MICROCRONTROLLER

Milestone 1

Task 1-5

Documentation

https://github.com/MoeezMufti/Microcontroller--Moeez-Mufti

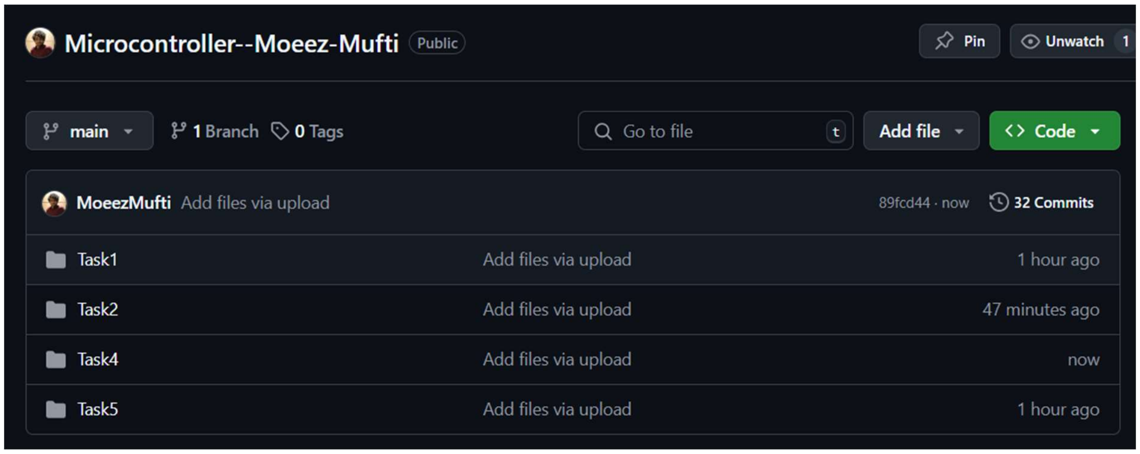
Moeez Mufti
moeez.mufti@stud.hshl.de

Table of Contents

GitHub:

Link:

https://github.com/MoeezMufti/Microcontroller--Moeez-Mufti



Microcontroller--Moeez-Mufti Public

Pin | Unwatch 1

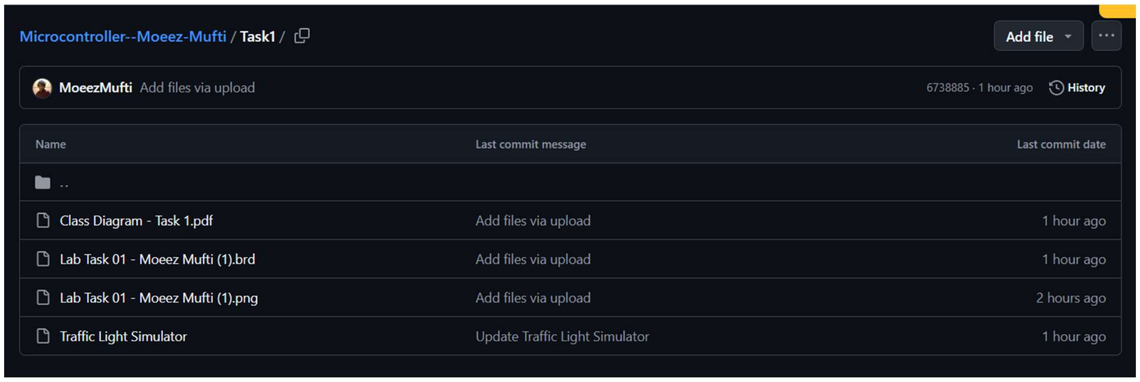
main | 1 Branch | 0 Tags | Go to file | Add file | Code

MoeezMufti Add files via upload 89fcd44 · now 32 Commits

Task1	Add files via upload	1 hour ago
Task2	Add files via upload	47 minutes ago
Task4	Add files via upload	now
Task5	Add files via upload	1 hour ago

Task 1

GitHub:



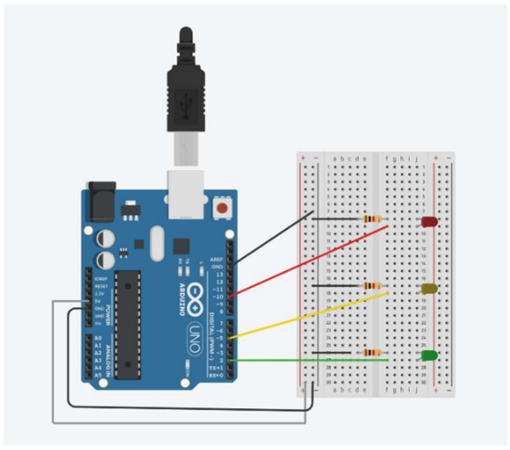
Microcontroller--Moeez-Mufti / Task1 /

Add file

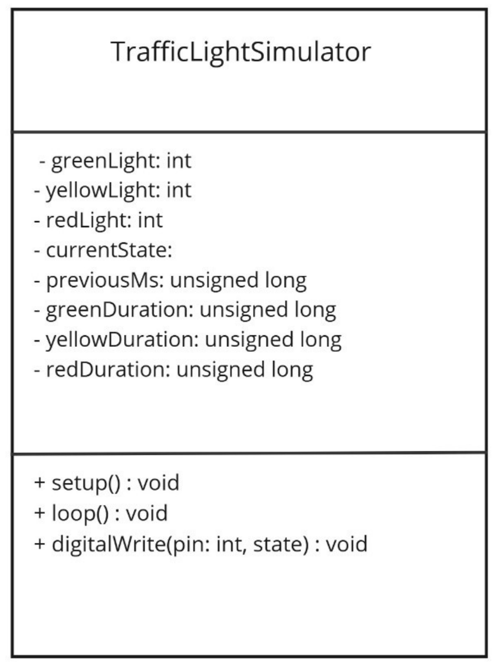
MoeezMufti Add files via upload 6738885 · 1 hour ago History

Name	Last commit message	Last commit date
..		
Class Diagram - Task 1.pdf	Add files via upload	1 hour ago
Lab Task 01 - Moeez Mufti (1).brd	Add files via upload	1 hour ago
Lab Task 01 - Moeez Mufti (1).png	Add files via upload	2 hours ago
Traffic Light Simulator	Update Traffic Light Simulator	1 hour ago

TinkerCAD Simulation:



State Machine Diagram:


TrafficLightSimulator
- greenLight: int
- yellowLight: int
- redLight: int
- currentState:
- previousMs: unsigned long
- greenDuration: unsigned long
- yellowDuration: unsigned long
- redDuration: unsigned long
+ setup() : void
+ loop() : void
+ digitalWrite(pin: int, state) : void

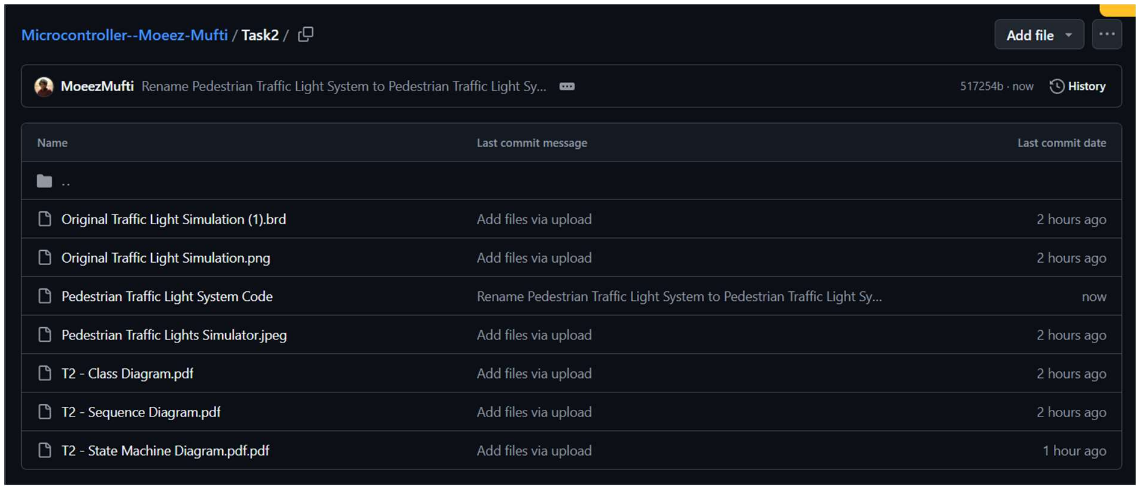
Code:

```
// Pin delegations
const int greenLight = 2;
const int yellowLight = 3;
const int redLight = 4;

// State definitions
enum TrafficState { Green, Yellow, Red };
TrafficState currentState = Green;

/*setting durations as constant so that they
don't change with with other condition*/
unsigned long previousMs = 0;
const unsigned long greenDuration = 10000; //in ms
const unsigned long yellowDuration = 3000;
const unsigned long redDuration = 5000;

void setup() {
  // Set variables to their modes
  pinMode(greenLight, OUTPUT);
  pinMode(yellowLight, OUTPUT);
  pinMode(redLight, OUTPUT);

  // System always comes back to Green Light
  digitalWrite(greenLight, HIGH);
  previousMs = millis(); //tells how much ms has passed since beginning
}
```

```
void loop() {
  unsigned long currentMs = millis();

  switch (currentState) {
    case Green:
      // Green light is on for 10s
      if (currentMs - previousMs >= greenDuration) {
        //how much time has passed from beginning to current
        digitalWrite(greenLight, LOW);
        digitalWrite(yellowLight, HIGH);
        currentState = Yellow;
        previousMs = currentMs;
      }
      break;

    case Yellow:
      // Yellow light is on for 3s
      if (currentMs - previousMs >= yellowDuration) {
        digitalWrite(yellowLight, LOW);
        digitalWrite(redLight, HIGH);
        currentState = Red;
        previousMs = currentMs;
      }
      break;

    case Red:
      // Red light is on for 5s
      if (currentMs - previousMs >= redDuration) {
        digitalWrite(redLight, LOW);
        digitalWrite(greenLight, HIGH);
        currentState = Green;
        previousMs = currentMs;
      }
      break;
  }
}
```

Task 2

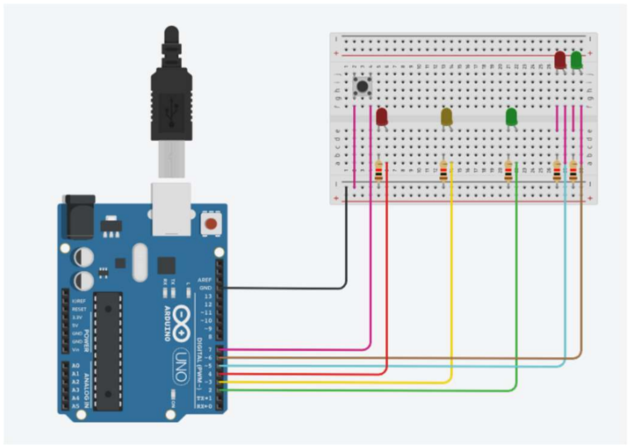
GitHub:


Microcontroller--Moeez-Mufti / Task2 /

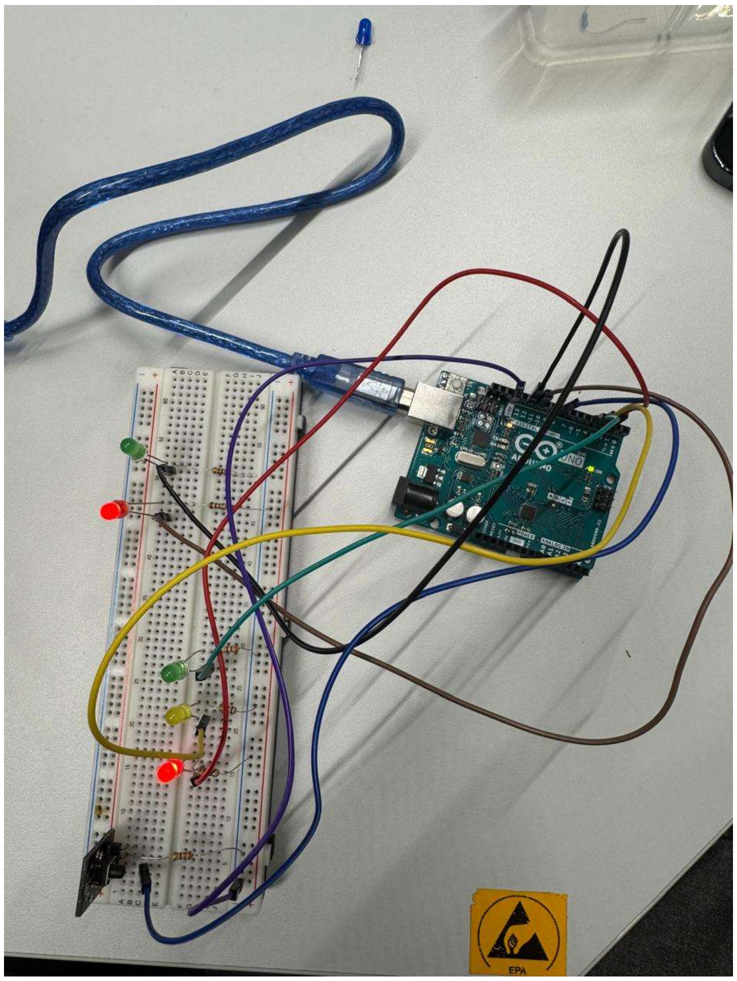
MoeezMufti Rename Pedestrian Traffic Light System to Pedestrian Traffic Light Sy... 517254b · now

Name	Last commit message	Last commit date
..		
Original Traffic Light Simulation (1).brd	Add files via upload	2 hours ago
Original Traffic Light Simulation.png	Add files via upload	2 hours ago
Pedestrian Traffic Light System Code	Rename Pedestrian Traffic Light System to Pedestrian Traffic Light Sy...	now
Pedestrian Traffic Lights Simulator.jpeg	Add files via upload	2 hours ago
T2 - Class Diagram.pdf	Add files via upload	2 hours ago
T2 - Sequence Diagram.pdf	Add files via upload	2 hours ago
T2 - State Machine Diagram.pdf.pdf	Add files via upload	1 hour ago

TinkerCAD Simulation:



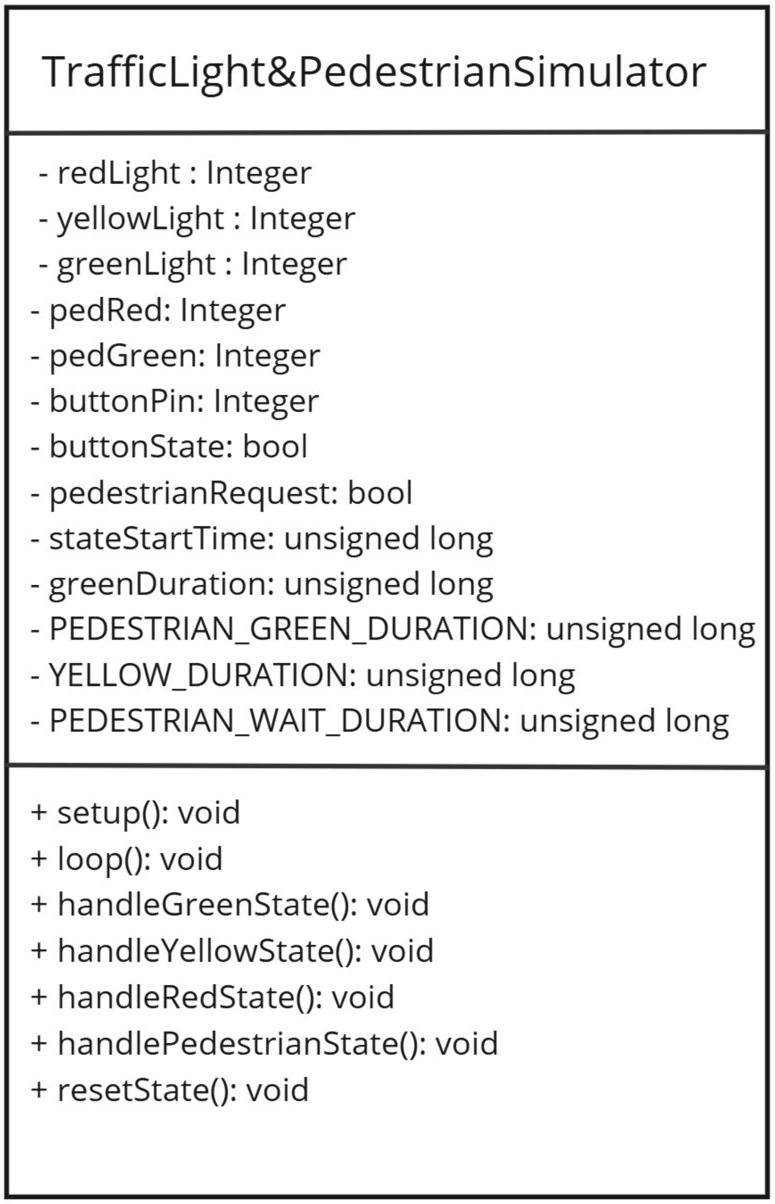
Physical Simulation:



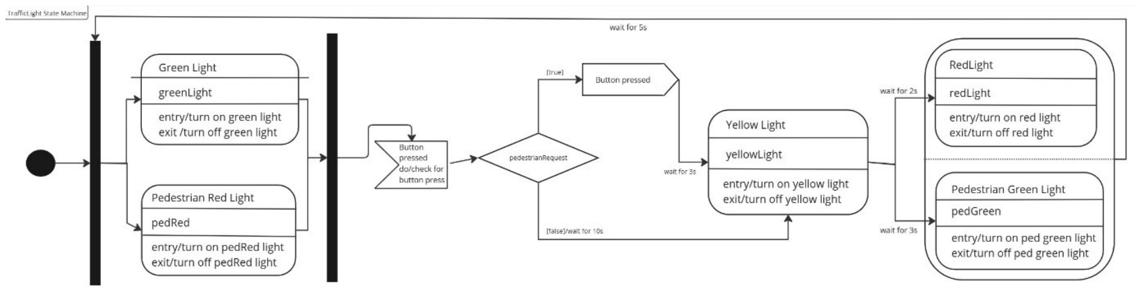
Code:

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

// State definitions
enum TrafficState { GREEN, YELLOW, RED, RESET };
TrafficState currentState = GREEN;

//setting timer to zero for initi setup and button to false
unsigned long greenTimer = 0;
bool buttonPressed = false;

void setup() {
  // Set pin AND modes
  pinMode(greenLight, OUTPUT);
  pinMode(yellowLight, OUTPUT);
  pinMode(redLight, OUTPUT);
  pinMode(pedRed, OUTPUT);
  pinMode(pedGreen, OUTPUT);
  pinMode(button, INPUT_PULLUP);

  // Always start and loop back to Green
  digitalWrite(greenLight, HIGH);
  digitalWrite(pedRed, HIGH);
  greenTimer = millis();
}

void loop() {
  // Check whether or not the button is pressed
  if (digitalRead(button) == LOW) {
    buttonPressed = true;
  }

  // State machine for traffic light control
  switch (currentState) {
    case GREEN:
      // Stay in GREEN state until 10 seconds pass or button is pressed
      if ((buttonPressed && digitalRead(greenLight) == HIGH) ||
          (millis() - greenTimer >= 10000)) {
        if (buttonPressed) {
          delay(3000);  // Green light will stay on for 3s after button press
        }
        currentState = YELLOW;
      }
      break;

    case YELLOW:
      digitalWrite(greenLight, LOW);
      digitalWrite(yellowLight, HIGH);
      delay(2000);
      digitalWrite(yellowLight, LOW);
      currentState = RED;
      break;

    case RED:
      // Red light and pedestrian crossing sequence
      digitalWrite(redLight, HIGH);
      digitalWrite(pedRed, LOW);
      digitalWrite(pedGreen, HIGH);
      delay(5000);
      currentState = RESET;
      break;

    case RESET:
      // Reset to initial state
      digitalWrite(pedGreen, LOW);
      digitalWrite(pedRed, HIGH);
      digitalWrite(redLight, LOW);
      digitalWrite(greenLight, HIGH);
      buttonPressed = false;
      greenTimer = millis();
      currentState = GREEN;
      break;
  }
}
```

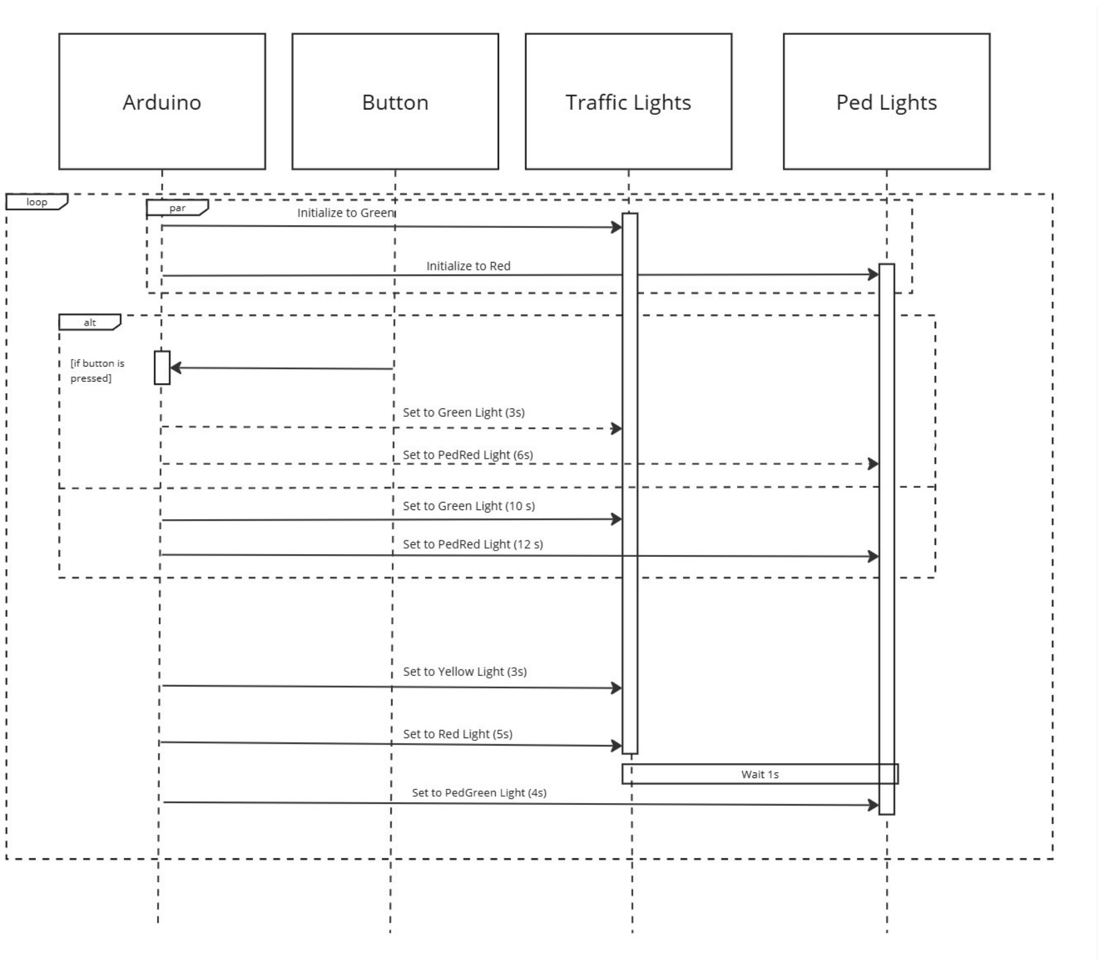
Class Machine Diagram:


TrafficLight&PedestrianSimulator
- redLight : Integer
- yellowLight : Integer
- greenLight : Integer
- pedRed: Integer
- pedGreen: Integer
- buttonPin: Integer
- buttonState: bool
- pedestrianRequest: bool
- stateStartTime: unsigned long
- greenDuration: unsigned long
- PEDESTRIAN_GREEN_DURATION: unsigned long
- YELLOW_DURATION: unsigned long
- PEDESTRIAN_WAIT_DURATION: unsigned long
+ setup(): void
+ loop(): void
+ handleGreenState(): void
+ handleYellowState(): void
+ handleRedState(): void
+ handlePedestrianState(): void
+ resetState(): void

State Machine:

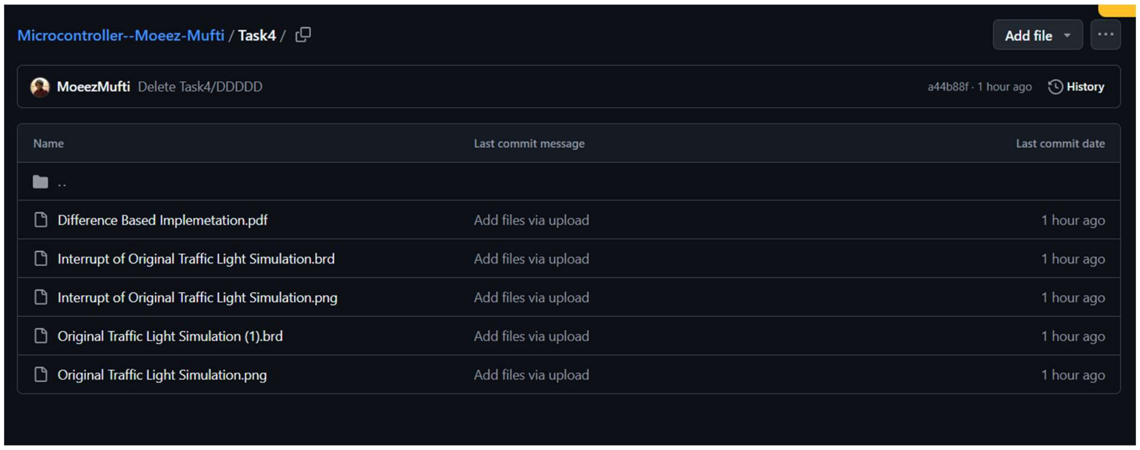

TrafficLight State Machine
wait for 5s
Green Light
greenLight
entry/turn on green light
exit /turn off green light
Pedestrian Red Light
pedRed
entry/turn on pedRed light
exit/turn off pedRed light
Button pressed do/check for button press
pedestrianRequest
[true]
Button pressed
[false]/wait for 10s
wait for 3s
Yellow Light
yellowLight
entry/turn on yellow light
exit/turn off yellow light
wait for 2s
wait for 3s
RedLight
redLight
entry/turn on red light
exit/turn off red light
Pedestrian Green Light
pedGreen
entry/turn on ped green light
exit/turn off ped green light

Sequence Diagram:



Task 4:

Github:

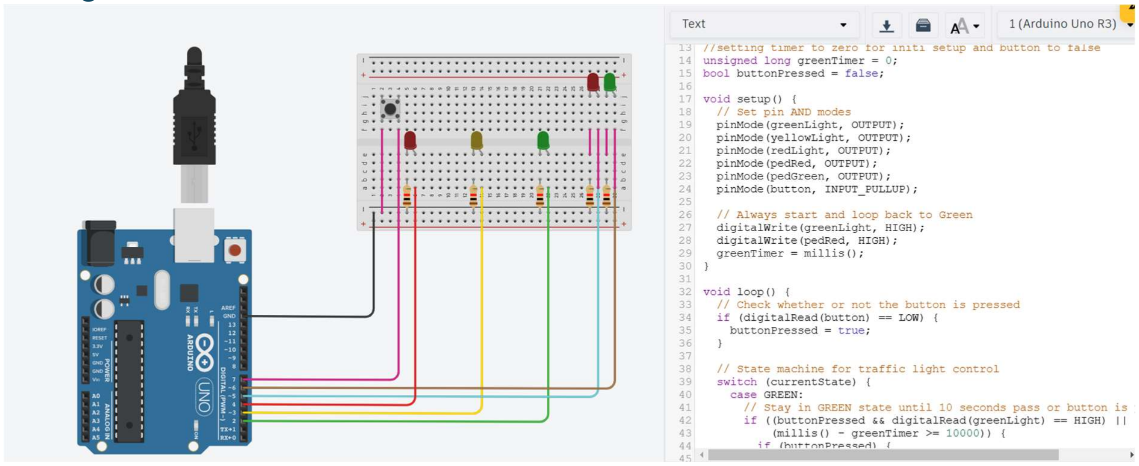


Microcontroller--Moeez-Mufti / Task4 /

MoeezMufti Delete Task4/DDDDD — a44b88f · 1 hour ago — History

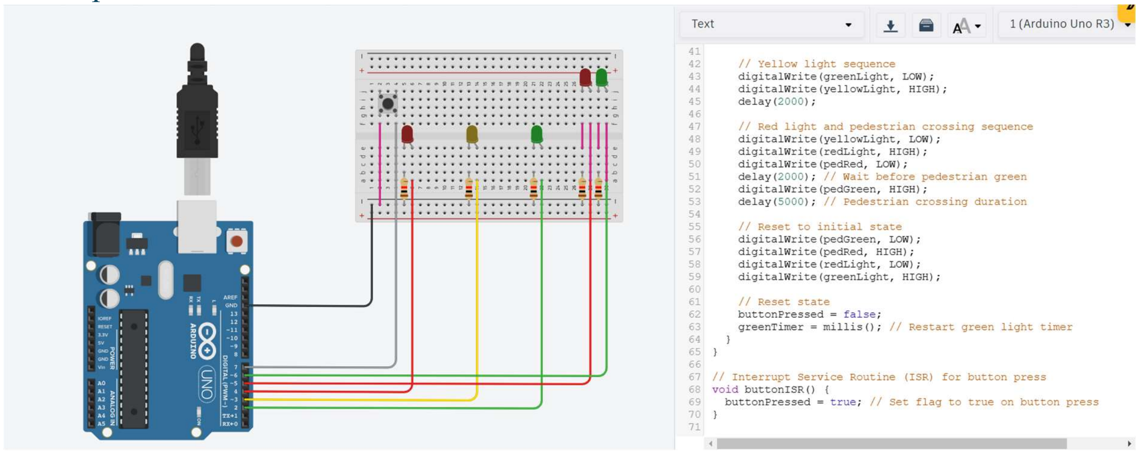
Name	Last commit message	Last commit date
..		
Difference Based Implemetation.pdf	Add files via upload	1 hour ago
Interrupt of Original Traffic Light Simulation.brd	Add files via upload	1 hour ago
Interrupt of Original Traffic Light Simulation.png	Add files via upload	1 hour ago
Original Traffic Light Simulation (1).brd	Add files via upload	1 hour ago
Original Traffic Light Simulation.png	Add files via upload	1 hour ago

Polling-Based:



```
//setting timer to zero for initi setup and button to false
unsigned long greenTimer = 0;
bool buttonPressed = false;

void setup() {
  // Set pin AND modes
  pinMode(greenLight, OUTPUT);
  pinMode(yellowLight, OUTPUT);
  pinMode(redLight, OUTPUT);
  pinMode(pedRed, OUTPUT);
  pinMode(pedGreen, OUTPUT);
  pinMode(button, INPUT_PULLUP);

  // Always start and loop back to Green
  digitalWrite(greenLight, HIGH);
  digitalWrite(pedRed, HIGH);
  greenTimer = millis();
}

void loop() {
  // Check whether or not the button is pressed
  if (digitalRead(button) == LOW) {
    buttonPressed = true;
  }

  // State machine for traffic light control
  switch (currentState) {
    case GREEN:
      // Stay in GREEN state until 10 seconds pass or button is
      if ((buttonPressed && digitalRead(greenLight) == HIGH) ||
          (millis() - greenTimer >= 10000)) {
        if (buttonPressed) {
```

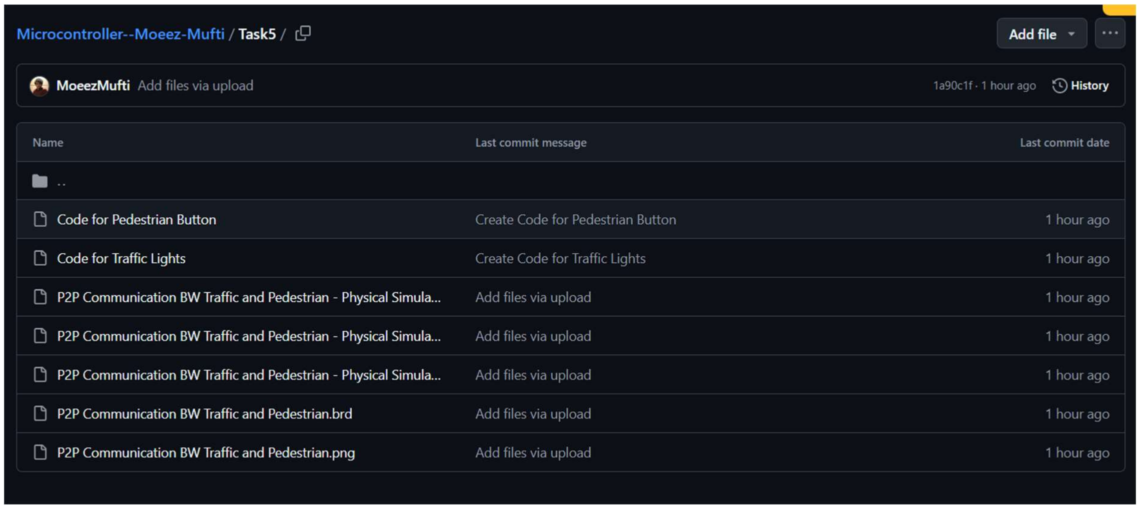
Interrupt-based:



```

    // Yellow light sequence
    digitalWrite(greenLight, LOW);
    digitalWrite(yellowLight, HIGH);
    delay(2000);

    // Red light and pedestrian crossing sequence
    digitalWrite(yellowLight, LOW);
    digitalWrite(redLight, HIGH);
    digitalWrite(pedRed, LOW);
    delay(2000); // Wait before pedestrian green
    digitalWrite(pedGreen, HIGH);
    delay(5000); // Pedestrian crossing duration

    // Reset to initial state
    digitalWrite(pedGreen, LOW);
    digitalWrite(pedRed, HIGH);
    digitalWrite(redLight, LOW);
    digitalWrite(greenLight, HIGH);

    // Reset state
    buttonPressed = false;
    greenTimer = millis(); // Restart green light timer
  }
}

// Interrupt Service Routine (ISR) for button press
void buttonISR() {
  buttonPressed = true; // Set flag to true on button press
}

```

Task 5

GitHub:

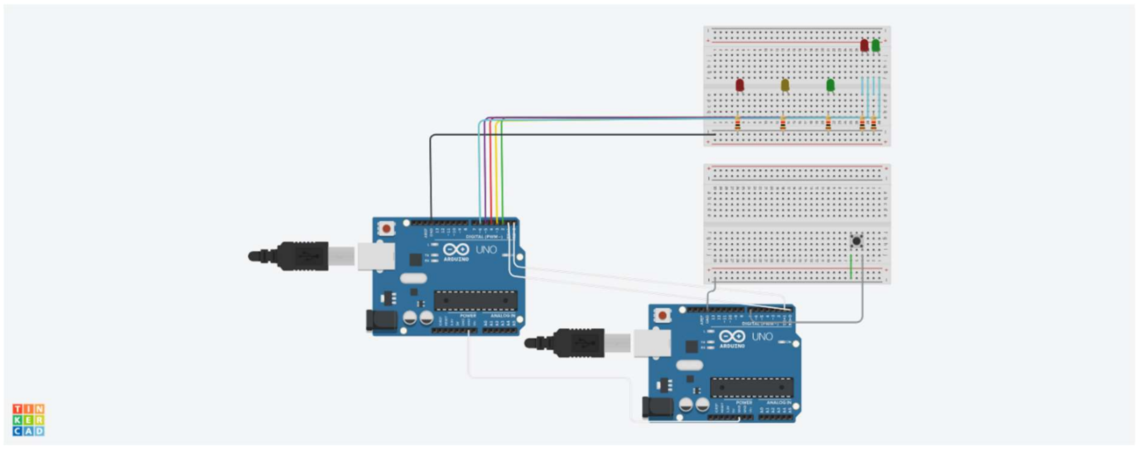


Microcontroller--Moeez-Mufti / Task5 /

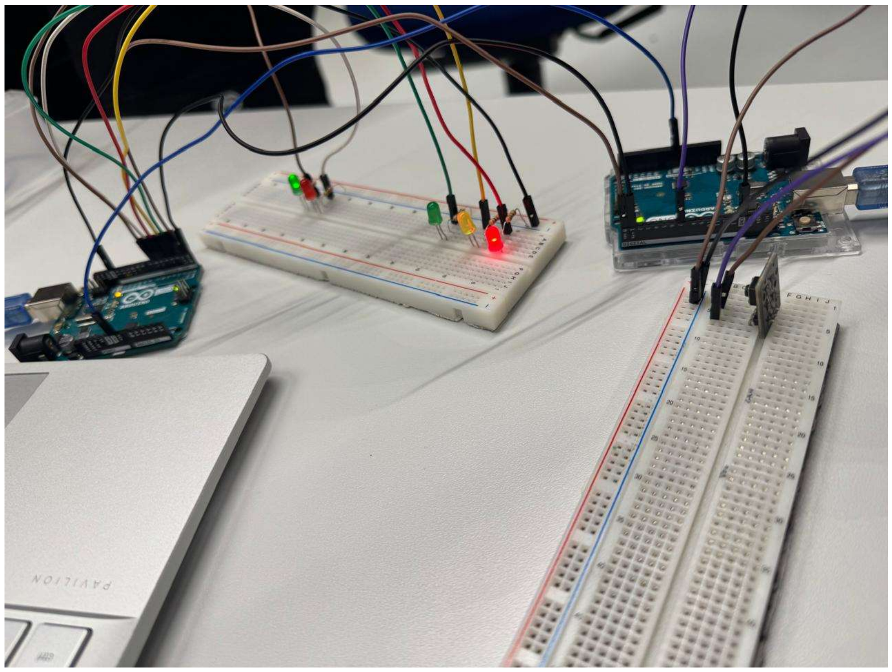
MoeezMufti Add files via upload 1a90c1f · 1 hour ago History

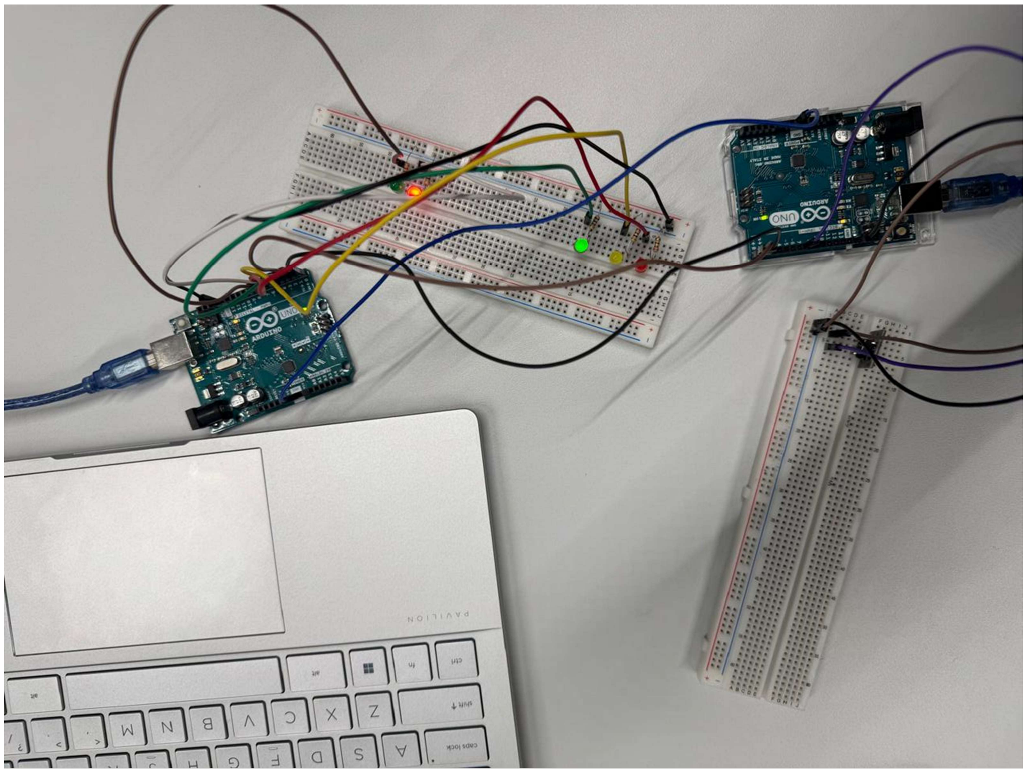
Name	Last commit message	Last commit date
..		
Code for Pedestrian Button	Create Code for Pedestrian Button	1 hour ago
Code for Traffic Lights	Create Code for Traffic Lights	1 hour ago
P2P Communication BW Traffic and Pedestrian - Physical Simula...	Add files via upload	1 hour ago
P2P Communication BW Traffic and Pedestrian - Physical Simula...	Add files via upload	1 hour ago
P2P Communication BW Traffic and Pedestrian - Physical Simula...	Add files via upload	1 hour ago
P2P Communication BW Traffic and Pedestrian.brd	Add files via upload	1 hour ago
P2P Communication BW Traffic and Pedestrian.png	Add files via upload	1 hour ago

TinkerCAD Simulation:



Physical Simulation:





Code:

Master Code (Pedestrian Button):

```
const int buttonPin = 7;
int buttonState = 0;

void setup() {
  Serial.begin(9600); // Start serial communication
  pinMode(buttonPin, INPUT_PULLUP); // Use internal pull-up resistor for the
button
}

void loop() {
  buttonState = digitalRead(buttonPin);

  if (buttonState == LOW) { // if Button is pressed
    Serial.println(1); // Send signal "1" to the Slave to communicate
    delay(500); // Debounce delay
  }
}
```

Slave Code (Traffic Lights):

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

enum TrafficState { GREEN, YELLOW, RED, PEDESTRIAN, RESET };
TrafficState currentState = GREEN;

bool pedestrianRequest = false;
unsigned long stateStartTime = 0;
unsigned long greenDuration = 10000;  // Default green light duration
const unsigned long PEDESTRIAN_GREEN_DURATION = 5000; //ms
const unsigned long YELLOW_DURATION = 3000;
const unsigned long PEDESTRIAN_WAIT_DURATION = 1000;

void setup() {
  Serial.begin(9600);
  pinMode(greenLight, OUTPUT);
  pinMode(yellowLight, OUTPUT);
  pinMode(redLight, OUTPUT);
  pinMode(pedRed, OUTPUT);
  pinMode(pedGreen, OUTPUT);

  // Start with green light and pedestrian red
  digitalWrite(greenLight, HIGH);
  digitalWrite(pedRed, HIGH);
  stateStartTime = millis();
}

void loop() {
  // Check for pedestrian request via serial input
  if (Serial.available() > 0) {
    char signal = Serial.read();
    if (signal == '1') {
      pedestrianRequest = true;
      // Adjust green duration to extend by 3 seconds from the moment of
button press
      unsigned long currentTime = millis();
      if (currentTime - stateStartTime < greenDuration - 3000) {
        greenDuration = (currentTime - stateStartTime) + 3000;
      }
    }
  }

  unsigned long currentTime = millis();

  switch (currentState) {
    case GREEN:
      if (currentTime - stateStartTime >= greenDuration) {
        currentState = YELLOW;
        stateStartTime = currentTime;
      }
      break;

    case YELLOW:
      digitalWrite(greenLight, LOW);
      digitalWrite(yellowLight, HIGH);
      if (currentTime - stateStartTime >= YELLOW_DURATION) {
        digitalWrite(yellowLight, LOW);
        digitalWrite(redLight, HIGH);
        currentState = RED;
        stateStartTime = currentTime;
      }
      break;

    case RED:
      // Handle pedestrian request in RED state
      if (pedestrianRequest) {
        if (currentTime - stateStartTime >= PEDESTRIAN_WAIT_DURATION) {
          digitalWrite(pedRed, LOW);
          digitalWrite(pedGreen, HIGH);
          currentState = PEDESTRIAN;
          stateStartTime = currentTime;
        }
      } else {
        currentState = RESET;
        stateStartTime = currentTime;
      }
      break;

    case PEDESTRIAN:
      if (currentTime - stateStartTime >= PEDESTRIAN_GREEN_DURATION) {
        digitalWrite(pedGreen, LOW);
        digitalWrite(pedRed, HIGH);
        pedestrianRequest = false; // Reset pedestrian request
        currentState = RESET;
        stateStartTime = currentTime;
      }
      break;

    case RESET:
      digitalWrite(redLight, LOW);
      digitalWrite(greenLight, HIGH);
      greenDuration = 10000; // Reset green duration to default timing
      currentState = GREEN;
      stateStartTime = currentTime;
      break;
  }
}
```