Department of ECE

19ECE304 Microcontrollers and Interfacing Lab

Project Report- December 2024

Project Title: Traffic Light System Simulation Using LPC2138 Microcontroller

Team Members

AM.EN.U4ECE22114 – BATTULA PUJITH
AM.EN.U4ECE22117 – DHEERAJ M
AM.EN.U4ECE22144 – TEENA S
AM.EN.U4ECE22152 – MANASA MURALI
AM.EN.U4ECE22156 – KALYANI SHIBU

Signature of faculty with Date

Traffic Light System Simulation Using LPC2138 Microcontroller

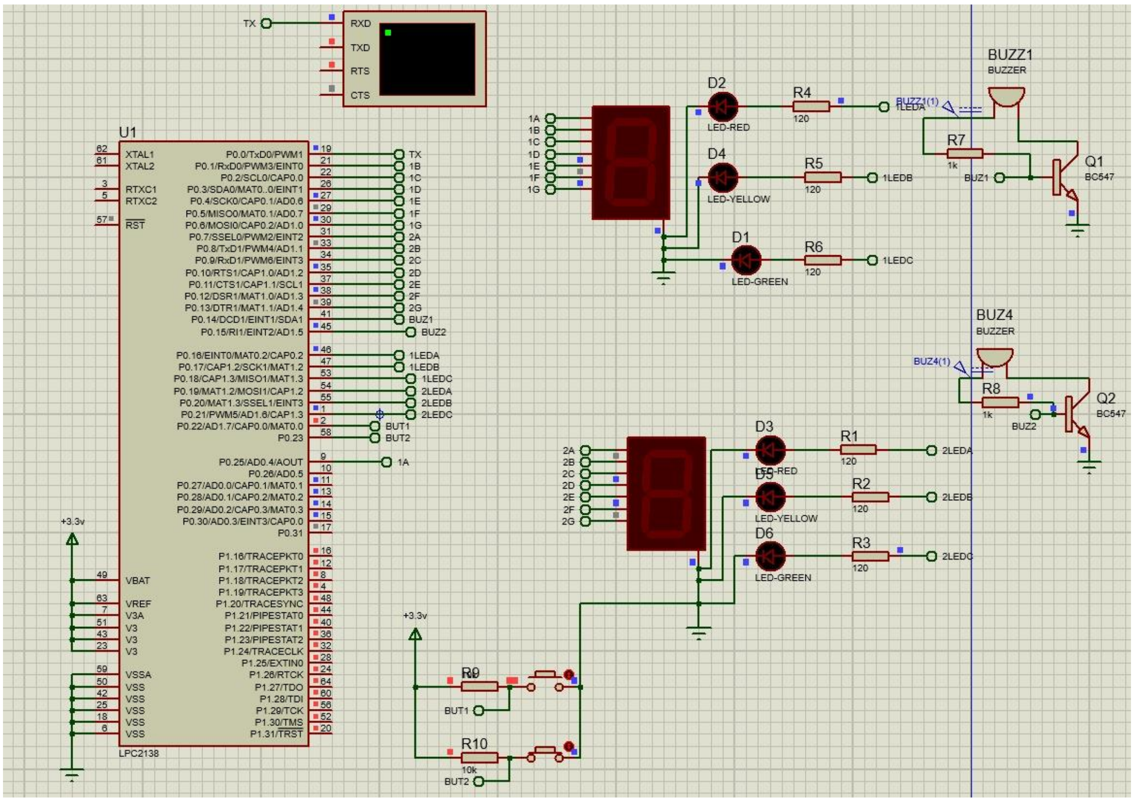
Objective

To design and simulate a traffic light control system using the LPC2138 microcontroller, LEDs, 7-segment displays, buzzers, and a serial monitor to enhance traffic management and emphasize safety.

Components Required

1. LPC2138 Microcontroller
2. LEDs (Red, Yellow, Green)
3. 7-Segment Displays
4. Resistors (120 Ω, 1 kΩ)
5. Transistors (BC547)
6. Buzzers
7. Push Buttons
8. Power Supply (+3.3 V)
9. Breadboard and Connecting Wires
10. Serial Monitor Software

Block Diagram/Circuit Diagram



Theory:

In this project, a traffic light system that controls traffic flow at crossings is simulated using the LPC2138 microcontroller. The microprocessor regulates each of the three phases that the system cycles through: red, yellow, and green. Traffic light signals are represented by LEDs, and the remaining time for each phase is shown via a countdown timer on 7-segment displays. In order to encourage road safety, the system additionally has a serial monitor that continuously shows "Drive Safe" and a buzzer for audio alerts.

The microcontroller manages the LEDs, displays, and buzzer using its GPIO pins, ensuring synchronized operation. The buzzer operates by a transistor, and the current flowing through the LEDs is limited by resistors. Clear textual, audio, and visual signals are provided for traffic control by the looping system. This study shows how microcontroller-based systems could be used effectively for safety awareness and real-time traffic light control.

Description of Peripherals Used:

LPC2138 Microcontroller:

- LEDs, 7-segment displays, and serial communication are all controlled by the LPC2138 microcontroller.
- LEDs and displays are driven via GPIO pins, while serial connectivity is enabled via UART.

LEDs:

- LEDs that are red, yellow, and green mimic traffic signals.

7-Segment Displays:

- To assist drivers in anticipating changes, show the countdown timer for each traffic phase.

Buzzer:

- Offers an audible warning when phases are changing.

Serial Monitor:

- Safety warning "Drive Safe" is displayed on the serial monitor.

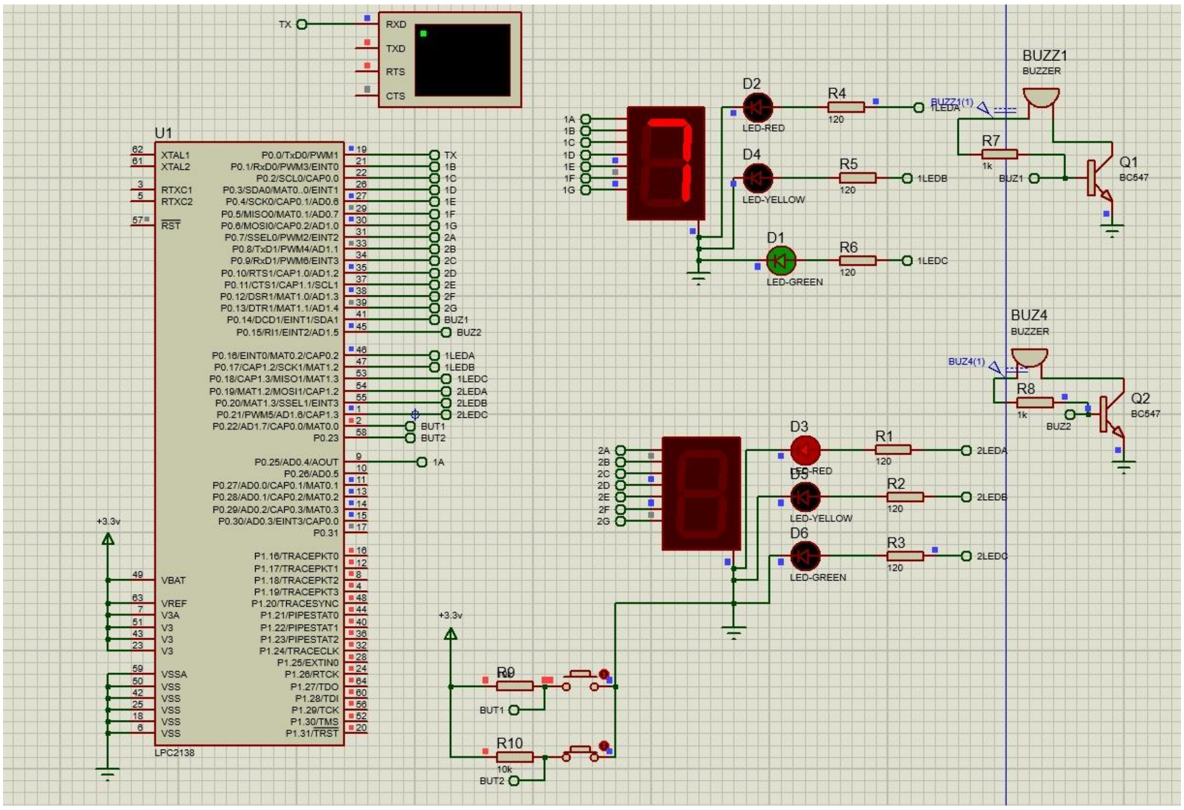
Project Description/ Working:

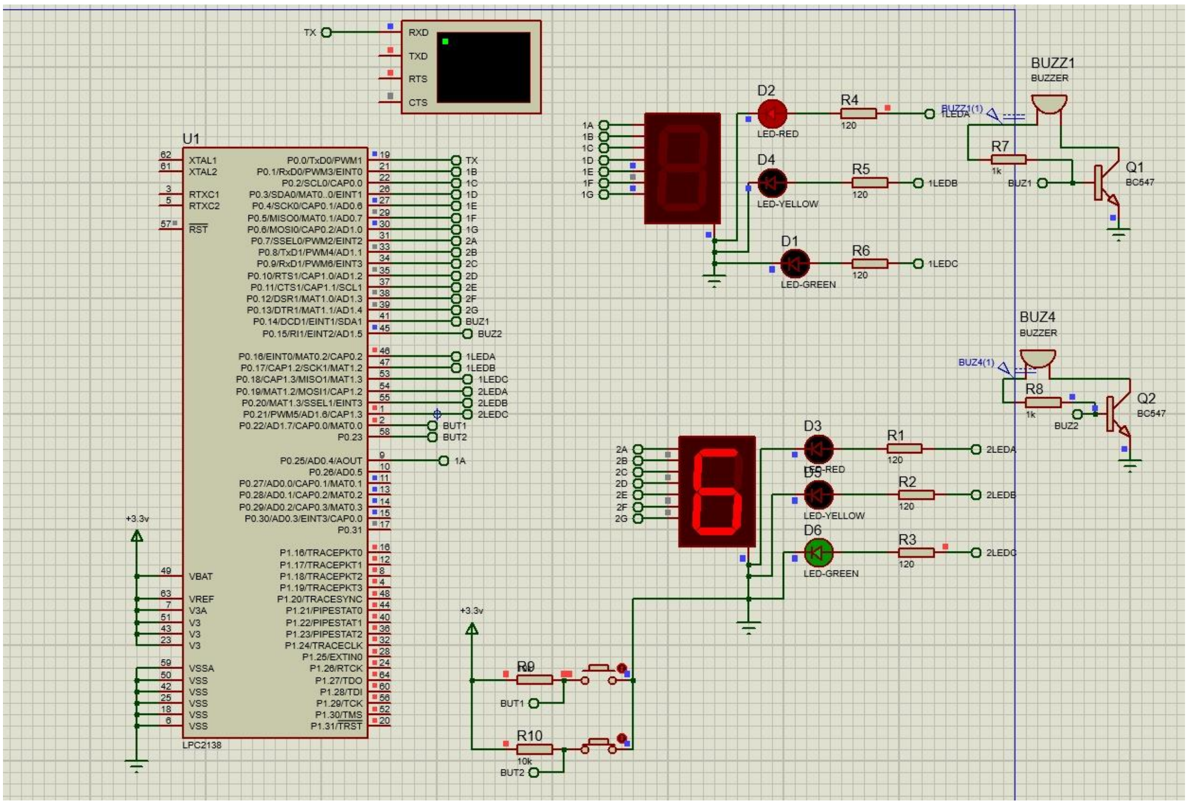
This microcontroller functions as the central processing unit, powering two multiplexed 7-segment countdown timer displays, a buzzer, and LEDs. The Red, Yellow, and Green phases of traffic lights are replicated in the system to resemble real-world conditions. In order to emphasize safety and promote responsible driving, it also incorporates a serial monitor that continuously shows the “Drive Safe”. There are three phases that the system goes through when it is operating. The red LED turns on during the Red-Light Phase, while the green and yellow LEDs stay off. The countdown timer indicates how long the stop signal will last, and drivers can choose to be alerted by the buzzer beeping. During the Yellow Light Phase, the timer is updated in accordance with the yellow LED turning on to indicate readiness.

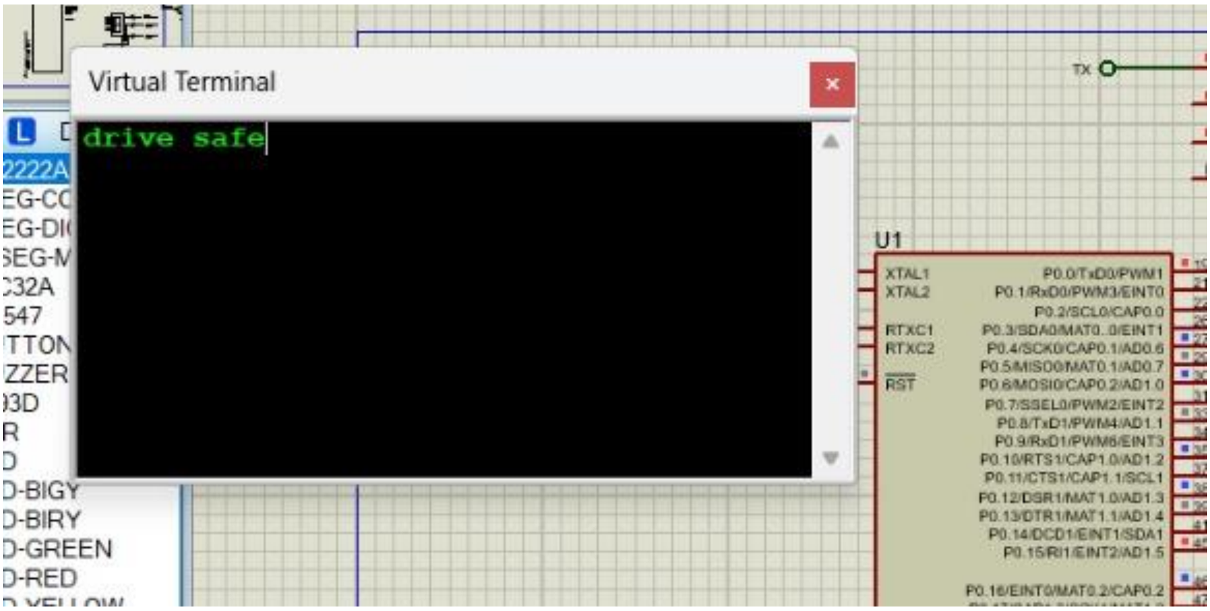
Finally, the green LED turns on to let cars pass during the Green Light Phase. The safety message "Drive Safe" is continuously displayed on the serial monitor, giving drivers constant reminders. The LPC2138 microcontroller's UART module is set up for serial connection, and it initializes GPIO pins for LED and display control. The multiplexed 7-segment displays are updated in real-time, and the countdown timer is controlled by delay functions. A transistor (BC547) powers the buzzer, while resistors control the amount of current flowing through the LEDs.

Result/Output:

A countdown timer is shown on seven-segment monitors, and corresponding LEDs cycle through the Red, Yellow, and Green phases of the traffic light system simulation. "Drive Safe" is continuously shown on the serial monitor to encourage safety awareness, and an buzzer is used to provide audio alerts during transitions.







Conclusion:

This project successfully demonstrates a microcontroller-based traffic light system with countdown timers and audio alerts. Utilizing the LPC2138 microcontroller, LEDs, buzzers, seven-segment displays, and a serial monitor, the system provides reliable and efficient traffic signal control. This configuration can be further developed or adapted for real-world applications, including intelligent traffic control systems, which can assist in reducing traffic jams and increase safety.

Reference

https://www.youtube.com/watch?v=I4Cwy-dhdFo

https://github.com/Satvik1712/Traffic_light_interfacing_using_LPC2148/blob/main/19EAC285_DA_EAC21075_EAC21076_EAC21085_2023.pdf

Acknowledgement:

We would like to express our sincere gratitude to mentor Mrs. Nisha KL and other lab staff for their invaluable guidance, support, and encouragement throughout the development of this project. Their expertise and insights were crucial in refining the concept and ensuring its successful implementation. We also extend our thanks to our peers for their constructive suggestions and collaborative efforts, which greatly enhanced the quality of the work. Finally, we are deeply grateful to our institution for providing the necessary resources and a conducive environment for research and innovation.

Appendix: Code with Comments

```
#include <lpc214x.h>

#include <stdio.h>


// Function to create a delay

void delay(unsigned int count) {

   unsigned int i = 0, j = 0;

   for (i = 0; i < 1000; i++) {

      for (j = 0; j < 1000; j++);

   }

}


// Function to transmit a character via UART0

void transmit(unsigned char ch) {

   while (!(U0LSR & 0x20)); // Wait until the transmitter is ready

   U0THR = ch;             // Load the character into the transmit register

}
```

```
int main(void) {
   int i;
   unsigned char temp[30] = "drive safe"; // Message to be transmitted via UART

   // UART0 Initialization
   PINSEL0 = 5;         // Select UART0 function for pins P0.0 (TXD) and P0.1 (RXD)
   U0LCR = 0x83;        // Enable DLAB (Divisor Latch Access Bit) and set 8-bit character
length
   U0DLM = 0x00;        // Set baud rate divisor (higher byte)
   U0DLL = 97;          // Set baud rate divisor (lower byte) for 9600 baud rate
   U0LCR = 0x03;        // Disable DLAB, retain 8-bit character length

   // Transmit the message character by character
   for (i = 0; temp[i] != '\0'; i++) {
      transmit(temp[i]);
   }

   // GPIO Initialization
   PINSEL0 = 0;         // Configure pins P0.0 to P0.15 as GPIO
   IODIR0 = 0x023FFFFF;  // Set P0.0 to P0.21 as output

   while (1) {
      if ((IOPIN0 & (1 << 22)) == 0) {
         IOSET0 |= (1 << 19);
         IOCLR0 |= (1 << 7) | (1 << 8) | (1 << 9) | (1 << 10) | (1 << 11) | (1 << 12) | (1 <<
13);

         // segment A //9
         IOSET0 |= (1 << 18);
```

```
IOCLR0 |= (1 << 16);
IOSET0 |= (1 << 25) | (1 << 1) | (1 << 2) | (1 << 3) | (1 << 5) | (1 << 6);
IOCLR0 |= (1 << 4);
delay(100);

// segment A //8
IOSET0 |= (1 << 25) | (1 << 1) | (1 << 2) | (1 << 3) | (1 << 4) | (1 << 5) | (1 << 6);
delay(100);

// segment A //7
IOSET0 |= (1 << 25) | (1 << 1) | (1 << 2);
IOCLR0 |= (1 << 3) | (1 << 4) | (1 << 5) | (1 << 6);
delay(100);

// segment A //6
IOSET0 |= (1 << 25) | (1 << 2) | (1 << 3) | (1 << 4) | (1 << 5) | (1 << 6);
IOCLR0 |= (1 << 1);
delay(100);

// segment A //5
IOSET0 |= (1 << 25) | (1 << 2) | (1 << 3) | (1 << 5) | (1 << 6);
IOCLR0 |= (1 << 1) | (1 << 4);
delay(100);
IOCLR0 |= (1 << 18);
delay(100);

// segment A //4
IOSET0 |= (1 << 18) | (1 << 1) | (1 << 2) | (1 << 5) | (1 << 6);
IOCLR0 |= (1 << 25) | (1 << 3) | (1 << 4);
```

```
delay(100);
IOCLR0 |= (1 << 18);
delay(100);

// segment A //3
IOSET0 |= (1 << 18) | (1 << 14) | (1 << 25) | (1 << 1) | (1 << 2) | (1 << 3) | (1 << 6);
IOCLR0 |= (1 << 4) | (1 << 5);
delay(100);
IOCLR0 |= (1 << 18) | (1 << 14);
delay(100);

// segment A //2
IOSET0 |= (1 << 18) | (1 << 14) | (1 << 25) | (1 << 1) | (1 << 3) | (1 << 4) | (1 << 6);
IOCLR0 |= (1 << 2) | (1 << 5);
delay(100);
IOCLR0 |= (1 << 18) | (1 << 14);
delay(100);

// segment A //1
IOSET0 |= (1 << 18) | (1 << 14) | (1 << 1) | (1 << 2);
IOCLR0 |= (1 << 25) | (1 << 3) | (1 << 4) | (1 << 5) | (1 << 6);
delay(100);
IOCLR0 |= (1 << 18) | (1 << 14);
delay(100);

// segment A //0
IOSET0 |= (1 << 18) | (1 << 14) | (1 << 25) | (1 << 1) | (1 << 2) | (1 << 3) | (1 << 4) | (1 << 5);
```

```
IOCLR0 |= (1 << 6);
delay(100);
IOCLR0 |= (1 << 18) | (1 << 14);
delay(100);

// YELLOW A
IOSET0 |= (1 << 17);
IOCLR0 |= (1 << 25) | (1 << 1) | (1 << 2) | (1 << 3) | (1 << 4) | (1 << 5) | (1 << 6);
delay(100);
IOCLR0 |= (1 << 17) | (1 << 19);

IOSET0 |= (1 << 16);
IOCLR0 |= (1 << 25) | (1 << 1) | (1 << 2) | (1 << 3) | (1 << 4) | (1 << 5) | (1 << 6);

// SEGMENT B //9
IOSET0 |= (1 << 21);
IOCLR0 |= (1 << 15);
IOSET0 |= (1 << 7) | (1 << 8) | (1 << 9) | (1 << 10) | (1 << 12) | (1 << 13);
IOCLR0 |= (1 << 11);
delay(100);

// segment B //8
IOSET0 |= (1 << 7) | (1 << 8) | (1 << 9) | (1 << 10) | (1 << 11) | (1 << 12) | (1 << 13);
delay(100);

// segment B //7
IOSET0 |= (1 << 7) | (1 << 8) | (1 << 9);
```

```
IOCLR0 |= (1 << 10) | (1 << 11) | (1 << 12) | (1 << 13);
delay(100);

// segment B //6
IOSET0 |= (1 << 7) | (1 << 9) | (1 << 10) | (1 << 11) | (1 << 12) | (1 << 13);
IOCLR0 |= (1 << 8);
delay(100);

// segment B //5
IOSET0 |= (1 << 7) | (1 << 9) | (1 << 10) | (1 << 12) | (1 << 13);
IOCLR0 |= (1 << 8) | (1 << 11);
delay(100);
IOCLR0 |= (1 << 21);
delay(100);

// segment B //4
IOSET0 |= (1 << 21) | (1 << 8) | (1 << 9) | (1 << 12) | (1 << 13);
IOCLR0 |= (1 << 7) | (1 << 10) | (1 << 11);
delay(100);
IOCLR0 |= (1 << 21);
delay(100);

// segment B //3
IOSET0 |= (1 << 21) | (1 << 15) | (1 << 7) | (1 << 8) | (1 << 9) | (1 << 10) | (1 << 
13);
IOCLR0 |= (1 << 11) | (1 << 12);
delay(100);
IOCLR0 |= (1 << 21) | (1 << 15);
delay(100);
```

```
// segment B //2
IOSET0 |= (1 << 21) | (1 << 15) | (1 << 7) | (1 << 8) | (1 << 10) | (1 << 11) | (1
<< 13);
IOCLR0 |= (1 << 9) | (1 << 12);
delay(100);
IOCLR0 |= (1 << 21) | (1 << 15);
delay(100);

// segment B //1
IOSET0 |= (1 << 21) | (1 << 15) | (1 << 8) | (1 << 9);
IOCLR0 |= (1 << 7) | (1 << 10) | (1 << 11) | (1 << 12) | (1 << 13);
delay(100);
IOCLR0 |= (1 << 21) | (1 << 15);
delay(100);

// segment B //0
IOSET0 |= (1 << 21) | (1 << 15) | (1 << 7) | (1 << 8) | (1 << 9) | (1 << 10) | (1 <<
11) | (1 << 12);
IOCLR0 |= (1 << 13);
delay(100);
IOCLR0 |= (1 << 21) | (1 << 15);
delay(100);

// YELLOW B
IOSET0 |= (1 << 20);
IOCLR0 |= (1 << 7) | (1 << 8) | (1 << 9) | (1 << 10) | (1 << 11) | (1 << 12) | (1 <<
13);
delay(100);
IOCLR0 |= (1 << 20) | (1 << 16);
```

```
        }
        else if ((IOPIN0 & (1 << 23)) == 0) {
            IOCLR0 = 0x023FFFFF;  // Clear all bits from 0 to 21a
        }
    }
        return 0;
}
```