Traffic light Control System

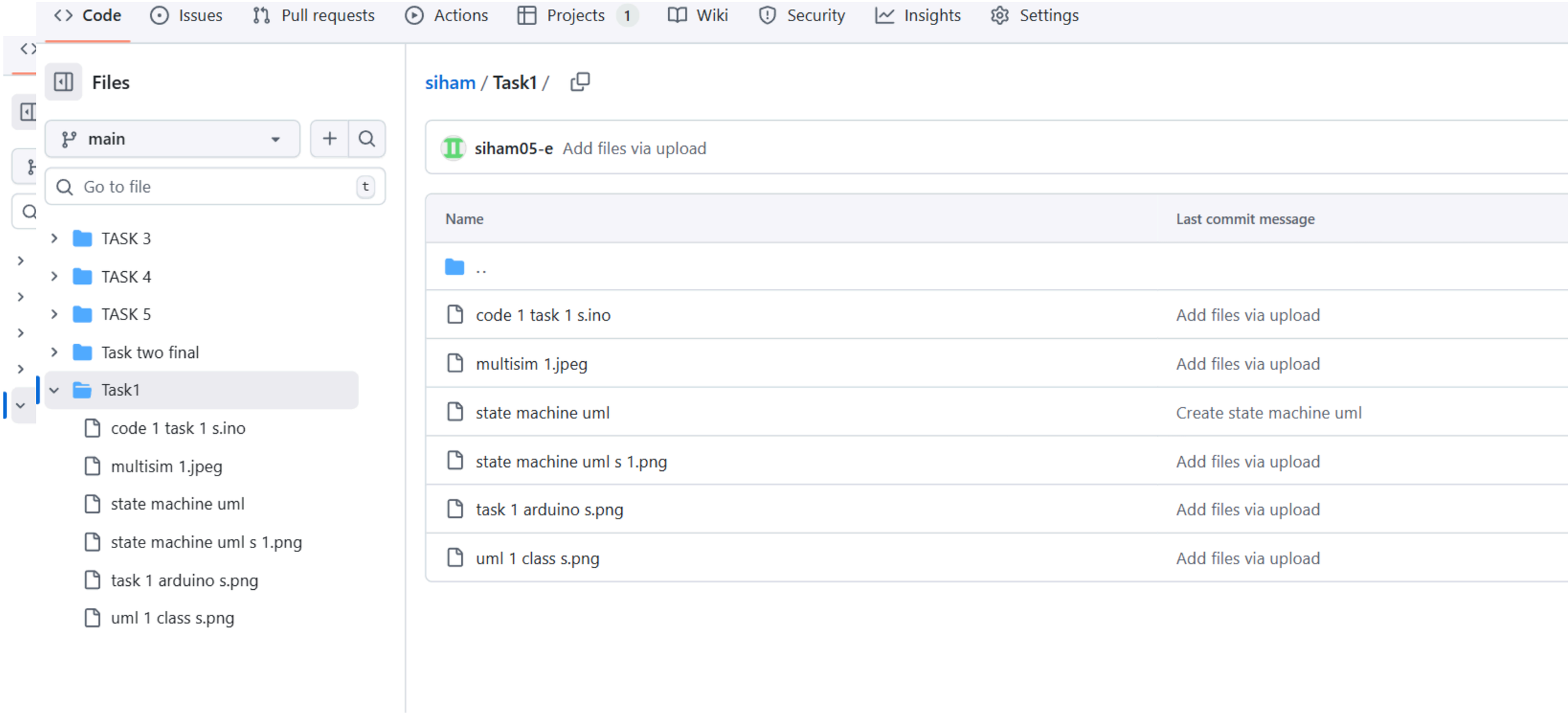
Embeded Systems

Siham El Rhazouany
Trima Clara Halder
Abdullah Hossameldin Mohammed Mostafa

GET STARTED

Content overview

TASK 1	scenario of a traffic light system
TASK 2	pedestrian lights
TASK 3	Real Arduino of task 1
TASK 4	real Arduino of task 2
TASK 5	UCS


Code
Issues
Pull requests
Actions
Projects 1
Wiki
Security
Insights
Settings
Files
main
Go to file t
TASK 3
TASK 4
TASK 5
Task two final
Task1
code 1 task 1 s.ino
multisim 1.jpeg
state machine uml
state machine uml s 1.png
task 1 arduino s.png
uml 1 class s.png
siham / Task1 /
siham05-e Add files via upload
Name
Last commit message
..
code 1 task 1 s.ino
Add files via upload
multisim 1.jpeg
Add files via upload
state machine uml
Create state machine uml
state machine uml s 1.png
Add files via upload
task 1 arduino s.png
Add files via upload
uml 1 class s.png
Add files via upload

Task 1:

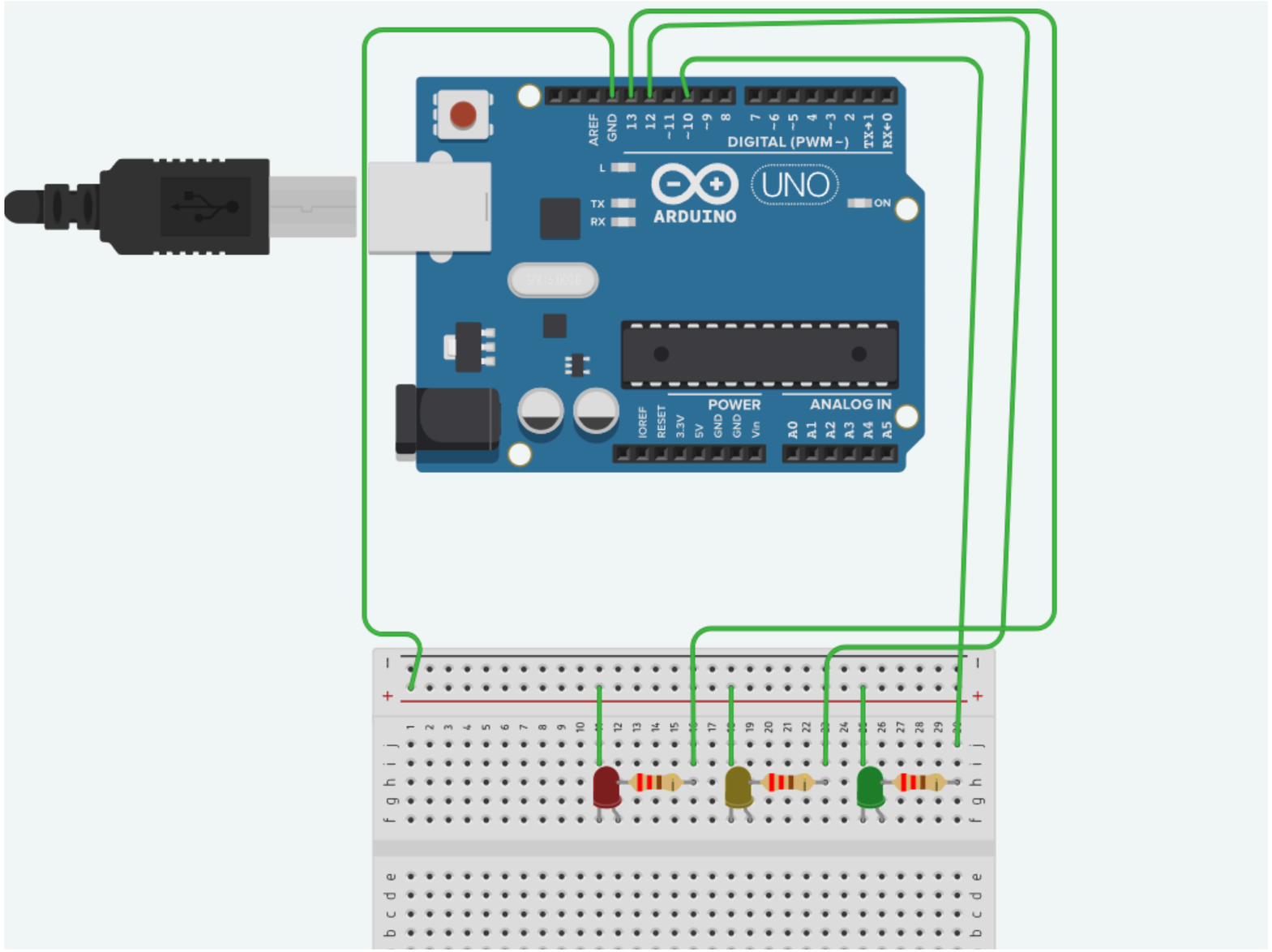
traffic lights

ts Actions Projects 1 Wiki

siham / Task1 /

siham05-e Add files via upload

Name
..
code 1 task 1 s.ino
multisim 1.jpeg
state machine uml
state machine uml s 1.png
task 1 arduino s.png
uml 1 class s.png



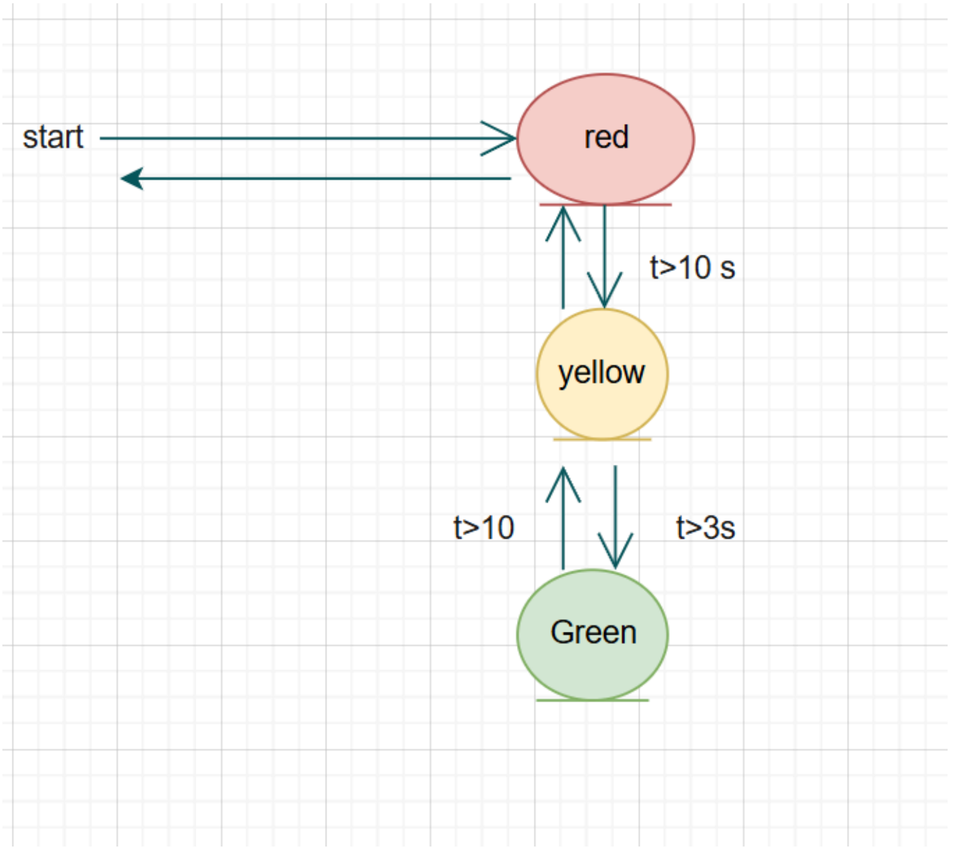
UML Diagrams

Traffic light system

int PIN_GREEN
int PIN_YELLOW
int PIN_RED

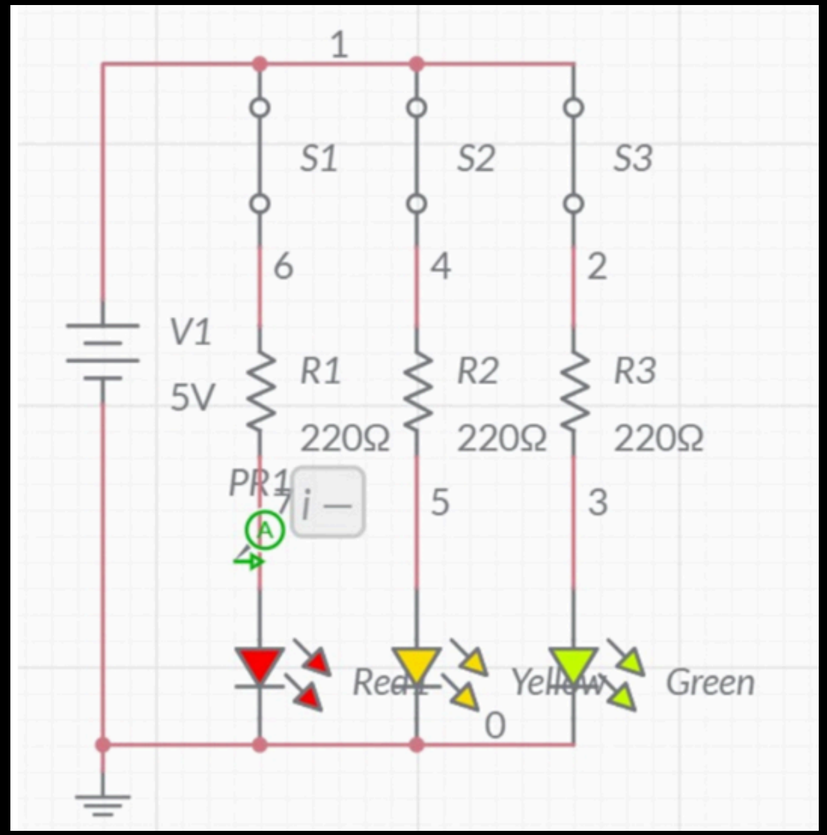
state(Green, yellow, Red)
+unsigned long timer yellow 3s
+unsigned long timer green, red 10s
currentState = RED;
targetState = GREEN;
previousMillis = 0
void loop:
unsigned long currentMillis = millis

class diagram



state machine

Multisim



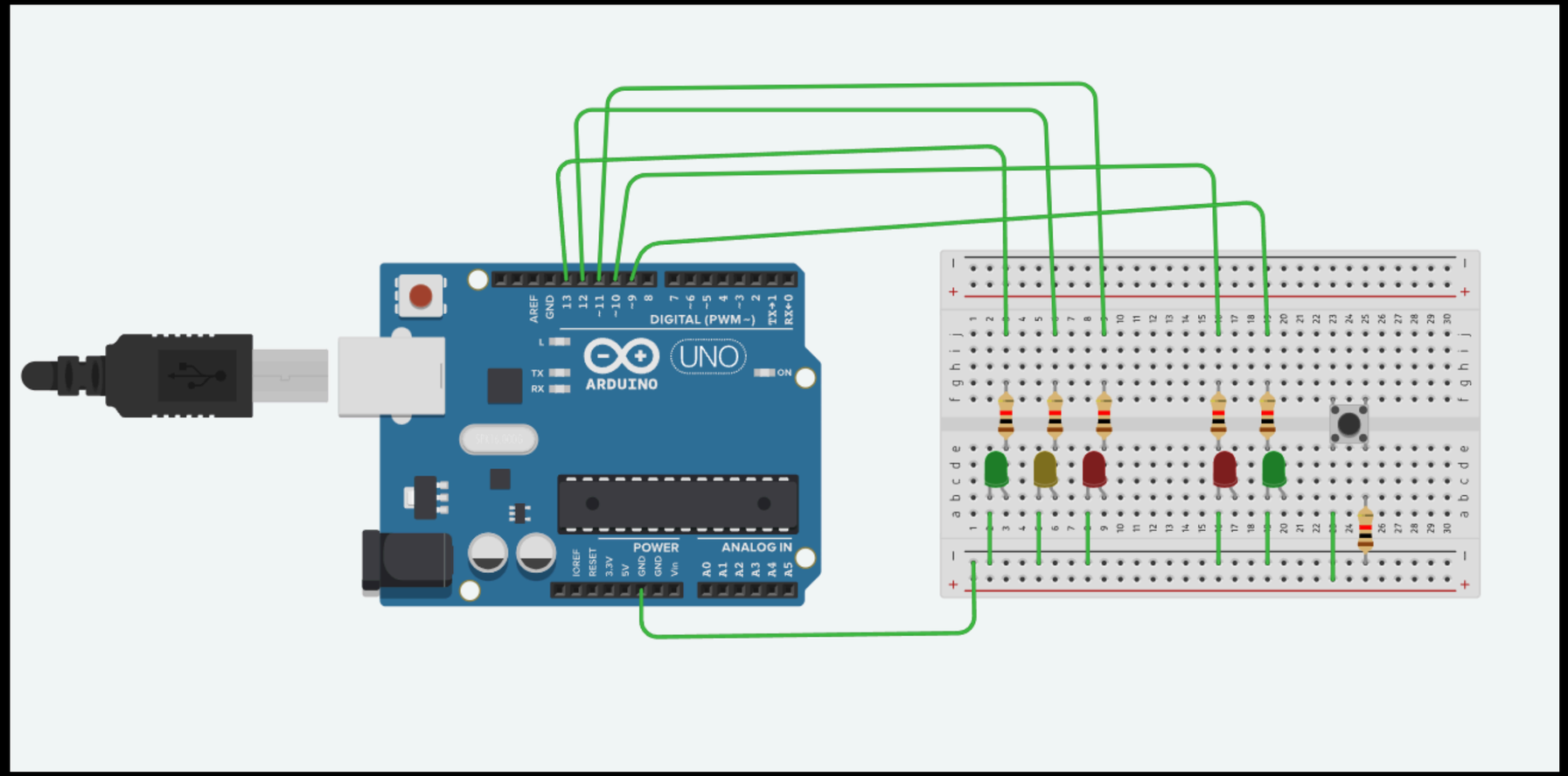
Task :1

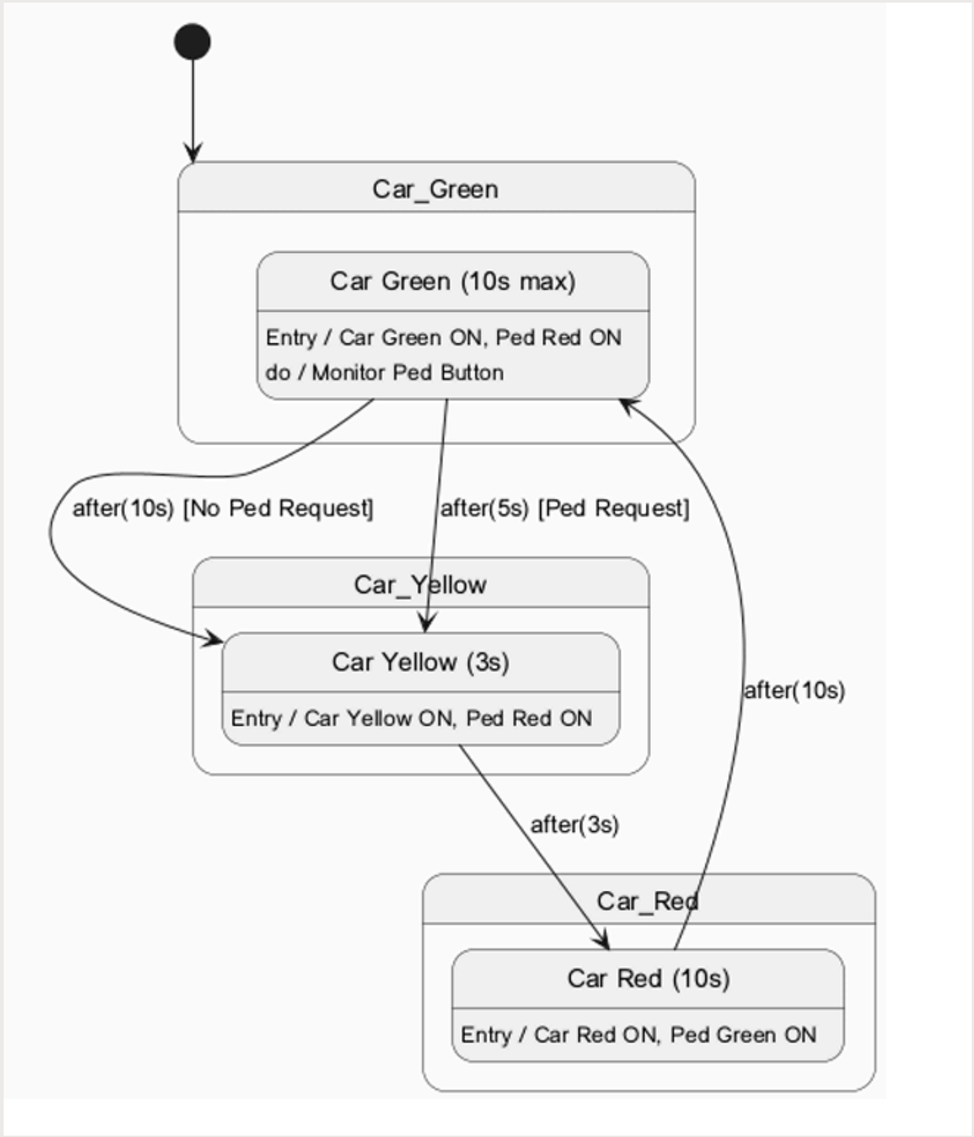
Code

```
// Traffic Light Control System
// Using millis() for non-blocking timing

const int PIN_GREEN  = 11;
const int PIN_YELLOW = 12;
const int PIN_RED    = 13;

class TrafficLight {
  // Timer durations in milliseconds
  const unsigned long TIMER_YELLOW = 3000;   // 3 seconds for yellow
  const unsigned long TIMER_MAIN   = 10000;  // 10 seconds for red/green

  // Pin numbers
  int pinGreen;
  int pinYellow;
  int pinRed;

  // Timing
  unsigned long previousMillis;

  // Traffic light states
  enum State {
    GREEN,
```

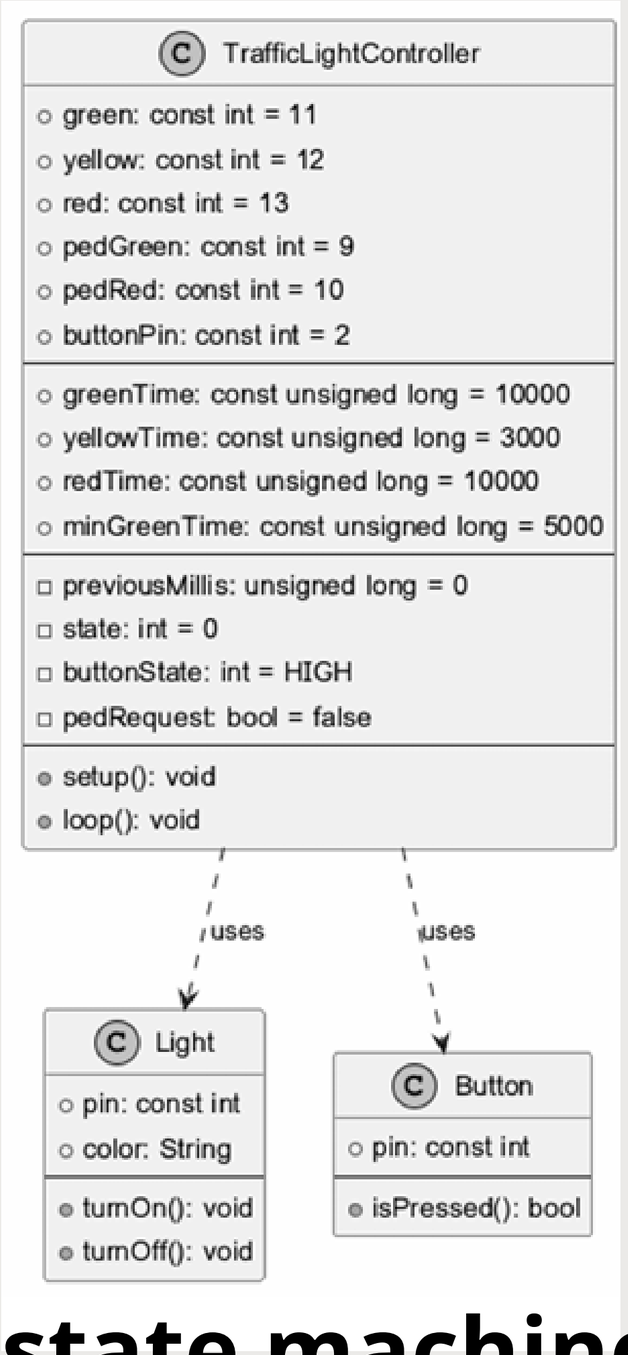
```
    YELLOW,
    RED
  };

  State currentState;
  State targetState;

public:
  TrafficLight(int greenLED, int yellowLED, int redLED) {
    pinGreen  = greenLED;
    pinYellow = yellowLED;
    pinRed    = redLED;

    pinMode(pinGreen, OUTPUT);
    pinMode(pinYellow, OUTPUT);
    pinMode(pinRed, OUTPUT);

    // Start on RED, will move to GREEN after cycle
    currentState   = RED;
    targetState    = GREEN;
    previousMillis = 0;     // will switch after TIMER_MAIN ms
    red();
  }

  void loop() {
    unsigned long currentMillis = millis();

    switch (currentState) {
      case RED:
        if (currentMillis - previousMillis >= TIMER_MAIN) {
          previousMillis = currentMillis;
          currentState   = YELLOW;
          targetState    = GREEN;
          yellow();
        }
        break;

      case GREEN:
        if (currentMillis - previousMillis >= TIMER_MAIN) {
          previousMillis = currentMillis;
          currentState   = YELLOW;
          targetState    = RED;
          yellow();
        }
        break;

      case YELLOW:
        if (currentMillis - previousMillis >= TIMER_YELLOW) {
          previousMillis = currentMillis;
          if (targetState == GREEN) {
            currentState = GREEN;
            green();
          } else {
            currentState = RED;
            red();
          }
        }
        break;
    }
  }

private:
  void green() {
    digitalWrite(pinGreen, HIGH);
    digitalWrite(pinYellow, LOW);
    digitalWrite(pinRed, LOW);
  }

  void yellow() {
    digitalWrite(pinGreen, LOW);
    digitalWrite(pinYellow, HIGH);
    digitalWrite(pinRed, LOW);
  }

  void red() {
    digitalWrite(pinGreen, LOW);
    digitalWrite(pinYellow, LOW);
    digitalWrite(pinRed, HIGH);
  }

  void off() { // unused, but available if needed
    digitalWrite(pinGreen, LOW);
    digitalWrite(pinYellow, LOW);
    digitalWrite(pinRed, LOW);
  }
};

// Create traffic light object
TrafficLight trafficLight(PIN_GREEN, PIN_YELLOW, PIN_RED);

void setup() {
  // nothing extra needed for now
}

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

Task :2





Class Diagram



state machine

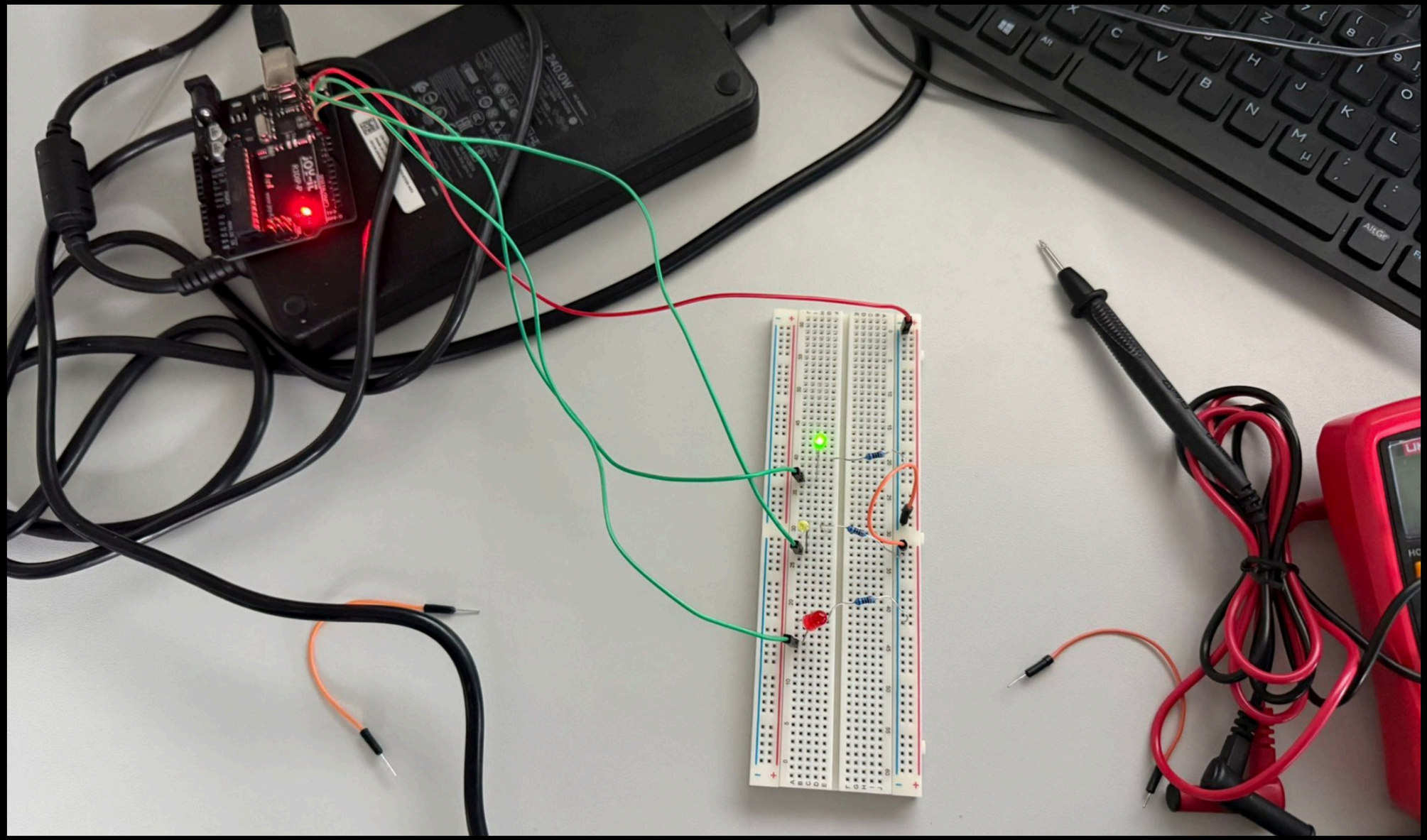
code

```
const int carGreen  = 11;
const int carYellow = 12;
const int carRed    = 13;

const int pedGreen  = 9;
const int pedRed    = 10;

const int buttonPin = 8;  // pedestrian button

// ----- Times -----
const unsigned long carGreenTime = 10000;  // 10 s green for cars
const unsigned long carYellowTime = 5000;  // 5 s yellow
const unsigned long pedGreenTime  = 5000;  // 5 s pedestrian green
const unsigned long allRedTime    = 2000;  // 2 s all red

// ----- States -----
enum State {
  CAR_GREEN,
  CAR_YELLOW,
  PED_GREEN,
  CAR_RED_ALL
};

State state = CAR_GREEN;

unsigned long previousMillis = 0;
bool pedRequest = false;

// ------------------------------------
void setup() {
  pinMode(carGreen, OUTPUT);
  pinMode(carYellow, OUTPUT);
  pinMode(carRed, OUTPUT);

  pinMode(pedGreen, OUTPUT);
  pinMode(pedRed, OUTPUT);

  pinMode(buttonPin, INPUT_PULLUP);
```

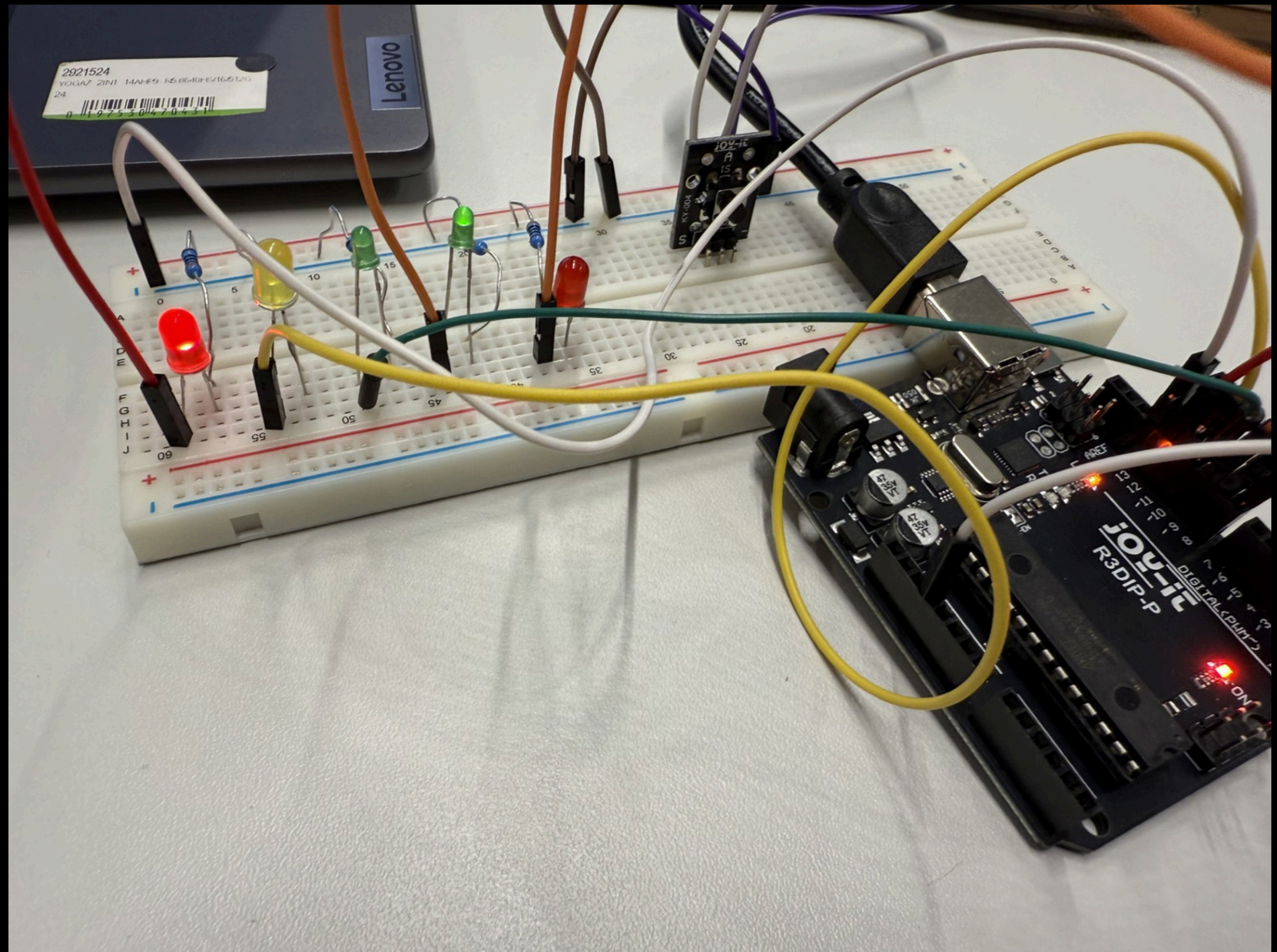
```
  digitalWrite(carGreen, HIGH);
  digitalWrite(carYellow, LOW);
  digitalWrite(carRed, LOW);

  digitalWrite(pedGreen, LOW);
  digitalWrite(pedRed, HIGH);

  previousMillis = millis();
}

// ------------------------------------
void loop() {
  unsigned long currentMillis = millis();

  // Read button
  bool buttonPressed = (digitalRead(buttonPin) == LOW);
  if (buttonPressed) {
    pedRequest = true;
  }

  switch (state) {

    // ----- CARS GREEN -----
    case CAR_GREEN:
      digitalWrite(carGreen, HIGH);
      digitalWrite(carYellow, LOW);
      digitalWrite(carRed, LOW);
      digitalWrite(pedGreen, LOW);
      digitalWrite(pedRed, HIGH);

      // Immediate switch if button pressed
      if (pedRequest) {
        state = CAR_YELLOW;
        previousMillis = currentMillis;
      }
      // Normal timing
      else if (currentMillis - previousMillis >= carGreenTime) {
        state = CAR_YELLOW;
        previousMillis = currentMillis;
      }
      break;

    // ----- CARS YELLOW -----
    case CAR_YELLOW:
      digitalWrite(carGreen, LOW);
      digitalWrite(carYellow, HIGH);
      digitalWrite(carRed, LOW);
      digitalWrite(pedGreen, LOW);
      digitalWrite(pedRed, HIGH);

      if (currentMillis - previousMillis >= carYellowTime) {
        state = PED_GREEN;
        previousMillis = currentMillis;
      }
      break;

    // ----- PEDESTRIAN GREEN -----
```

```
    case PED_GREEN:
      digitalWrite(carGreen, LOW);
      digitalWrite(carYellow, LOW);
      digitalWrite(carRed, HIGH);
      digitalWrite(pedGreen, HIGH);
      digitalWrite(pedRed, LOW);

      if (currentMillis - previousMillis >= pedGreenTime) {
        state = CAR_RED_ALL;
        previousMillis = currentMillis;
        pedRequest = false;
      }
      break;

    // ----- ALL RED -----
    case CAR_RED_ALL:
      digitalWrite(carGreen, LOW);
      digitalWrite(carYellow, LOW);
      digitalWrite(carRed, HIGH);
      digitalWrite(pedGreen, LOW);
      digitalWrite(pedRed, HIGH);

      if (currentMillis - previousMillis >= allRedTime) {
        state = CAR_GREEN;
        previousMillis = currentMillis;
      }
      break;
  }
} // ← this closes loop()
```

task 3



task 4



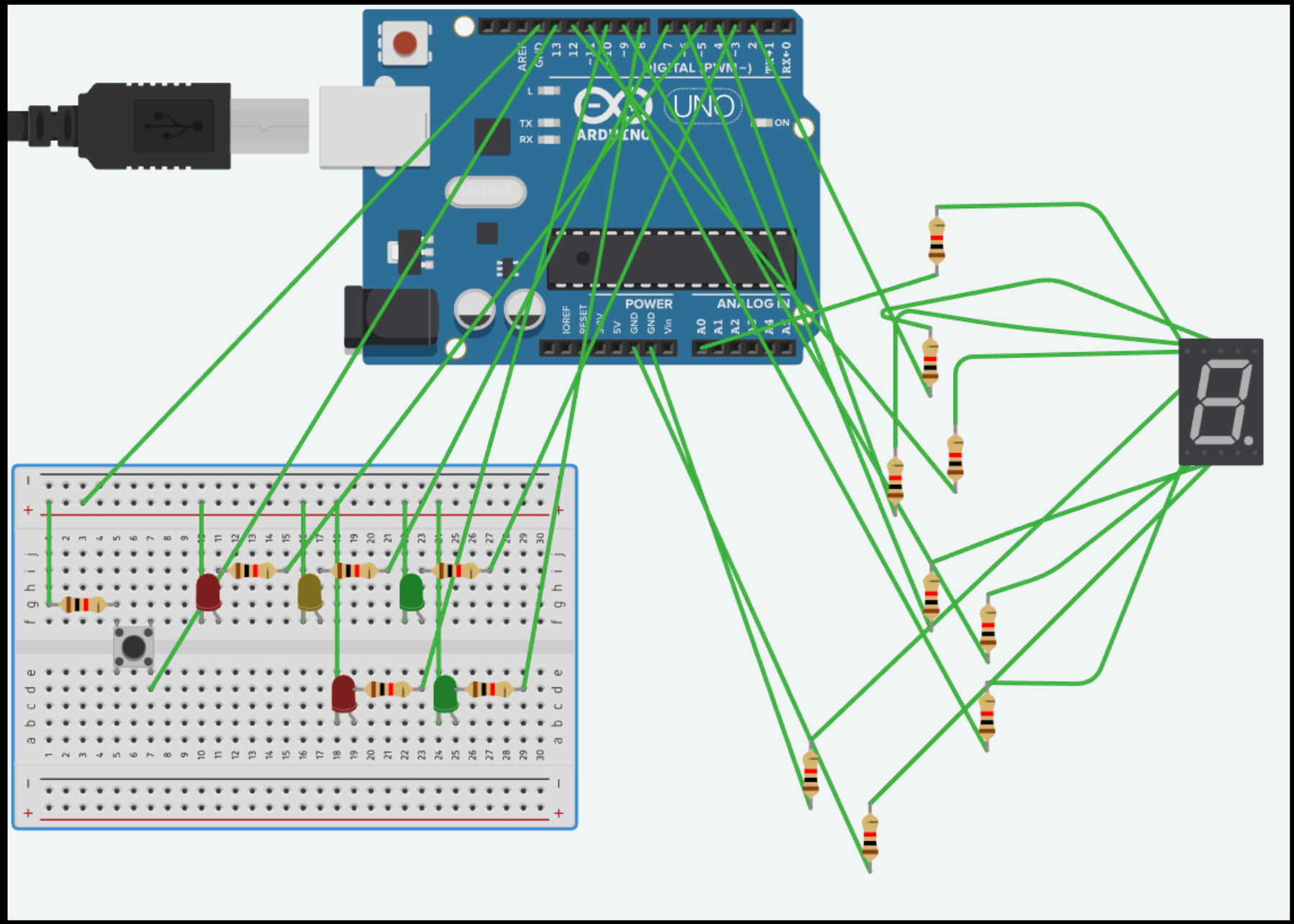
code

```
const int PIN_GREEN  = 11;
const int PIN_YELLOW = 12;
const int PIN_RED    = 13;

class TrafficLight {
  // Timer durations in milliseconds
  const unsigned long TIMER_YELLOW = 3000;   // 3 seconds for yellow
  const unsigned long TIMER_MAIN   = 10000;  // 10 seconds for red/green

  // Pin numbers
  int pinGreen;
  int pinYellow;
  int pinRed;

  // Timing
  unsigned long previousMillis;

  // Traffic light states
  enum State {
    GREEN,
    YELLOW,
    RED
  };

  State currentState;
  State targetState;

public:
  TrafficLight(int greenLED, int yellowLED, int redLED) {
    pinGreen  = greenLED;
    pinYellow = yellowLED;
    pinRed    = redLED;

    pinMode(pinGreen, OUTPUT);
    pinMode(pinYellow, OUTPUT);
    pinMode(pinRed, OUTPUT);

    // Start on RED, will move to GREEN after cycle
    currentState   = RED;
    targetState    = GREEN;
    previousMillis = 0;     // will switch after TIMER_MAIN ms
    red();
  }

  void loop() {
    unsigned long currentMillis = millis();

    switch (currentState) {
      case RED:
        if (currentMillis - previousMillis >= TIMER_MAIN) {
          previousMillis = currentMillis;
          currentState   = YELLOW;
          targetState    = GREEN;
          yellow();
        }
        break;

      case GREEN:
        if (currentMillis - previousMillis >= TIMER_MAIN) {
          previousMillis = currentMillis;
          currentState   = YELLOW;
          targetState    = RED;
          yellow();
        }
        break;
```

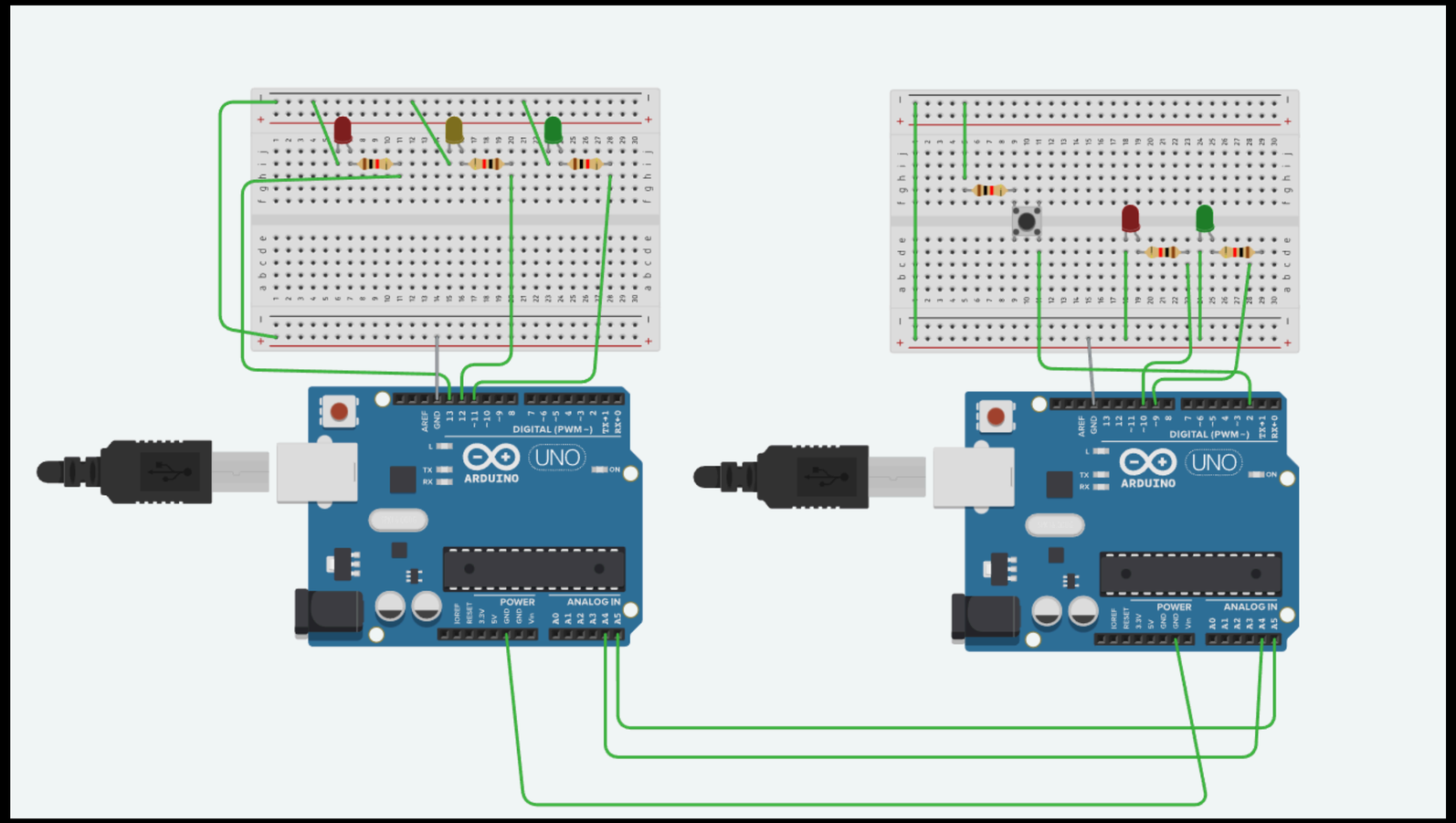
```
      case YELLOW:
        if (currentMillis - previousMillis >= TIMER_YELLOW) {
          previousMillis = currentMillis;
          if (targetState == GREEN) {
            currentState = GREEN;
            green();
          } else {
            currentState = RED;
            red();
          }
        }
        break;
    }
  }

private:
  void green() {
    digitalWrite(pinGreen, HIGH);
    digitalWrite(pinYellow, LOW);
    digitalWrite(pinRed, LOW);
  }

  void yellow() {
    digitalWrite(pinGreen, LOW);
    digitalWrite(pinYellow, HIGH);
    digitalWrite(pinRed, LOW);
  }

  void red() {
    digitalWrite(pinGreen, LOW);
    digitalWrite(pinYellow, LOW);
    digitalWrite(pinRed, HIGH);
  }

  void off() { // unused, but available if needed
    digitalWrite(pinGreen, LOW);
    digitalWrite(pinYellow, LOW);
    digitalWrite(pinRed, LOW);
  }
};

// Create traffic light object
TrafficLight trafficLight(PIN_GREEN, PIN_YELLOW, PIN_RED);

void setup() {
  // nothing extra needed for now
}

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

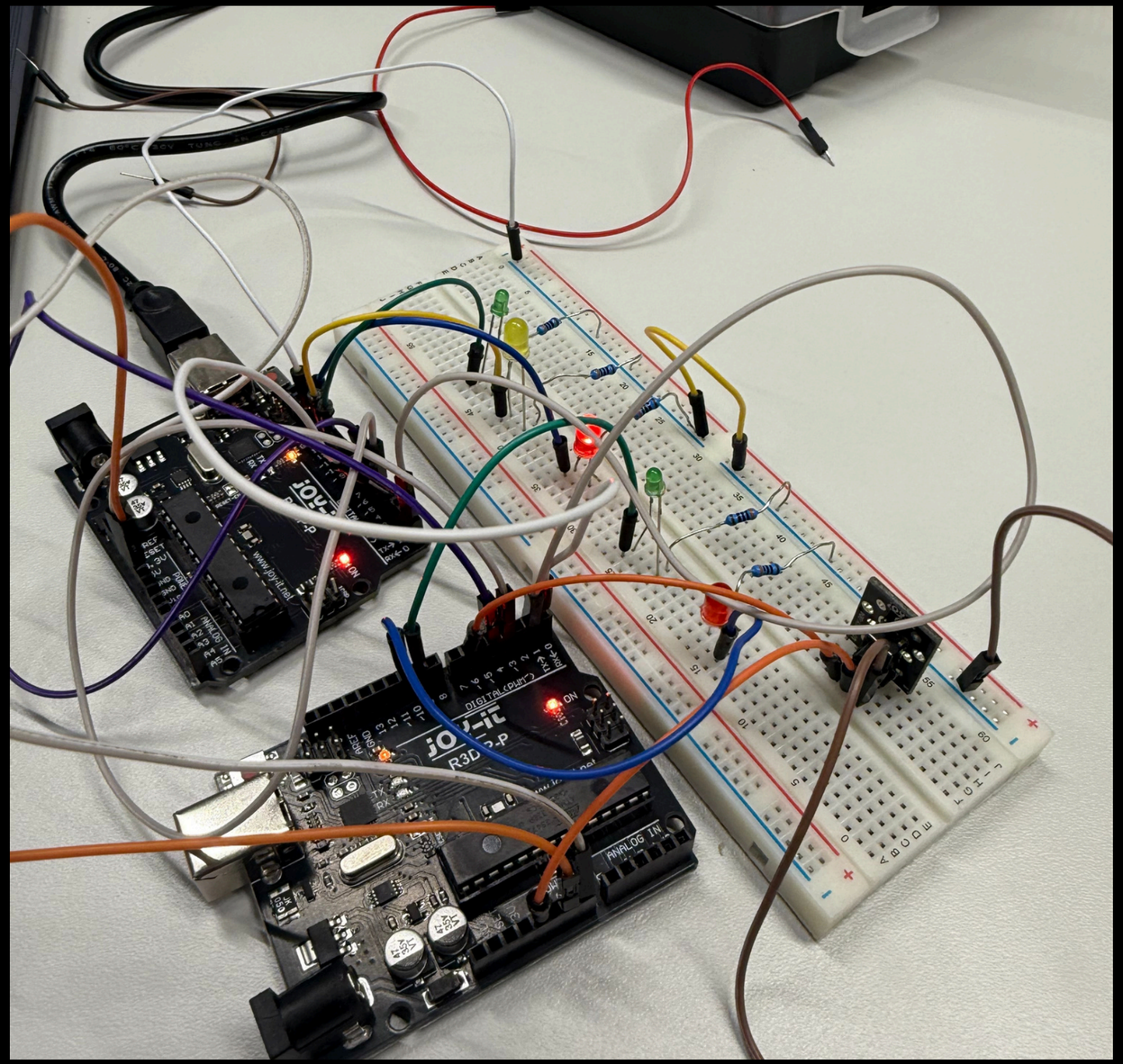
code

```
const int green = 11;
const int yellow = 12;
const int red = 13;
const int pedGreen = 9;
const int pedRed = 10;

unsigned long previousMillis = 0;
int state = 0;
const unsigned long greenTime = 10000;   // 10 seconds green for cars
const unsigned long yellowTime = 3000;   // 3 seconds yellow
const unsigned long redTime = 10000;     // 10 seconds red for cars (pedestrians walk)

void setup() {
  pinMode(green, OUTPUT);
  pinMode(yellow, OUTPUT);
  pinMode(red, OUTPUT);
  pinMode(pedGreen, OUTPUT);
  pinMode(pedRed, OUTPUT);
}

void loop() {
  unsigned long currentMillis = millis();

  switch(state) {
    case 0:  // Green cars, red pedestrians
      digitalWrite(green, HIGH);
      digitalWrite(yellow, LOW);
      digitalWrite(red, LOW);
      digitalWrite(pedGreen, LOW);
      digitalWrite(pedRed, HIGH);
      if (currentMillis - previousMillis >= greenTime) {
        state = 1;
        previousMillis = currentMillis;
      }
      break;

    case 1:  // Yellow cars, red pedestrians
      digitalWrite(green, LOW);
      digitalWrite(yellow, HIGH);
      digitalWrite(red, LOW);
      digitalWrite(pedGreen, LOW);
      digitalWrite(pedRed, HIGH);
      if (currentMillis - previousMillis >= yellowTime) {
        state = 2;
        previousMillis = currentMillis;
      }
      break;

    case 2:  // Red cars, green pedestrians
      digitalWrite(green, LOW);
      digitalWrite(yellow, LOW);
      digitalWrite(red, HIGH);
      digitalWrite(pedGreen, HIGH);
      digitalWrite(pedRed, LOW);
      if (currentMillis - previousMillis >= redTime) {
        state = 0;
        previousMillis = currentMillis;
      }
      break;
  }
}
```


UNO
ON
TX
RX
POWER
ANALOG IN
IOREF
RESET
3V3
5V
GND
GND
Vin
A0
A1
A2
RX←0
TX→1

task 5


DIGITAL (PWM~)
UNO
ARDUINO
POWER
ANALOG IN
DIGITAL (PWM~)
UNO
ARDUINO
POWER
ANALOG IN



TX.RX

RX;TX