Documentation

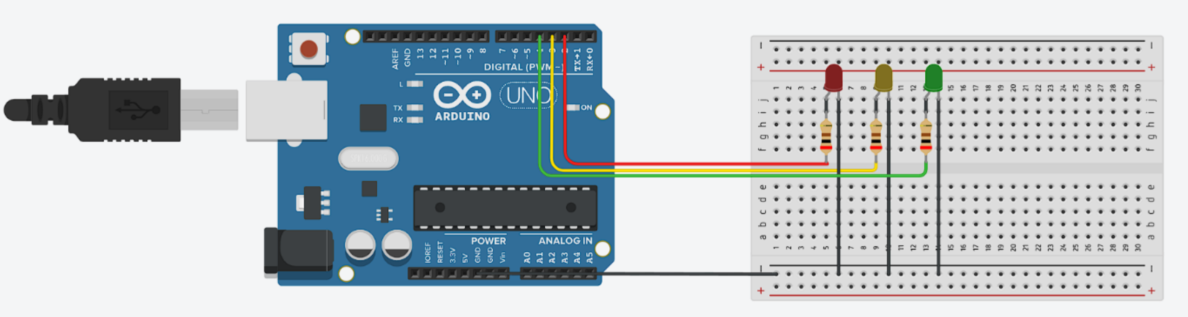
by Andrey Sysoev

Traffic light (task.1)

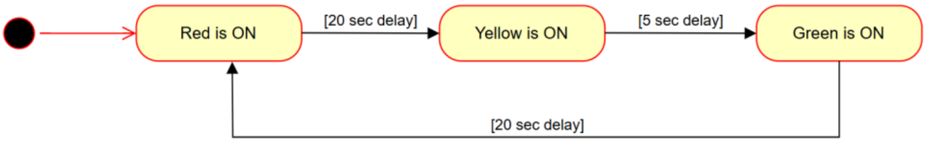
- Objective: Implement a basic traffic light system using Arduino.

The traffic light cycles through RED, YELLOW, and GREEN states with fixed durations.

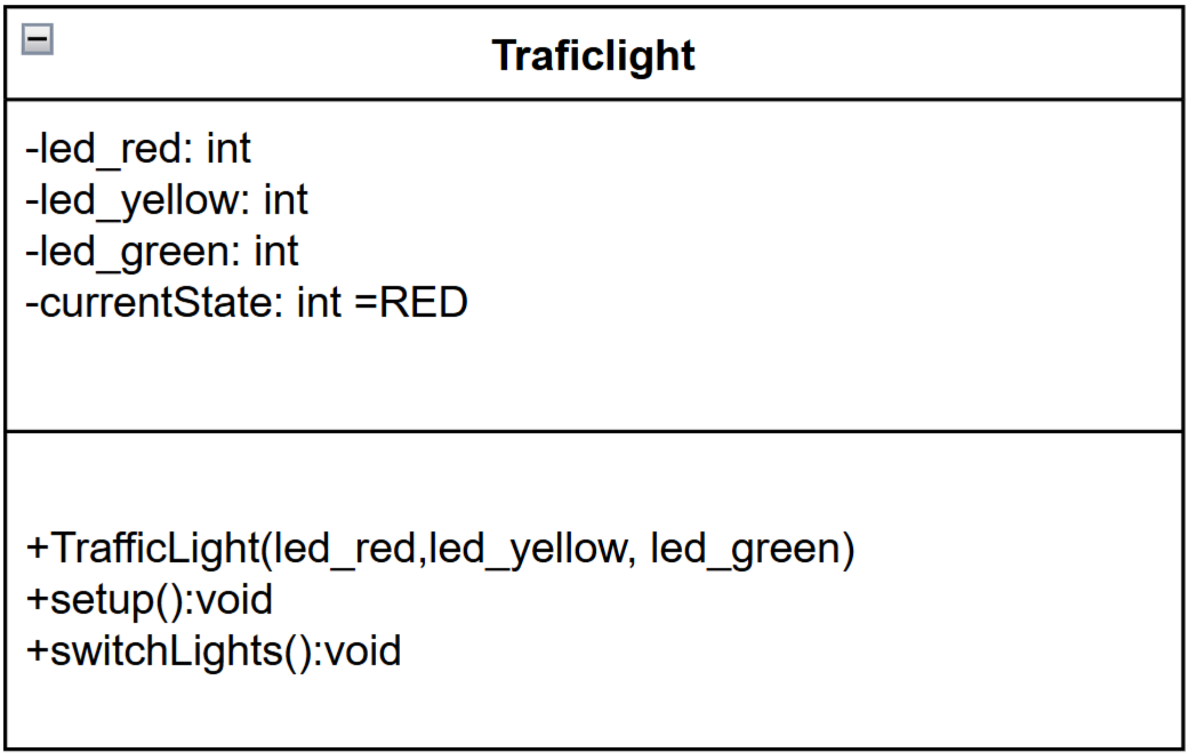
- Circuit model includes: Arduino board, 3 LEDs,3 220 ohm resistors, breadboard, wires.



- State machine diagram:



- Class diagram:



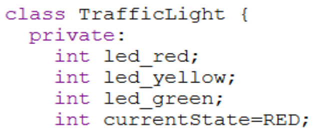
Traficlight

-led_red: int
-led_yellow: int
-led_green: int
-currentState: int =RED

+TrafficLight(led_red,led_yellow, led_green)
+setup():void
+switchLights():void

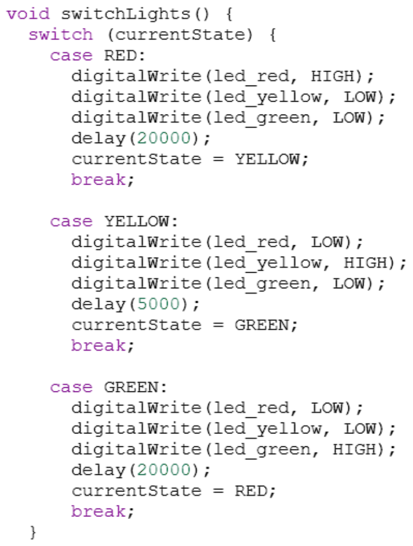
- Explanation of a code:

The traffic light class initialize with 3 pins and a starting state (RED):



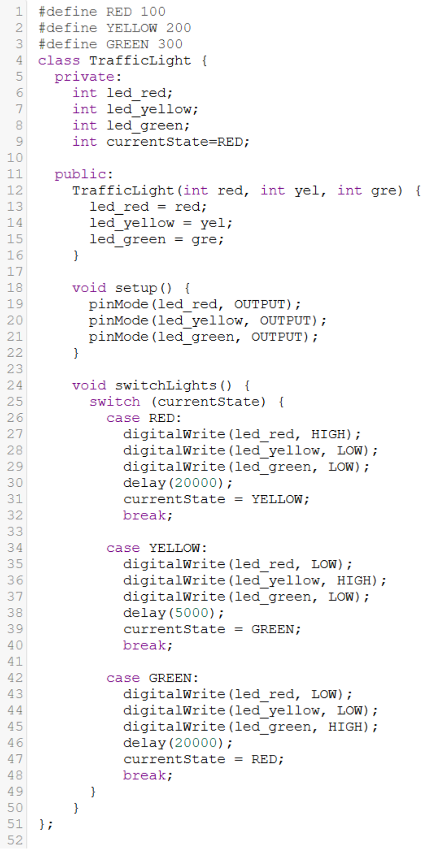
```
class TrafficLight {
  private:
    int led_red;
    int led_yellow;
    int led_green;
    int currentState=RED;
```

Traffic light cycles between 3 states red- yellow – green, with timing red(20 sec) yellow(5 sec) green(20sec):

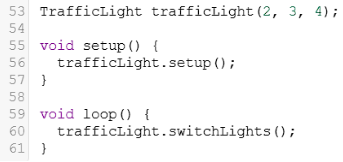


```
void switchLights() {
  switch (currentState) {
    case RED:
      digitalWrite(led_red, HIGH);
      digitalWrite(led_yellow, LOW);
      digitalWrite(led_green, LOW);
      delay(20000);
      currentState = YELLOW;
      break;

    case YELLOW:
      digitalWrite(led_red, LOW);
      digitalWrite(led_yellow, HIGH);
      digitalWrite(led_green, LOW);
      delay(5000);
      currentState = GREEN;
      break;

    case GREEN:
      digitalWrite(led_red, LOW);
      digitalWrite(led_yellow, LOW);
      digitalWrite(led_green, HIGH);
      delay(20000);
      currentState = RED;
      break;
  }
```

- Code:



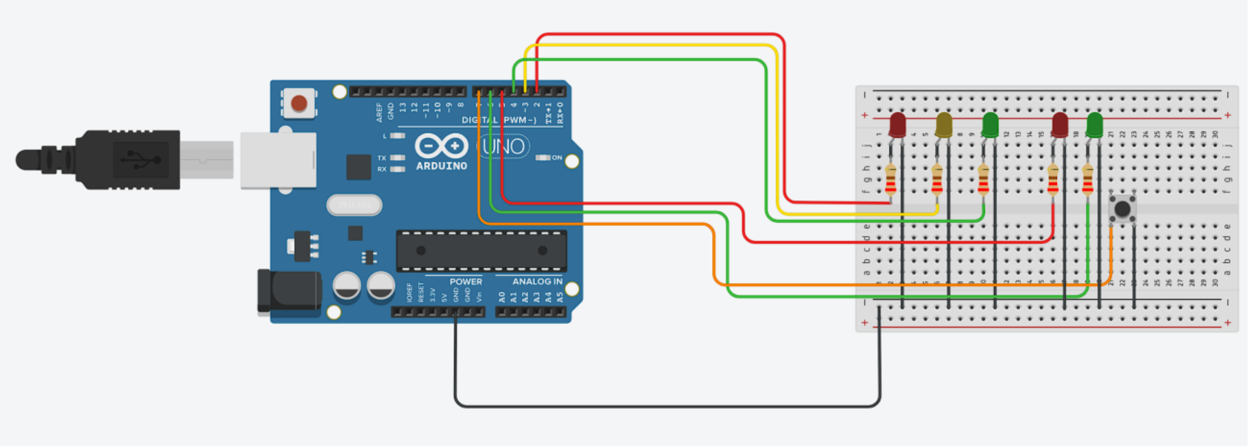
```
#define RED 100
#define YELLOW 200
#define GREEN 300
class TrafficLight {
  private:
    int led_red;
    int led_yellow;
    int led_green;
    int currentState=RED;

  public:
    TrafficLight(int red, int yel, int gre) {
      led_red = red;
      led_yellow = yel;
      led_green = gre;
    }

    void setup() {
      pinMode(led_red, OUTPUT);
      pinMode(led_yellow, OUTPUT);
      pinMode(led_green, OUTPUT);
    }

    void switchLights() {
      switch (currentState) {
        case RED:
          digitalWrite(led_red, HIGH);
          digitalWrite(led_yellow, LOW);
          digitalWrite(led_green, LOW);
          delay(20000);
          currentState = YELLOW;
          break;

        case YELLOW:
          digitalWrite(led_red, LOW);
          digitalWrite(led_yellow, HIGH);
          digitalWrite(led_green, LOW);
          delay(5000);
          currentState = GREEN;
          break;

        case GREEN:
          digitalWrite(led_red, LOW);
          digitalWrite(led_yellow, LOW);
          digitalWrite(led_green, HIGH);
          delay(20000);
          currentState = RED;
          break;
      }
    }
};
```



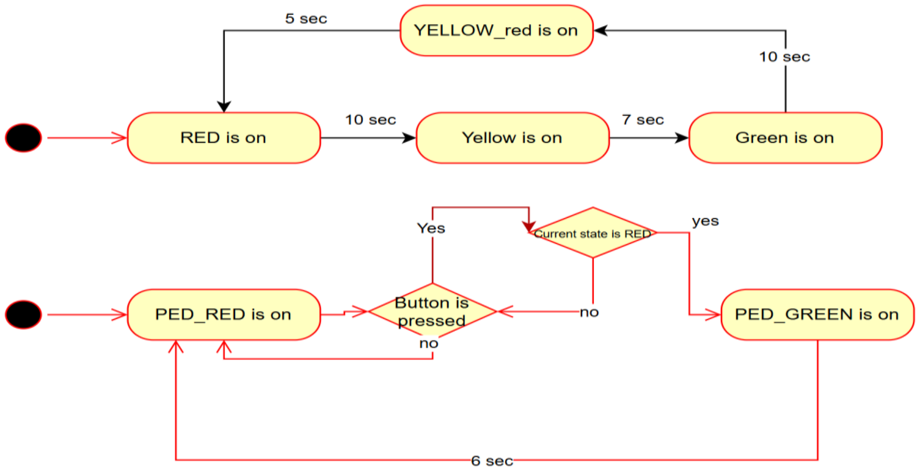
```
TrafficLight trafficLight(2, 3, 4);

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

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

Traffic light with pedestrian light (task.2)

- Objective: extend traffic light system by adding pedestrian light, by pushing button, light for pedestrian sidewalk should be turned green only if current traffic light is red.
- Circuit model includes: Arduino board, 5 LEDs, 5 220 ohm resistors, breadboard, wires.



- State machine diagram:



- Class diagram:

Trafficlight

-led_red: int
-led_yellow: int
-led_green: int
-redDuration: const long
-yellowDuration: const long
-greenDuration: const long
-previousMillis: unsigned long
-current state: int (RED)

+ Trafficlight(int red, yellow, green)
+ setup(): void
+update(): void
+getCurrentstate(): int

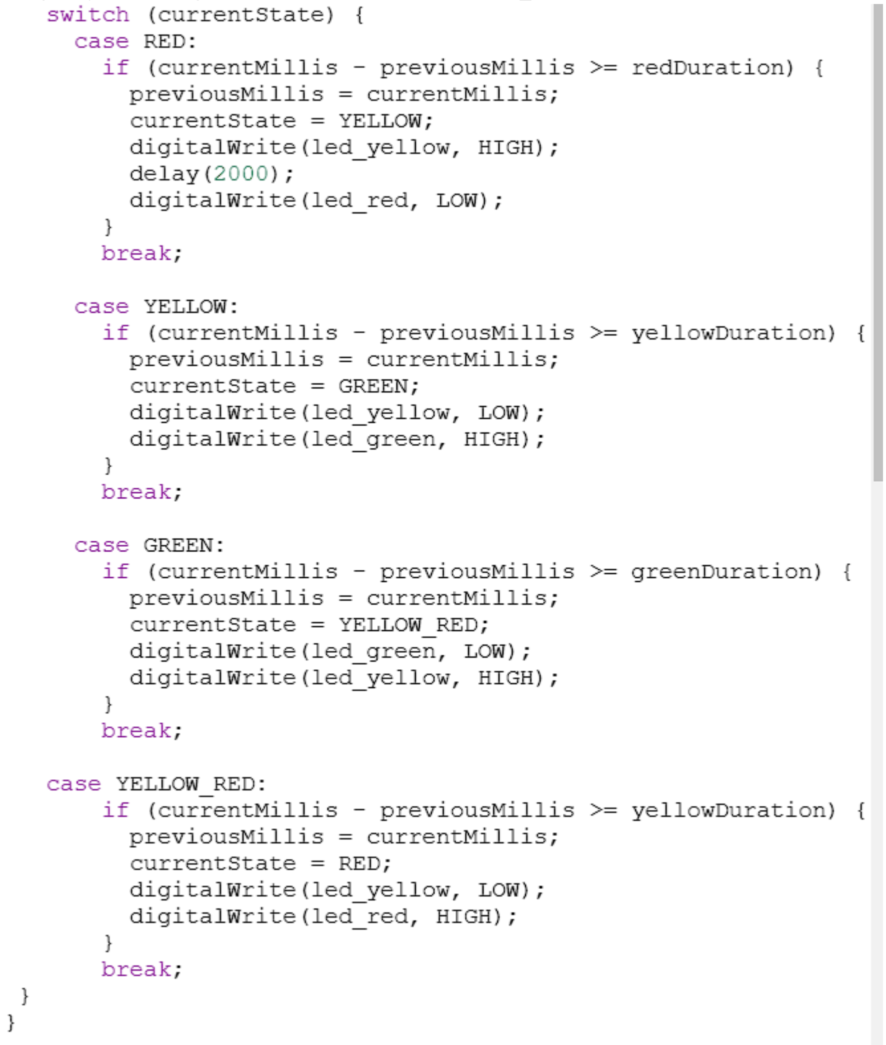
PedestrianLight

-pRed: int
-pGreen: int
-buttonPin: int
-currentPedState: int (PED_RED)
-buttonpressed: bool

+ PedestrianLight(int pRed,Pgreen, button)
+setup(): void
+checkButtonpressed(): void
+update(int Trafficstate): void

Explanation of a code:

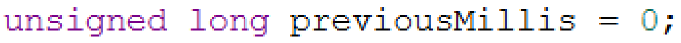
Traffic light cycles through RED, YELLOW, GREEN, YELLOW_RED states:



```
   switch (currentState) {
     case RED:
       if (currentMillis - previousMillis >= redDuration) {
         previousMillis = currentMillis;
         currentState = YELLOW;
         digitalWrite(led_yellow, HIGH);
         delay(2000);
         digitalWrite(led_red, LOW);
       }
       break;

     case YELLOW:
       if (currentMillis - previousMillis >= yellowDuration) {
         previousMillis = currentMillis;
         currentState = GREEN;
         digitalWrite(led_yellow, LOW);
         digitalWrite(led_green, HIGH);
       }
       break;

     case GREEN:
       if (currentMillis - previousMillis >= greenDuration) {
         previousMillis = currentMillis;
         currentState = YELLOW_RED;
         digitalWrite(led_green, LOW);
         digitalWrite(led_yellow, HIGH);
       }
       break;

   case YELLOW_RED:
       if (currentMillis - previousMillis >= yellowDuration) {
         previousMillis = currentMillis;
         currentState = RED;
         digitalWrite(led_yellow, LOW);
         digitalWrite(led_red, HIGH);
       }
       break;
 }
}
```

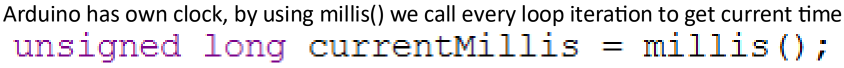
Time between states cycle defines by millis() function.

A variable priviuousMillis is used to store time after last state happened.



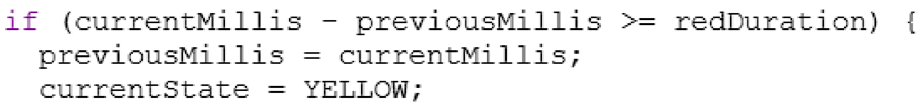
```
unsigned long previousMillis = 0;
```

Arduino has own clock, by using millis() we call every loop iteration to get current time.



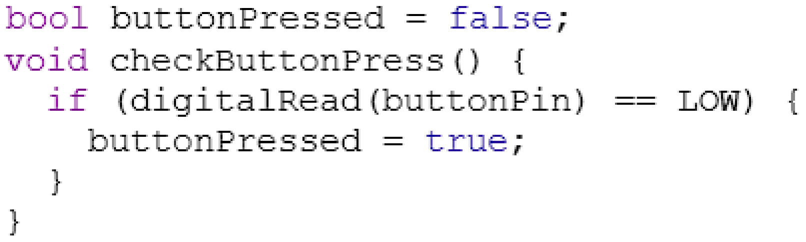
```
unsigned long currentMillis = millis();
```

If a time between current and stored (for example red state duration should be 10 sec) is greater or equal, it resets and moves to next state.



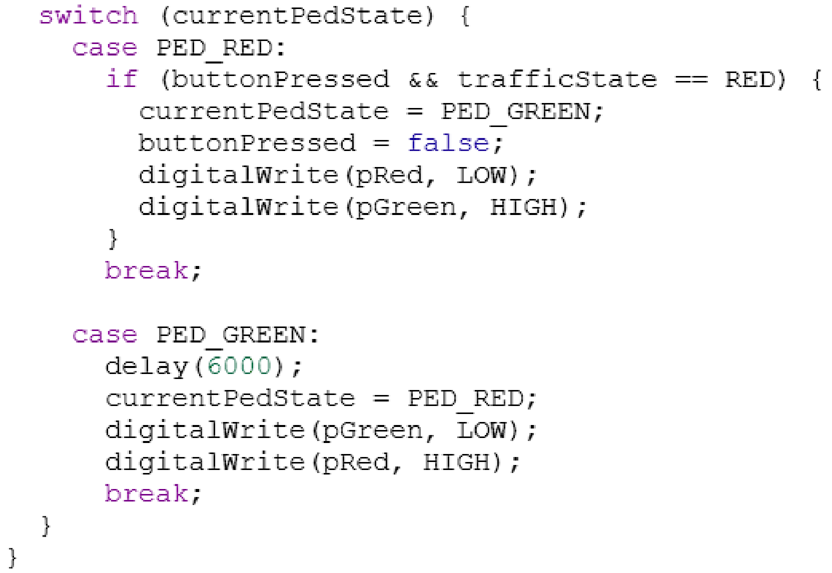
```
if (currentMillis - previousMillis >= redDuration) {
  previousMillis = currentMillis;
  currentState = YELLOW;
```

For button is used Boolean function, if it has been pressed button changes from false to true.



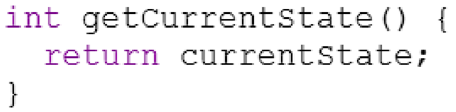
```
bool buttonPressed = false;
void checkButtonPress() {
  if (digitalRead(buttonPin) == LOW) {
    buttonPressed = true;
  }
}
```

If button is pressed and current state of traffic light is red, pedestrian light turns green for 6 sec.



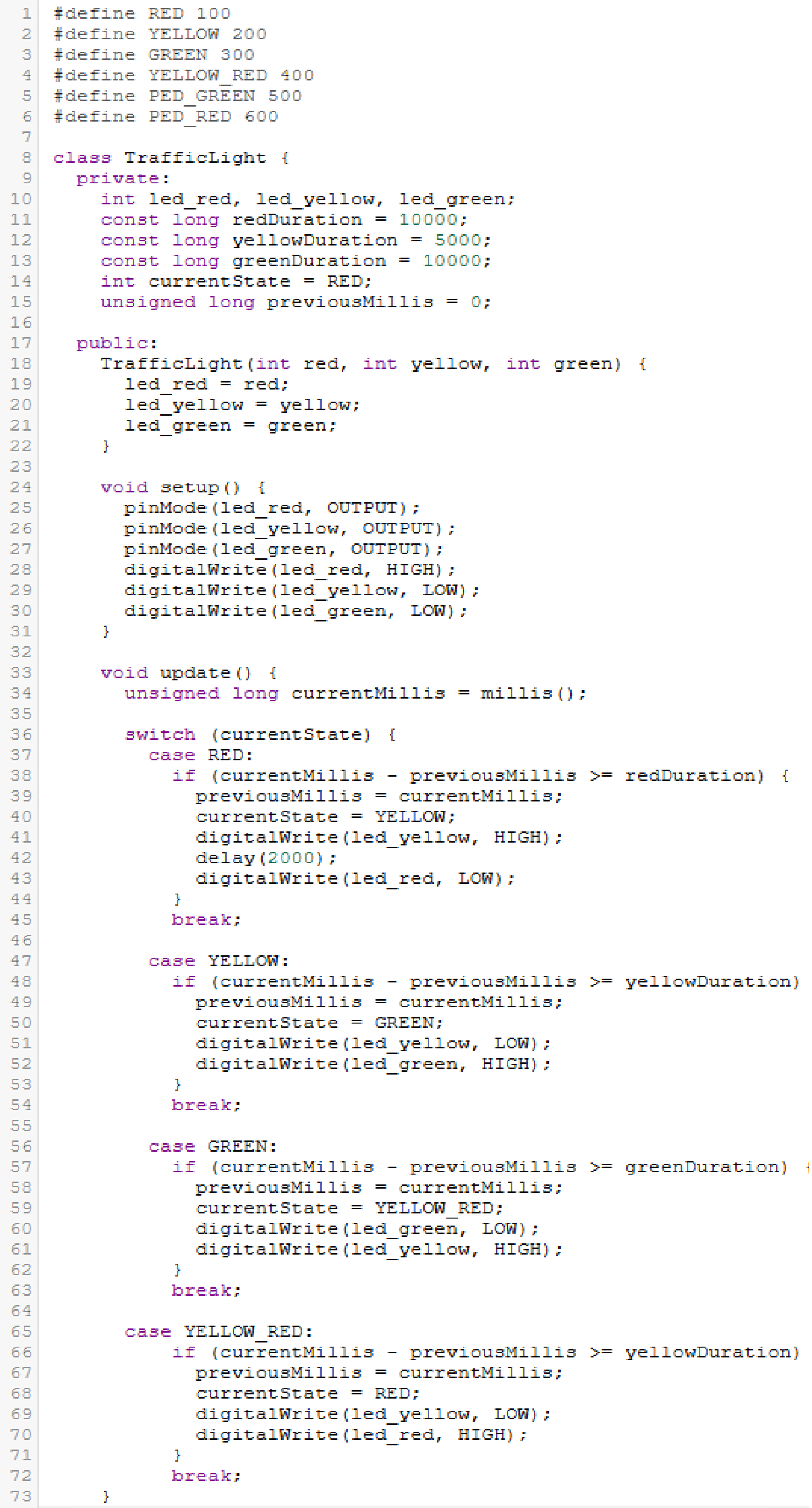
```
  switch (currentPedState) {
    case PED_RED:
      if (buttonPressed && trafficState == RED) {
        currentPedState = PED_GREEN;
        buttonPressed = false;
        digitalWrite(pRed, LOW);
        digitalWrite(pGreen, HIGH);
      }
      break;

    case PED_GREEN:
      delay(6000);
      currentPedState = PED_RED;
      digitalWrite(pGreen, LOW);
      digitalWrite(pRed, HIGH);
      break;
  }
}
```

To get current state of traffic light.

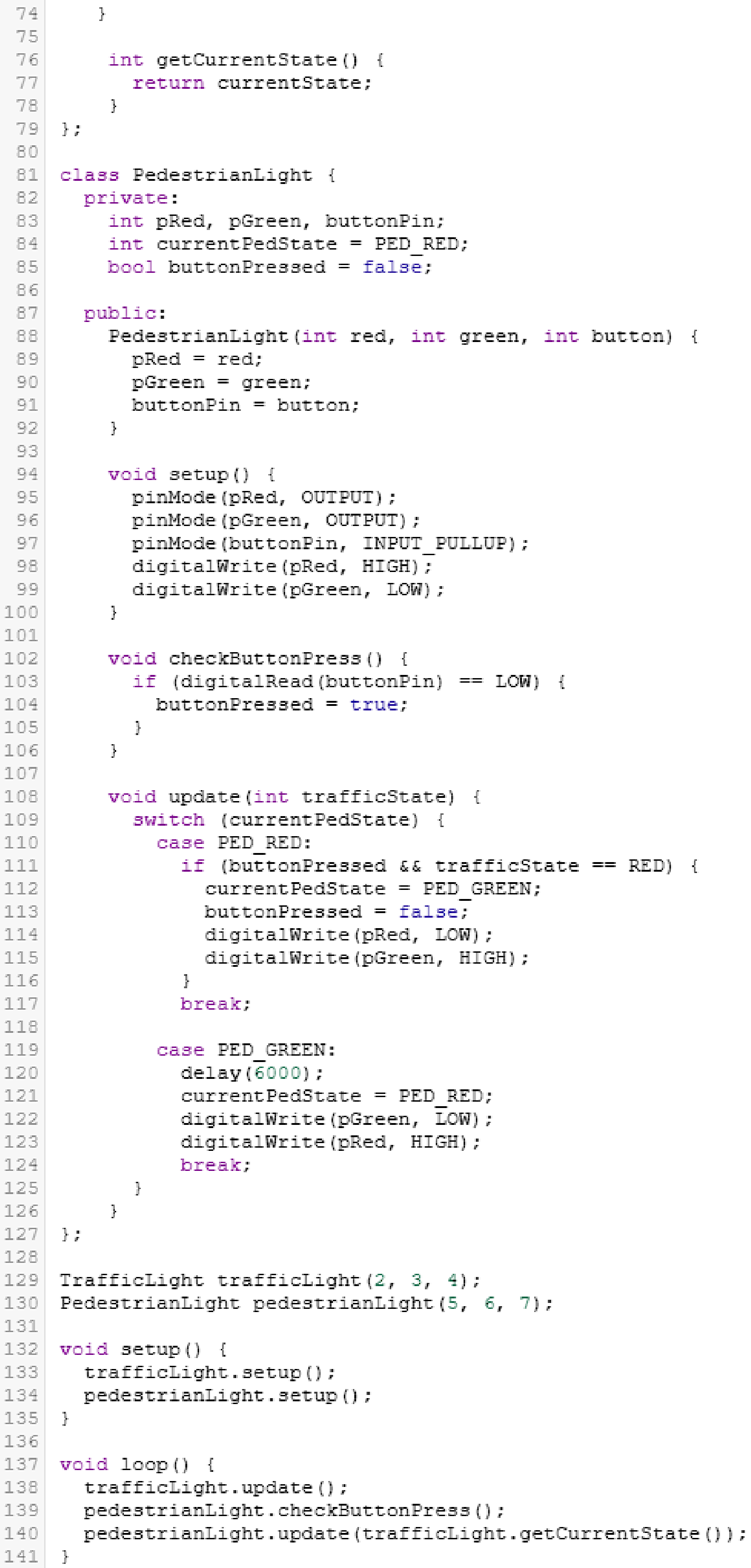


```
int getCurrentState() {
  return currentState;
}
```

- Code:



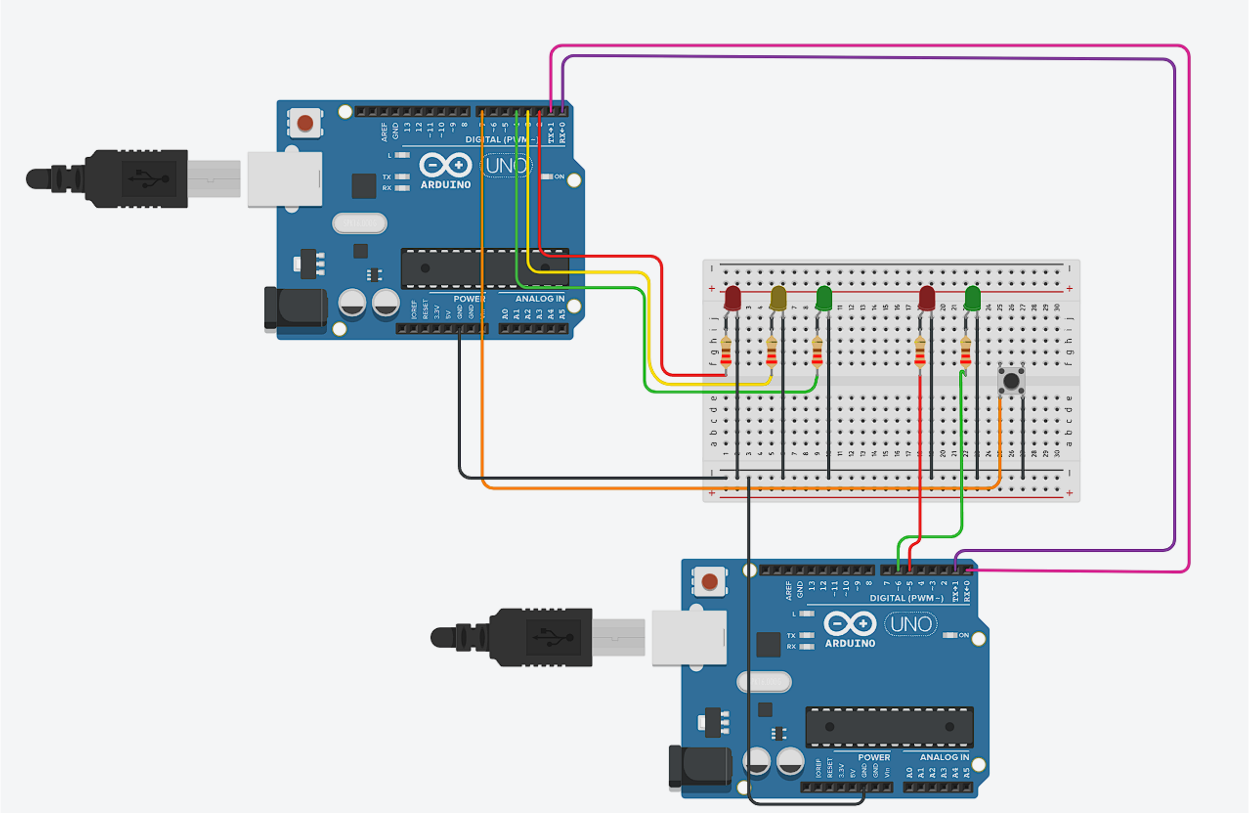
```
#define RED 100
#define YELLOW 200
#define GREEN 300
#define YELLOW_RED 400
#define PED_GREEN 500
#define PED_RED 600

class TrafficLight {
  private:
    int led_red, led_yellow, led_green;
    const long redDuration = 10000;
    const long yellowDuration = 5000;
    const long greenDuration = 10000;
    int currentState = RED;
    unsigned long previousMillis = 0;

  public:
    TrafficLight(int red, int yellow, int green) {
      led_red = red;
      led_yellow = yellow;
      led_green = green;
    }

    void setup() {
      pinMode(led_red, OUTPUT);
      pinMode(led_yellow, OUTPUT);
      pinMode(led_green, OUTPUT);
      digitalWrite(led_red, HIGH);
      digitalWrite(led_yellow, LOW);
      digitalWrite(led_green, LOW);
    }

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

      switch (currentState) {
        case RED:
          if (currentMillis - previousMillis >= redDuration) {
            previousMillis = currentMillis;
            currentState = YELLOW;
            digitalWrite(led_yellow, HIGH);
            delay(2000);
            digitalWrite(led_red, LOW);
          }
          break;

        case YELLOW:
          if (currentMillis - previousMillis >= yellowDuration)
            previousMillis = currentMillis;
            currentState = GREEN;
            digitalWrite(led_yellow, LOW);
            digitalWrite(led_green, HIGH);
          }
          break;

        case GREEN:
          if (currentMillis - previousMillis >= greenDuration) {
            previousMillis = currentMillis;
            currentState = YELLOW_RED;
            digitalWrite(led_green, LOW);
            digitalWrite(led_yellow, HIGH);
          }
          break;

      case YELLOW_RED:
          if (currentMillis - previousMillis >= yellowDuration)
            previousMillis = currentMillis;
            currentState = RED;
            digitalWrite(led_yellow, LOW);
            digitalWrite(led_red, HIGH);
          }
          break;
    }
```



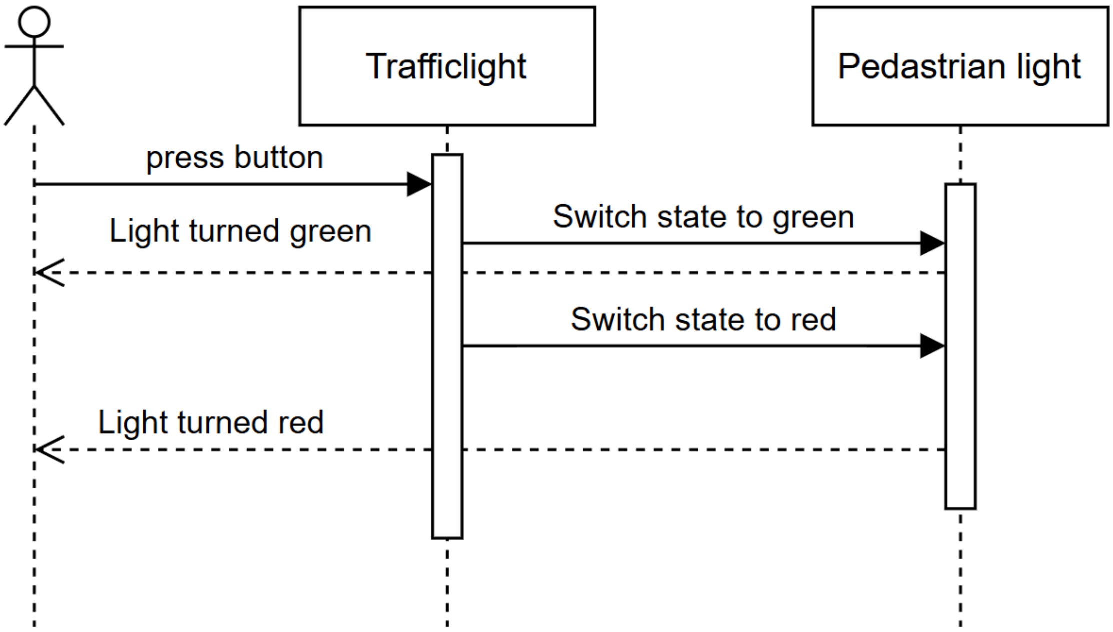
```
   }

    int getCurrentState() {
      return currentState;
    }
};

class PedestrianLight {
  private:
    int pRed, pGreen, buttonPin;
    int currentPedState = PED_RED;
    bool buttonPressed = false;

  public:
    PedestrianLight(int red, int green, int button) {
      pRed = red;
      pGreen = green;
      buttonPin = button;
    }

    void setup() {
      pinMode(pRed, OUTPUT);
      pinMode(pGreen, OUTPUT);
      pinMode(buttonPin, INPUT_PULLUP);
      digitalWrite(pRed, HIGH);
      digitalWrite(pGreen, LOW);
    }

    void checkButtonPress() {
      if (digitalRead(buttonPin) == LOW) {
        buttonPressed = true;
      }
    }

    void update(int trafficState) {
      switch (currentPedState) {
        case PED_RED:
          if (buttonPressed && trafficState == RED) {
            currentPedState = PED_GREEN;
            buttonPressed = false;
            digitalWrite(pRed, LOW);
            digitalWrite(pGreen, HIGH);
          }
          break;

        case PED_GREEN:
          delay(6000);
          currentPedState = PED_RED;
          digitalWrite(pGreen, LOW);
          digitalWrite(pRed, HIGH);
          break;
      }
    }
};

TrafficLight trafficLight(2, 3, 4);
PedestrianLight pedestrianLight(5, 6, 7);

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

void loop() {
  trafficLight.update();
  pedestrianLight.checkButtonPress();
  pedestrianLight.update(trafficLight.getCurrentState());
}
```

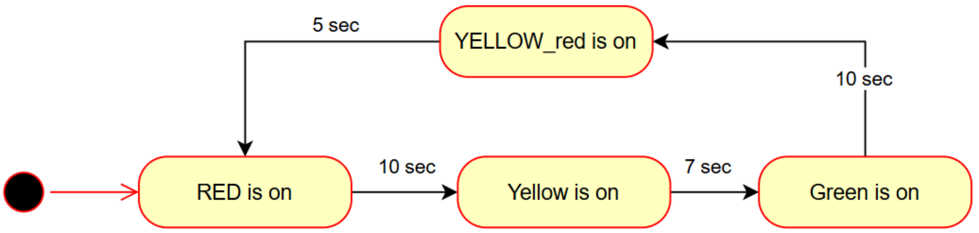
Traffic and Pedestrian Light System with Serial Communication (Task 5)

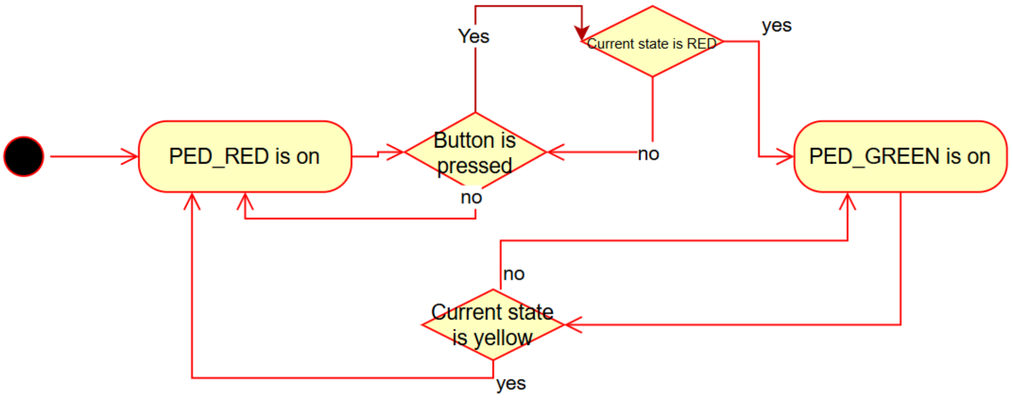
- Objective: establish communication between two separate µC for the car traffic light and the pedestrian traffic light.
- Circuit model includes: 2 Arduino board, 5 LEDs, 5 220 ohm resistors, breadboard, wires.



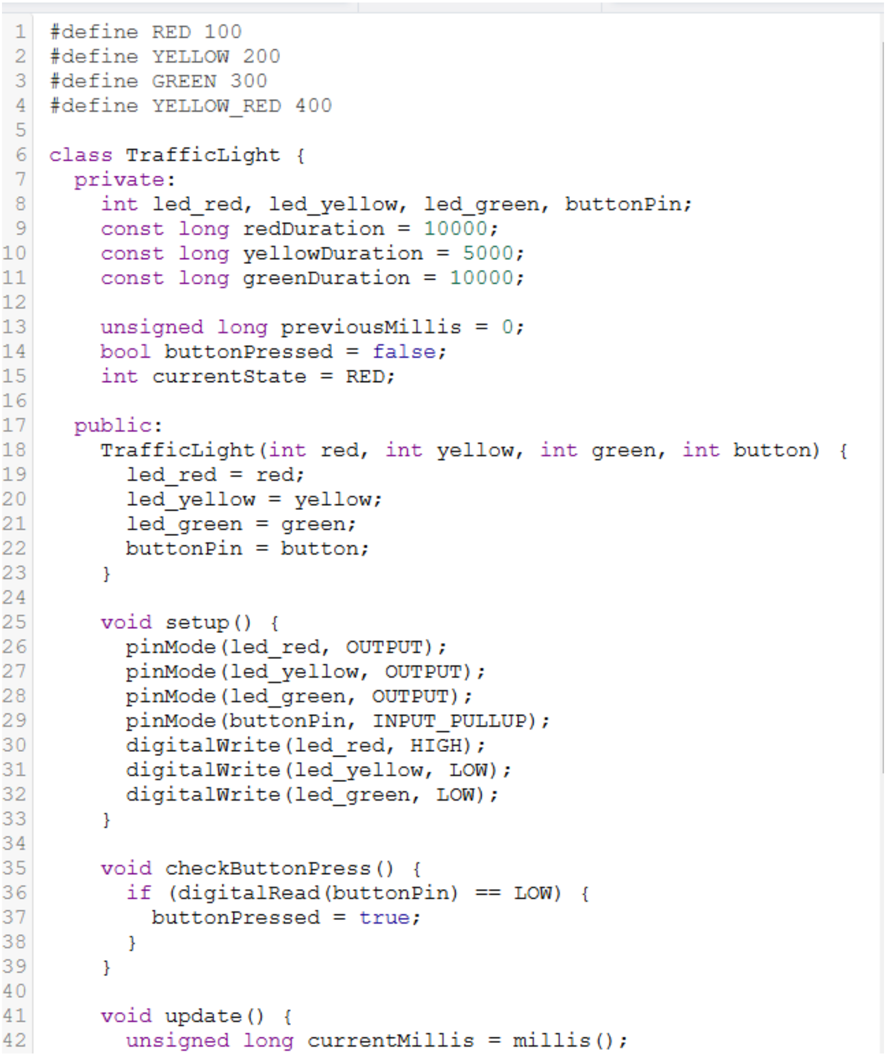
- Diagrams:



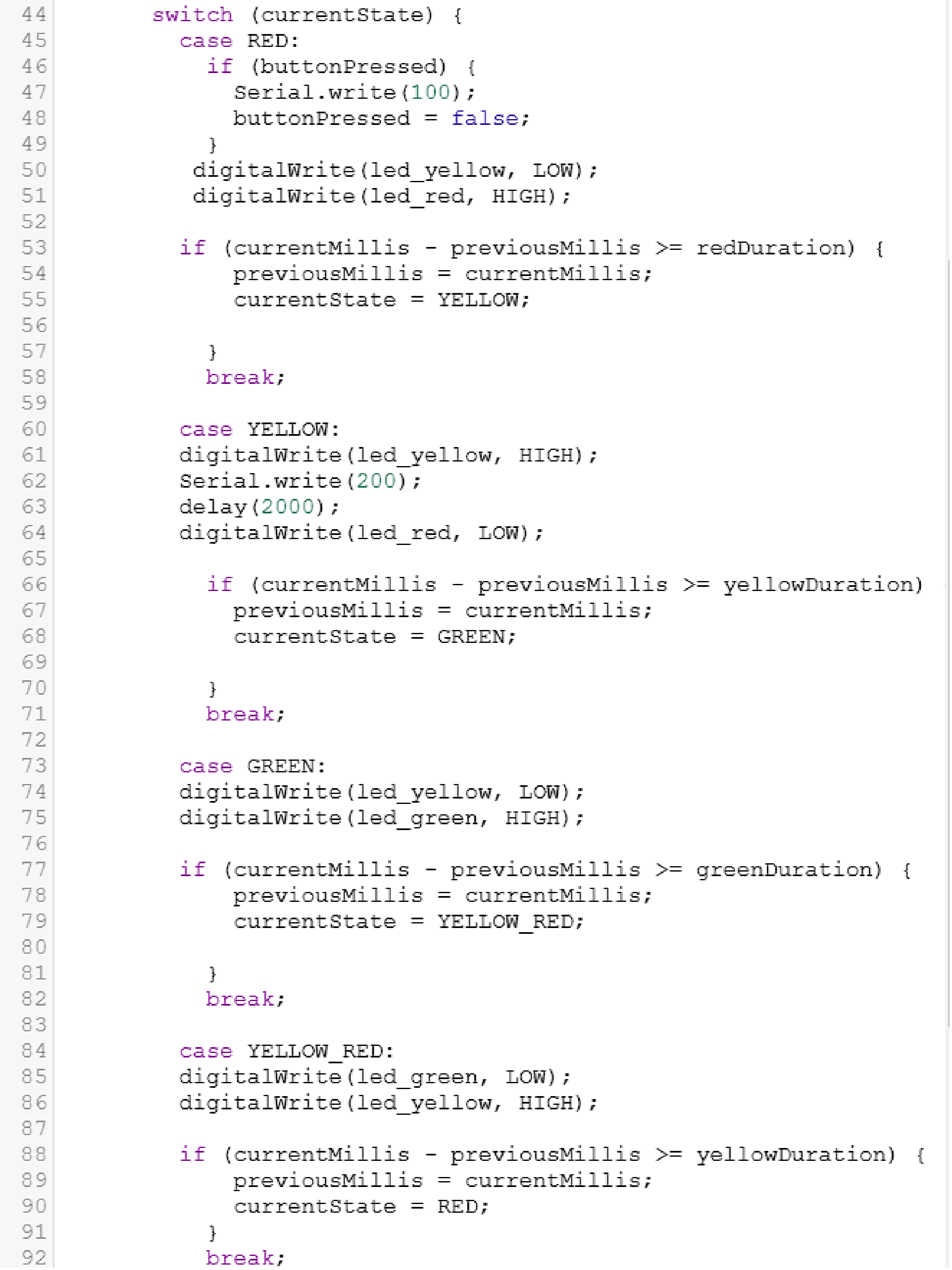




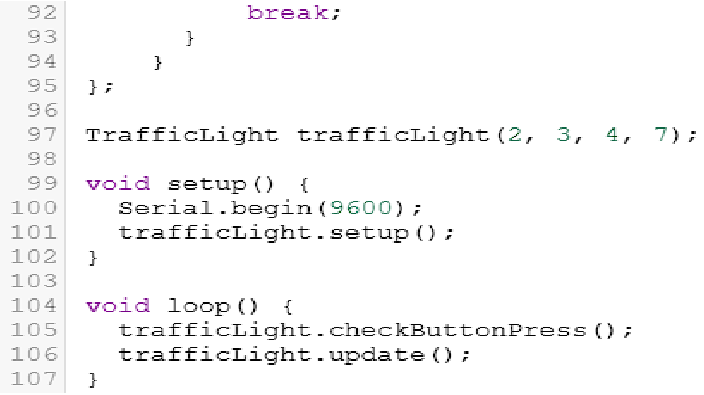
- Code:



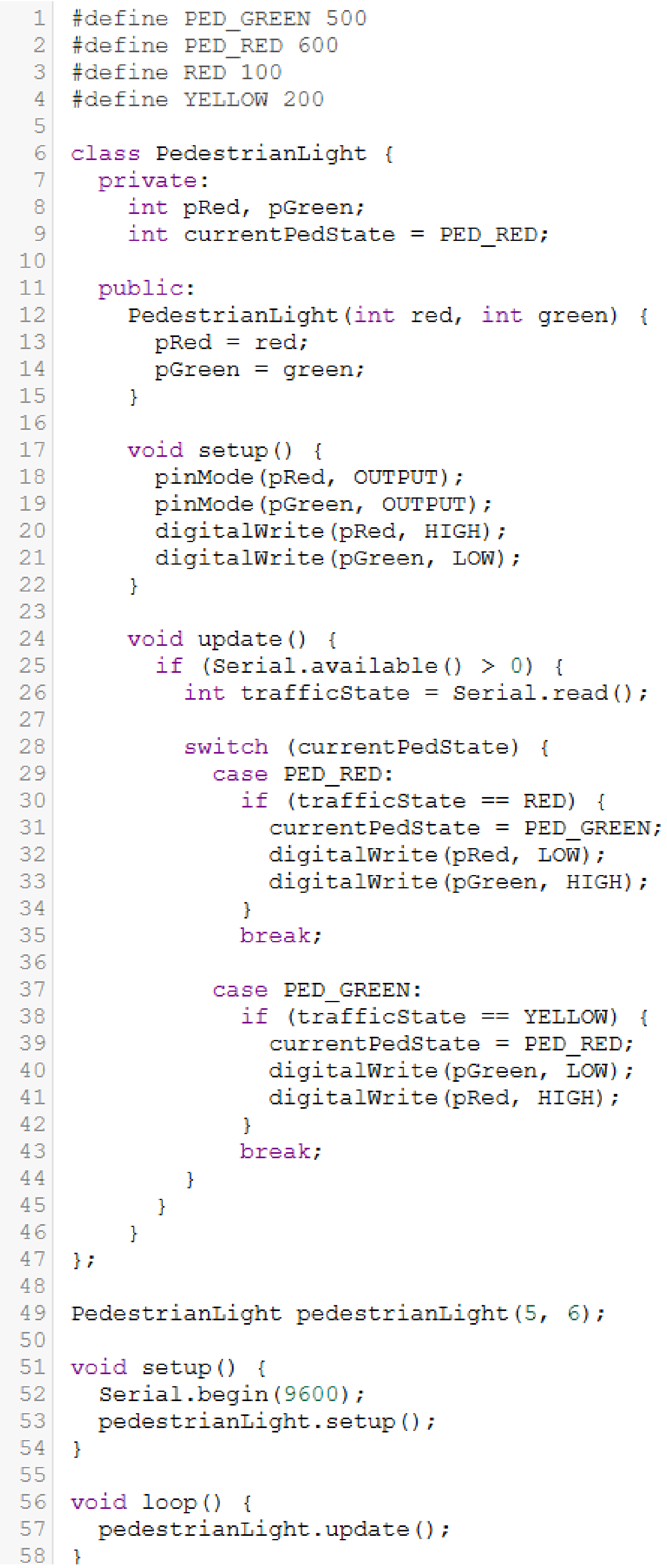
```
#define RED 100
#define YELLOW 200
#define GREEN 300
#define YELLOW_RED 400

class TrafficLight {
  private:
    int led_red, led_yellow, led_green, buttonPin;
    const long redDuration = 10000;
    const long yellowDuration = 5000;
    const long greenDuration = 10000;

    unsigned long previousMillis = 0;
    bool buttonPressed = false;
    int currentState = RED;

  public:
    TrafficLight(int red, int yellow, int green, int button) {
      led_red = red;
      led_yellow = yellow;
      led_green = green;
      buttonPin = button;
    }

    void setup() {
      pinMode(led_red, OUTPUT);
      pinMode(led_yellow, OUTPUT);
      pinMode(led_green, OUTPUT);
      pinMode(buttonPin, INPUT_PULLUP);
      digitalWrite(led_red, HIGH);
      digitalWrite(led_yellow, LOW);
      digitalWrite(led_green, LOW);
    }

    void checkButtonPress() {
      if (digitalRead(buttonPin) == LOW) {
        buttonPressed = true;
      }
    }

    void update() {
      unsigned long currentMillis = millis();
```



```
      switch (currentState) {
        case RED:
          if (buttonPressed) {
            Serial.write(100);
            buttonPressed = false;
          }
         digitalWrite(led_yellow, LOW);
         digitalWrite(led_red, HIGH);

        if (currentMillis - previousMillis >= redDuration) {
            previousMillis = currentMillis;
            currentState = YELLOW;

          }
          break;

        case YELLOW:
        digitalWrite(led_yellow, HIGH);
        Serial.write(200);
        delay(2000);
        digitalWrite(led_red, LOW);

          if (currentMillis - previousMillis >= yellowDuration)
            previousMillis = currentMillis;
            currentState = GREEN;

          }
          break;

        case GREEN:
        digitalWrite(led_yellow, LOW);
        digitalWrite(led_green, HIGH);

        if (currentMillis - previousMillis >= greenDuration) {
            previousMillis = currentMillis;
            currentState = YELLOW_RED;

          }
          break;

        case YELLOW_RED:
        digitalWrite(led_green, LOW);
        digitalWrite(led_yellow, HIGH);

        if (currentMillis - previousMillis >= yellowDuration) {
            previousMillis = currentMillis;
            currentState = RED;
          }
          break;
```



```
          break;
      }
    }
};

TrafficLight trafficLight(2, 3, 4, 7);

void setup() {
  Serial.begin(9600);
  trafficLight.setup();
}

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



```
#define PED_GREEN 500
#define PED_RED 600
#define RED 100
#define YELLOW 200

class PedestrianLight {
  private:
    int pRed, pGreen;
    int currentPedState = PED_RED;

  public:
    PedestrianLight(int red, int green) {
      pRed = red;
      pGreen = green;
    }

    void setup() {
      pinMode(pRed, OUTPUT);
      pinMode(pGreen, OUTPUT);
      digitalWrite(pRed, HIGH);
      digitalWrite(pGreen, LOW);
    }

    void update() {
      if (Serial.available() > 0) {
        int trafficState = Serial.read();

        switch (currentPedState) {
          case PED_RED:
            if (trafficState == RED) {
              currentPedState = PED_GREEN;
              digitalWrite(pRed, LOW);
              digitalWrite(pGreen, HIGH);
            }
            break;

          case PED_GREEN:
            if (trafficState == YELLOW) {
              currentPedState = PED_RED;
              digitalWrite(pGreen, LOW);
              digitalWrite(pRed, HIGH);
            }
            break;
        }
      }
    }
};

PedestrianLight pedestrianLight(5, 6);

void setup() {
  Serial.begin(9600);
  pedestrianLight.setup();
}

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