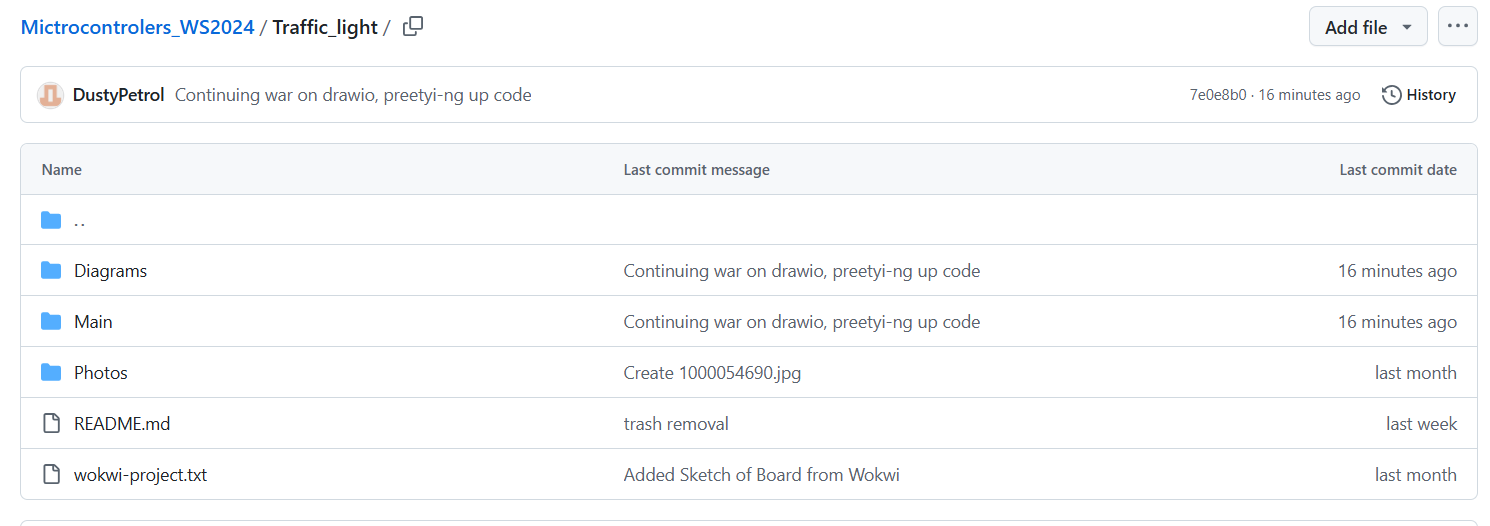
Documentation for Microcontrolers WS2024 project

By Petr Lavrenov

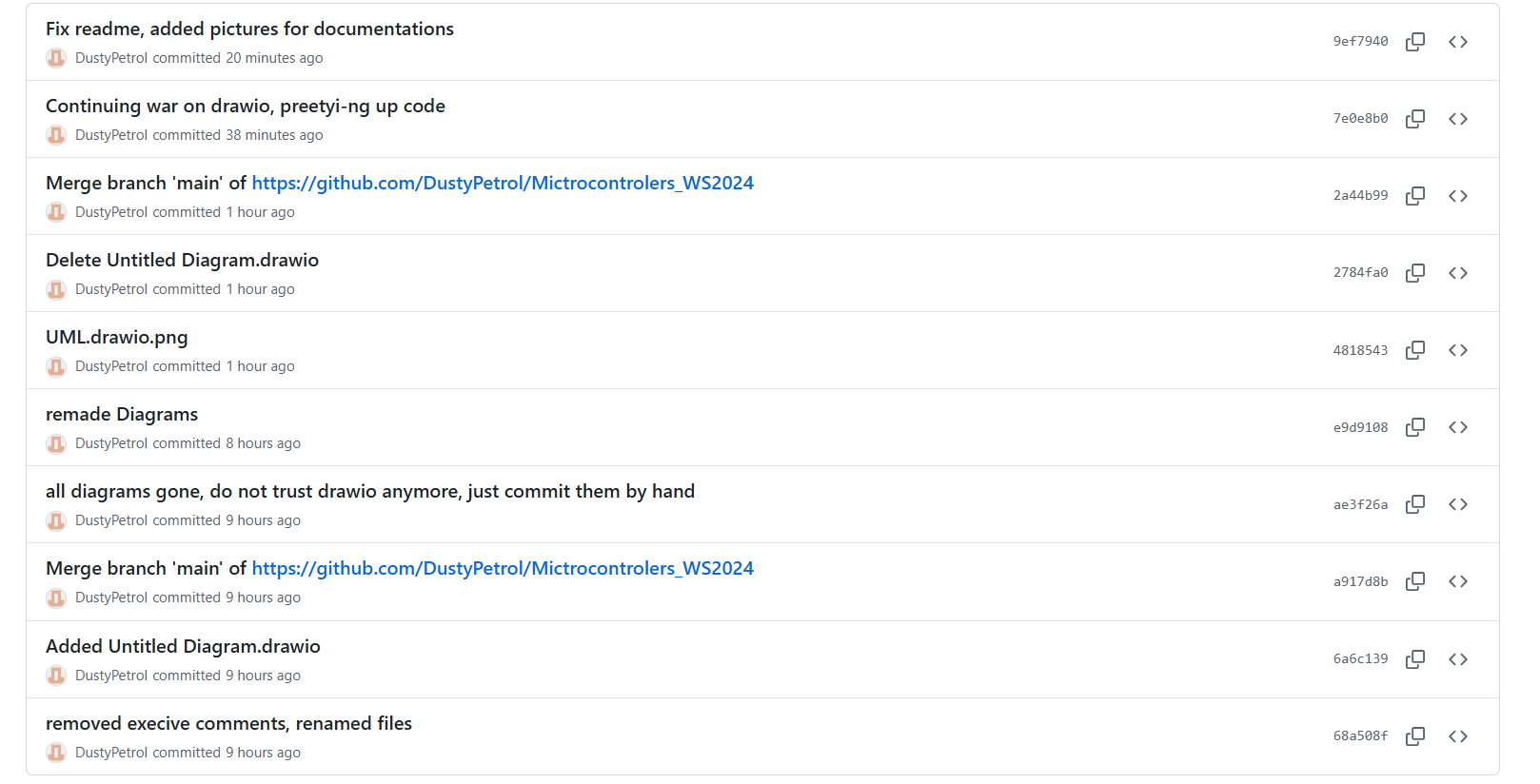
1. Git Usage
2. Diagrams
3. Code

# Git usage

Folder Structure:

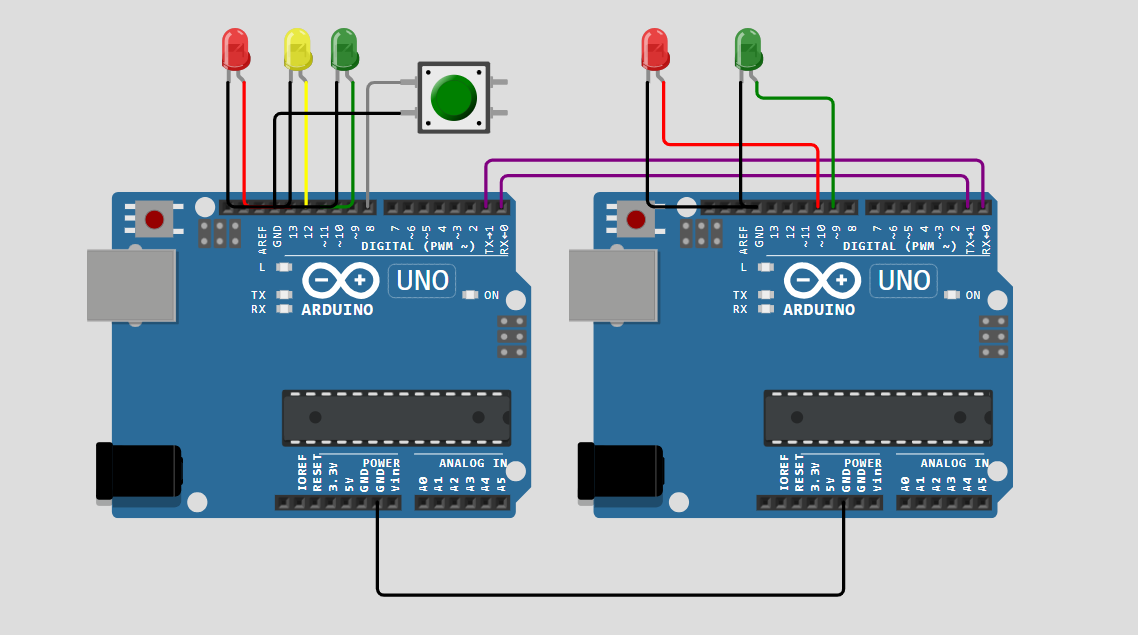


An example of a day of commits:

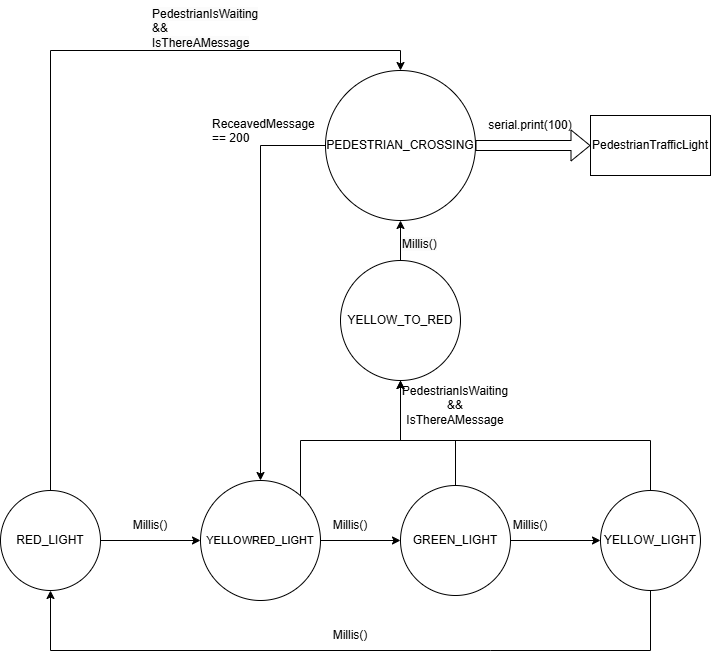


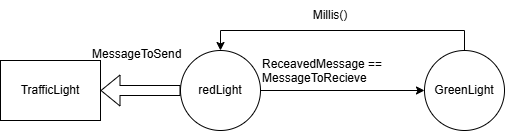
# Diagrams

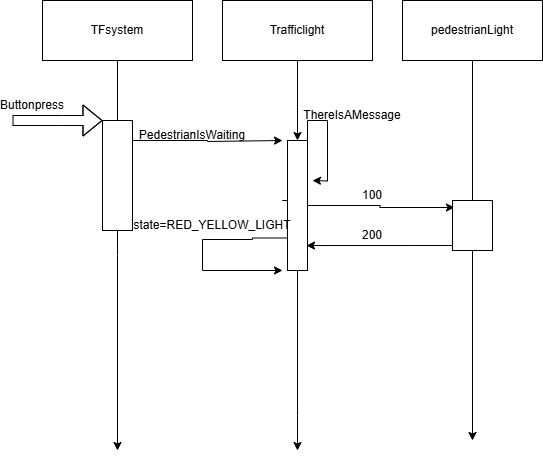
## Wokwi Diagram of pin mapping:



UML diagram of classes:

State machine trafficLight:

State machine Pedestrian

Sequence diagram TFsystem:

# Relevant parts of code:

## TrafficLight:

class TrafficLight {

  private:

    int redPin, yellowPin, greenPin;

    unsigned long previousMillis;

    int state;

    bool pedestrianWaiting;

    uint8\_t ReceavedMessage=0;

    bool IsThereAMessage = false;

  public:

    TrafficLight(int rPin, int yPin, int gPin) :

      redPin(rPin),

      yellowPin(yPin),

      greenPin(gPin),

      state(RED\_LIGHT),

      previousMillis(millis()),

      pedestrianWaiting(false) {

      pinMode(redPin, OUTPUT);

      pinMode(yellowPin, OUTPUT);

      pinMode(greenPin, OUTPUT);

      redLight();   }

    void update(unsigned long currentMillis) {

      if (pedestrianWaiting && state != PEDESTRIAN\_CROSSING && state != YELLOW\_TO\_RED) {

        if (state == RED\_LIGHT) {

          state = PEDESTRIAN\_CROSSING;

          previousMillis = currentMillis;

          pedestrianWaiting = false;

        } else {

          state = YELLOW\_TO\_RED;

          previousMillis = currentMillis;

          pedestrianWaiting = false;

          yellowLight();}}

      switch (state) {

        case GREEN\_LIGHT:

          if (currentMillis - previousMillis >= greenTime) {

            state = YELLOW\_LIGHT;

            previousMillis = currentMillis;

            yellowLight();}

          break;

        case YELLOW\_LIGHT:

          if (currentMillis - previousMillis >= yellowTime) {

            state = RED\_LIGHT;

            previousMillis = currentMillis;

            redLight(); }

          break;

        case RED\_LIGHT:

          if (currentMillis - previousMillis >= redTime) {

            state = RED\_YELLOW\_LIGHT;

            previousMillis = currentMillis;

            redYellowLight();  }

          break;

        case RED\_YELLOW\_LIGHT:

          if (currentMillis - previousMillis >= redYellowTime) {

            state = GREEN\_LIGHT;

            previousMillis = currentMillis;

            greenLight(); }

          break;

        case YELLOW\_TO\_RED:

          if (currentMillis - previousMillis >= yellowTime) {

            state = PEDESTRIAN\_CROSSING;

            previousMillis = currentMillis;

            redLight();}

          break;

        case PEDESTRIAN\_CROSSING:

          SendMessage();

          if (ReceavedMessage == MessageToRecieve) {

            ReceavedMessage = 0;

            state = RED\_YELLOW\_LIGHT;

            previousMillis = currentMillis;

            redYellowLight();

            pedestrianWaiting = false; }

          break;  } }

    void ThereIsAMessage() {

      IsThereAMessage = true;}

   void recieveMessage(uint8\_t message){

    ReceavedMessage=message; }

    void SendMessage() {

      if (IsThereAMessage) {

        Serial.write(MessageToSend);

        IsThereAMessage = false; }}

    void redLight() {

      digitalWrite(redPin, HIGH);

      digitalWrite(yellowPin, LOW);

      digitalWrite(greenPin, LOW); }

    void yellowLight() {

      digitalWrite(redPin, LOW);

      digitalWrite(yellowPin, HIGH);

      digitalWrite(greenPin, LOW); }

    void greenLight() {

      digitalWrite(redPin, LOW);

      digitalWrite(yellowPin, LOW);

      digitalWrite(greenPin, HIGH); }

    void redYellowLight() {

      digitalWrite(redPin, HIGH);

      digitalWrite(yellowPin, HIGH);

      digitalWrite(greenPin, LOW);  }

    void triggerPedestrianCrossing() {

      pedestrianWaiting = true;

    }

    bool isPedestrianCrossing() {

      return state == PEDESTRIAN\_CROSSING }};

class TFSystem {

  private:

    TrafficLight traffic;

    uint8\_t MessageToPass;

  public:

    TFSystem(int trafficRed, int trafficYellow, int trafficGreen)

    : traffic(trafficRed, trafficYellow, trafficGreen) {

      pinMode(buttonPin, INPUT\_PULLUP); }

    void update(unsigned long currentMillis) {

      if (digitalRead(buttonPin) == LOW && !traffic.isPedestrianCrossing()) {

        traffic.ThereIsAMessage();

        traffic.triggerPedestrianCrossing();}

       while (Serial.available() > 0) {

        MessageToPass=Serial.read();

        traffic.recieveMessage(MessageToPass);}

      traffic.update(currentMillis); };

TFSystem trafficSystem(trafficRedPin, trafficYellowPin, trafficGreenPin);

void setup() {

  Serial.begin(9600);  }

void loop() {

  unsigned long currentMillis = millis();

  trafficSystem.update(currentMillis);  }

## Pedestrian:

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

class PedestrianLight {

private:

  uint8\_t redPin, greenPin;

  bool active;

  unsigned long previousMillis;

  byte ReceavedMessage=0;

public:

  PedestrianLight(uint8\_t rPin, uint8\_t gPin)

    : redPin(rPin), greenPin(gPin), active(false), previousMillis(0) {

    pinMode(rPin, OUTPUT);

    pinMode(gPin, OUTPUT);

    redLight();     }

  void update(unsigned long currentMillis) {

    while (Serial.available() > 0){

      ReceavedMessage = Serial.read(); }

    if (ReceavedMessage == 100 && !active) {

      while

      activate(currentMillis);

      ReceavedMessage = 0;

      greenLight();     }

    if (active && currentMillis - previousMillis >= pedestrianGreenTime) {

      active = false;

      redLight();

      Serial.write(200);      }  }

  void redLight() {

    digitalWrite(redPin, HIGH);

    digitalWrite(greenPin, LOW);}

  void greenLight() {

    digitalWrite(redPin, LOW);

    digitalWrite(greenPin, HIGH);}

  void activate(unsigned long currentMillis) {

    active = true;                        // Set active flag

    previousMillis = currentMillis;       // Store the current time for timing reference

  }

  bool isActive() {

    return active;}};

PedestrianLight pedestrian(pedestrianRedPin, pedestrianGreenPin);

void loop() {

  unsigned long currentMillis = millis();

  pedestrian.update(currentMillis);        }