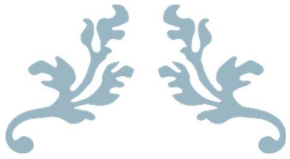


MICROCONTROLLER

Milestone 2 - Documentation

20.12.2024

Moeez Mufti – 1233013
moeez.mufti@stud.hshl.de

Contents

Abstract

This project demonstrates a traffic light T-Junction using two Arduino Uno microcontrollers to manage a traffic intersection with pedestrian crossing capabilities. The system comprises a master controller managing primary intersection control and pedestrian signals, and a slave controller handling synchronized secondary traffic signals. The implementation features real-time communication between controllers, dynamic timing adjustments through pedestrian input, and fail-safe state transitions, demonstrating an effective solution for small to medium-sized intersection management.

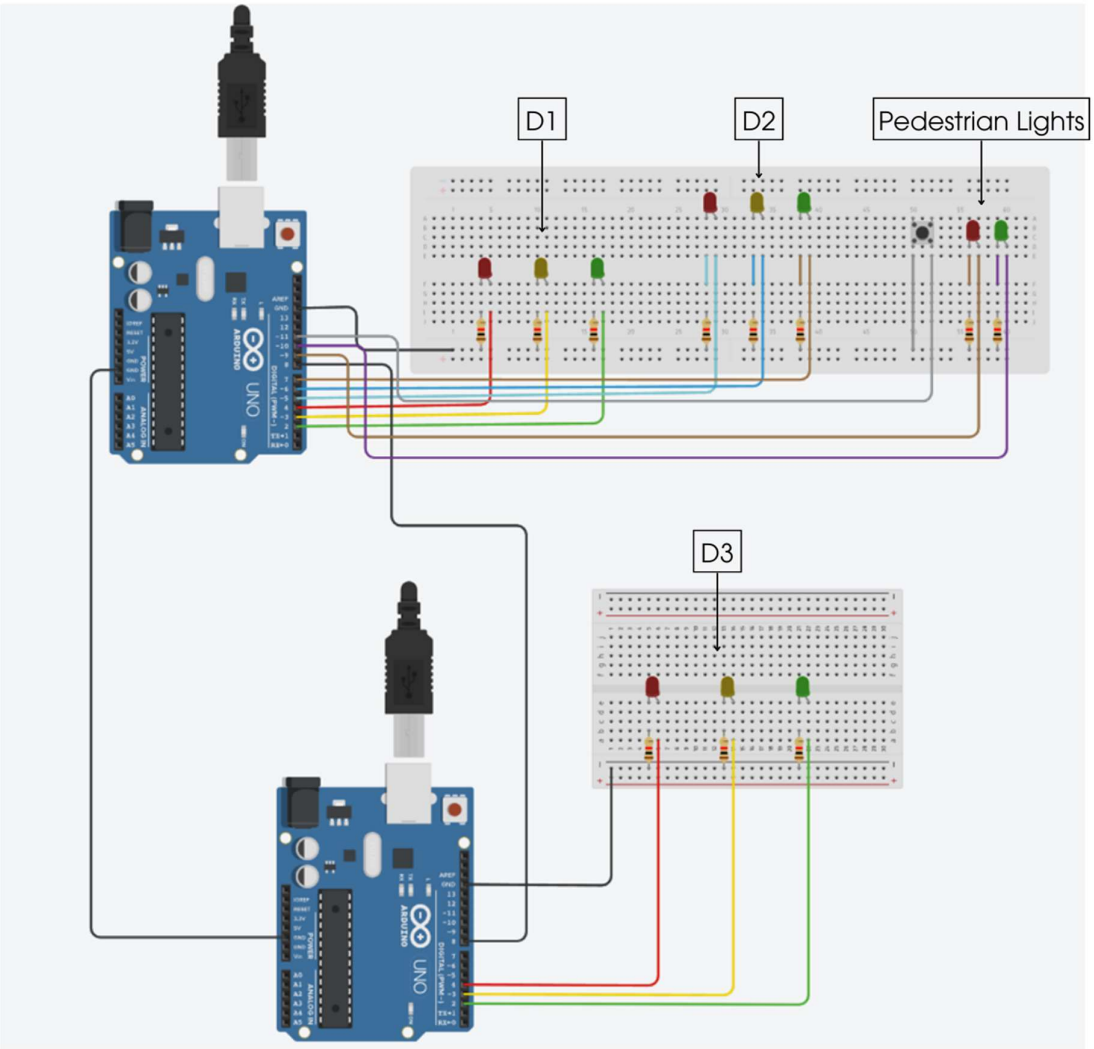
System Architecture:

- Master Controller (D2):
    - Controls main intersection lights (2 sets)
    - Manages pedestrian crossing signals with a push button
    - Processes pedestrian crossing requests
    - Coordinates timing with slave controller
- Slave Controller (D1):
    - Controls secondary traffic lights
    - Synchronizes with master controller
    - Maintains consistent timing patterns

Github Link:

https://github.com/MoeezMufti/Microcontroller--Moeez-Mufti/tree/main/Task6

TinkerCAD Simulation



Code

Master (Top Arduino):

```
//MASTER (Top Arduino)
const int d1Green = 2;
const int d1Yellow = 3;
const int d1Red = 4;
const int d2Red = 5;
const int d2Yellow = 6;
const int d2Green = 7;
const int pedRed = 9;
const int pedGreen = 10;
const int button = 11;         // Button input
const int buttonSignal = 8; // Output to D1

enum TrafficState { GREEN, YELLOW, RED };
TrafficState currentState = GREEN;  // Start with green
unsigned long stateTimer = 0;
bool buttonPressed = false;

void setup() {
  pinMode(d1Green, OUTPUT);
  pinMode(d1Yellow, OUTPUT);
  pinMode(d1Red, OUTPUT);
  pinMode(d2Green, OUTPUT);
  pinMode(d2Yellow, OUTPUT);
  pinMode(d2Red, OUTPUT);
  pinMode(pedRed, OUTPUT);
  pinMode(pedGreen, OUTPUT);
  pinMode(button, INPUT_PULLUP);
  pinMode(buttonSignal, OUTPUT);

  // Initial state: Both sets Green, Ped Red
  digitalWrite(d1Green, HIGH);
  digitalWrite(d2Green, HIGH);
  digitalWrite(d1Red, LOW);
  digitalWrite(d2Red, LOW);
  digitalWrite(pedRed, HIGH);
  digitalWrite(buttonSignal, LOW);
  stateTimer = millis();
}

void loop() {
  // Check button press during GREEN state
  if (currentState == GREEN && !buttonPressed && digitalRead(button) == LOW) {
    buttonPressed = true;
    digitalWrite(buttonSignal, HIGH);  // Signal D1
    delay(50); // Debounce
  }
```

```
  switch (currentState) {
    case GREEN:
      if (millis() - stateTimer >= (buttonPressed ? 5000 : 10000)) {
        // Transition to Yellow
        digitalWrite(d1Green, LOW);
        digitalWrite(d2Green, LOW);
        digitalWrite(d1Yellow, HIGH);
        digitalWrite(d2Yellow, HIGH);
        currentState = YELLOW;
        stateTimer = millis();
      }
      break;

    case YELLOW:
      if (millis() - stateTimer >= 2000) {
        // Transition to Red
        digitalWrite(d1Yellow, LOW);
        digitalWrite(d2Yellow, LOW);
        digitalWrite(d1Red, HIGH);
        digitalWrite(d2Red, HIGH);
        digitalWrite(pedGreen, HIGH);  // Pedestrian can now cross
        digitalWrite(pedRed, LOW);
        currentState = RED;
        stateTimer = millis();
      }
      break;

    case RED:
      if (millis() - stateTimer >= 10000) {
        // Back to Green
        digitalWrite(d1Red, LOW);
        digitalWrite(d2Red, LOW);
        digitalWrite(d1Green, HIGH);
        digitalWrite(d2Green, HIGH);
        digitalWrite(pedGreen, LOW);   // Stop pedestrian crossing
        digitalWrite(pedRed, HIGH);
        digitalWrite(buttonSignal, LOW);  // Reset signal to D1
        currentState = GREEN;
        buttonPressed = false;
        stateTimer = millis();
      }
      break;
  }
}
```

Slave (Bottom Arduino):

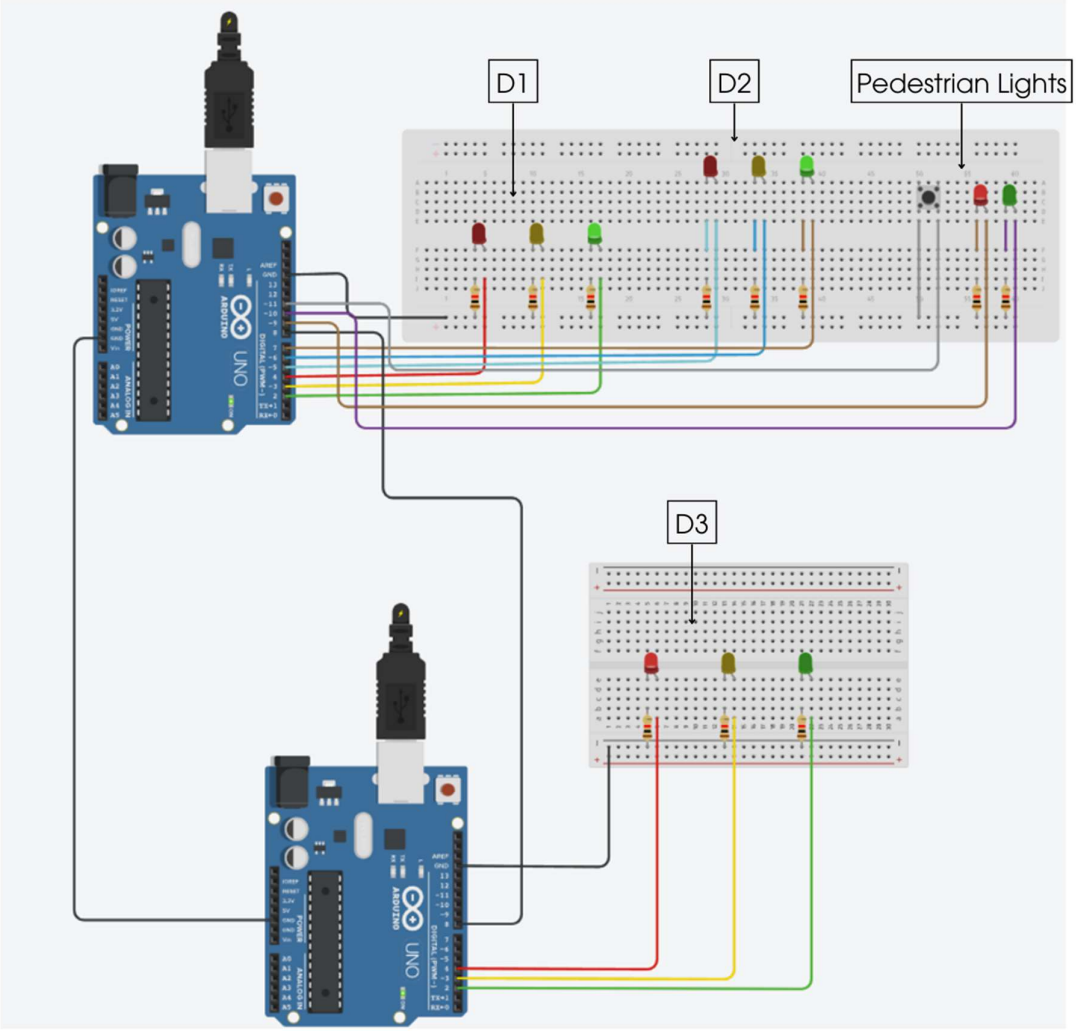
```
//SLAVE (Bottom Arduino)
const int d3Green = 2;
const int d3Yellow = 3;
const int d3Red = 4;
const int buttonSignal = 8;  // Input from D2

enum TrafficState { GREEN, YELLOW, RED };
TrafficState currentState = RED;  // Start with red
unsigned long stateTimer = 0;

void setup() {
  pinMode(d3Green, OUTPUT);
  pinMode(d3Yellow, OUTPUT);
  pinMode(d3Red, OUTPUT);
  pinMode(buttonSignal, INPUT);  // Input from D2

  // Initial state: Red
  digitalWrite(d3Red, HIGH);
  digitalWrite(d3Green, LOW);
  stateTimer = millis();
}

void loop() {
  // Read button signal from D2
  bool buttonActive = digitalRead(buttonSignal) == HIGH;

  switch (currentState) {
    case RED:
      if (millis() - stateTimer >= (buttonActive ? 5000 : 10000)) {
        digitalWrite(d3Red, LOW);
        digitalWrite(d3Yellow, HIGH);
        currentState = YELLOW;
        stateTimer = millis();
      }
      break;

    case YELLOW:
      if (millis() - stateTimer >= 2000) {
        digitalWrite(d3Yellow, LOW);
        digitalWrite(d3Green, HIGH);
        currentState = GREEN;
        stateTimer = millis();
      }
      break;

    case GREEN:
      if (millis() - stateTimer >= 10000) {
```

```
        digitalWrite(d3Green, LOW);
        digitalWrite(d3Red, HIGH);
        currentState = RED;
        stateTimer = millis();
      }
      break;
  }
}
```

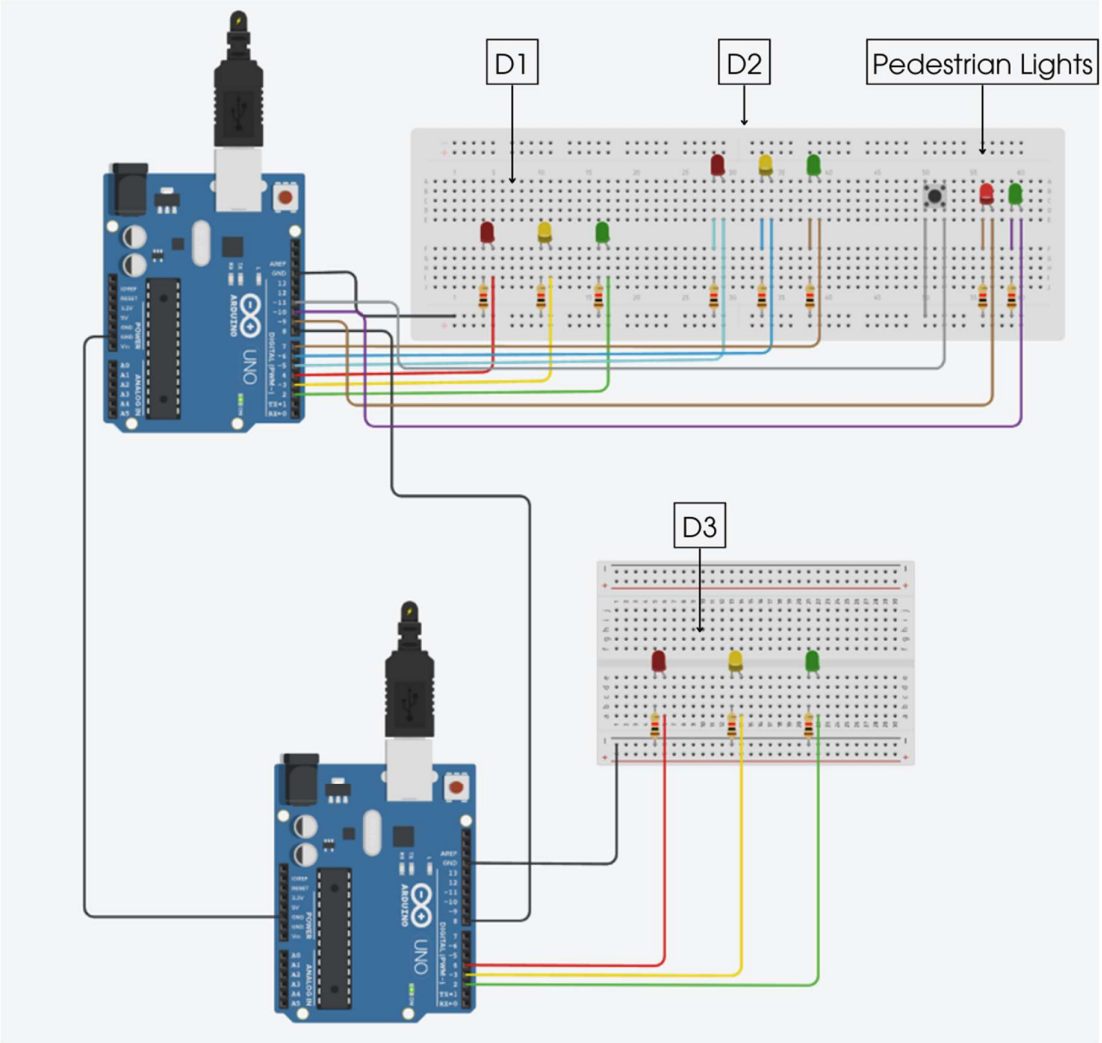
Scenarios

Scenario 1: When button is not pressed

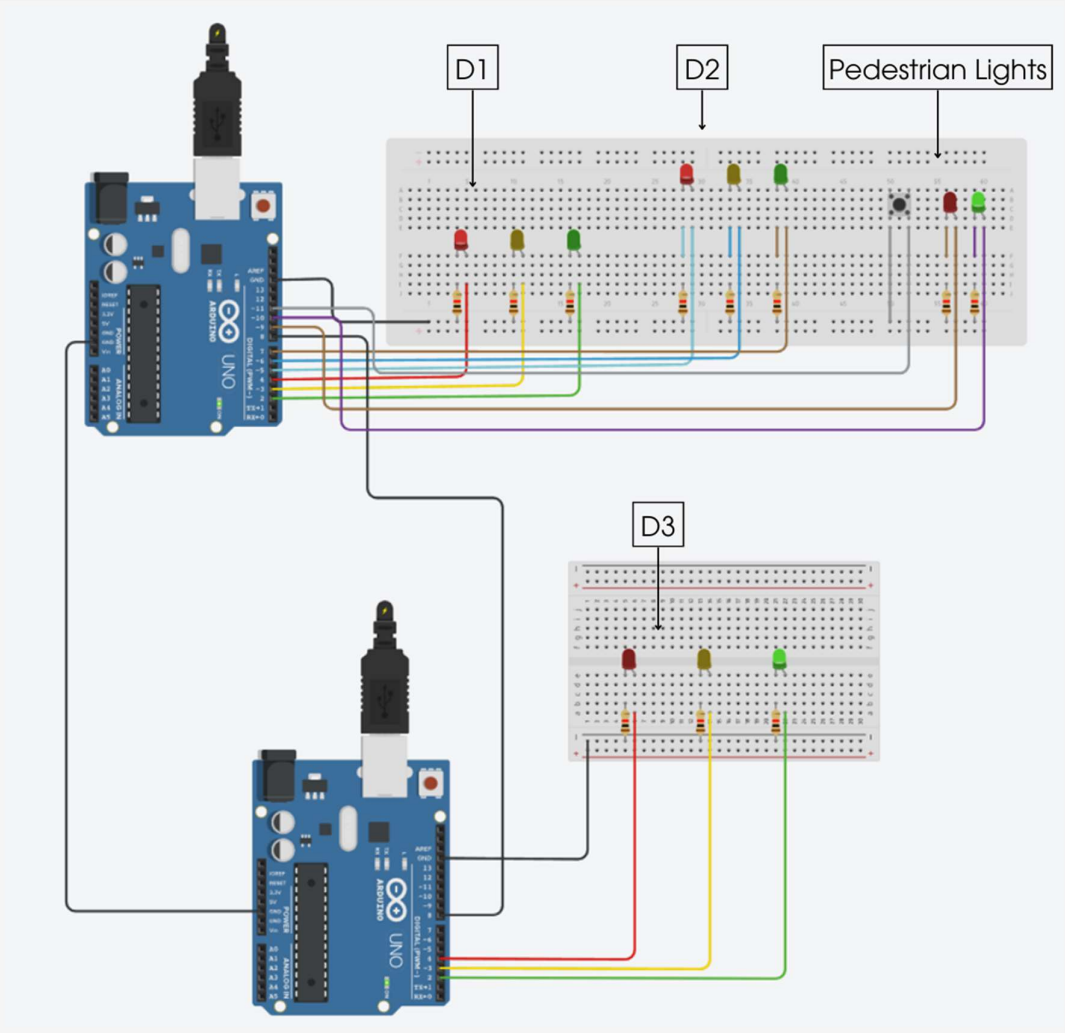
- D1 and D2 are in sync and are at green light state
- D3 and pedestrian lights are in red state



- After 10s, both D1 & D2 and D3 are at yellow.



- After 2s, D1 & D2 are at red and D3 and pedestrian are at green.
- The cycle continues



Scenario 2: When button is pressed:

- When the button is pressed initially, the timer changes from 10s to 4s.
- So, D1 & D2 changes within 4s from green to yellow. At the same time, D3 also changes while pedestrian remains at red.
- The cycle then continues as normal with D1 & D2 at red for 10s.

Safety Features:

- Synchronized yellow phases
- Fail-safe state transitions
- Clear pedestrian crossing indicators
- Debounced button input