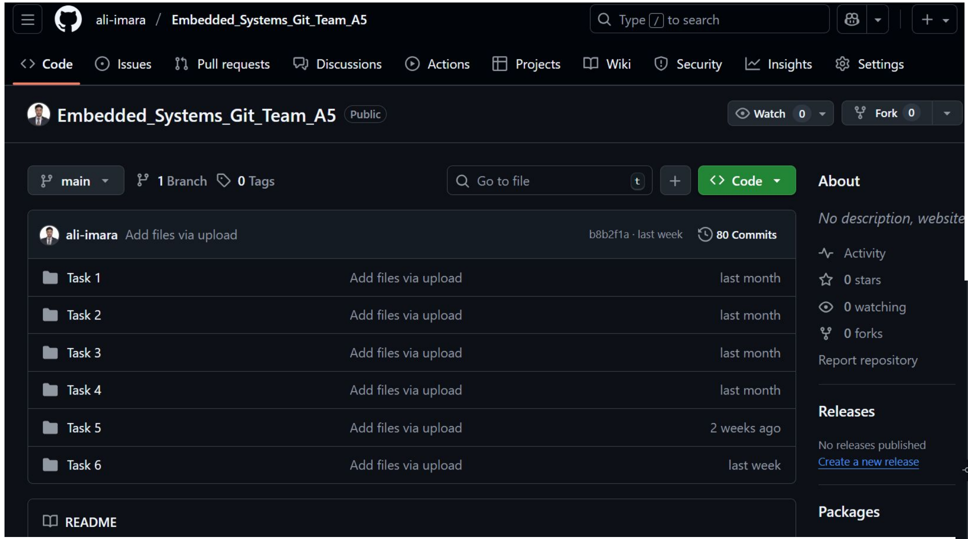
Presentation

TEAM A5

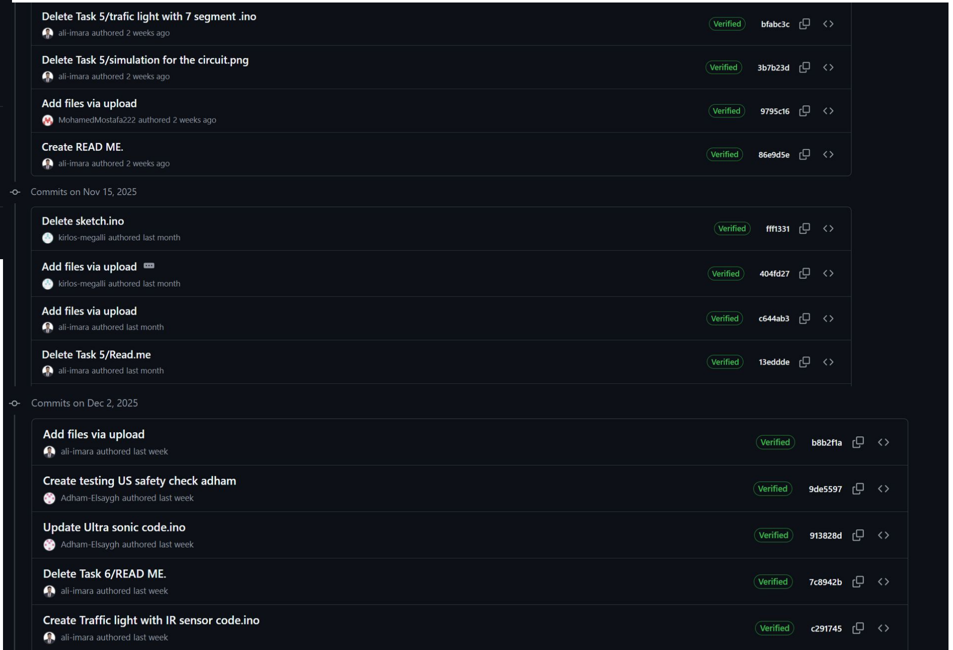
Adham Elsaygh-Ali Abdelkhalek-Mohamed Awis-Kirlos Megalli

Organisation



ali-imara Add files via upload b8b2f1a · last week 80 Commits

Task 1	Add files via upload	last month
Task 2	Add files via upload	last month
Task 3	Add files via upload	last month
Task 4	Add files via upload	last month
Task 5	Add files via upload	2 weeks ago
Task 6	Add files via upload	last week



Delete Task 5/trafic light with 7 segment .ino
ali-imara authored 2 weeks ago — Verified bfabc3c

Delete Task 5/simulation for the circuit.png
ali-imara authored 2 weeks ago — Verified 3b7b23d

Add files via upload
MohamedMostafa222 authored 2 weeks ago — Verified 9795c16

Create READ ME.
ali-imara authored 2 weeks ago — Verified 86e9d5e

Commits on Nov 15, 2025

Delete sketch.ino
kirlos-megalli authored last month — Verified fff1331

Add files via upload
kirlos-megalli authored last month — Verified 404fd27

Add files via upload
ali-imara authored last month — Verified c644ab3

Delete Task 5/Read.me
ali-imara authored last month — Verified 13eddde

Commits on Dec 2, 2025

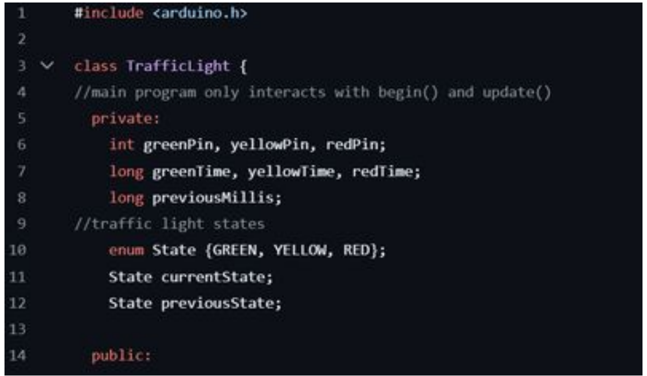
Add files via upload
ali-imara authored last week — Verified b8b2f1a

Create testing US safety check adham
Adham-Elsaygh authored last week — Verified 9de5597

Update Ultra sonic code.ino
Adham-Elsaygh authored last week — Verified 913828d

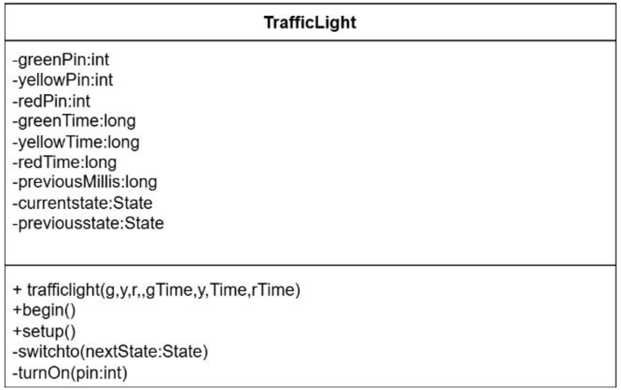
Delete Task 6/READ ME.
ali-imara authored last week — Verified 7c8942b

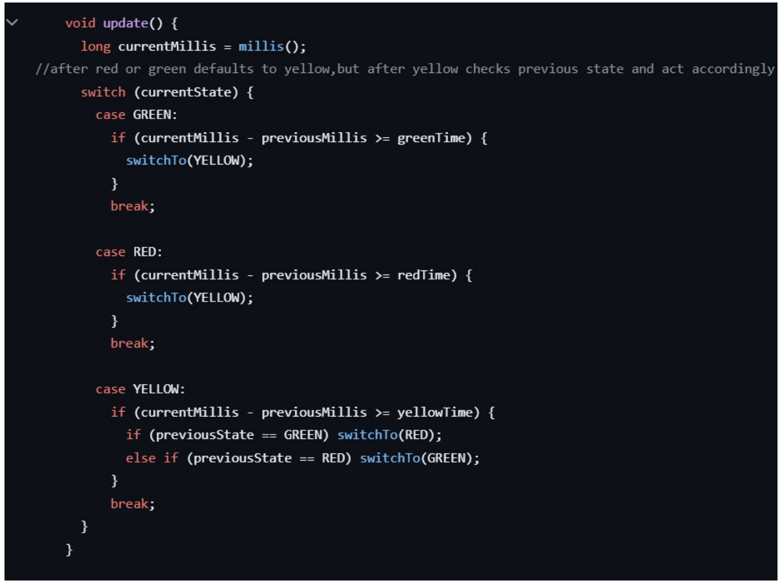
Create Traffic light with IR sensor code.ino
ali-imara authored last week — Verified c291745



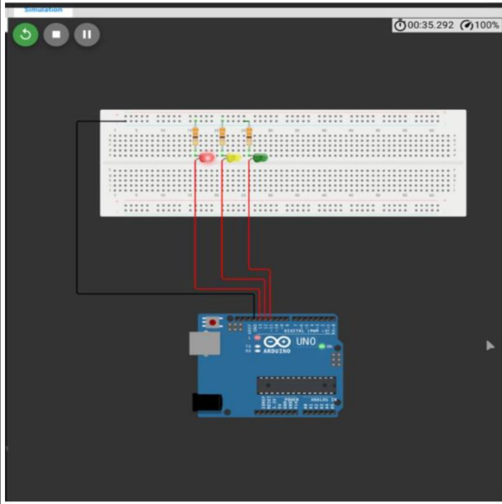
```
#include <arduino.h>

class TrafficLight {
//main program only interacts with begin() and update()
  private:
    int greenPin, yellowPin, redPin;
    long greenTime, yellowTime, redTime;
    long previousMillis;
//traffic light states
    enum State {GREEN, YELLOW, RED};
    State currentState;
    State previousState;

  public:
```

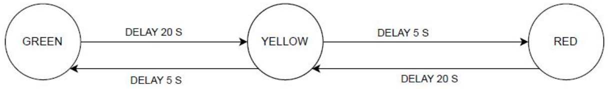
Task 1





```
    void update() {
      long currentMillis = millis();
//after red or green defaults to yellow,but after yellow checks previous state and act accordingly
      switch (currentState) {
        case GREEN:
          if (currentMillis - previousMillis >= greenTime) {
            switchTo(YELLOW);
          }
          break;

        case RED:
          if (currentMillis - previousMillis >= redTime) {
            switchTo(YELLOW);
          }
          break;

        case YELLOW:
          if (currentMillis - previousMillis >= yellowTime) {
            if (previousState == GREEN) switchTo(RED);
            else if (previousState == RED) switchTo(GREEN);
          }
          break;
      }
    }
```

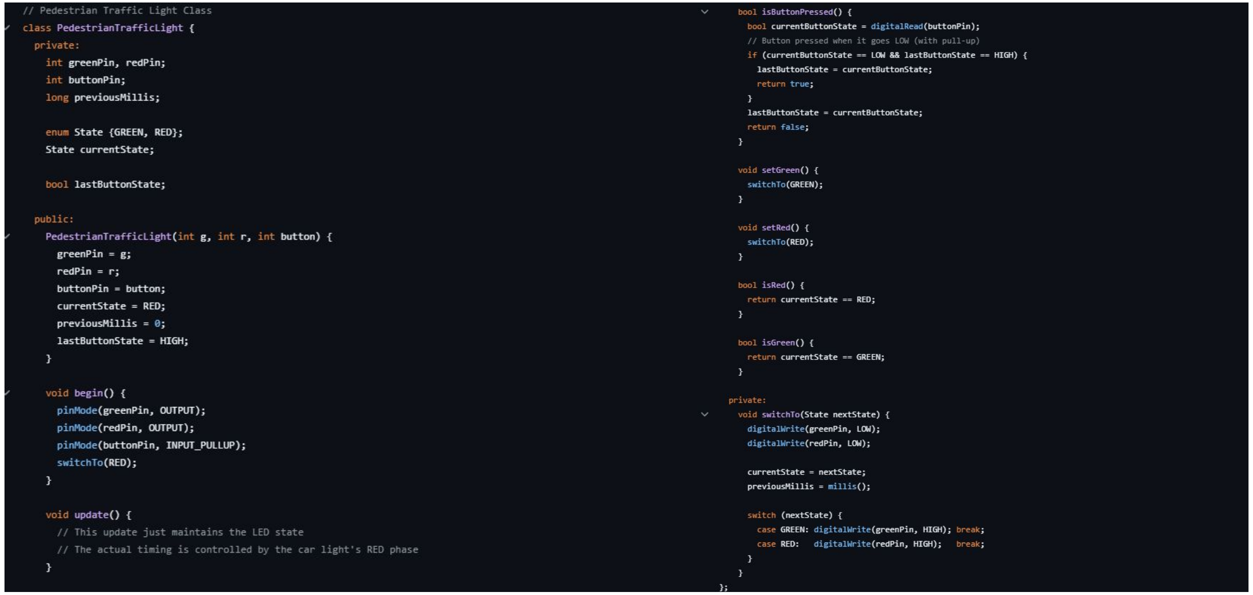




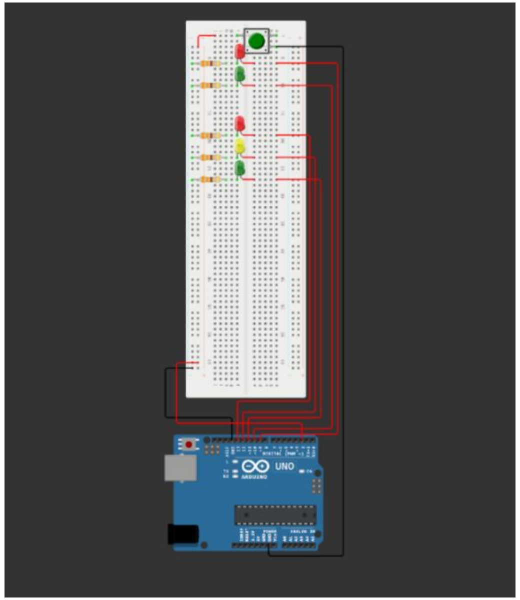
Implementation is structured using Object-Oriented Programming (OOP) principles, with the main function detailing execution flow after hardware initialization

Task 2

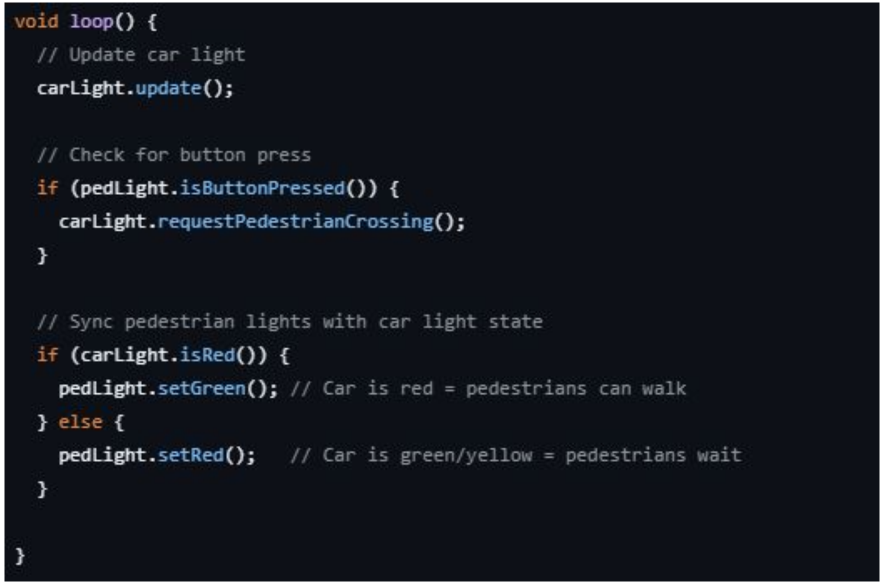
A new class was created for the pedestrian traffic light.



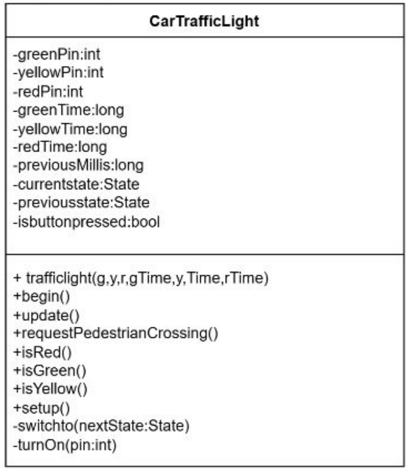
```cpp
// Pedestrian Traffic Light Class
class PedestrianTrafficLight {
  private:
    int greenPin, redPin;
    int buttonPin;
    long previousMillis;

    enum State {GREEN, RED};
    State currentState;

    bool lastButtonState;

  public:
    PedestrianTrafficLight(int g, int r, int button) {
      greenPin = g;
      redPin = r;
      buttonPin = button;
      currentState = RED;
      previousMillis = 0;
      lastButtonState = HIGH;
    }

    void begin() {
      pinMode(greenPin, OUTPUT);
      pinMode(redPin, OUTPUT);
      pinMode(buttonPin, INPUT_PULLUP);
      switchTo(RED);
    }

    void update() {
      // This update just maintains the LED state
      // The actual timing is controlled by the car light's RED phase
    }

    bool isButtonPressed() {
      bool currentButtonState = digitalRead(buttonPin);
      // Button pressed when it goes LOW (with pull-up)
      if (currentButtonState == LOW && lastButtonState == HIGH) {
        lastButtonState = currentButtonState;
        return true;
      }
      lastButtonState = currentButtonState;
      return false;
    }

    void setGreen() {
      switchTo(GREEN);
    }

    void setRed() {
      switchTo(RED);
    }

    bool isRed() {
      return currentState == RED;
    }

    bool isGreen() {
      return currentState == GREEN;
    }

  private:
    void switchTo(State nextState) {
      digitalWrite(greenPin, LOW);
      digitalWrite(redPin, LOW);

      currentState = nextState;
      previousMillis = millis();

      switch (nextState) {
        case GREEN: digitalWrite(greenPin, HIGH); break;
        case RED:   digitalWrite(redPin, HIGH);   break;
      }
    }
};
```

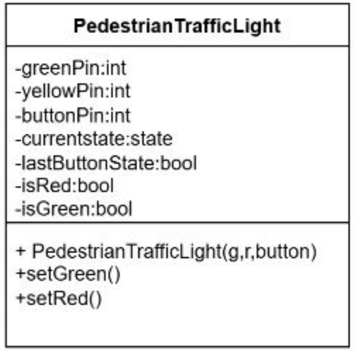


Task 2



```
void loop() {
  // Update car light
  carLight.update();

  // Check for button press
  if (pedLight.isButtonPressed()) {
    carLight.requestPedestrianCrossing();
  }

  // Sync pedestrian lights with car light state
  if (carLight.isRed()) {
    pedLight.setGreen(); // Car is red = pedestrians can walk
  } else {
    pedLight.setRed();   // Car is green/yellow = pedestrians wait
  }

}
```



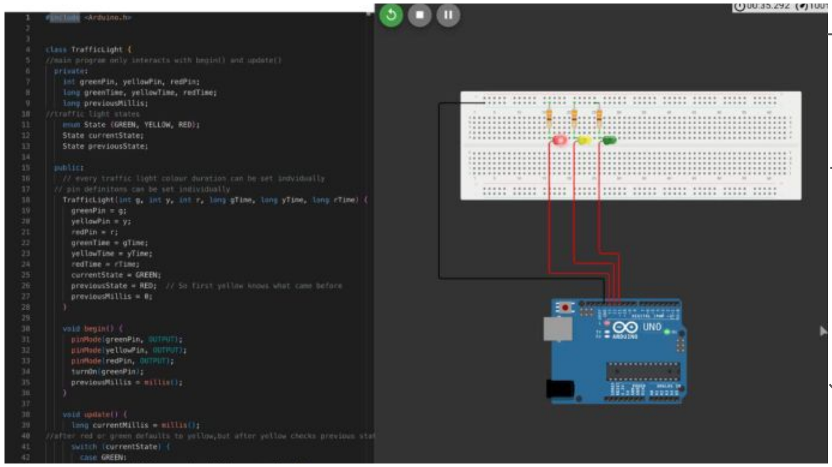


Task 3,4

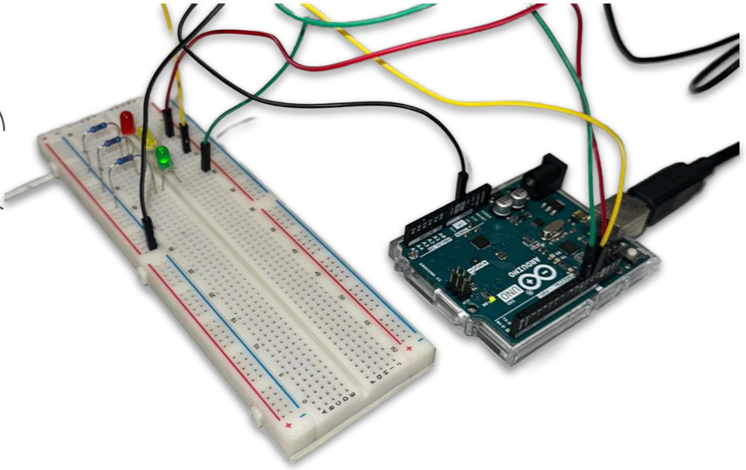
- Mapping of task 1 to a (real) Arduino
- Mapping of task 2 to a (real) Arduino

Task 3

The simulation of Task 1 results are consistent with the behavior of the real-life circuit.



```
#include <Arduino.h>


class TrafficLight {
//main program only interacts with begin() and update()
  private:
    int greenPin, yellowPin, redPin;
    long greenTime, yellowTime, redTime;
    long previousMillis;
//traffic light states
    enum State {GREEN, YELLOW, RED};
    State currentState;
    State previousState;

  public:
    // every traffic light colour duration can be set indvidually
  // pin definitons can be set individually
    TrafficLight(int g, int y, int r, long gTime, long yTime, long rTime) {
      greenPin = g;
      yellowPin = y;
      redPin = r;
      greenTime = gTime;
      yellowTime = yTime;
      redTime = rTime;
      currentState = GREEN;
      previousState = RED;  // So first yellow knows what came before
      previousMillis = 0;
    }

    void begin() {
      pinMode(greenPin, OUTPUT);
      pinMode(yellowPin, OUTPUT);
      pinMode(redPin, OUTPUT);
      turnOn(greenPin);
      previousMillis = millis();
    }

    void update() {
      long currentMillis = millis();
//after red or green defaults to yellow,but after yellow checks previous stat
      switch (currentState) {
        case GREEN:
```



Task 3

The measured voltage from the real red LED closely corresponds to the calculated voltage.

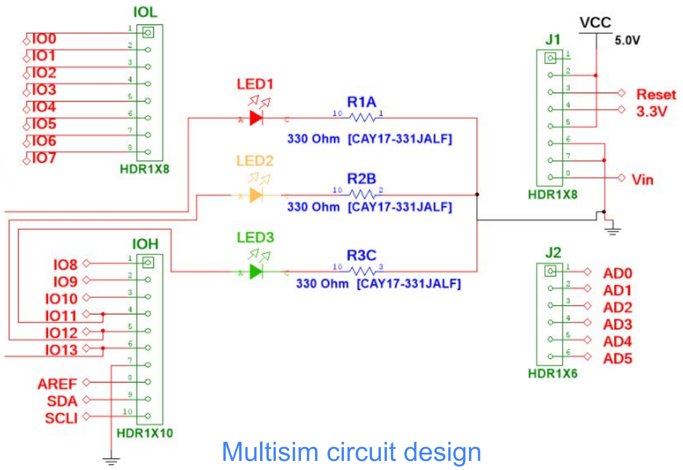
Voltage:

1- Pin13,12,11: V=5.0v

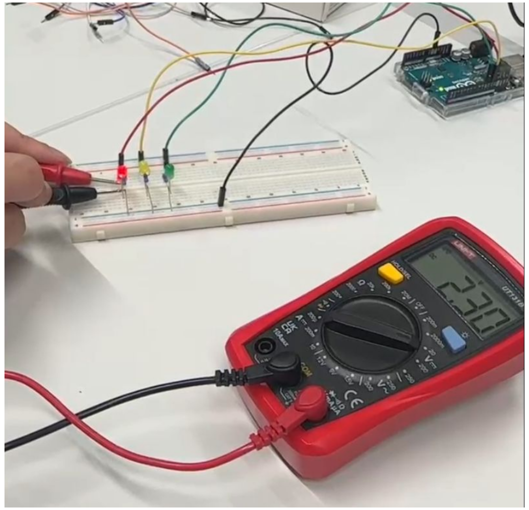
2- Voltage at Red led= 2.0v (1.8-2.2)

3- Voltage at yellow led= 2.1v (2.0-2.4)

4- Voltage at green led= 2.2v (2.0-3.0)



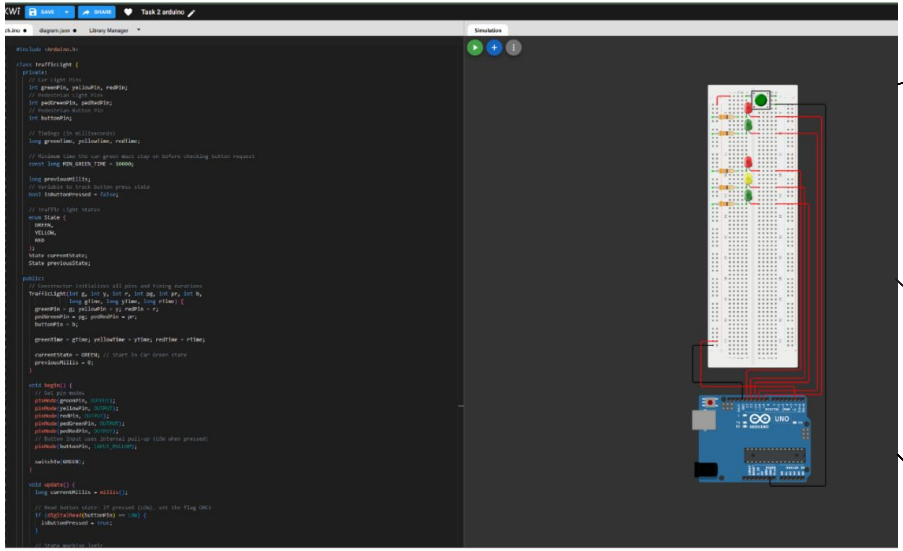
Multisim circuit design



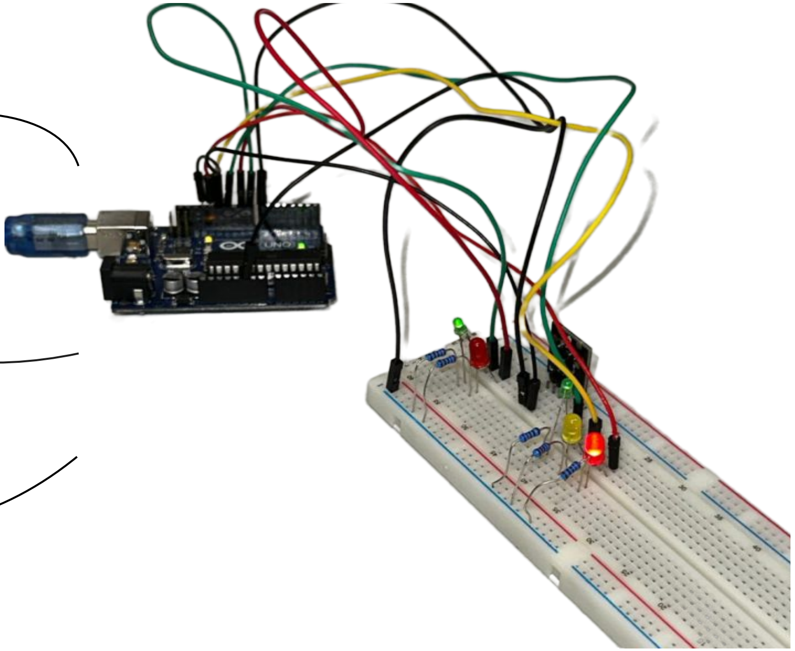
Real life calculation

Task 4

The simulation of Task 2 results are consistent with the behavior of the real-life circuit.



```
#include <Arduino.h>

class TrafficLight {
  private:
    // Car Light Pins
    int greenPin, yellowPin, redPin;
    // Pedestrian Light Pins
    int pedGreenPin, pedRedPin;
    // Pedestrian Button Pin
    int buttonPin;

    // Timings (in milliseconds)
    long greenTime, yellowTime, redTime;

    // Minimum time the car green must stay on before checking button request
    const long MIN_GREEN_TIME = 10000;

    long previousMillis;
    // Variable to track button press state
    bool isButtonPressed = false;

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

  public:
    // Constructor initializes all pins and timing durations
    TrafficLight(int g, int y, int r, int pg, int pr, int b,
                 long gTime, long yTime, long rTime) {
      greenPin = g; yellowPin = y; redPin = r;
      pedGreenPin = pg; pedRedPin = pr;
      buttonPin = b;

      greenTime = gTime; yellowTime = yTime; redTime = rTime;

      currentState = GREEN; // Start in Car Green state
      previousMillis = 0;
    }

    void begin() {
      // Set pin modes
      pinMode(greenPin, OUTPUT);
      pinMode(yellowPin, OUTPUT);
      pinMode(redPin, OUTPUT);
      pinMode(pedGreenPin, OUTPUT);
      pinMode(pedRedPin, OUTPUT);
      // Button input uses internal pull-up (LOW when pressed)
      pinMode(buttonPin, INPUT_PULLUP);

      switchTo(GREEN);
    }

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

      // Read button state: if pressed (LOW), set the flag ONCE
      if (digitalRead(buttonPin) == LOW) {
        isButtonPressed = true;
      }

      // State machine logic
```



Task 5

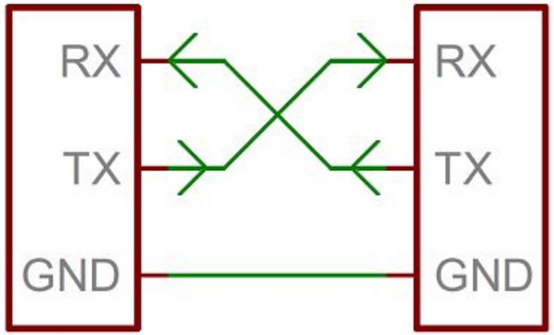
Connections Techniques to connect µCs

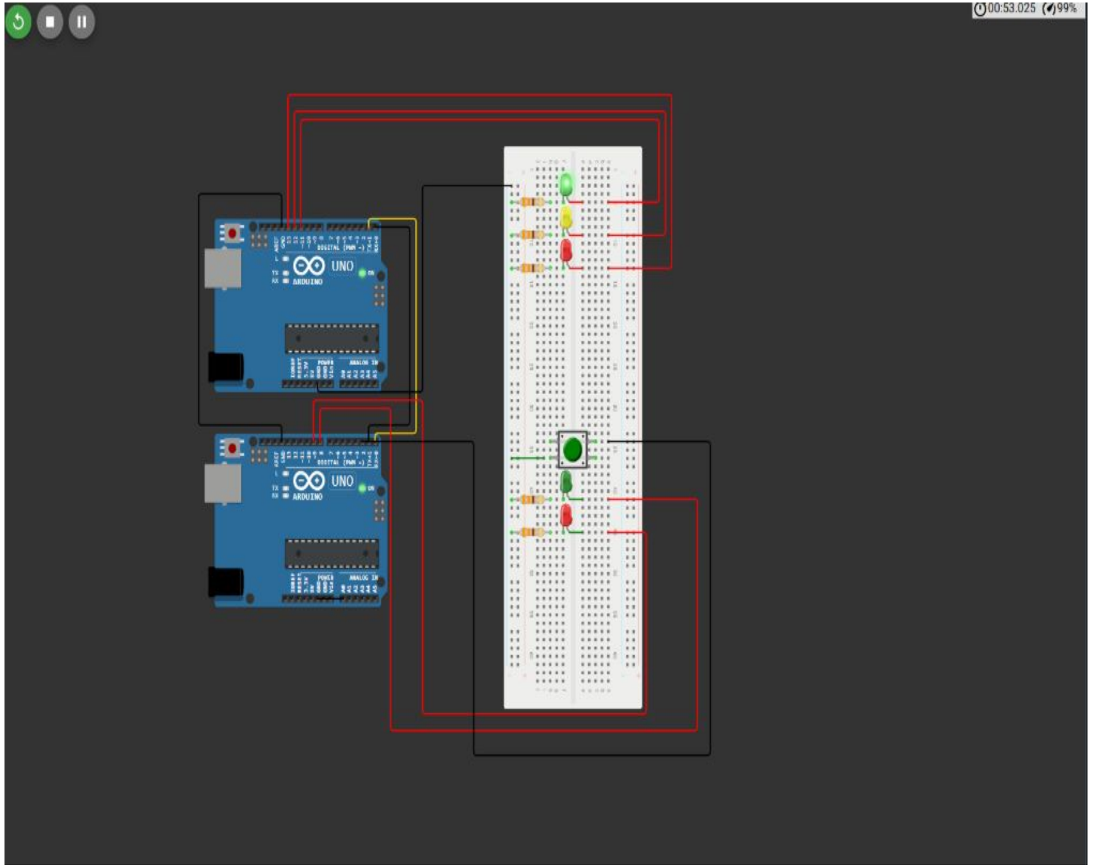
UART SPI

I2C CAN

Task 5

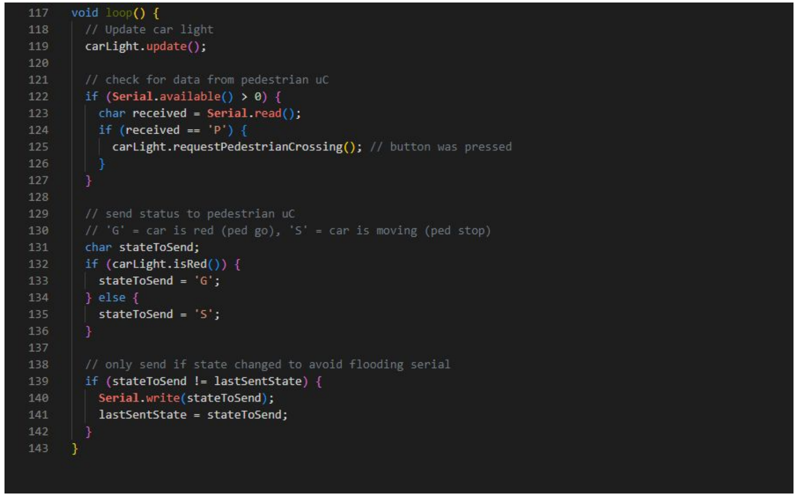
Adjusting the circuit from task 4 by adding a second microcontroller, and wiring the two microcontrollers according to the UART wiring.



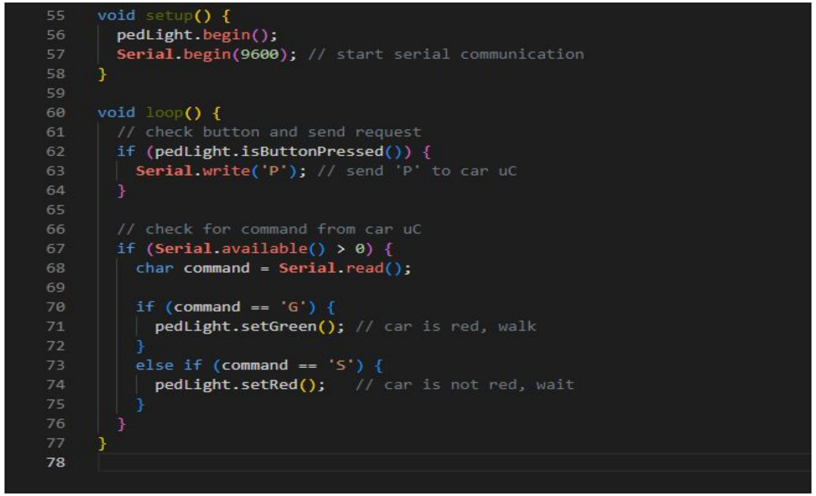


Task 5

Starting the serial communication Between the car traffic light and the pedestrian light



```
void loop() {
  // Update car light
  carLight.update();

  // check for data from pedestrian uC
  if (Serial.available() > 0) {
    char received = Serial.read();
    if (received == 'P') {
      carLight.requestPedestrianCrossing(); // button was pressed
    }
  }

  // send status to pedestrian uC
  // 'G' = car is red (ped go), 'S' = car is moving (ped stop)
  char stateToSend;
  if (carLight.isRed()) {
    stateToSend = 'G';
  } else {
    stateToSend = 'S';
  }

  // only send if state changed to avoid flooding serial
  if (stateToSend != lastSentState) {
    Serial.write(stateToSend);
    lastSentState = stateToSend;
  }
}
```



```
void setup() {
  pedLight.begin();
  Serial.begin(9600); // start serial communication
}

void loop() {
  // check button and send request
  if (pedLight.isButtonPressed()) {
    Serial.write('P'); // send 'P' to car uC
  }

  // check for command from car uC
  if (Serial.available() > 0) {
    char command = Serial.read();

    if (command == 'G') {
      pedLight.setGreen(); // car is red, walk
    }
    else if (command == 'S') {
      pedLight.setRed();   // car is not red, wait
    }
  }
}
```

Thanks For Your Attention