

Presentation

TEAM A5

Adham Elsaygh-Ali Abdelkhalek-Mohamed Awis-Kirlos Megalli

Organisation

Delete Task 5/traffic light with 7 segment.ino	Verified	bfa6c3c	ali-imara authored 2 weeks ago
Delete Task 5/simulation for the circuit.png	Verified	3b7b23d	ali-imara authored 2 weeks ago
Add files via upload	Verified	9795c16	MohamedMostafa222 authored 2 weeks ago
Create README	Verified	86a7d5e	ali-imara authored 2 weeks ago

Commits on Nov 15, 2025

Delete sketch.ino	Verified	ff1331	kirios-megali authored last month
Add files via upload	Verified	4046d27	kirios-megali authored last month
Add files via upload	Verified	c644ab3	ali-imara authored last month
Delete Task 5/Read.me	Verified	13edddc	ali-imara authored last month

Commits on Dec 2, 2025

Add files via upload	Verified	b8b2f1a	ali-imara authored last week
Create testing US safety check adham	Verified	9de5597	Adham-Elsaygh authored last week
Update Ultra sonic code.ino	Verified	91382bd	Adham-Elsaygh authored last week
Delete Task 6/READ ME	Verified	7c8942b	ali-imara authored last week
Create Traffic light with IR sensor code.ino	Verified	c291745	ali-imara authored last week

```
1  #include <arduino.h>
2
3  class TrafficLight {
4      //main program only interacts with begin() and update()
5      private:
6          int greenPin, yellowPin, redPin;
7          long greenTime, yellowTime, redTime;
8          long previousMillis;
9      //traffic light states
10     enum State {GREEN, YELLOW, RED};
11     State currentState;
12     State previousState;
13
14     public:
```

ali-imara / Embedded_Systems_Git_Team_A5

Code Issues Pull requests Discussions Actions Projects Wiki Security Insights

Embedded_Systems_Git_Team_A5 (Public)

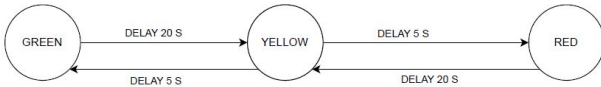
main	1 Branch	0 Tags	Go to file	+	Code
ali-imara	Add files via upload	b8b2f1a · last week	80 Commits		
Task 1	Add files via upload		last month		
Task 2	Add files via upload		last month		
Task 3	Add files via upload		last month		
Task 4	Add files via upload		last month		
Task 5	Add files via upload		2 weeks ago		
Task 6	Add files via upload		last week		

Task 1

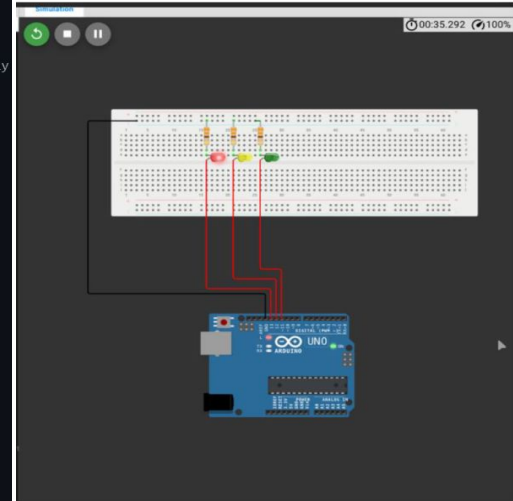
TrafficLight

-greenPin:int
-yellowPin:int
-redPin:int
-greenTime:long
-yellowTime:long
-redTime:long
-previousMillis:long
-currentstate:State
-previousstate:State

+ trafficlight(g,y,r,,gTime,yTime,rTime)
+ begin()
+ setup()
- switchto(nextState:State)
- turnOn(pin:int)



```
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.

```
// 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);
    }

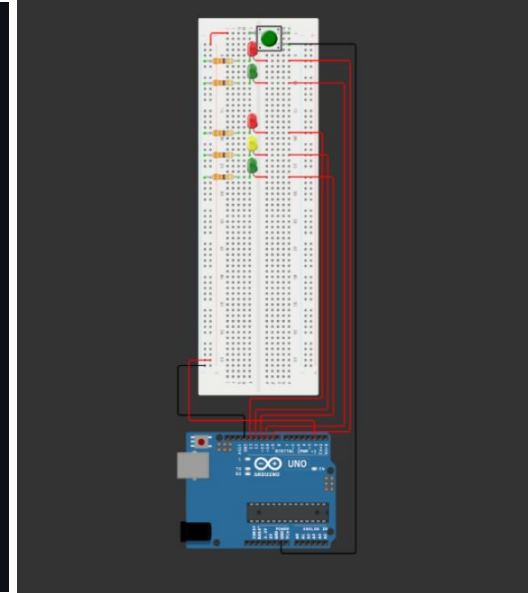
    bool isRed() {
        return currentState == RED;
    }

    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

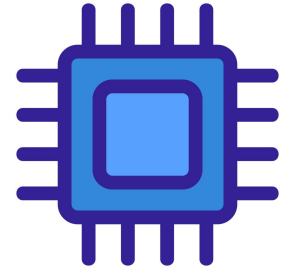
```
void loop() {  
  // Update car light  
  carLight.update();  
  
  // Check for button press  
  if (pedLight.isButtonPressed()) {  
    carLight.requestPedestrianCrossing();  
  }  
  
  // Sync pedestrian lights with car light state  
  if (carLight.isRed()) {  
    pedLight.setGreen(); // Car is red = pedestrians can walk  
  } else {  
    pedLight.setRed();   // Car is green/yellow = pedestrians wait  
  }  
}
```

CarTrafficLight
-greenPin:int -yellowPin:int -redPin:int -greenTime:long -yellowTime:long -redTime:long -previousMillis:long -currentstate:State -previousstate:State -isbuttonpressed:bool
+ trafficlight(g,y,r,gTime,y,Time,rTime) +begin() +update() +requestPedestrianCrossing() +isRed() +isGreen() +isYellow() +setup() +switchto(nextState:State) +turnOn(pin:int)

PedestrianTrafficLight
-greenPin:int -yellowPin:int -buttonPin:int -currentstate:state -lastButtonState:bool -isRed:bool -isGreen:bool
+ PedestrianTrafficLight(g,r,button) +setGreen() +setRed()

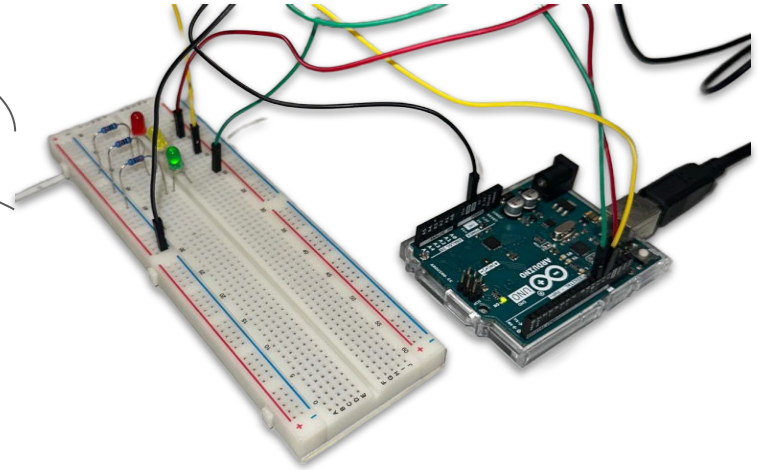
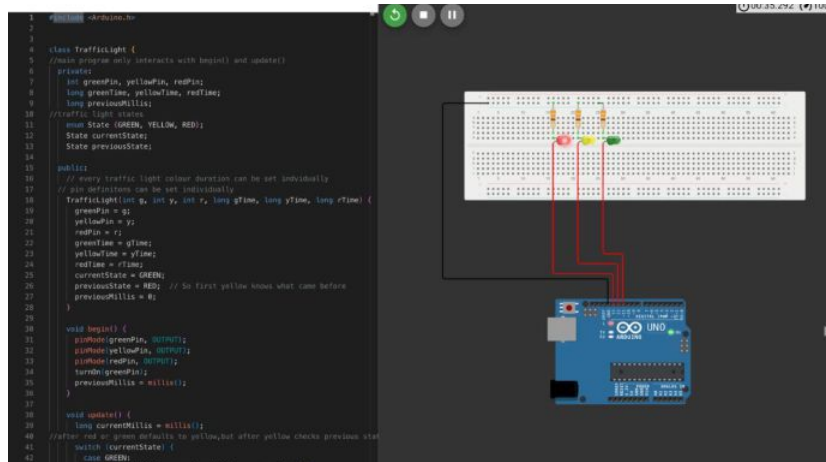
Task 3,4

- 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.

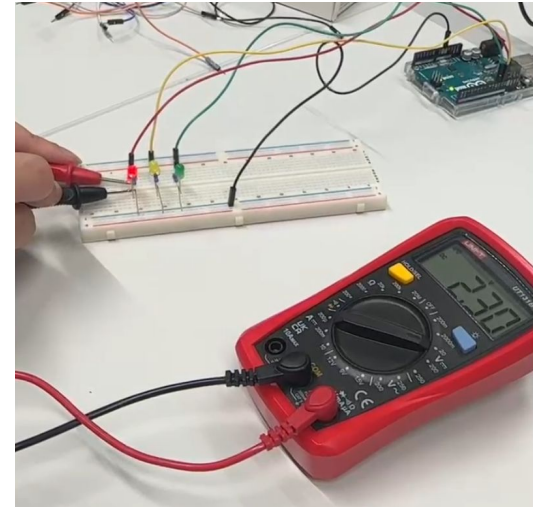
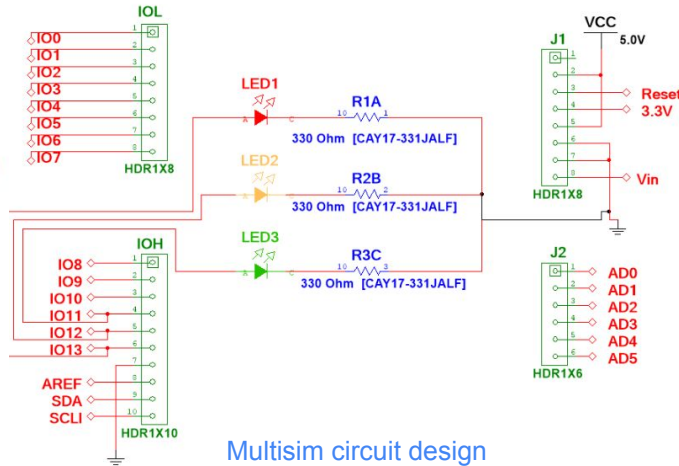


Task 3

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

Voltage:

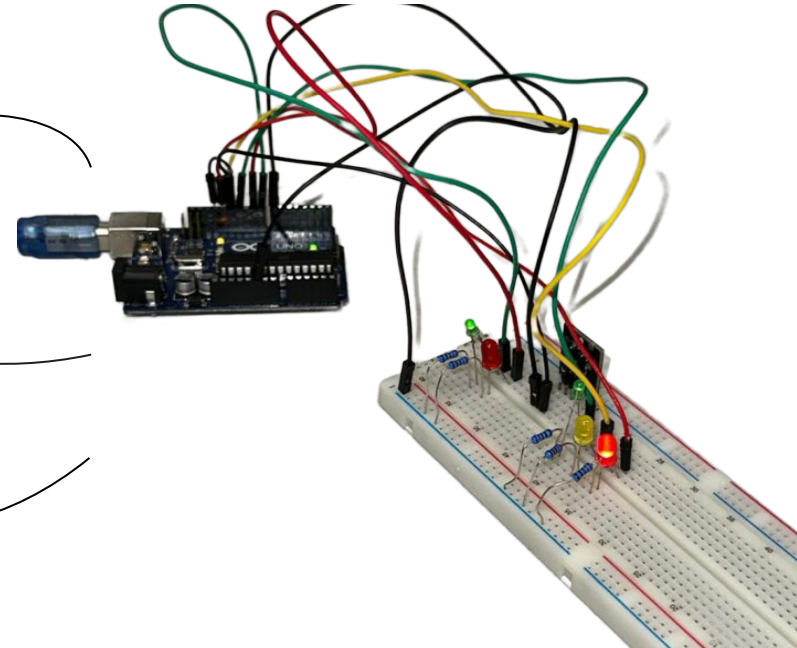
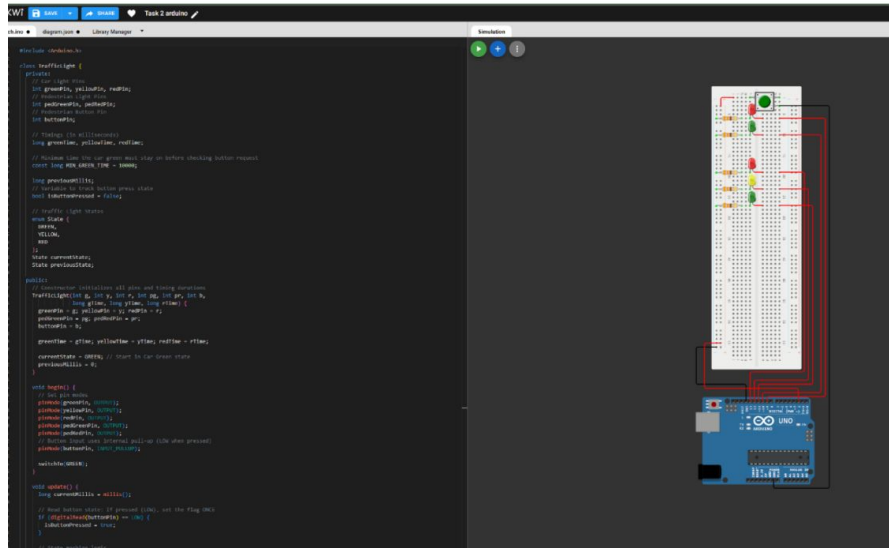
- 1- Pin13,12,11: V=5.0v
- 2- Voltage at Red led= 2.0v (1.8-2.2)
- 3- Voltage at yellow led= 2.1v (2.0-2.4)
- 4- Voltage at green led= 2.2v (2.0-3.0)



Real life calculation

Task 4

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



Task 5

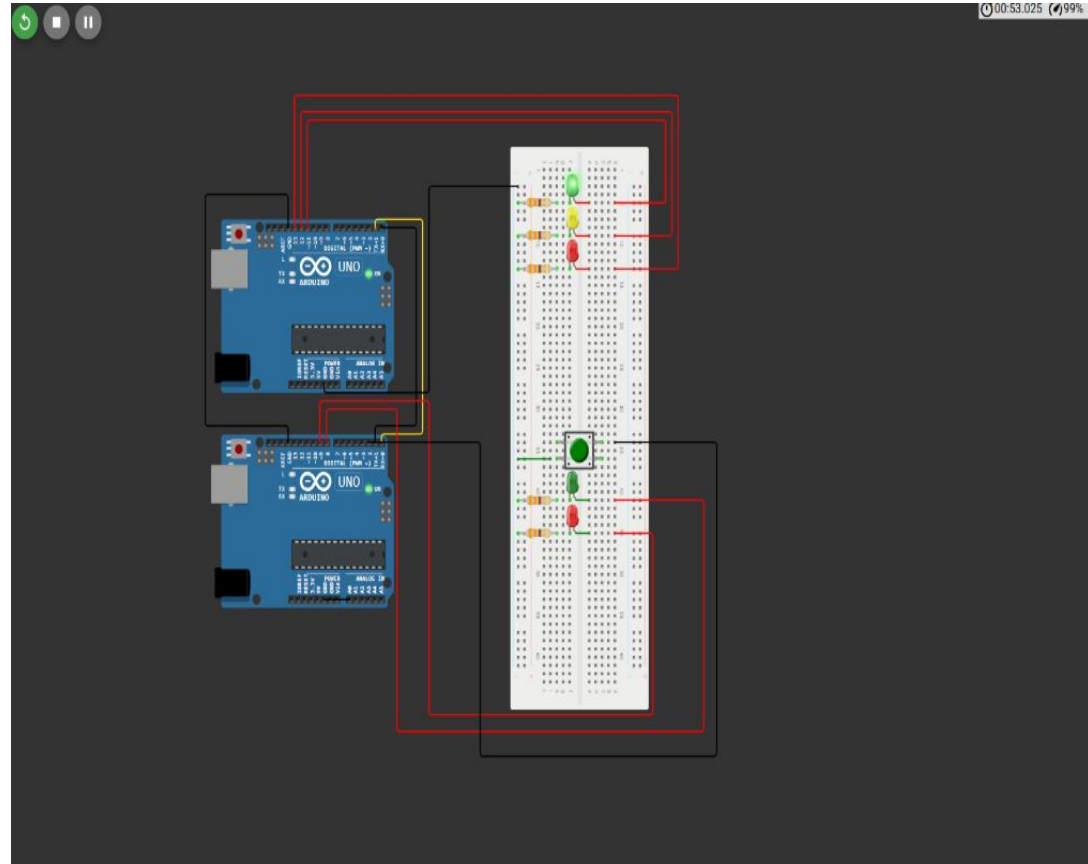
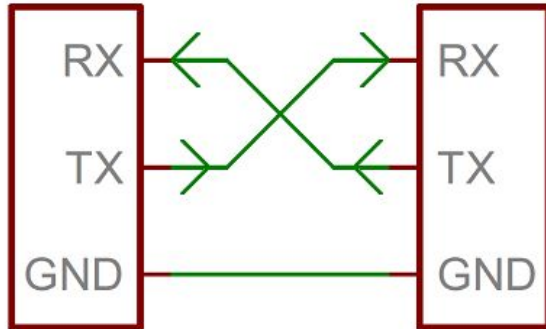
Connections Techniques to connect μ Cs

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

```
117 void loop() {
118     // Update car light
119     carLight.update();
120
121     // check for data from pedestrian uC
122     if (Serial.available() > 0) {
123         char received = Serial.read();
124         if (received == 'P') {
125             carLight.requestPedestrianCrossing(); // button was pressed
126         }
127     }
128
129     // send status to pedestrian uC
130     // 'G' = car is red (ped go), 'S' = car is moving (ped stop)
131     char stateToSend;
132     if (carLight.isRed()) {
133         stateToSend = 'G';
134     } else {
135         stateToSend = 'S';
136     }
137
138     // only send if state changed to avoid flooding serial
139     if (stateToSend != lastSentState) {
140         Serial.write(stateToSend);
141         lastSentState = stateToSend;
142     }
143 }
```

```
55 void setup() {
56     pedLight.begin();
57     Serial.begin(9600); // start serial communication
58 }
59
60 void loop() {
61     // check button and send request
62     if (pedLight.isButtonPressed()) {
63         Serial.write('P'); // send 'P' to car uC
64     }
65
66     // check for command from car uC
67     if (Serial.available() > 0) {
68         char command = Serial.read();
69
70         if (command == 'G') {
71             pedLight.setGreen(); // car is red, walk
72         }
73         else if (command == 'S') {
74             pedLight.setRed(); // car is not red, wait
75         }
76     }
77 }
78
```

Thanks For Your Attention