {
    // Pedestrian crossing sequence
    digitalWrite(9, LOW);  // Turn off Stop light
    digitalWrite(8, HIGH); // Turn on Go light
    digitalWrite(4, HIGH); // Turn on Red traffic light
    digitalWrite(3, LOW);
    digitalWrite(2, LOW);
    delay(5000);           // Pedestrian Go time
    digitalWrite(8, LOW);  // Turn off Go light
    digitalWrite(9, HIGH); // Turn Stop light back on
    buttonPressed = false; // Reset button flag
```

```
  } else {
    // Normal traffic light cycle
    digitalWrite(9, HIGH); // Ensure Stop light is on
    digitalWrite(4, HIGH); // Red traffic light
    digitalWrite(3, LOW);
    digitalWrite(2, LOW);
    delay(5000);

    digitalWrite(4, LOW);  // Yellow traffic light
    digitalWrite(3, HIGH);
    digitalWrite(2, LOW);
    delay(1000);

    digitalWrite(4, LOW);  // Green traffic light
    digitalWrite(3, LOW);
    digitalWrite(2, HIGH);
    delay(5000);

    digitalWrite(4, LOW);  // Yellow light for reset
    digitalWrite(3, HIGH);
    digitalWrite(2, LOW);
    delay(1000);
  }
}

void handleButtonPress() {
  buttonPressed = true;
}
```

simulation



master code

```
// Master Arduino
void setup() {
  // Set LED pins as outputs
  pinMode(4, OUTPUT); // Red LED
  pinMode(3, OUTPUT); // Yellow LED
  pinMode(2, OUTPUT); // Green LED
  pinMode(5, OUTPUT); // Orange LED (Pedestrian Stop)
  pinMode(6, OUTPUT); // White LED (Pedestrian Go)

  // Set initial states
  digitalWrite(4, LOW);  // Red off
  digitalWrite(3, LOW);  // Yellow off
  digitalWrite(2, LOW);  // Green off
  digitalWrite(5, HIGH); // Pedestrian stop light on
  digitalWrite(6, LOW);  // Pedestrian go light off

  pinMode(7, INPUT); // Button signal from slave
}

void loop() {
  if (digitalRead(7) == HIGH) { // Button pressed
    handlePedestrianCrossing();
  } else {
    trafficCycle();
  }
}

void trafficCycle() {
  // Red light
  digitalWrite(4, HIGH);
  digitalWrite(3, LOW);
  digitalWrite(2, LOW);
  digitalWrite(5, HIGH); // Pedestrian stop light on
  delay(5000);

  // Yellow light
  digitalWrite(4, LOW);
  digitalWrite(3, HIGH);
  digitalWrite(2, LOW);
  delay(5000);

  // Green light
  digitalWrite(4, LOW);
  digitalWrite(3, LOW);
  digitalWrite(2, HIGH);
  delay(5000);
}

void handlePedestrianCrossing() {
  // Stop traffic
  digitalWrite(4, HIGH);  // Red on
  digitalWrite(3, LOW);   // Yellow off
  digitalWrite(2, LOW);   // Green off

  // Allow pedestrians to cross
  digitalWrite(5, LOW);   // Pedestrian stop light off
  digitalWrite(6, HIGH);  // Pedestrian go light on
  delay(5000);

  // Restore pedestrian stop light
  digitalWrite(5, HIGH);  // Pedestrian stop light on
  digitalWrite(6, LOW);   // Pedestrian go light off
}
```

slave code

```
void setup() {
  pinMode(2, INPUT_PULLUP); // Button with internal pull-up
  pinMode(7, OUTPUT);       // Signal to Master
}

void loop() {
  int buttonState = digitalRead(2); // Read the state of the button

  if (buttonState == LOW) {         // Button is pressed
    digitalWrite(7, HIGH);          // Send signal to Master
    delay(500);                     // Hold the signal for 500 ms
    digitalWrite(7, LOW);           // Reset signal
  }
}
```