



**AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH  
(AIUB)**

**FACULTY OF SCIENCE & TECHNOLOGY  
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
MICROPROCESSOR AND EMBEDDED SYSTEMS**

**Spring 2024-2025**

**Section: Q, Group: 08**

**LAB REPORT ON**

Familiarization with an STM32, the study of blink test and implementation of a light-controlling system using microcontrollers.

**Supervised By**

Protik Parvez Sheikh

**Submitted By**

Sanjukta Das

**Group Member**

No.	Name	ID
1.	SANJUKTA DAS	22-47801-2
2.	MD. AL-IMRAN SAYEM	22-48023-2
3.	MD. ABRAR RAFID SHARIAR	22-48055-2
4.	MD. TANJIL TASHRIK ZIM	22-48021-2
5.	HUMAIRA ZANNAT	22-47789-2

**Date of Submission:** March 11, 2025

## Table of Contents

Experiment Title	3
Objectives	3
Apparatus	3
Experimental Setup	4
Code/Program	5
Experimental Results	8
Answers to the Questions in the Lab Manual	11
Discussion	14
References	15

**Experiment Title:** Familiarization with an STM32, the study of blink test and implementation of a light-controlling system using microcontrollers.

## **Objectives:**

The objectives of an experiment focused on familiarizing with an STM32 microcontroller, performing a "blink test," and implementing a light-controlling system could include the following:

### **1. Introduction to STM32 Microcontrollers**

- Understand the architecture and basic features of STM32 microcontrollers.
- Learn to set up and configure the development environment for STM32 (such as STM32CubeIDE).
- Get familiar with basic STM32 programming and GPIO configuration.

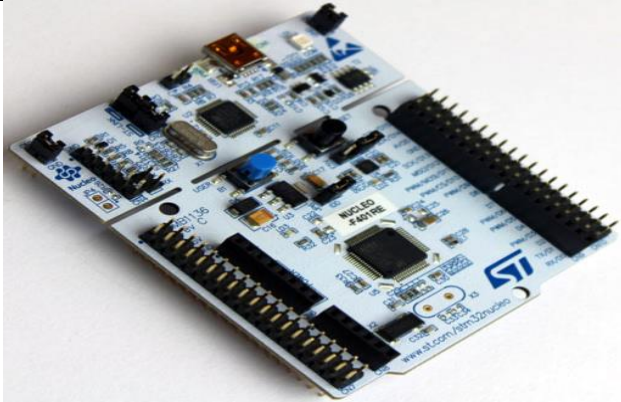
### **2. Performing the Blink Test**

- Develop a simple program to toggle an LED on and off, commonly known as a "blink test."
- Test GPIO initialization and timing control for accurate LED blinking.
- Verify correct setup and functionality of the development environment by observing the LED behavior.

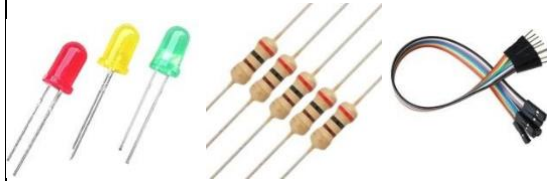
### **3. Implementing a Light-Control System**

- Design and implement a light-controlling system using GPIO and PWM (Pulse Width Modulation) for brightness control, if required.
- Learn to connect and control external LEDs or other light sources using the STM32 microcontroller.
- Use input methods (e.g., push buttons, sensors) to interact with the light control system.
- Explore advanced configurations like dimming control or multiple light settings.

## **Apparatus:**

1) STM32 Cube IDE (1.0.1 or any recent version)	Software
2) STM32 Nucleo-F401RE Board	

3) LED lights (RED, GREEN and YELLOW) and three 200 ohms resistors and jumper wires



### **Experimental Setup:**

Make the circuits first using the following connection system between all the elements:

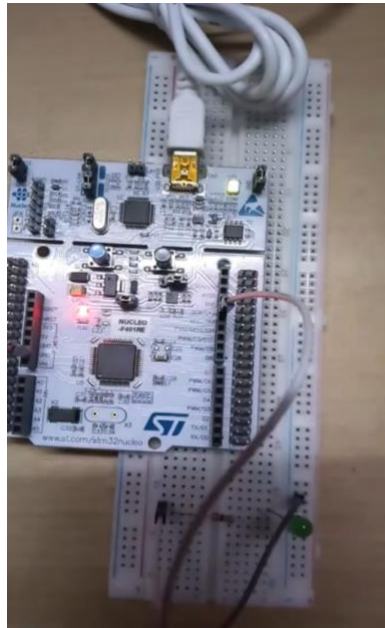


Fig.1 Experimental Setup of LED Blink Test using a STM32 Nucleo board.

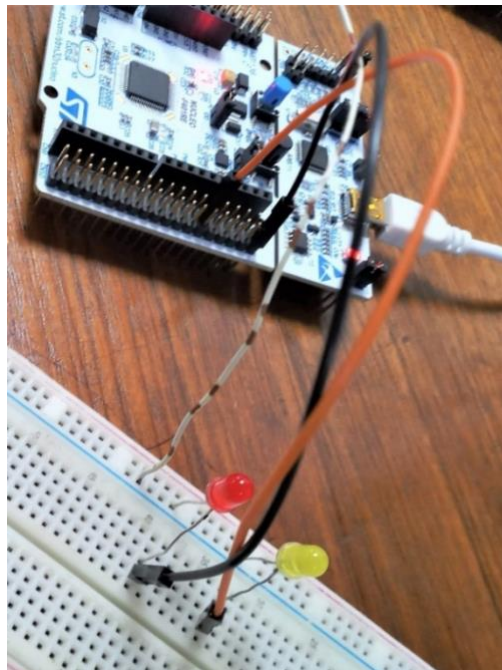


Fig.2 Experimental Setup of two LEDs Blink Test using a STM32 Nucleo board.

## **Code/Program:**

**The following code demonstrates the implementation of the LED blink test:**

```
int main(void)
{
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
/* MCU Configuration-----*/
/* Reset of all peripherals, Initializes the Flash interface and the Systick. */ HAL_Init();
/* USER CODE BEGIN Init */
/* USER CODE END Init */
/* Configure the system clock */
SystemClock_Config();
/* USER CODE BEGIN SysInit */
/* USER CODE END SysInit */
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USART2_UART_Init();
/* USER CODE BEGIN 2 */
/* USER CODE END 2 */
/* Infinite loop */

/* USER CODE BEGIN WHILE */

while (1)
{ HAL_GPIO_TogglePin(GPIOA,GPIO_PIN_5);
  HAL_Delay(1000);
}

void SystemClock_Config(void)
{
RCC_OscInitTypeDef RCC_OscInitStruct {0};
RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
/** Configure the main internal regulator output voltage
*/
HAL_RCC_PWR_CLK_ENABLE();
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE2);
/** Initializes the CPU, AHB and APB busses clocks
*/
}
```

**The following code demonstrates the implementation of the Traffic Control System test:**

```
int main(void)
{
    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */

    /* MCU Configuration-----*/

    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();

    /* USER CODE BEGIN Init */

    /* USER CODE END Init */

    /* Configure the system clock */
    SystemClock_Config();

    /* USER CODE BEGIN SysInit */

    /* USER CODE END SysInit */

    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_USART2_UART_Init();
    /* USER CODE BEGIN 2 */

    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        #define RED_PIN GPIO_PIN_5
        #define YELLOW_PIN GPIO_PIN_6
        #define GREEN_PIN GPIO_PIN_7

        int red_on = 4000;
```

```

int yellow_on = 1000;
int green_on = 4000;
int green_blink = 500;


    HAL_GPIO_WritePin(GPIOA,RED_PIN,1);
    HAL_Delay(red_on);
    HAL_GPIO_WritePin(GPIOA,RED_PIN,0);


    HAL_GPIO_WritePin(GPIOA,YELLOW_PIN,1);
    HAL_Delay(yellow_on);
    HAL_GPIO_WritePin(GPIOA,YELLOW_PIN,0);


    HAL_GPIO_WritePin(GPIOA,GREEN_PIN,1);
        HAL_Delay(green_on);
    for(int i = 0; i < 3; i = i+1)
    {
        HAL_GPIO_WritePin(GPIOA,GREEN_PIN,1);
        HAL_Delay(green_blink);
        HAL_GPIO_WritePin(GPIOA,GREEN_PIN,0);
        HAL_Delay(green_blink);
    }


    HAL_GPIO_WritePin(GPIOA,YELLOW_PIN,1);


        HAL_Delay(yellow_on);
        HAL_GPIO_WritePin(GPIOA,YELLOW_PIN,0);
    }
/* USER CODE END 3 */
}

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

```



## Experimental Results:

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

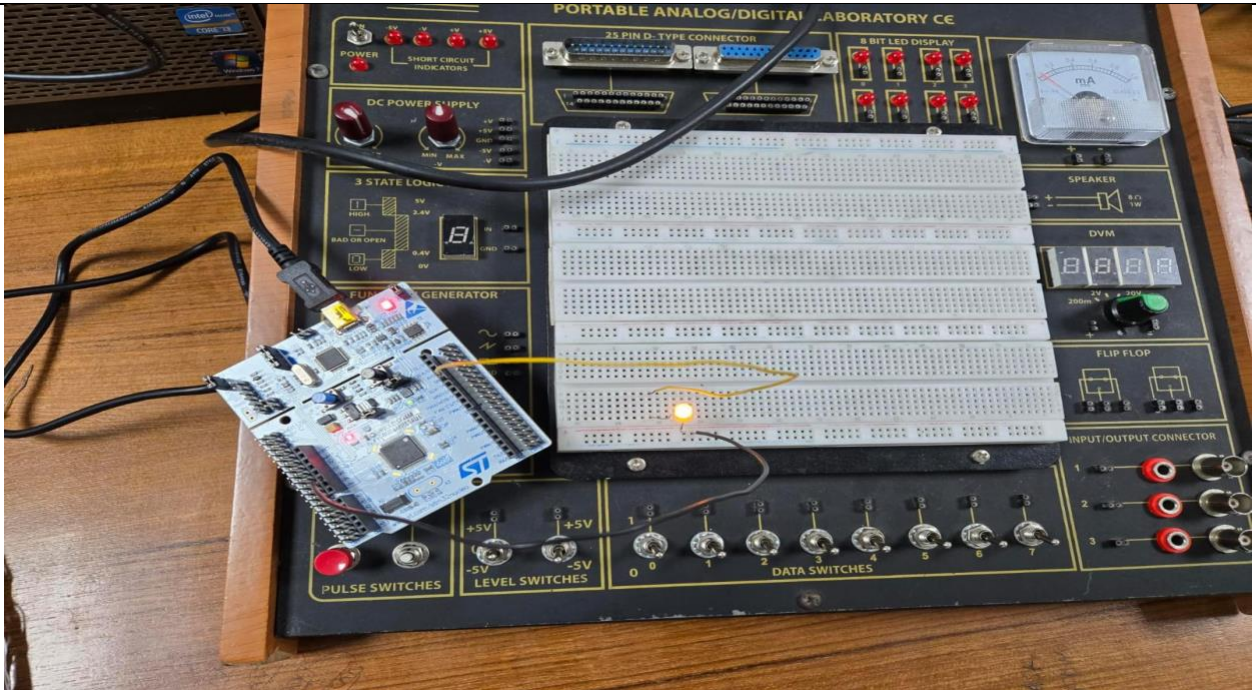


Fig.3 LED is ON in Light Blink Test.

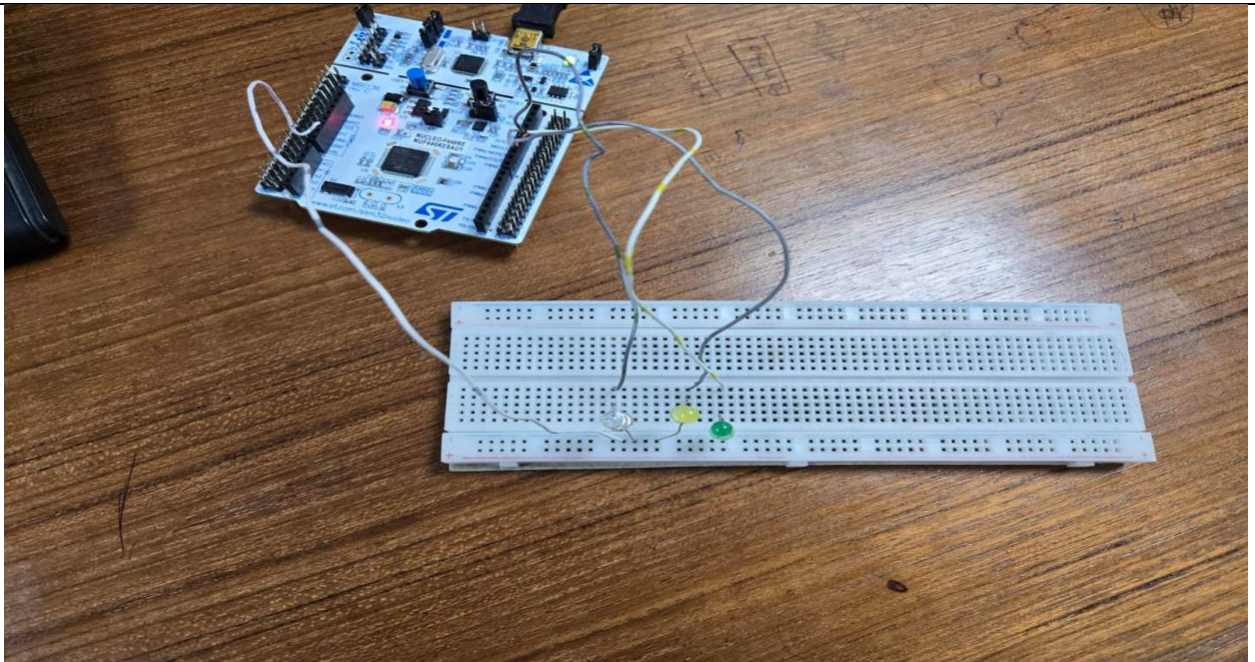


Fig.4 LED is OFF in Light Blink Test.



**Explanation :** In this setup, we have an STM32 Nucleo board connected to a breadboard with an LED and a resistor. The board is powered and programmed via a USB connection to a computer.

A green LED is placed on the breadboard, with its anode connected to a current-limiting resistor. This resistor prevents excessive current flow, protecting both the LED and the microcontroller's GPIO pin. The other end of the resistor is connected to GPIO pin A5 on the STM32 board, which is configured in the code to toggle on and off, making the LED blink.

To complete the circuit, a ground (GND) wire runs from the STM32 board to the breadboard. Another wire connects GPIOA pin 5 to the resistor and LED, allowing the microcontroller to control the LED's state.

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

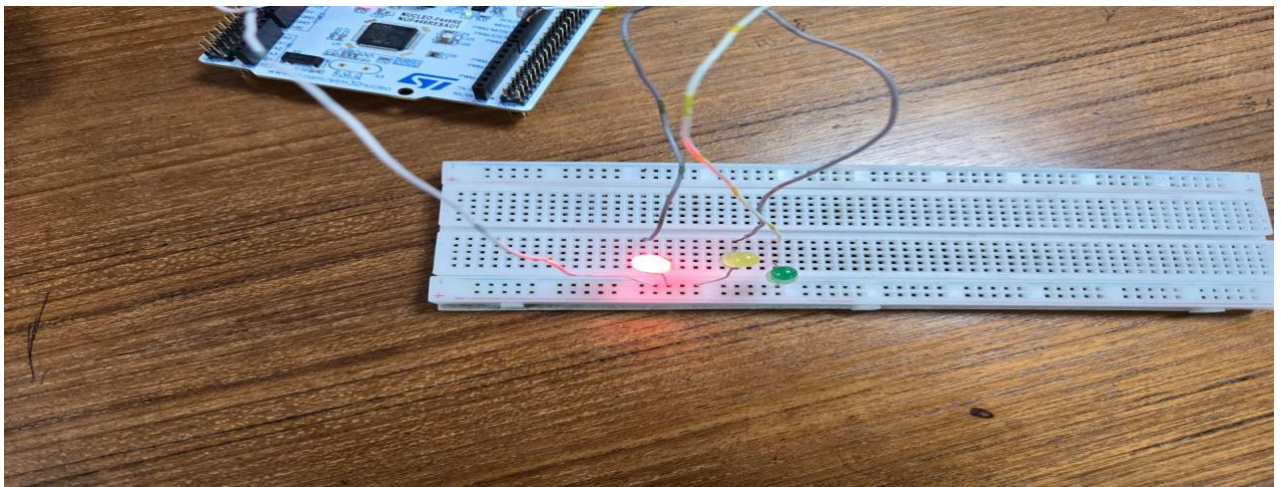


Fig.5 Red LED is ON in the Traffic Light System

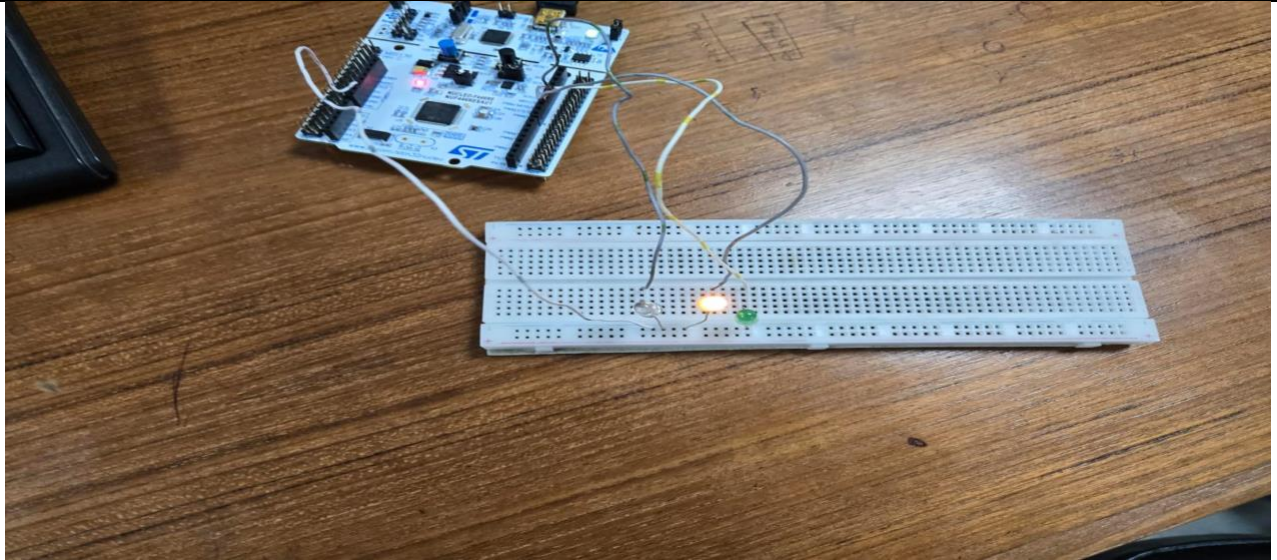


Fig.6 Yellow LED is ON in the Traffic Light System

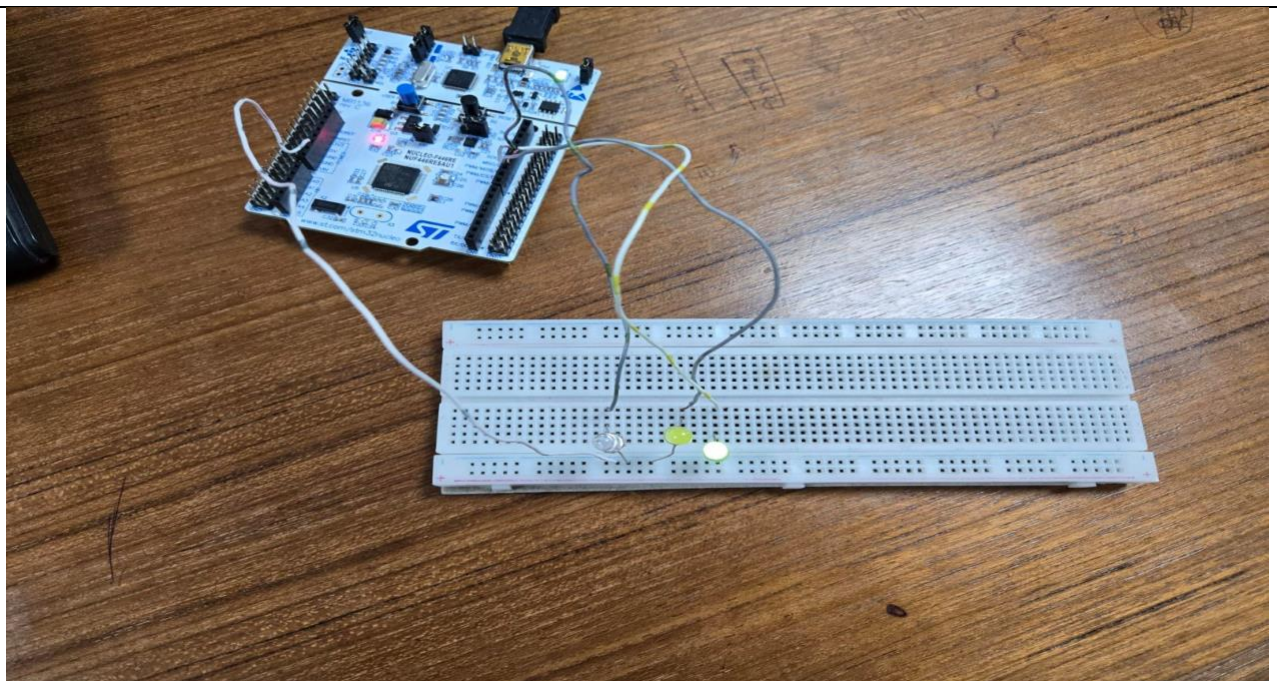


Fig.7 Green LED is ON in the Traffic Light System



### Explanation:

This setup is an STM32-based traffic light system, where each LED represents a different traffic light signal: red, yellow, and green. The STM32 microcontroller board is connected to the computer via USB, which powers the system and enables programming. There are three LEDs on the breadboard: red, yellow, and green, each connected with a current-limiting resistor. The red LED is controlled by **GPIOA pin 5**, the yellow LED is controlled by **GPIOA pin 6** and the green LED is controlled by **GPIOA pin 7**. Wires connect each LED to the corresponding GPIO pin on the STM32 board. Each LED also has a resistor in series to prevent excessive current, which could damage the LED or microcontroller pin. A common ground connection is made to complete each LED circuit.

### Answers to the Questions in the Lab Manual:

#### Simulation:

1: Here are the simulation output results of the LED Blink Test simulation:

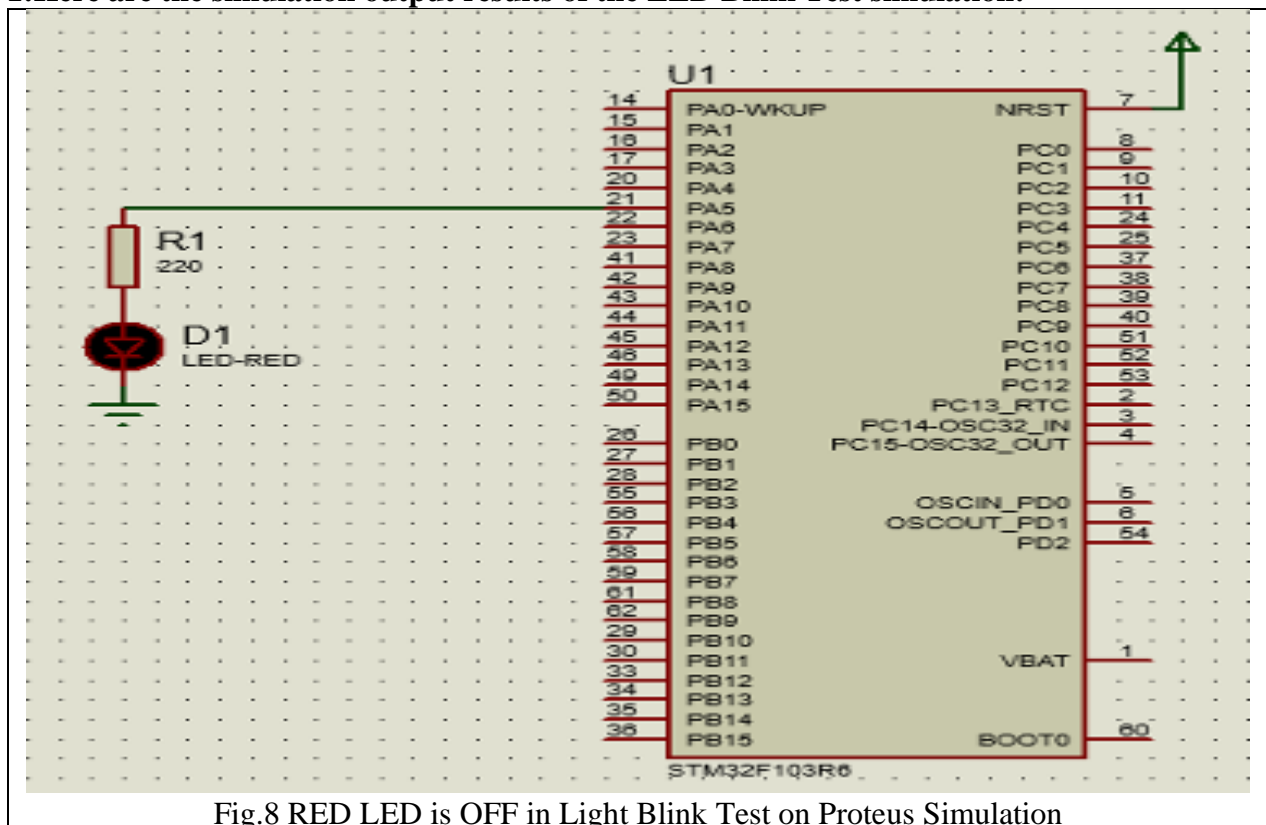


Fig.8 RED LED is OFF in Light Blink Test on Proteus Simulation

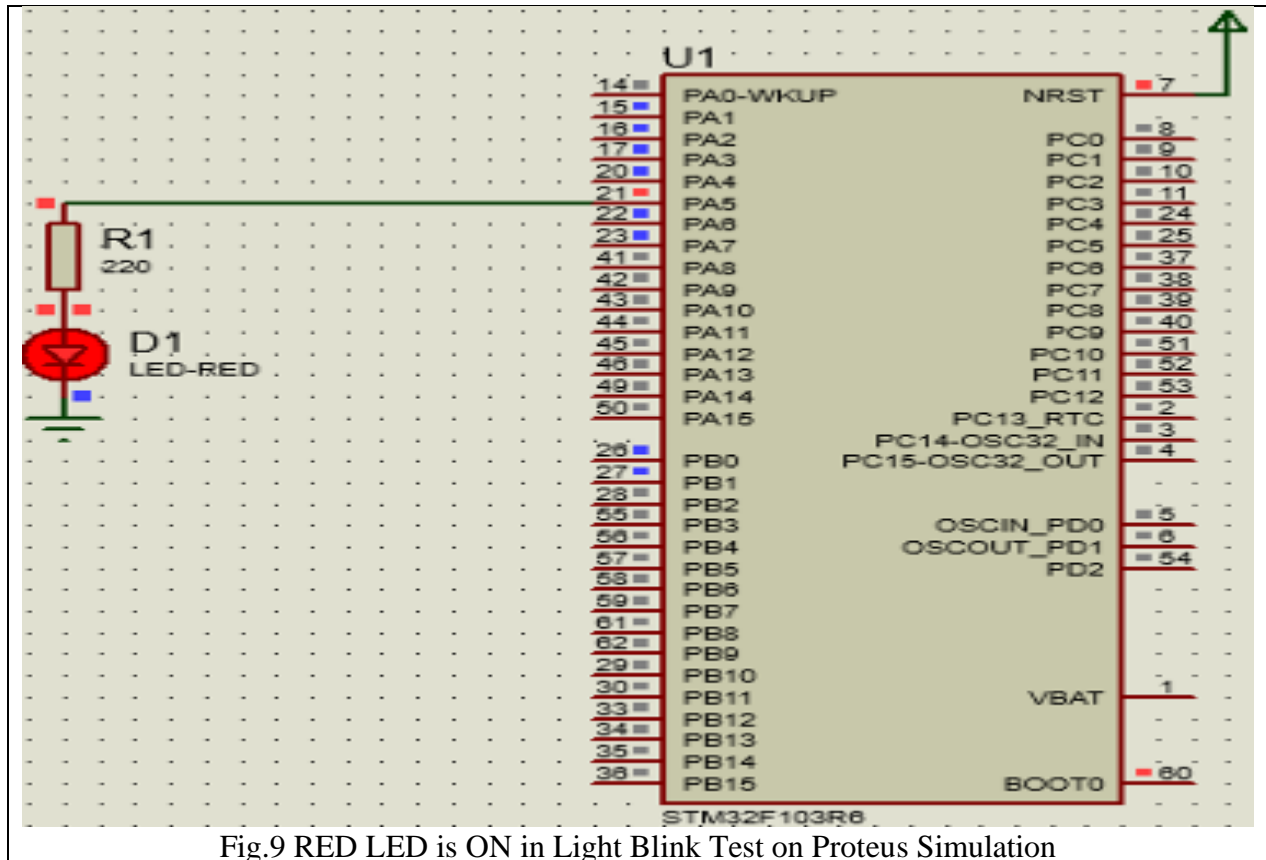


Fig.9 RED LED is ON in Light Blink Test on Proteus Simulation

### **Explanation:**

In this simulation, we set up an STM32F401RE board, a resistor, and a red LED to replicate the physical hardware configuration we previously created. (Note: In our actual hardware setup, we used a green LED, but in Proteus, we used a red LED.)

We developed the program using STM32 IDE, carefully selecting the correct pins for our setup. After configuring the project, we generated the HEX file, which we then loaded into Proteus for simulation.

Once the simulation was running, we observed the LED's behavior and recorded our findings. Finally, we compared the simulated results with those from the actual hardware setup to ensure consistency and accuracy.

2:Here are the simulation output results of the Traffic Light System simulation:

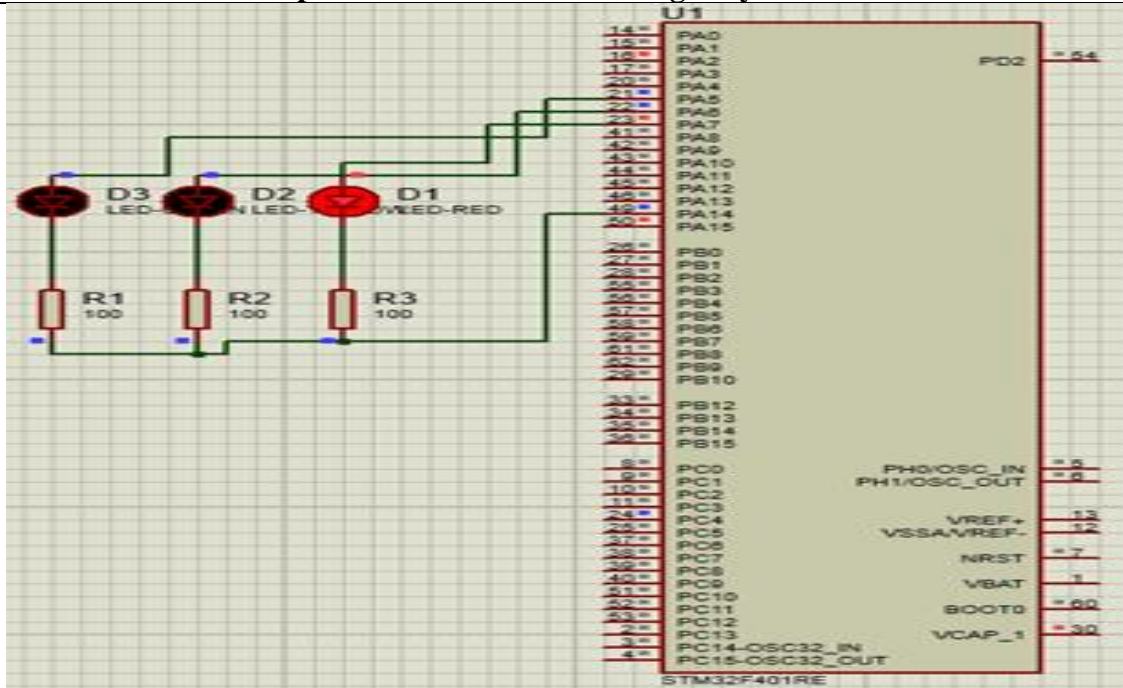


Fig.10 RED LED is ON in Traffic Light System on Proteus Simulation

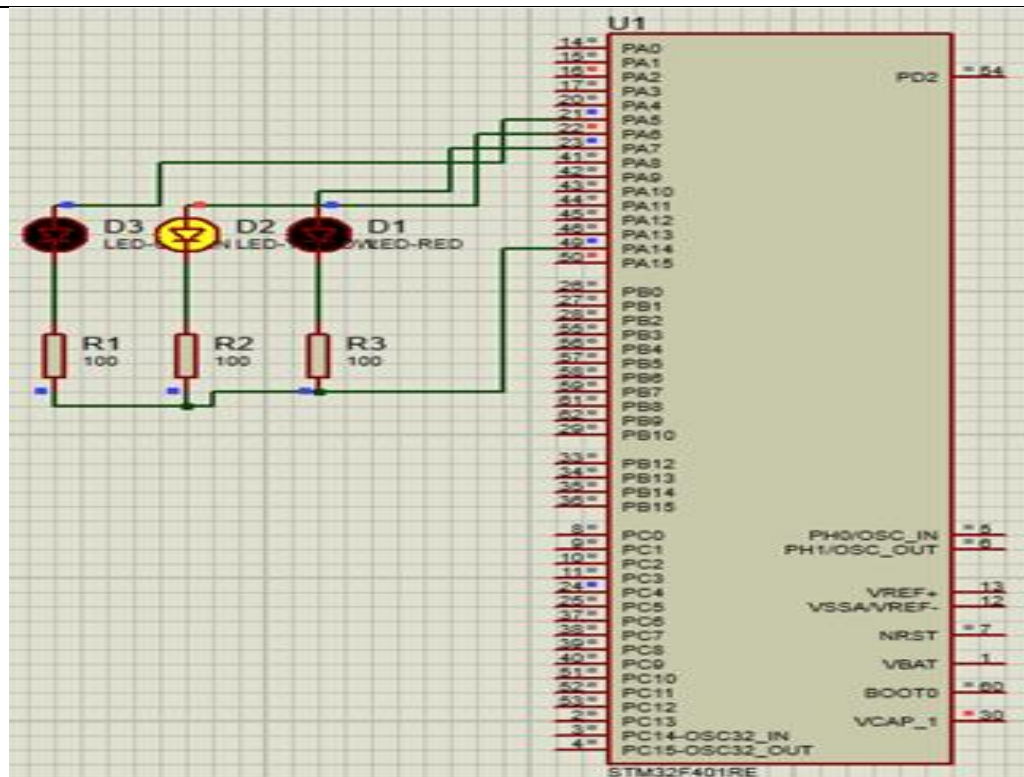


Fig.11 YELLOW LED is ON in Traffic Light System on Proteus Simulation

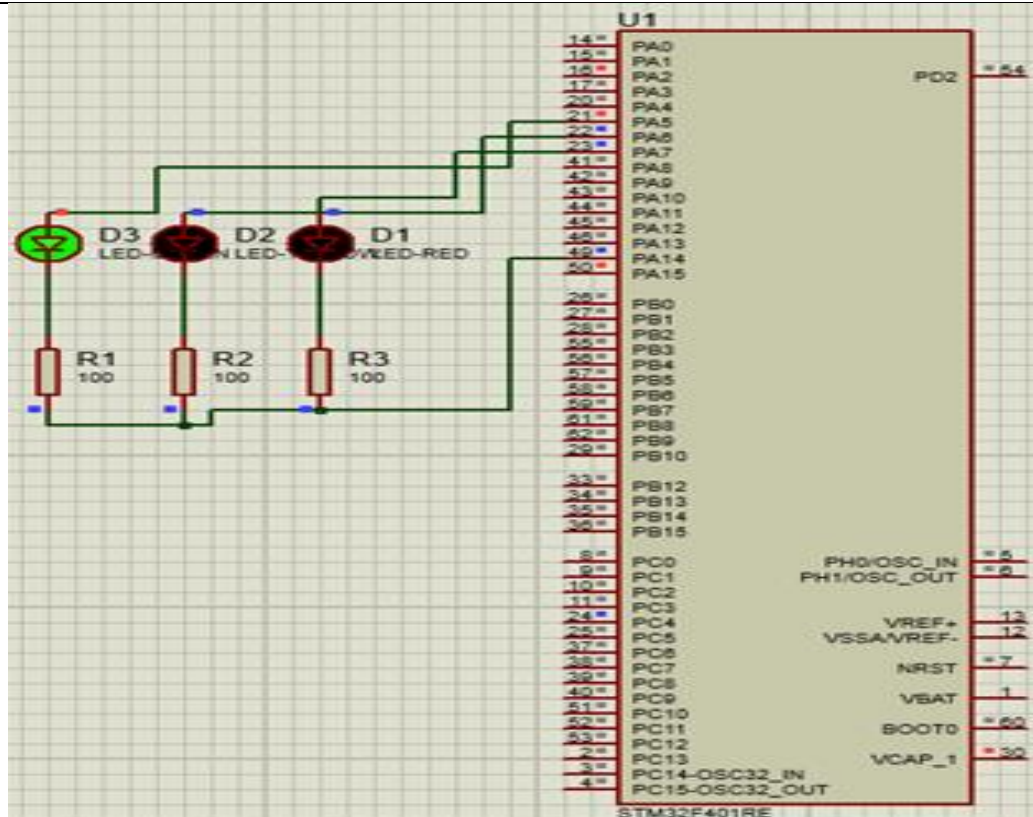


Fig.12 GREEN LED is ON in Traffic Light System on Proteus Simulation

### **Explanation:**

In this simulation, we set up an STM32F401RE board along with resistors and red, green, and yellow LEDs to match our physical hardware setup. Using the STM32 IDE, we wrote the program and carefully selected the appropriate pins for each component. After finalizing the setup, we generated a HEX file, which we then imported into the Proteus simulation to execute the operational instructions.

Once the HEX file was loaded, we ran the simulation, observed the results, and documented key findings. Finally, we compared these simulated results with the results from our hardware setup to ensure everything functioned as expected.



## **Discussion:**

The objective of this experiment was to gain hands-on experience with an STM32 microcontroller, perform a basic LED blink test, and develop a traffic light control system. Each goal was successfully achieved, providing valuable insights into the microcontroller's functionality.

Familiarizing ourselves with the STM32 microcontroller was crucial for understanding its architecture, features, and operation. This exploration built a strong foundation in key areas such as pin configuration, input/output functions, and communication interfaces—essential knowledge for future programming tasks.

The LED blink test served as a practical introduction, allowing us to apply what we learned by programming the microcontroller to control an LED. Observing the LED blink at a fixed frequency reinforced our understanding of GPIO operations and coding syntax in the STM32 environment.

The highlight of the experiment was designing and implementing a traffic light control system on the STM32 board. Through careful planning and programming, we successfully simulated the timing and sequencing of traffic signals. This hands-on exercise deepened our understanding of embedded systems and real-time control applications.

Overall, this experiment provided a solid grasp of both fundamental and intermediate applications of the STM32 microcontroller, ensuring a strong foundation for more advanced projects.

4o

## **References:**

- 1) <https://www.st.com/en/evaluation-tools/nucleo-f401re.html> for STM32F401RE,datasheet
- 2) [www.st.com](http://www.st.com)
- 3) AIUB Microprocessor and Embedded Systems Lab Manual 2
- 4) [https://www.st.com/resource/en/user\\_manual/dm00105879-description-of-stm32f4-hal-and-ll-drivers-stmicroelectronics.pdf](https://www.st.com/resource/en/user_manual/dm00105879-description-of-stm32f4-hal-and-ll-drivers-stmicroelectronics.pdf)
- 5) [www.st.com/en/development-tools/stm32cubeide.html](http://www.st.com/en/development-tools/stm32cubeide.html)