AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Engineering

Lab Report

Experiment # 03

Experiment Title: Familiarization with the Timers of an Arduino Microcontroller Board, the study of LED blink test, and implementation of a simple traffic control system using its Timer() function.

Date of Perform:	17 March 2025	**Date of Submission:**	7 April 2025
Course Title:	Microprocessor and Embedded Systems Lab		
Course Code:	EE4103	**Section:**	P
Semester:	Spring 2024-25	**Degree Program:**	BSc in CSE
Course Teacher:	**Prof. Dr. Engr. Muhibul Haque Bhuyan**		

Declaration and Statement of Authorship:

1. I/we hold a copy of this Assignment/Case Study, which can be produced if the original is lost/damaged.
2. This Assignment/Case Study is my/our original work; no part has been copied from any other student's work or any other source except where due acknowledgment is made.
3. No part of this Assignment/Case Study has been written for me/us by any other person except where such collaboration has been authorized. by the concerned teacher and is acknowledged in the assignment.
4. I/we have not previously submitted or am submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared, and archived to detect plagiarism.
6. I/we permit a copy of my/our marked work to be retained by the Faculty Member for review by any internal/external examiners.
7. I/we understand that Plagiarism is the presentation of the work, idea, or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offense that may lead to expulsion from the University. Plagiarized material can be drawn from, and presented in, written, graphic, and visual forms, including electronic data, and oral presentations. Plagiarism occurs when the origin of the source is not appropriately cited.
8. I/we also understand that enabling plagiarism is the act of assisting or allowing another person to plagiarize or copy my/our work.

** Student(s) must complete all details except the faculty use part.*
** Please submit all assignments to your course teacher or the office of the concerned teacher.

Group # 01

Sl No	Name	ID	PROGRAM	SIGNATURE
1.	Md.Saikat Hossain	23-51242-1	BSc in CSE	
2.	Md.Mosharof Hossain Khan	23-51259-1	BSc in CSE	
3.	Rimal Banik	23-51260-1	BSc in CSE	
4.	Md.Rahidul Islam	23-51269-1	BSc in CSE	
5.	Rahat Ahmed	21-44911-2	BSc in CSE	

Faculty use only

FACULTY COMMENTS	**Marks Obtained**	
	Total Marks	

Contents

Objectives:

The objectives of this experiment are to-

a) Study the Timers of an Arduino Microcontroller Board.
b) Learn basic programming commands of the Timer functions.
c) Apply the coding techniques of the Timer functions.
d) Implement the LED blink test using Timer0 of an Arduino Microcontroller Board.
e) Implement a traffic light control system using an Arduino Microcontroller Board.

Apparatus:

1. Arduino IDE 2.3.5
2. Arduino Microcontroller board
3. LED lights (RED, GREEN, and YELLOW)
4. Three 100 ohms resistors, and
5. Jumper wires

Circuit Diagram:

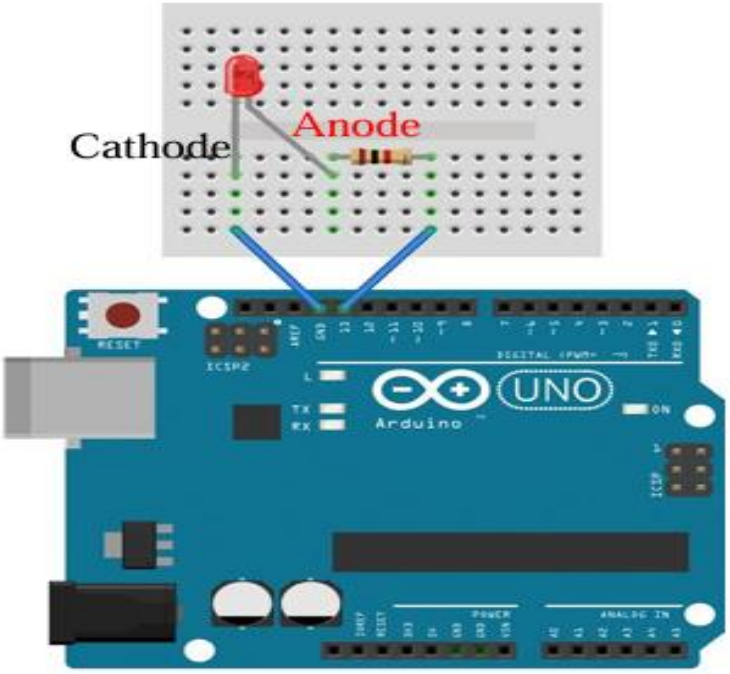

Figure 01: Experimental Setup of Blink Test using an Arduino Microcontroller Board.

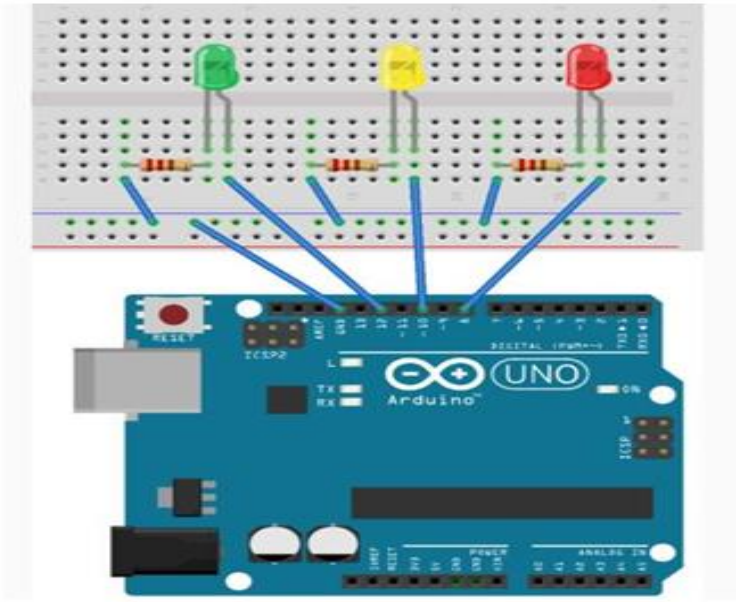

Figure 02: Experimental Setup of a Traffic Control System using an Arduino Microcontroller Board.

Code Explanation:

1.Program for LED blink test using Timer() function:

```
// Pin numbers are defined by the corresponding LED colors
#define RED_PIN 10
// The delay amounts are declared in ms; 1 s = 1000 ms
int red_on = 2000;
// Declaring the timer function to count 1 ms of delay instead of calling the delay function
int delay_timer (int milliseconds){
int count = 0;
while(1)
{
if(TCNT0 >= 16)  // Checking if 1 millisecond has passed
{
TCNT0=0;
       count++;
       if (count == milliseconds) //checking if required milliseconds delay has passed
       {
          count=0;
           break; // exits the loop
     }
     }
  }
  return 0;
}
```

```
void setup() {
   //define pins connected to LEDs as outputs
   pinMode(RED_PIN, OUTPUT);

   //set up the timer using timer registers
   TCCR0A = 0b00000000;
   TCCR0B = 0b00000101; //setting pre-scaler for timer clock
   TCNT0 = 0;
}

void loop() {
   //to make the red LED turned on
   digitalWrite(RED_PIN, HIGH);
   delay_timer(red_on);

   //turning off the red LED
   digitalWrite(RED_PIN, LOW);
   delay_timer(red_on);
   }
```

Explanation: The code defines a constant for the RED_PIN (pin 10).The delay_timer() function is a custom function created to implement a delay in milliseconds. The idea is to control how long the LED stays on or off without blocking the rest of the program (which the delay() function does). This function uses a hardware timer (TCNT0) that counts clock cycles in the Arduino board. When this timer reaches a value of 16, it means approximately 1 millisecond has passed. Every time the timer reaches 16, the counter (count) is incremented by 1. The function continues to loop until the count reaches the value of the milliseconds argument passed to the function (2000 milliseconds). Once that happens, the loop exits and the function ends. In the setup() function, the red LED pin (RED_PIN) is set as an output. This tells the Arduino that the pin will be used to send a signal (turn the LED on/off). The timer registers (TCCR0A and TCCR0B) are configured. These registers set up the timer to count at a specific rate (in this case, every 16 clock cycles, which is equivalent to 1 millisecond). In the loop() function, the red LED is turned on by setting the RED_PIN to HIGH. Then, the delay_timer(red_on) function is called to keep the red LED on for 2 seconds. The function waits for the specified delay (2000 milliseconds). After that, the red LED is turned off by setting the RED_PIN to LOW. The function then waits for another 2 seconds before repeating the process.

2. Program for Traffic Light Control System using Timer() function:

```
// Pin numbers are defined by the corresponding LED colors
#define GREEN_PIN 8
#define YELLOW_PIN 10
#define RED_PIN 12
//Define the delays for each traffic light color
int red_on = 6000; //3 s delay
int green_on = 3000;      //3 s delay
int yellow_blink = 500;   //0.5 s delay
```

```
// Declaring the timer function to count 1 ms of delay instead of calling the delay function
int delay_timer (int milliseconds){
  int count = 0;
  while(1)
  {
    if(TCNT0 >= 16)  // Checking if 1 millisecond has passed
      {
        TCNT0 = 0;
        count++;
        if (count == milliseconds) //checking if required milliseconds delay has passed
        {
           count = 0;
           break; // exits the loop
        }
      }
  }
  return 0;
}
 void setup() {
    //Define pins connected to LEDs as outputs
    pinMode(RED_PIN, OUTPUT);
    pinMode(YELLOW_PIN, OUTPUT);
    pinMode(GREEN_PIN, OUTPUT);
//Set up the Timer0
    TCCR0A = 0b00000000;
    TCCR0B = 0b00000101; //setting pre-scaler for timer clock
    TCNT0 = 0;
}
 void loop() {
    //to make the green LED turned on
    digitalWrite(GREEN_PIN, HIGH);
    delay_timer(green_on);
    //to make the green LED turned off
    digitalWrite(GREEN_PIN, LOW);
  //for turning the yellow LED on and off for 4 times
    for(int i = 0; i < 4; i = i+1)
    {
digitalWrite(YELLOW_PIN, HIGH);
delay_timer(yellow_blink);
digitalWrite(YELLOW_PIN, LOW);
    delay_timer(yellow_blink);
    }
      //to make the red LED turned on
    digitalWrite(RED_PIN, HIGH);
    delay_timer(red_on);
    //to make the red LED turned off
    digitalWrite(RED_PIN, LOW);
    }
```

Explanation: This program simulates a traffic light control system using an Arduino. It controls three LEDs (Green, Yellow, and Red) connected to pins 8, 10, and 12. The delay_timer() function is a custom function to handle delays in milliseconds without blocking other parts of the code. This is achieved using a hardware timer (TCNT0), which increments every 16 clock cycles. TCNT0 counts from 0 and increments by 1 every 16 clock cycles. The function checks if TCNT0 reaches 16, meaning 1 millisecond has passed. When the counter reaches the specified delay time(2000 milliseconds), the loop exits, and the function ends. In the setup() function the pinMode() function sets the LED pins (RED_PIN, YELLOW_PIN, GREEN_PIN) as OUTPUT, meaning they will control the LEDs by sending a HIGH or LOW signal. The timer is configured using the registers TCCR0B = 0b00000101 configures the timer to count every 16 clock cycles (which is about 1 millisecond).TCNT0 = 0 resets the timer counter at the start. In the loop() function the green LED(pin 8)is turned on by setting GREEN_PIN to HIGH, which sends 5V to the pin, lighting up the LED. The delay_timer(green_on) function is called, making the green LED stay on for the specified time (3000 milliseconds). After the delay, the green LED is turned off by setting GREEN_PIN to LOW.Then yellow LED(pin 10)blinks 4 times. A for loop runs 4 times, turning the yellow LED on and off, with each state lasting 500 milliseconds. This creates a blinking effect where the yellow LED stays on for 0.5 seconds, turns off for 0.5 seconds, and repeats this 4 times. Then red LED(pin 12)is turned on by setting RED_PIN to HIGH, turning the LED on. The delay_timer(red_on) function is called to keep the red LED on for 6000 milliseconds After the delay, the red LED is turned off by setting RED_PIN to LOW.

Hardware Implementation and Explanation:

1.**LED blink test:**

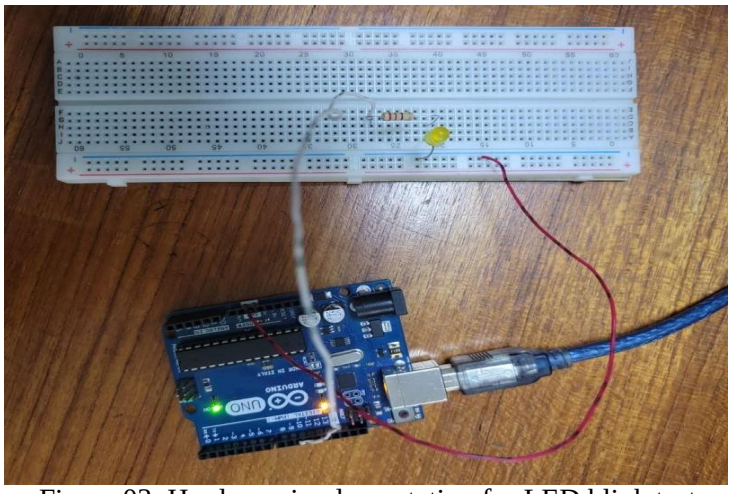

Figure 03: Hardware implementation for LED blink test

Explanation: In this implementation, a jumper wire was connected at the pin 10 of the Arduino Uno board. The wire was then connected on the breadboard. The cathode of an LED light was connected with the wire connected with pin 10. The anode of the LED light was connected with a 100 Ω resistor. The following resistor was then connected with the Ground (GND) of the Arduino Uno board. The Arduino Uno board was connected with a computer to compile and import required code.

2.**Traffic light control system:**

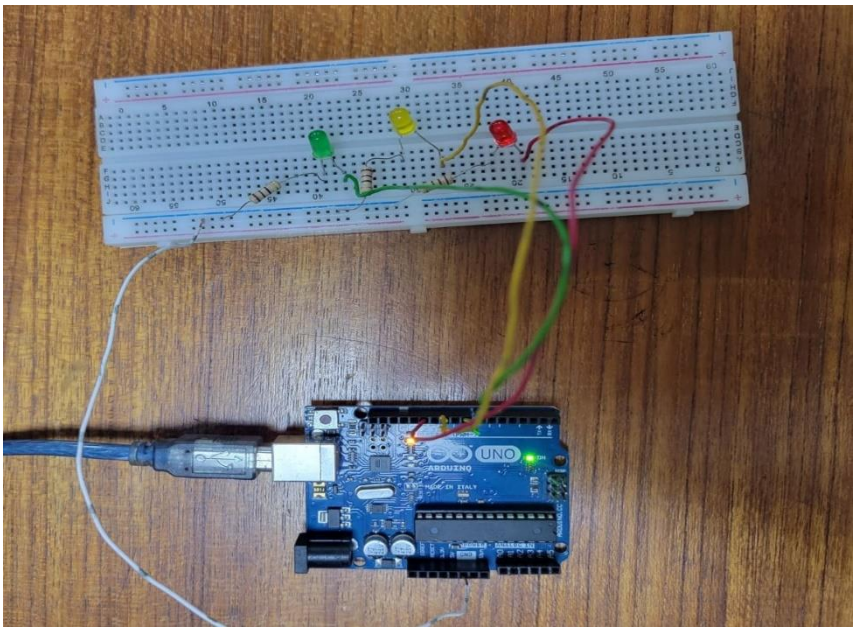

Figure 04: Hardware implementation for traffic light control system

Explanation: In this experiment, jumper wires were connected from pin 8, 10 and 12 of the Arduino Uno board to the anodes of the GREEN, YELLOW and RED LED sequentially. Three 100 Ω resistors were connected at each cathode of all the LED.The following resistor was then connected with the Ground of the Arduino board. The Arduino board was connected with a computer to compile and import required code.

Experimental Output Results:

1. **LED Blink Test:**

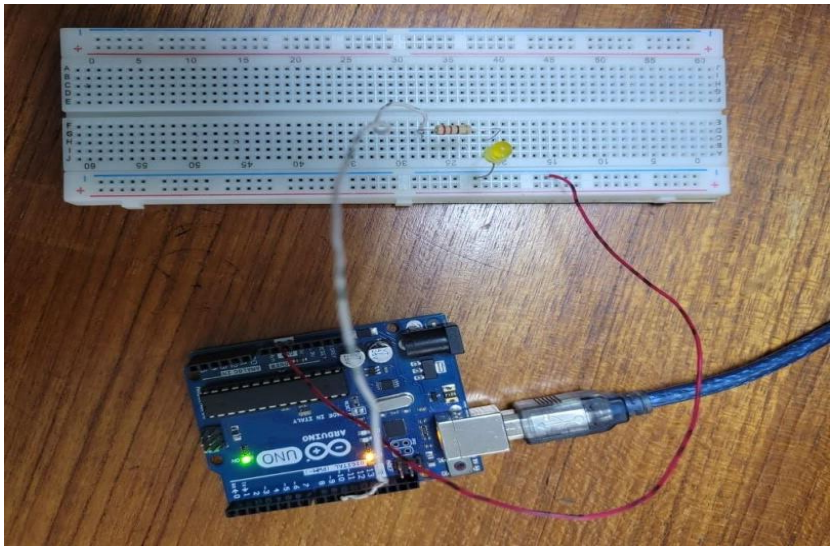

Figure 05: Light blink test (When LED is off)

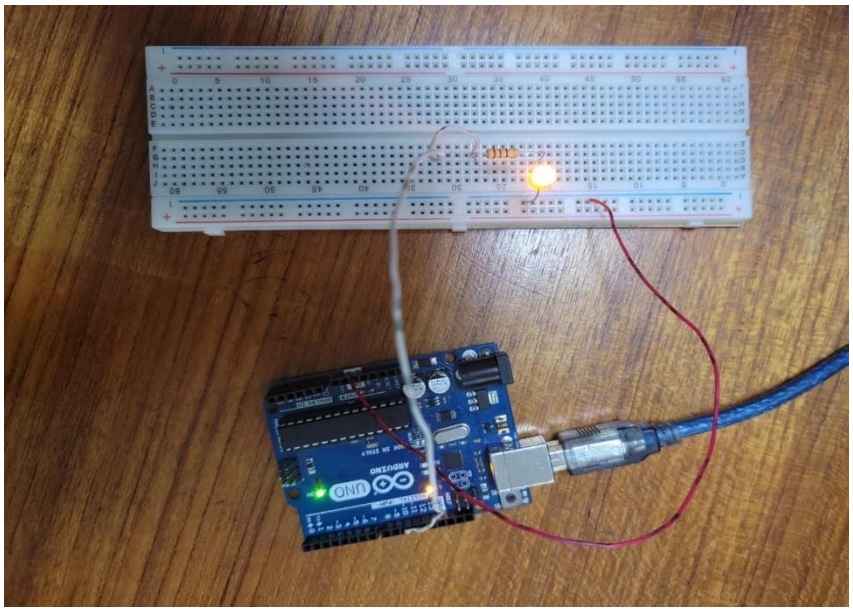

Figure 06: Light blink test (When LED is on)

Explanation: When the code runs, digitalWrite() sets the RED_PIN (pin 10) to HIGH, turning the red LED on. The delay_timer() function is then used to create a delay of 1000 milliseconds without blocking other code. It uses the hardware timer (TCNT0) to count time in 16-clock cycle increments. Once 1000 milliseconds have passed, digitalWrite() sets the RED_PIN to LOW, turning the LED off. The delay_timer() function waits again for another 1000 milliseconds. This cycle repeats, turning the LED on for 1000 ms, off for 1000 ms, until the microcontroller is powered off or reset.In the case of the red_on variable, the delay_timer() function would use the red_on value, which is set to 2000 milliseconds. This means that after the LED is turned on, it will stay on for 2 seconds (2000 milliseconds) before being turned off. The cycle would then repeat, with the red LED staying on for 2 seconds and off for 2 seconds, controlled by the delay_timer() function using the hardware timer.

2.Traffic Light Control System:

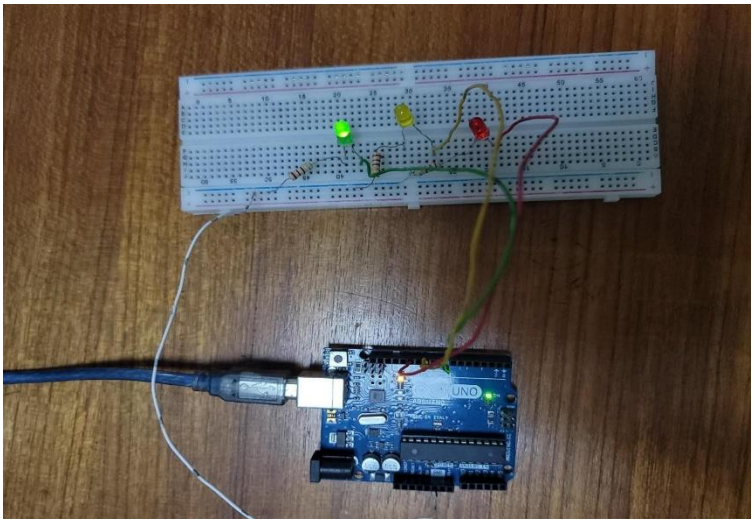

Figure 07: Traffic light control system(When GREEN LED is on)

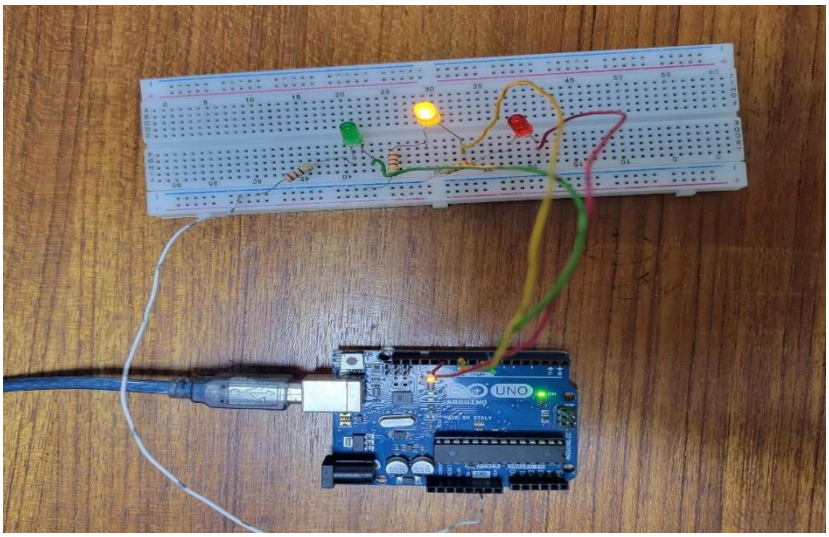

Figure 08: Traffic light control system(When YELLOW LED is on)

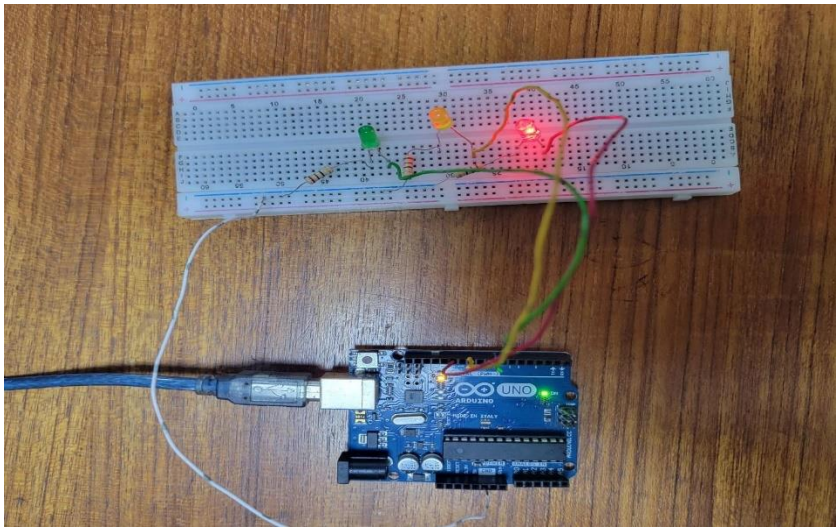

Figure 09: Traffic light control system(When RED LED is on)

Explanation: When the code is implemented and the loop() function starts, the code implements a digitalWrite() operation on the microcontroller and provides a high voltage at pin 8 as output. As a result, the green light turns ON. Then, the light stays on for 3000 milliseconds because the delay_timer() function is called with green_on (3000 ms). The delay_timer() function uses the hardware timer (TCNT0) to create the delay without blocking other code. Once the 3000 milliseconds have passed, the light is turned off by setting pin 8 to LOW.Next, a digitalWrite() operation on the microcontroller provides a high voltage at pin 10 as output. This turns the yellow light ON and OFF four times due to the for loop in the code. Each blink lasts 500 milliseconds due to the delay_timer() function being called with yellow_blink (500 ms). After four blinks, the yellow LED turns off.After the yellow LED blinks four times, a digitalWrite() operation on the microcontroller provides a high voltage at pin 12 as output. This turns the red light ON. Then, the light stays on for 6000 milliseconds due to the delay_timer() function being called with red_on (6000 ms). After the 6000 ms delay, the light turns off by setting pin 12 to LOW.All of these operations continue in a loop, allowing the traffic light sequence to repeat. The delay_timer() function ensures that each delay (green, yellow, red) occurs without blocking the program's other functions.

Simulation Output Results:

1.LED Blink Test:

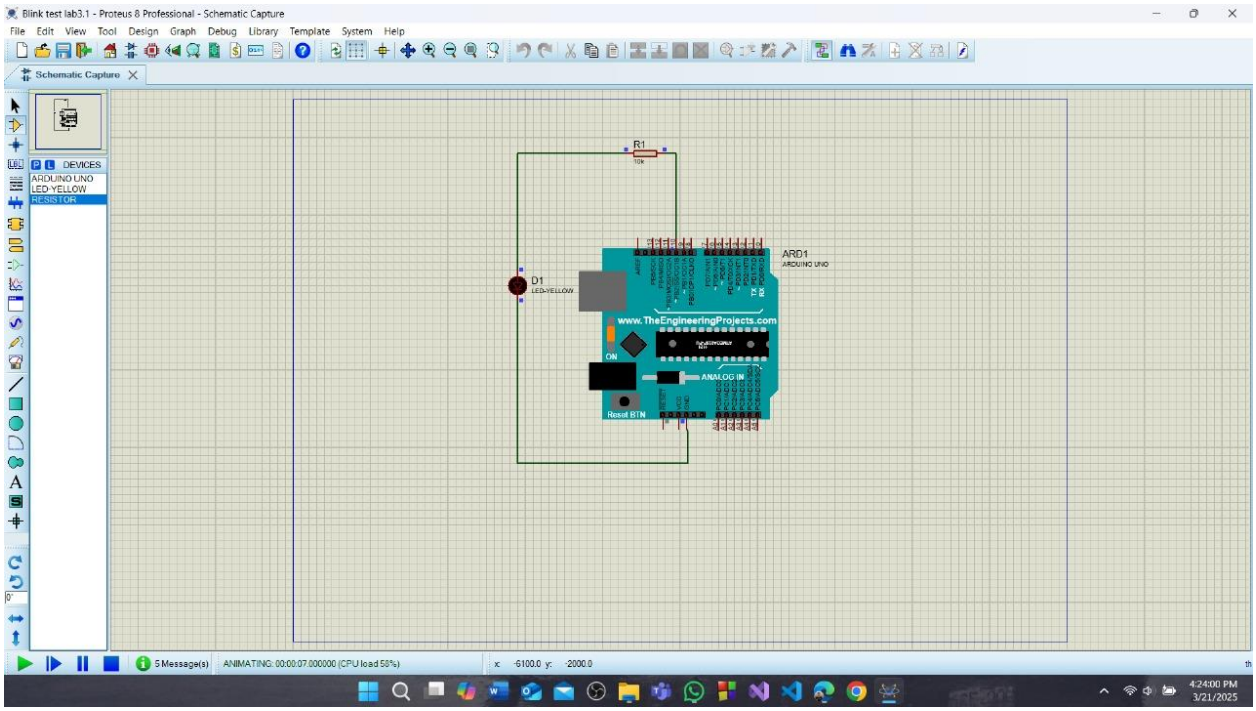

Figure 10: LED Blink Test on Simulation(When LED is off)

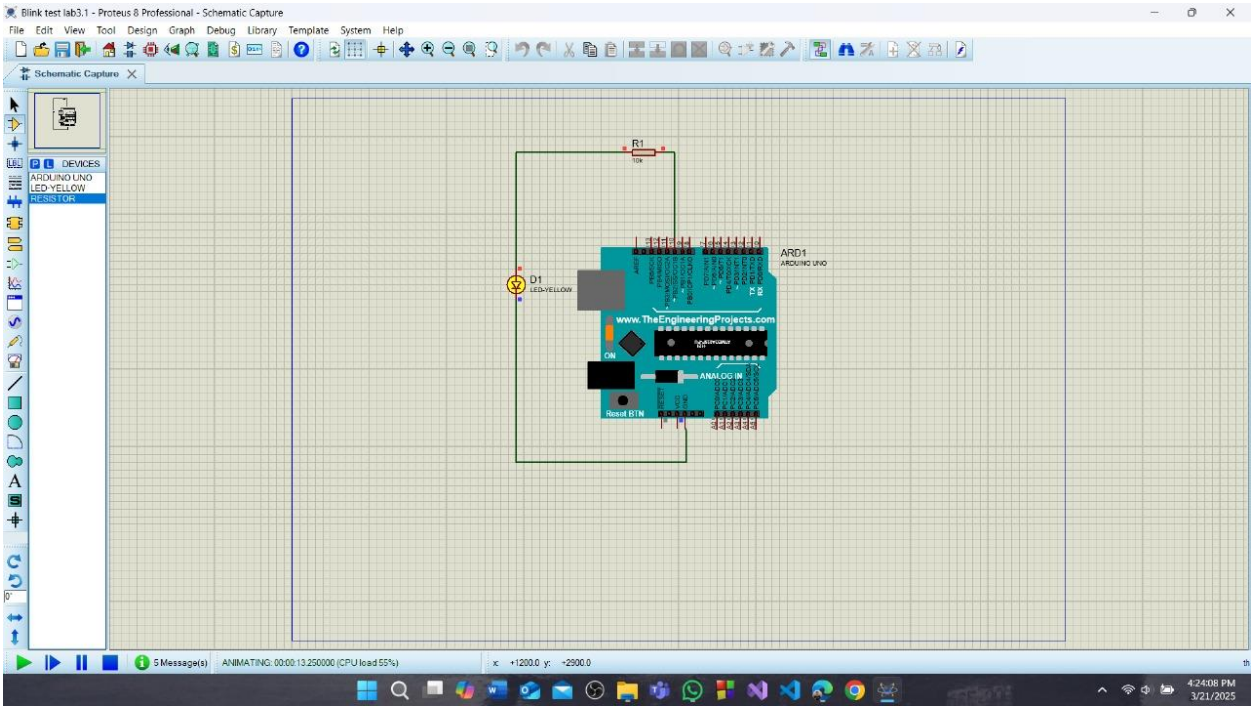

Figure 11: LED Blink Test on Simulation(When LED is on)

Explanation: In this simulation, the Arduino Uno board, 100Ω resistor, and LED were set up according to the hardware configuration used in the lab. The program was written and verified in the Arduino IDE 2.3.5, which generated a HEX file. This file was then loaded into the Proteus simulation to simulate the behavior of the circuit. After running the simulation, the results were observed and compared with the actual hardware results to check for consistency and verify the performance of the system.

2.Traffic Light System Control:

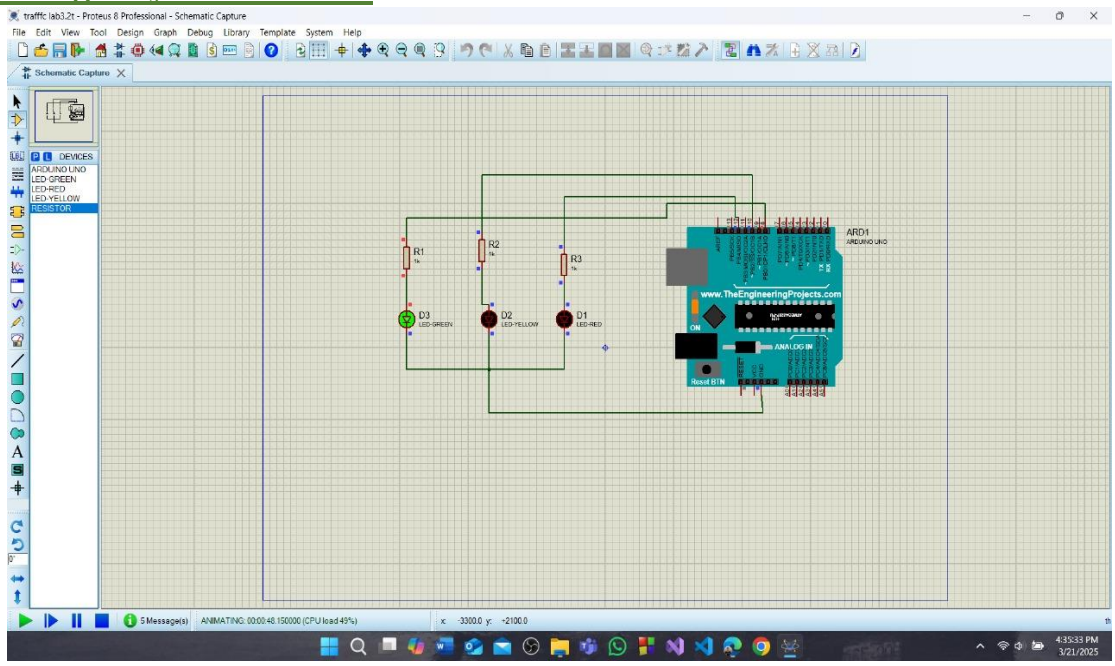

Figure 12: Traffic Light System Control on simulation(Green LED is on)

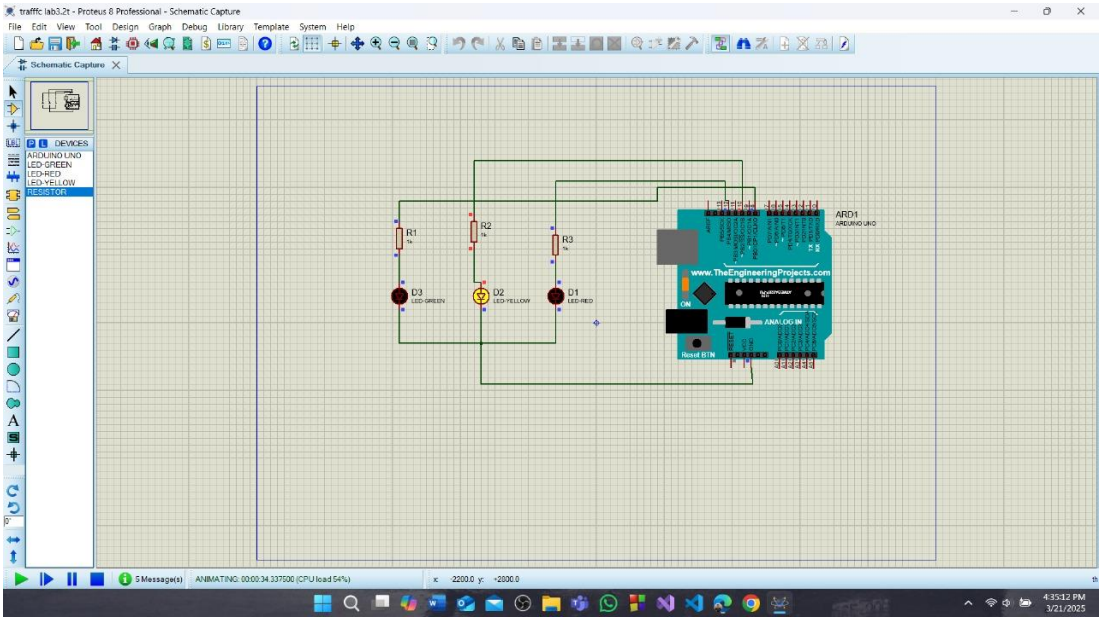

Figure 13: Traffic Light System Control on simulation(Yellow LED is on)

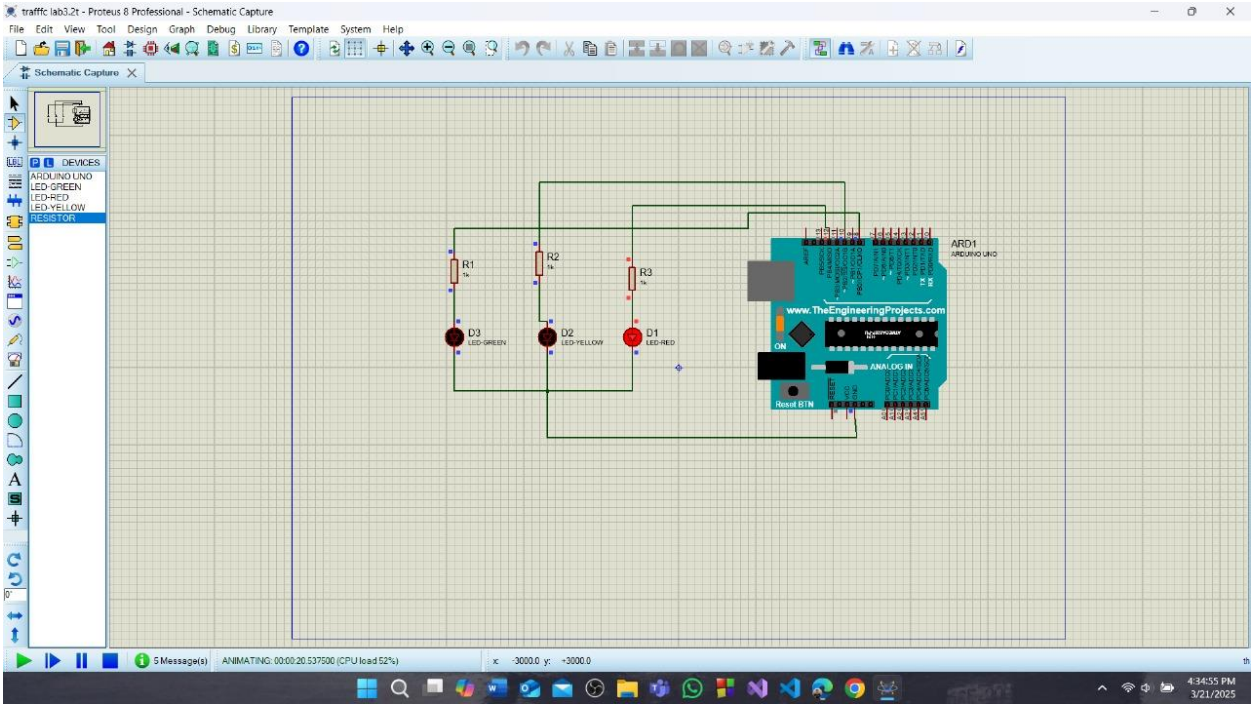

Figure 14: Traffic Light System Control on simulation(Red LED is on)

Explanation: In this simulation, the Arduino Uno board, three 100Ω resistors,Green, Yellow,Red LEDs were set up according to the hardware configuration used in the lab. The program was written and verified in the Arduino IDE 2.3.5, which generated a HEX file. This file was then loaded into the Proteus simulation to simulate the behavior of the circuit. After running the simulation, the results were observed and compared with the actual hardware results to check for consistency and verify the performance of the system.

Answer to Question:

3) My ID no- 23-51242-1, According to question A=2(2s delay for RED LED), B=4(4s delay for Yellow LED) and C=2(2s delay for Green LED)

Code:

```
// Pin numbers are defined by the corresponding LED colors
#define GREEN_PIN 8
#define YELLOW_PIN 10
#define RED_PIN 12

 //Define the delays for each traffic light color
int red_on = 2000; //2 s delay
int yellow_on = 4000;   //4 s delay
int green_on = 2000;      //2 s delay
```

```
// Declaring the timer function to count 1 ms of delay instead of calling the delay function
int delay_timer (int milliseconds){
  int count = 0;
  while(1)
  {
    if(TCNT0 >= 16)  // Checking if 1 millisecond has passed
      {
        TCNT0 = 0;
        count++;
        if (count == milliseconds) //checking if required milliseconds delay has passed
        {
            count = 0;
            break; // exits the loop
        }
      }
  }
  return 0;
}
 void setup() {
    //Define pins connected to LEDs as outputs
    pinMode(RED_PIN, OUTPUT);
    pinMode(YELLOW_PIN, OUTPUT);
    pinMode(GREEN_PIN, OUTPUT);
     //Set up the Timer0
    TCCR0A = 0b00000000;
    TCCR0B = 0b00000101; //setting pre-scaler for timer clock
    TCNT0 = 0;
}
 void loop() {
    //to make the green LED turned on
    digitalWrite(GREEN_PIN, HIGH);
    delay_timer(green_on);

    //to make the green LED turned off
    digitalWrite(GREEN_PIN, LOW);

    //for turning the yellow LED on
    digitalWrite(YELLOW_PIN, HIGH);
    delay_timer(yellow_on);

   //to make the yellow LED turned off
    digitalWrite(YELLOW_PIN, LOW);
    //to make the red LED turned on
    digitalWrite(RED_PIN, HIGH);
    delay_timer(red_on);
    //to make the red LED turned off
    digitalWrite(RED_PIN, LOW);
    }
```

Explanation: This program simulates a test using an Arduino. It controls three LEDs (Green, Yellow, and Red) connected to pins 8, 10, and 12. The delay_timer() function is a custom function to handle delays in milliseconds without blocking other parts of the code. This is achieved using a hardware timer (TCNT0), which increments every 16 clock cycles. TCNT0 counts from 0 and increments by 1 every 16 clock cycles. The function checks if TCNT0 reaches 16, meaning 1 millisecond has passed. When the counter reaches the specified delay time(2000 milliseconds), the loop exits, and the function ends. In the setup() function the pinMode() function sets the LED pins (RED_PIN, YELLOW_PIN, GREEN_PIN) as OUTPUT, meaning they will control the LEDs by sending a HIGH or LOW signal. The timer is configured using the registers TCCR0B = 0b00000101 configures the timer to count every 16 clock cycles (which is about 1 millisecond).TCNT0 = 0 resets the timer counter at the start. In the loop() function the green LED(pin 8)is turned on by setting GREEN_PIN to HIGH, which sends 5V to the pin, lighting up the LED. The delay_timer(green_on) function is called, making the green LED stay on for the specified time (2000 milliseconds). After the delay, the green LED is turned off by setting GREEN_PIN to LOW. Then creates a blinking effect where the yellow LED(pin 10) stays on for 4 seconds, turns off for 4 seconds. Then red LED(pin 12)is turned on by setting RED_PIN to HIGH, turning the LED on. The delay_timer(red_on) function is called to keep the red LED on for 2000 milliseconds After the delay, the red LED is turned off by setting RED_PIN to LOW.

Simulation Results:

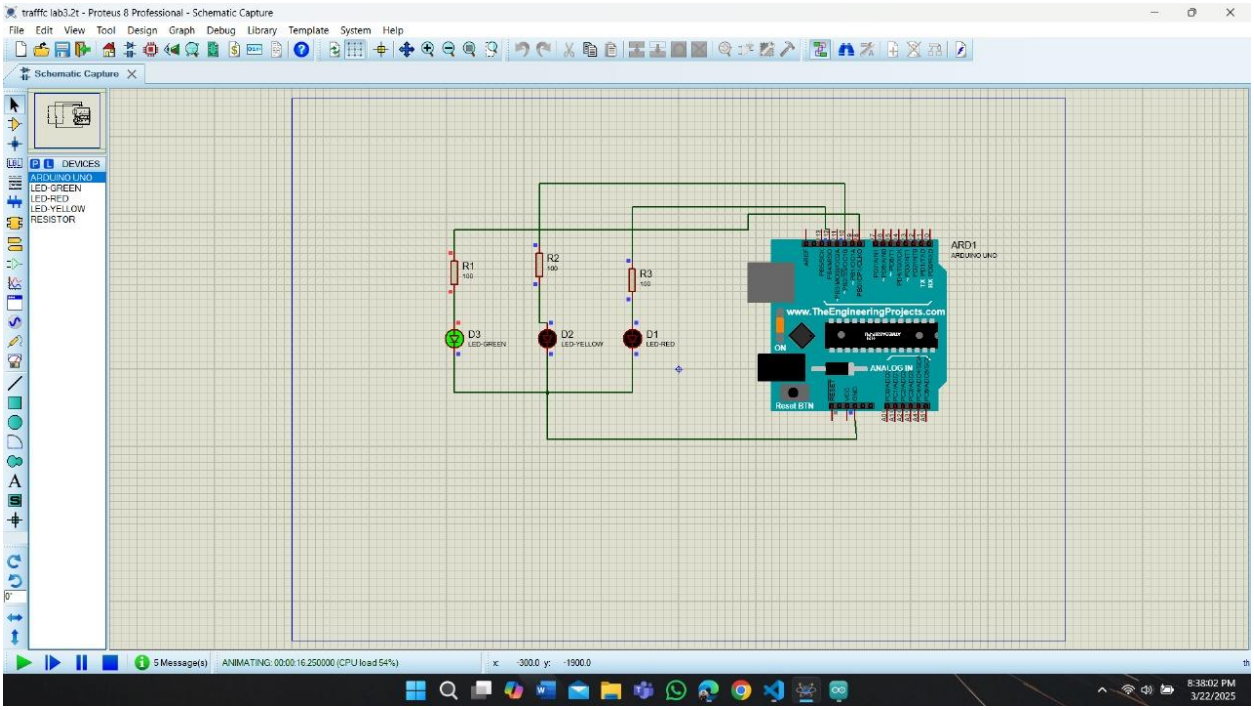

Figure 15: Proteus simulation(When Green LED is on)

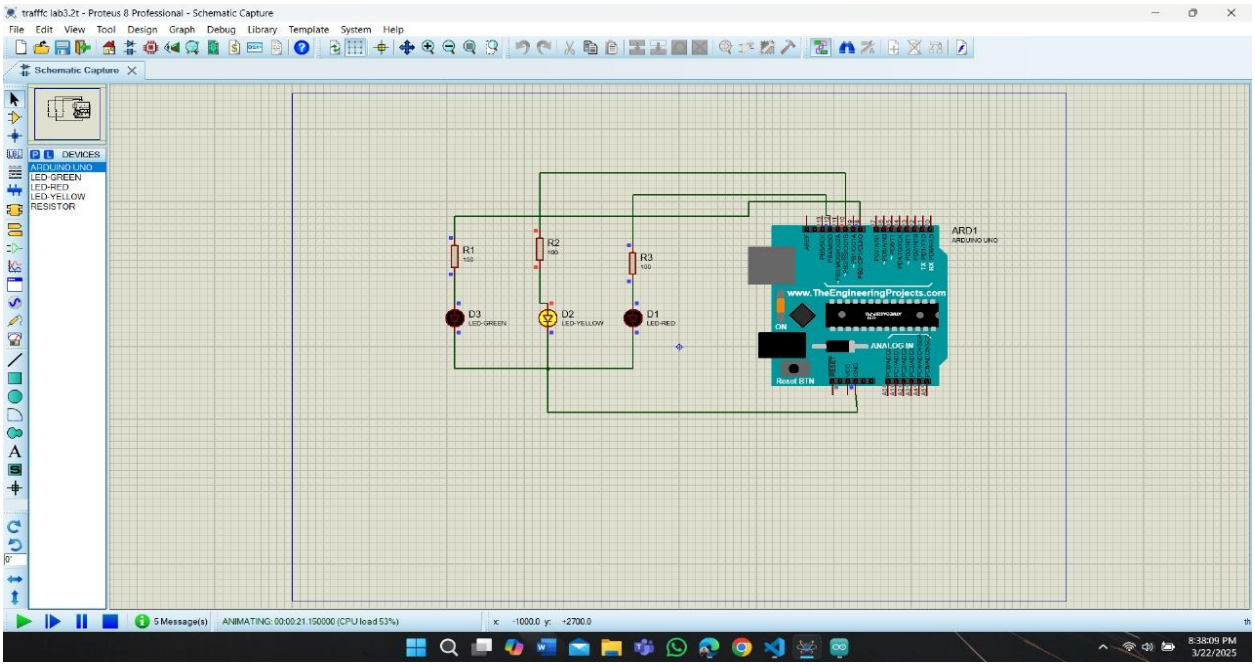

Figure 16: Proteus simulation(When Yellow LED is on)

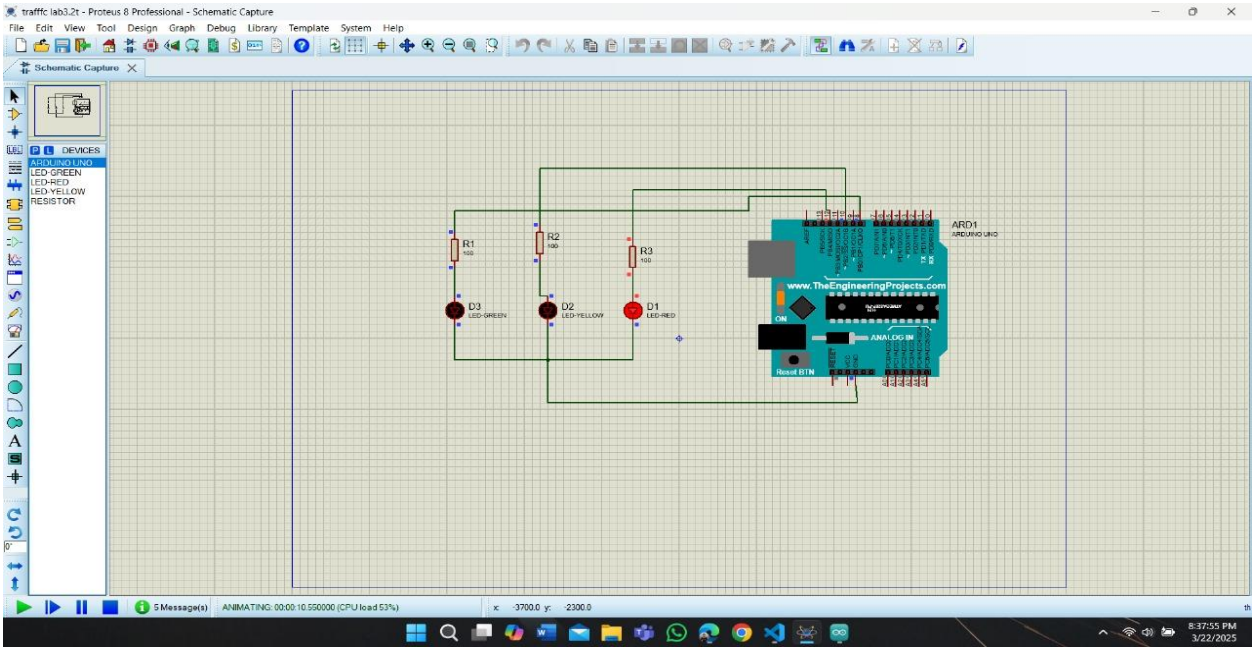

Figure 17: Proteus simulation(When Red LED is on)

Explanation: In this simulation, the Arduino Uno board, three 100Ω resistors,Green, Yellow,Red LEDs were set up according to the hardware configuration used in the lab. The program was written and verified in the Arduino IDE 2.3.5, which generated a HEX file. This file was then loaded into the Proteus simulation to simulate the behavior of the circuit. After running the simulation, the results were observed and compared with the actual hardware results to check for consistency and verify the performance of the system.

Discussion:

This experiment focused on gaining familiarity with the Timers of an Arduino Microcontroller Board by performing two practical implementations: an LED blink test and a simple traffic control system using the Timer0 function. The objective was to explore how timers work in generating precise delays and handling periodic events essential for real-time applications.

In the LED blink test, the Timer0 function was configured to generate consistent timing intervals for turning an LED ON and OFF. Unlike the conventional delay() function, which blocks the execution of other tasks, using Timer0 allowed the microcontroller to perform other operations concurrently. This technique is especially important for applications requiring multitasking, such as handling user inputs, monitoring sensors, or communication protocols while executing periodic tasks. The traffic control system was designed to simulate real-world traffic signal operations by using three LEDs representing green, yellow, and red lights. Timer0 was programmed to accurately control the duration of each LED state: green for 3 seconds, blinking yellow for 4 intervals of 0.5 seconds each, and red for 6 seconds. By utilizing Timer0, we achieved precise timing control, ensuring that the transitions between LEDs occurred as intended without interrupting other processes.

This experiment provided hands-on experience in configuring hardware timers, manipulating timer registers, and understanding how interrupt handling works to achieve efficient timing operations. By using interrupts and timers, we gained insights into non-blocking programming techniques, which are essential for developing real-time embedded systems. Furthermore, the experiment enhanced our skills in integrating hardware and software components, allowing us to design efficient, reliable, and responsive systems.

Here are some real life scenario and application of this experiment:

1. Pedestrian Crosswalk Signals:

Microcontroller timers manage LED signals for pedestrian crossings, providing safe walking intervals.

Example: Push-button systems at crosswalks activate LED signals that flash for a specific duration, allowing pedestrians to cross safely**.**

2. Emergency Vehicle Priority Systems:

Advanced traffic control systems use microcontroller timers to give priority to emergency vehicles.

Example: Traffic lights are programmed to turn green when emergency vehicles approach, using a timer function to reset normal operation after they pass**.**

3. Traffic Light Control Systems:

Microcontrollers with timers are widely used to control traffic signals, ensuring smooth traffic flow and improving road safety.

Example: In urban traffic systems, Arduino-based controllers use timers to alternate red, yellow, and green lights at intersections with precise timing intervals.

4. Railway Crossing Signals:

Microcontrollers control automatic railway crossing gates using timers to synchronize LED lights and barriers.

Example: When a train is detected, the microcontroller triggers red LED lights and lowers the gate for a predefined period before resetting the system.

5. School Zone Warning Systems:

Timers in microcontrollers are used to activate blinking warning lights during school hours.
Example: LED warning signs are turned on during specific times of the day when children are crossing the road, ensuring enhanced safety.

6. Automatic Traffic Counter Systems:

Microcontrollers with timers are used to measure traffic density and adjust signals accordingly.
Example: Timers manage how long traffic lights stay green based on the volume of vehicles passing through a specific area.

7.Smart Parking Systems:

Timers in microcontrollers are used to control LED indicators that show parking availability.
Example: Parking garages use microcontrollers to activate green LEDs for vacant spots and red LEDs for occupied spots, based on timed monitoring.

8. Highway Ramp Metering Systems:

Microcontroller timers are used in highway ramp metering systems to regulate the rate at which vehicles enter high-speed roadways, improving traffic flow and preventing congestion.
Example: Arduino-based systems control LED signals on highway entrance ramps, flashing green and red LEDs at calculated intervals to allow vehicles to merge safely and efficiently. The timing intervals are dynamically adjusted based on traffic density, improving overall road efficiency.

Conclusion:

In conclusion, this experiment deepened our understanding of microcontroller-based automation through the LED blink test and the implementation of a simple traffic control system using Timer0 functions. The LED blink test introduced fundamental digital output control, while the traffic control system demonstrated sequential execution, precise timing, and efficient real-world automation. By using Timer0 instead of delay functions, we achieved more responsive and efficient control over the LEDs. This experiment also enhanced our knowledge of hardware-software integration, loop execution, and embedded system design, which are essential for developing smart traffic management systems, industrial automation, and IoT-based applications. The skills gained from this experiment provide a solid foundation for future projects involving more complex and adaptive automation technologies.

References:

[1] **https://www.arduino.cc/.**
[2] **https://www.educba.com**
[3] **https://www.geeksforgeeks.org/led-blinking-using-arduino**