AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Engineering

Lab Report

Experiment # 01

Experiment Title: Familiarization with a microcontroller, the study of blink test and implementation of a traffic control system using microcontrollers.

Date of Perform:	Date Month 2023	**Date of Submission:**	12 June 2023
Course Title:	MICROPROCESSOR AND EMBEDDED SYSTEM LAB		
Course Code:	COE3104	**Section:**	N/A
Semester:	Summer 2022-23	**Degree Program:**	BSc in CSE/EEE
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 and no part of it has been copied from any other student's work or from 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 clearly acknowledged in the assignment.
4. I/we have not previously submitted or currently submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared, and archived for the purpose of detecting plagiarism.
6. I/we give permission for 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 #

Sl No	Name	ID	PROGRAM	SIGNATURE

Faculty use only		
FACULTY COMMENTS	**Marks Obtained**	
	Total Marks	

Table of Contents

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 Timer0 function.

Objectives:

The objective of familiarizing yourself with the timers of an Arduino microcontroller board and studying LED blink tests is to understand the timing capabilities of the microcontroller and how to use them effectively in your projects. By studying LED blink tests, you can learn how to control the timing and frequency of LED blinking patterns.

The objective of implementing a simple traffic control system using the Timer0 function of the Arduino board is to demonstrate how timers can be utilized for practical applications. The traffic control system example simulates a basic traffic signal with red, yellow, and green LEDs, and the Timer0 function is used to control the timing and sequencing of the LEDs to create the desired traffic signal behavior.

By achieving this objective, you will gain a deeper understanding of the timer functionality, timing concepts, and how to apply them to real-world projects. This knowledge can be extended to other applications that require precise timing and control, such as robotics, automation, and sensor-based systems.

Equipment List:
1) Arduino IDE (2.0.1 or any recent version)
2) Arduino Microcontroller board
3) LED lights (Red, Green, and Yellow)
4) Three 100 Ω resistors
5) Jumper wires

Circuit Diagram:
The Arduino platform is made up of the following components.

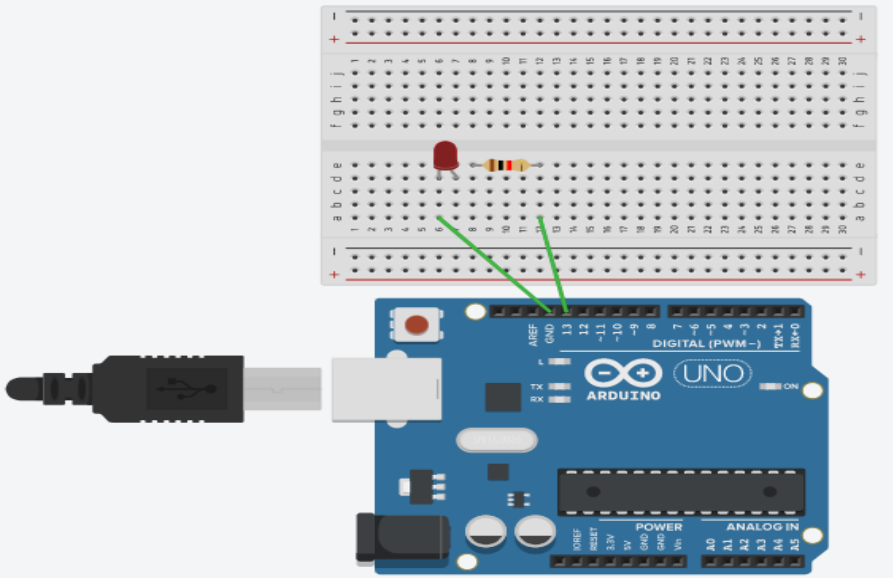

Fig. 1 Experimental Setup of Blink Test using an Arduino Microcontroller Board

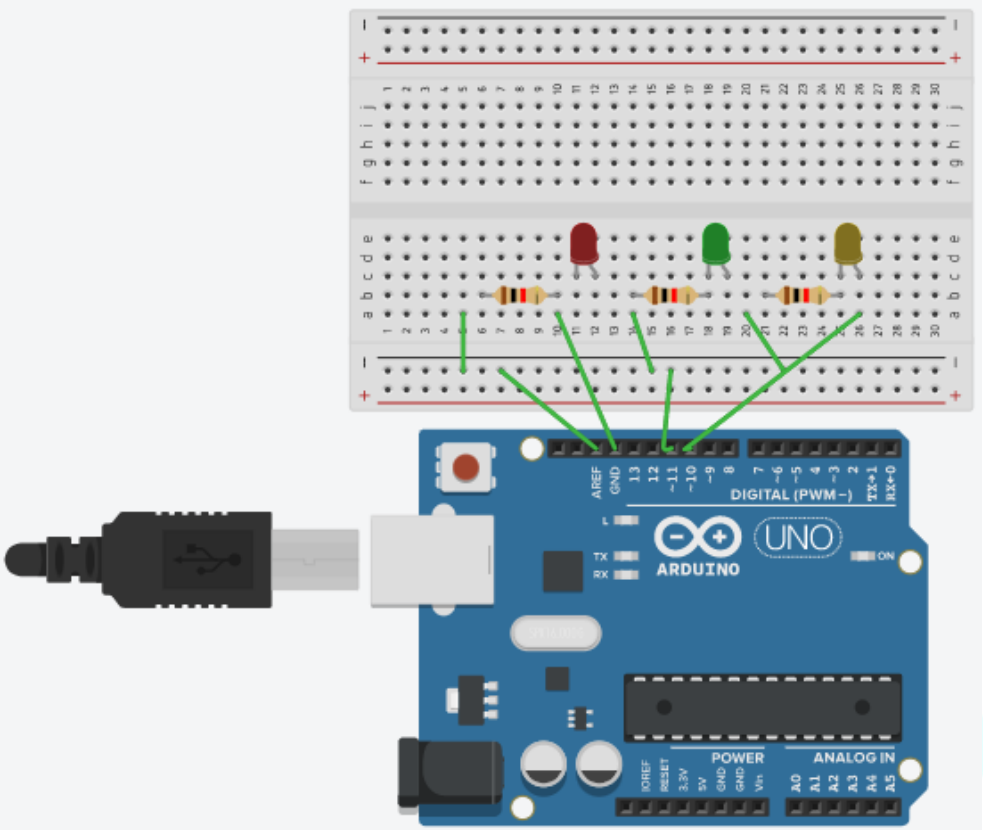

Fig. 2 Experimental Setup of a Traffic Control System using an Arduino Microcontroller Board

Code/Program:

The following is the code for the implementation of the LED blink test with the necessary code explanation:

```
// 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);
 }
```

The following is the code for the implementation of a simple traffic control system with the necessary code explanation:

```
// Pin numbers are defined by the corresponding LED colors
#define RED_PIN 8
#define YELLOW_PIN 10
#define GREEN_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);
 }
```


Hardware Output Results:

Here is the hardware implementation of the LED blink test and implement simple traffic control system test and the necessary explanation of the implementation:

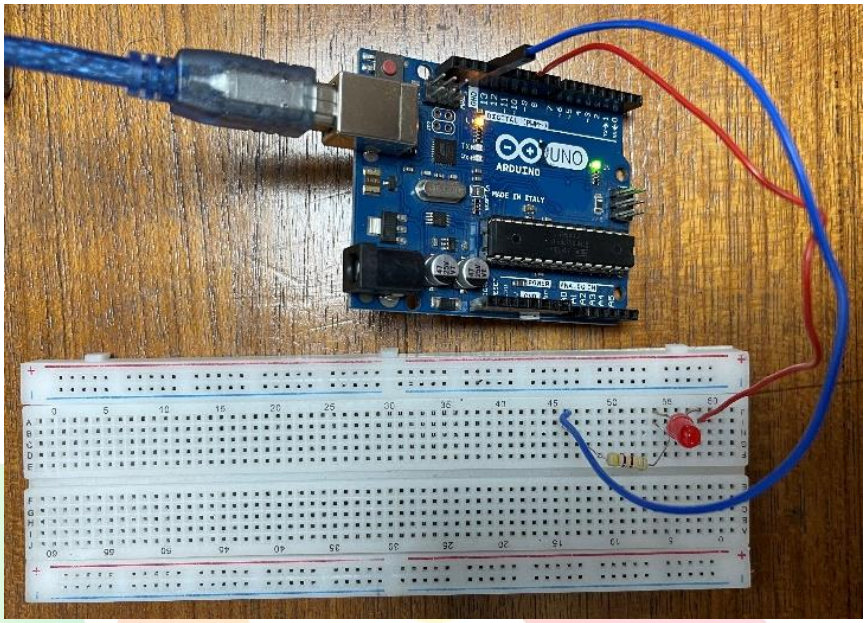

Fig.3 Hardware implementation for the blink test

Explanation: In the following implementation, a jumper wire was connected at the pin 8 of the Arduino Uno board. The wire was then connected on the breadboard. The anode of an LED light was connected with the wire connected with pin 8. The cathode 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 PC to compile and import necessary codes.

Here is the hardware implementation of Traffic Light System and the necessary explanation of the implementation:

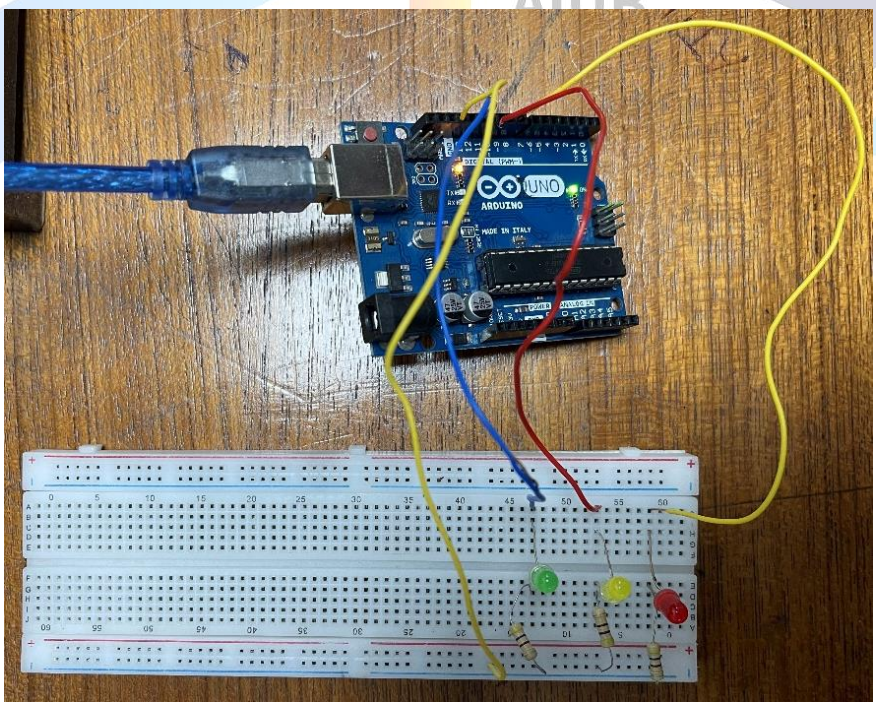

Fig.4 Hardware circuit diagram for the traffic light system

Explanation: In the following experiment, jumper wires were connected from pin 8, 10 and 12 of the Arduino Uno board to the anodes of the RED, YELLOW and GREEN LED consecutively. Three 100 Ω resistors were connected at each cathode of all the LEDs. Jumper wires were then connected at the negative common row of the breadboard. A jumper wire was connected then with the Ground (GND) of the Arduino Uno board from the common negative of the breadboard. The Arduino Uno board was connected with a PC to compile and import necessary codes.

Experimental Output Results:

Here are results of the light blink test and the necessary explanation of the results:

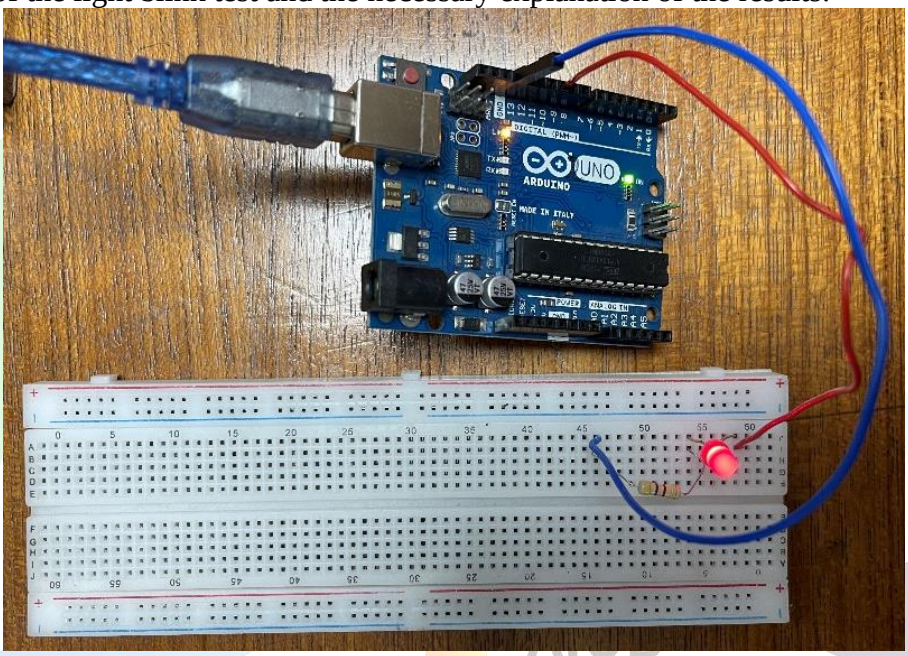

Fig.5 LED is ON in Light Blink Test

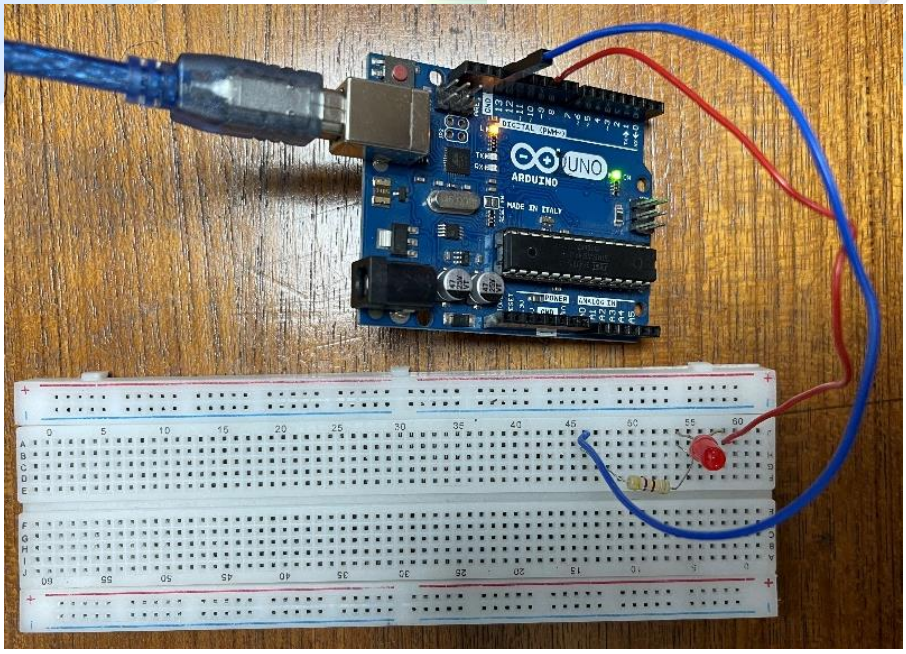

Fig.6 LED is OFF in Light Blink Test

Explanation: When the code is implemented and the loop () function starts, the code implements a digital write operation on the microcontroller and provides a high voltage at the pin 8 as output. Hence, the LED that was connected at pin 8 turns ON (Fig,5). A delay occurs afterwards for a certain time. This time was defined in milliseconds. According to the code that was implemented, a 2000 millisecond delay occurs after the light is turned ON. Then, another digital write operation occurs and the pin 8 was set to LOW as output. Therefore, the LED turns OFF (Fig. 6). Another 2000 milliseconds delay occurs after the LED was set to low. The delay operation was executed by using delay_timer () function in the loop () function. Thus, these results were happening one by one until the microcontroller was turned off or removed from the power source.

Here are results of the Traffic Light System and the necessary explanation of the results:

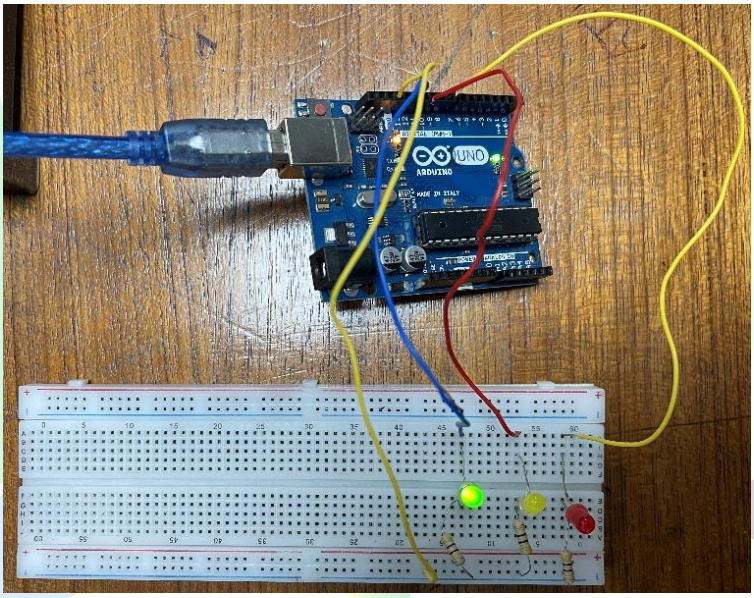

Fig.7 GREEN LED is ON in Traffic Light System

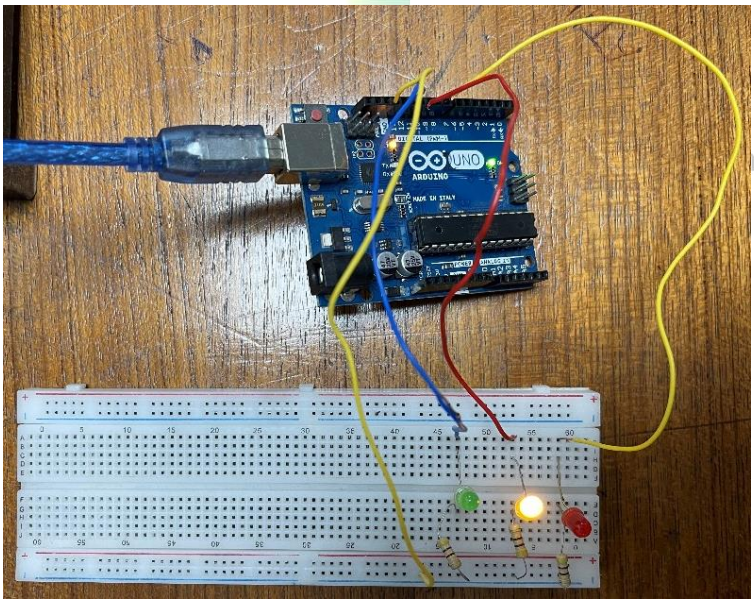

Fig.8 YELLOW LED is ON in Traffic Light System

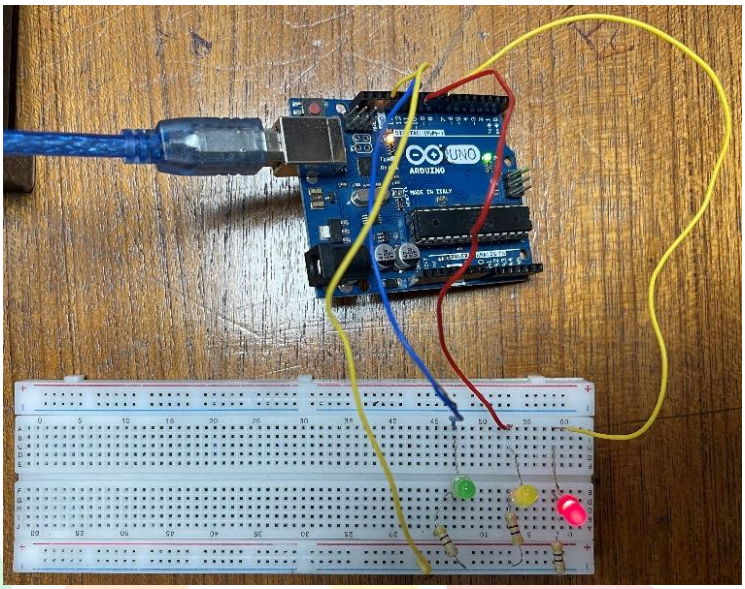

Fig.9 RED LED is ON in Traffic Light System

Explanation: When the code is implemented and the loop () function starts, the code implements a digital write operation on the microcontroller and provides a high voltage at the pin 12 as output. As a result, the green light turns ON (Fig.7). Then the light stayed on for 3000 milliseconds due to the delay function called 'green_on'. After that, the light turned off as a digital write was performed on the microcontroller and output at 12 was low. Then, a for loop was introduced in the code which made the microcontroller blink the yellow loght 4 times with an interval of 500 milliseconds (Fig.8) for the delay caused by the value set on 'yellow_blink'. When the yellow LED blinked for 4 times, then the red LED turns ON (Fig.9). It turned off after 6000 millisecond which was set in the 'red_on' variable and was used a delay function. All the operations kept occurring as all of the codes were performed in a loop.

Simulation Output Results:

Here are the simulation output results of Light Blink Test on Proteus simulation and explanation:

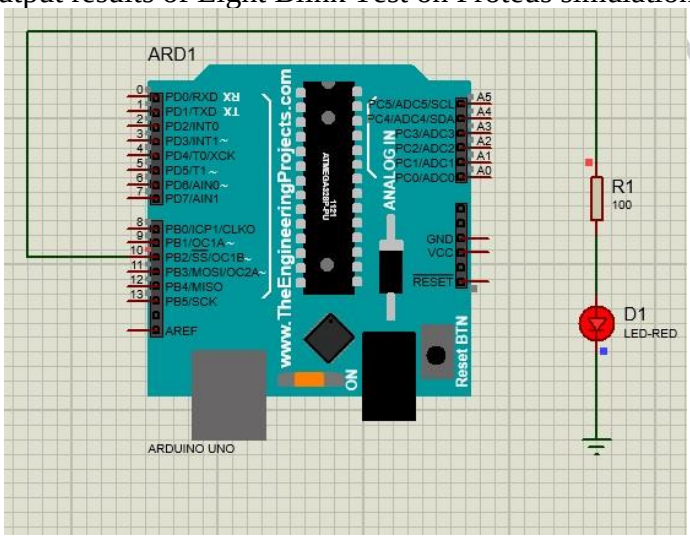

Fig.10 LED is ON in Light Blink Test on Proteus Simulation

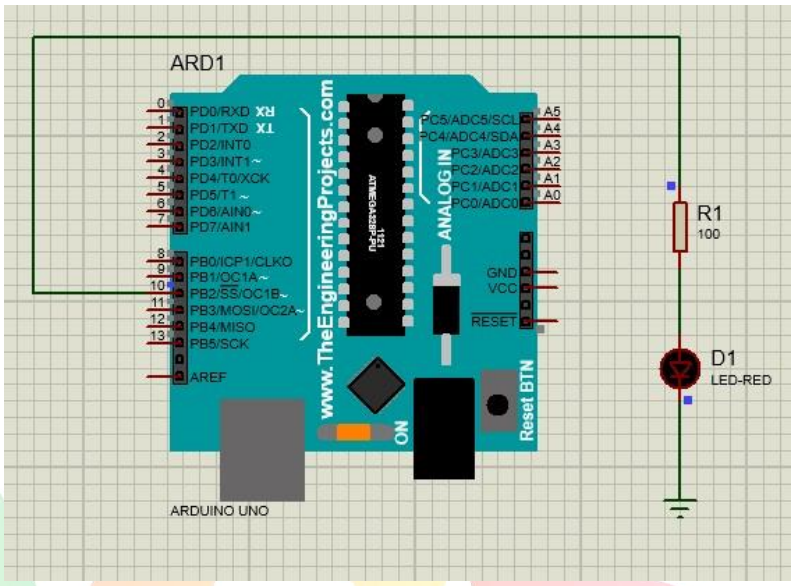

Fig.11 LED is OFF in Light Blink Test on Proteus Simulation

Explanation: In this simulation, an Arduino Uno Board, Resistor and LED was configured according to the hardware implementation that was performed. The program was made in the Arduino IDE was verified. As a result, a HEX file was generated. The following HEX file was implemented in the Proteus simulation for operation related instructions of the simulation. Afterwards, the simulation was performed and the results that were obtained were noted and compared with the hardware results.

Here are the simulation output results of Traffic Light System on Proteus simulation and explanation:

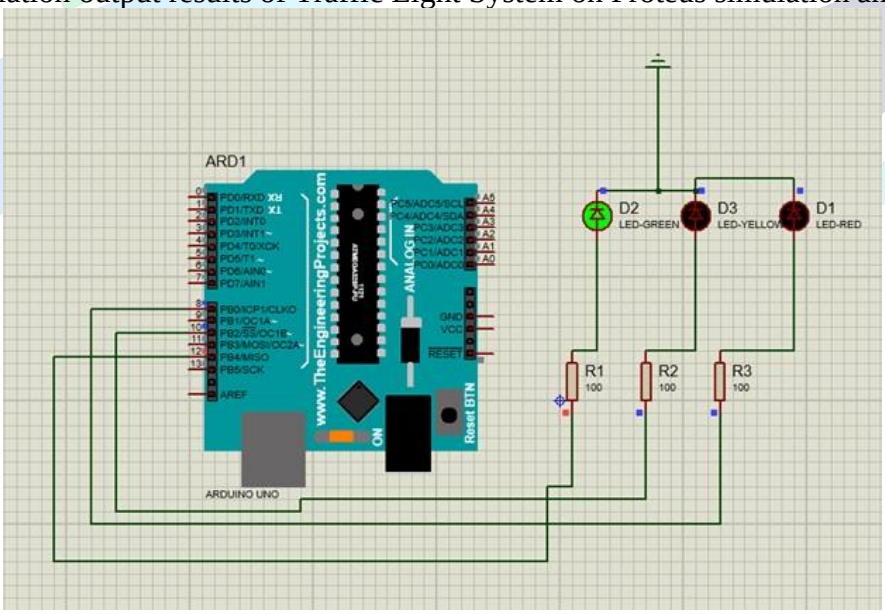

Fig.12 Green LED is ON in Traffic Light System on Proteus Simulation

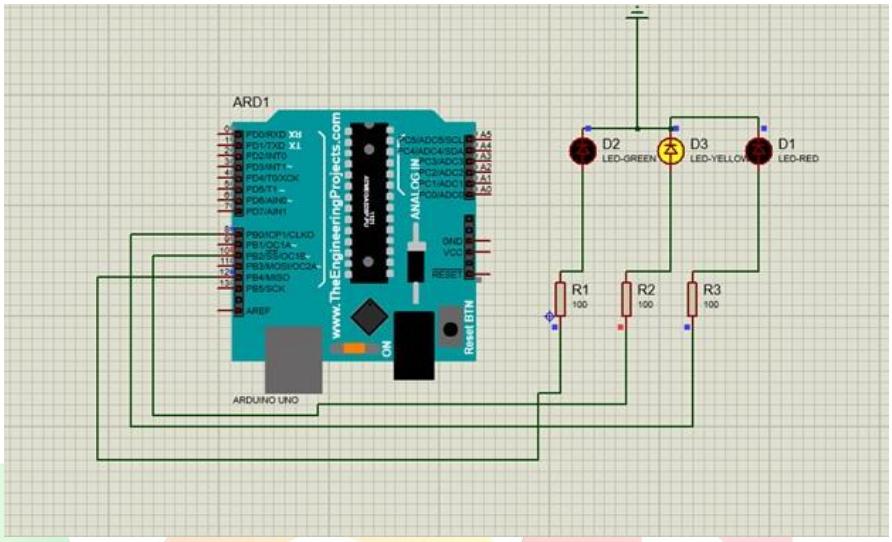

Fig.13 Yellow LED is ON in Traffic Light System on Proteus Simulation

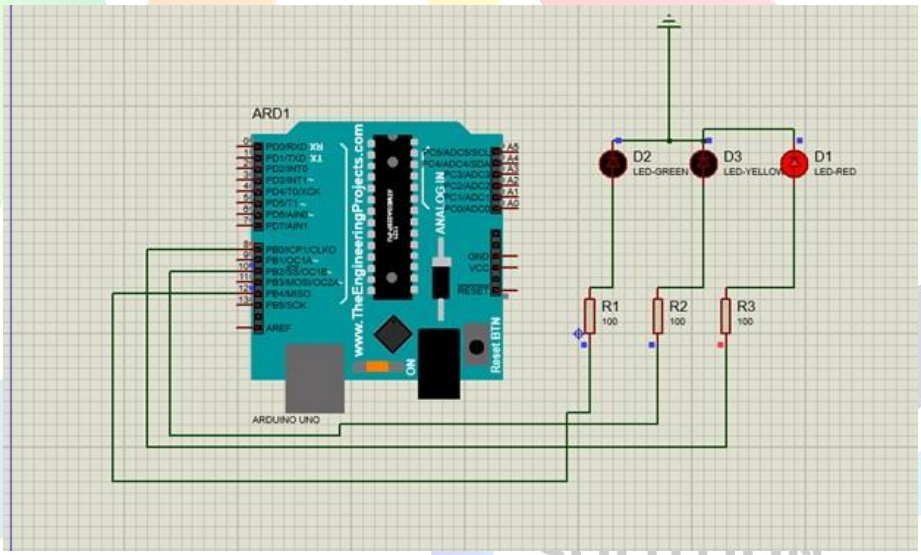

Fig.14 Red LED is ON in Traffic Light System on Proteus Simulation

Explanation: In this simulation, an Arduino Uno Board, Resistor, RED, GREEN and YELLOW LEDs were configured according to the hardware implementation that was performed. The program was made in the Arduino IDE was verified. As a result, a HEX file was generated. The following HEX file was implemented in the Proteus simulation for operation related instructions of the simulation. Afterwards, the simulation was performed and the results that were obtained were noted and compared with the hardware results.

Answers to the Questions in the Lab Manual:

3) Configure the system to have delays for outputs according to your ID. Consider the last three digits from your ID (if your ID is XY-PQABC-Z then consider A s delay for the RED LED, B s delay for the YELLOW LED, and C s delay for GREEN LED). In case any digit is zero then use Z s delay by default. Include the program and results within your lab report.

Solution: The following university ID was used:

2	1	-	4	4	6	4	9	-	1
X	Y	-	P	Q	A	B	C	-	Z

Here are the values that will be used:
Red LED Delay, A = 6 s
Yellow LED Delay, B = 4 s
Green LED Delay, C = 9 s

The following is the code for the implementation of a simple traffic control system with the necessary code explanation:

```
// Pin numbers are defined by the corresponding LED colors
#define RED_PIN 8
#define YELLOW_PIN 10
#define GREEN_PIN 12
//Define the delays for each traffic light color
int red_on = 6000; //6 s delay
int green_on = 9000; //9 s delay
int yellow_blink = 4000; //4 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);
 }
```

Here are the simulation output results of Traffic Light System according to the delay values set based on ID on Proteus simulation and explanation:

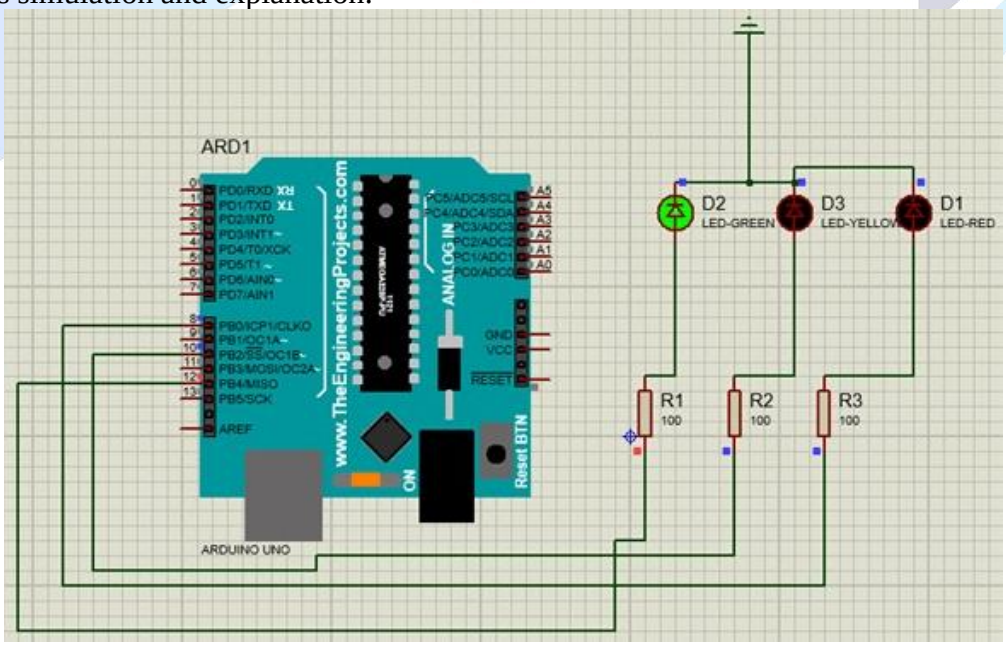

Fig.12 Green LED is ON in Traffic Light System on Proteus Simulation

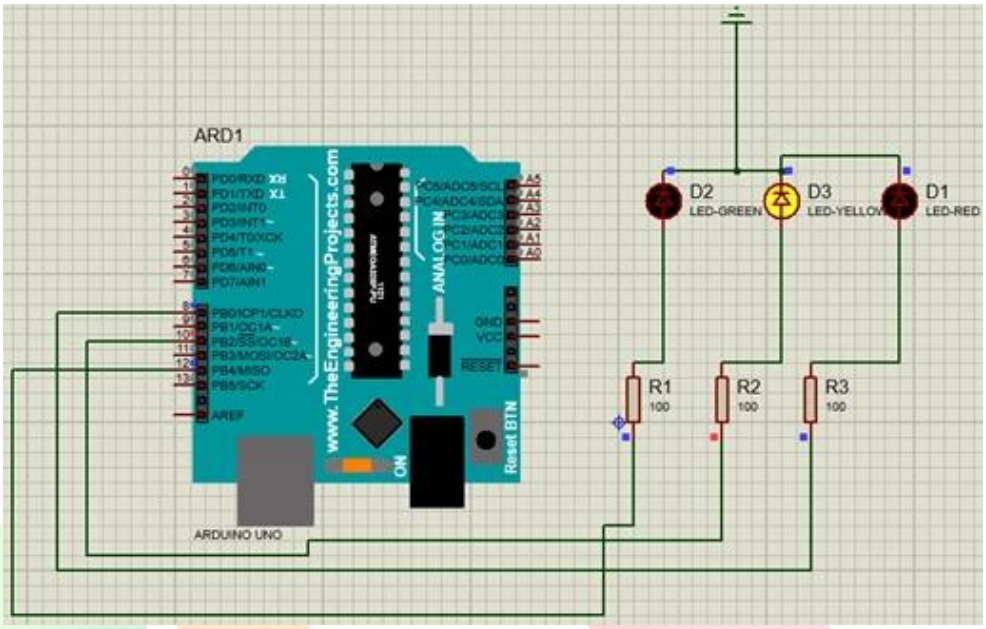

Fig.13 Yellow LED is ON in Traffic Light System on Proteus Simulation

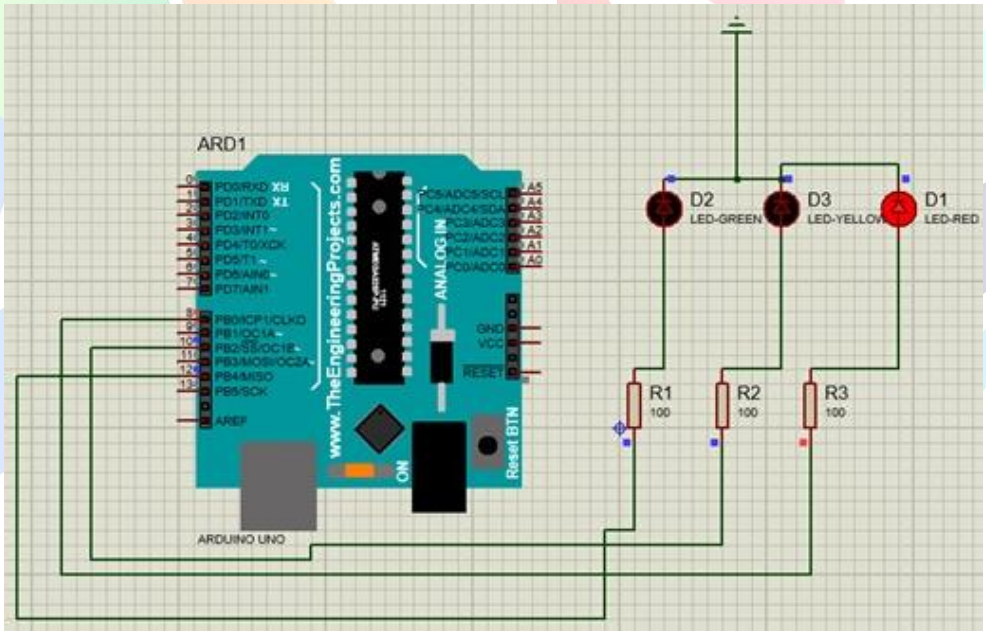

Fig.14 Red LED is ON in Traffic Light System on Proteus Simulation

Discussion:

The following experiment discussed with timers, LED blink test, and the implementation of a simple traffic control system using the Timer0 function of an Arduino microcontroller board. Timers in Arduino microcontroller boards are hardware components that allow precise timing and synchronization of events. They provide a way to generate interrupts or trigger actions at specific intervals. By understanding timers, you can leverage their capabilities to control the timing of your projects effectively. There are several timers available in Arduino boards, such as Timer0, Timer1, and Timer2. Each timer has its own set of registers and functions for configuration and control. It's important to refer to the documentation specific to your board to understand the details of the timers available. The LED blink test is a simple introductory

exercise to learn how to control the state and timing of an LED using an Arduino board. By blinking an LED at specific intervals, you can grasp the basics of digital output, delay functions, and timing control. The LED blink test typically involves setting a digital pin as an output, toggling its state between HIGH and LOW to turn the LED on and off, and adding delay functions to control the blinking frequency. This exercise helps you gain confidence in using digital pins and timing functions. Implementing a traffic control system using the Timer0 function demonstrates the practical application of timers. In this example, Timer0 is used to control the timing and sequencing of LEDs to simulate a basic traffic signal. By leveraging the timer's interrupt capability, you can precisely time the state changes of the LEDs without blocking the main program execution. Timer0 can generate interrupts at specific intervals, allowing you to define the timing behavior of the traffic control system. The implementation involves initializing the necessary pins as outputs, setting up Timer0 with the desired timing parameters, and using the interrupt service routine to control the LED states based on the defined traffic signal sequence. By combining timers, interrupts, and LED control, you can create a functioning traffic control system that can be further enhanced with additional features such as pedestrian signals or sensor integration. Overall, familiarizing yourself with timers, understanding LED blink tests, and implementing a simple traffic control system using the Timer0 function provide a solid foundation for using timers effectively in your Arduino projects. It enables you to control timing, synchronize events, and create more advanced applications that require precise timing control.

References:

1) Circuit Digest Website, [Online: March 21, 2022], [Cited: June 10, 2023] Available: https://circuitdigest.com/article/everything-you-need-to-know-about-arduino-uno-board-hardware

2) All About Circuits, *Introduction to Microcontroller Timers: Periodic Timers* [Online: June 15, 2022], [Cited: June 17, 2023] Available: https://www.allaboutcircuits.com/technical-articles/introduction-to-microcontroller-timers-periodic-timers/

3) Arduino CC Website, [Online] [Cited: June 10, 2023] Available: https://docs.arduino.cc/hardware/uno-rev3

4) Microchip Developer, *Timer0* [Online: 2021], [Cited: 18 June, 2023] Available: https://microchipdeveloper.com/8bit:timer0

3) AIUB Microprocessor and Embedded Systems Lab Manual 1