AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Engineering

Lab Report

Experiment 10

Experiment Title: Familiarization with the Raspberry Pi.

Date of Perform:	18 May 2025	**Date of Submission:**	25 May 2025
Course Title:	Microprocessor and Embedded Systems Lab		
Course Code:	EE4103	**Section:**	G
Semester:	Spring 2024-25	**Degree Program:**	BSc in CSE/EEE
Course Teacher:	**Prof. Dr. Engr. Muhibul Haque Bhuyan**		

Group # 02

Sl No	Name	ID	PROGRAM	SIGNATURE
1	MAHBUB HASAN	22-47419-3	BSc in CSE	
2	MUJTAHID TABASSUM MOHONA	22-47430-3	BSc in CSE	
3	MD.SIAM MEHEDI	22-48342-3	BSc in CSE	
4	MD TOYEB ALI	23-52819-2	BSc in CSE	

Faculty use only

FACULTY COMMENTS	**Marks Obtained**	
	Total Marks	

Marking Rubrics (to be filled by Faculty):

Level Category	Excellent [5]	Proficient [4]	Good [3]	Acceptable [2]	Unacceptable [1]	No Response [0]
Title and Objectives	Able to clarify the understanding of the lab, no issues are missing and formatting is good.	Able to clarify the understanding of the lab experiment, no issues are missing but its formatting is not good.	Able to clarify the understanding of the lab experiment, but a few issues are wrong, and its formatting is bad.	Able to clarify the understanding of the lab experiment, but it lacks a few important issues of the experiment without maintaining the format.	Unable to clarify the understanding of the lab experiment.	No Response/ copied from others/ identical submissions with gross errors/image file printed
Codes and Methods	Able to explain the experimental codes and simulation methods using Proteus very well.	Able to explain the experimental codes and simulation methods using Proteus but is not formatted well.	Able to explain the experimental codes but simulation method using Proteus is not explained well.	Presents the experimental codes but didn't explain simulation methods using Proteus clearly.	Presents the experimental codes but didn't explain simulation methods using Proteus.	
Results	Key results and images are there. Figures/Tables have all identifications and refer to them properly in the texts.	Key results and images are there. Figures/Tables have all identifications, such as the axis labels, numbers, and captions with a few minor errors; the texts refer them.	Key results and images are there. Figures/Tables lack a few identifications, such as the axis labels, numbers, and captions; the texts refer them.	Misses several key results and images. Figures/Tables lack identification, such as the axis labels, numbers, and captions; the texts don't refer them.	Major results, such as experimental and simulation results' images are not included. Figures and tables are poorly constructed or not presented.	
Discussion and Conclusion	Proper interpretation of results and summarizes the results to draw a conclusion, discusses its applications in real-life situations to connect with the report's conclusion.	Proper interpretation of results and summarizes the results to draw a conclusion but didn't discuss its applications in real-life situations to connect with the conclusion of the report.	Interpretation of results is presented. However, there is a disconnect between the results and discussion.	Misses the interpretation of key results. There is little connection between the results and discussion.	Very poor interpretation of the results. No connection between results and discussions.	
Question and Answer	Able to produce all questions' answers correctly maintaining the lab report format.	Able to produce all questions' answers but didn't maintain the lab report format.	Able to produce all questions' answers but wrong answers to a few questions.	Able to produce all questions' answers but wrong/missing answers to multiple questions.	Unable to produce all questions' answers and completely wrong answers.	
Comments						**Total Marks (25)**

Table of contents:

<u>Objectives:</u>

The objectives of this experiment are to-

1. Familiarize the students with the Raspberry Pi.
2. Make an LED blink using the Raspberry Pi and its time. Sleep () function.
3. Control the LEDs' ON/OFF using the input push switch.
4. Implement a traffic light control system.

<u>Equipment List:</u>

1. Activated Raspberry pi
2. LED
3. Push switch
4. Resistor (220 □)
5. Breadboard
6. Jumper wires

<u>Circuit Diagram:</u>

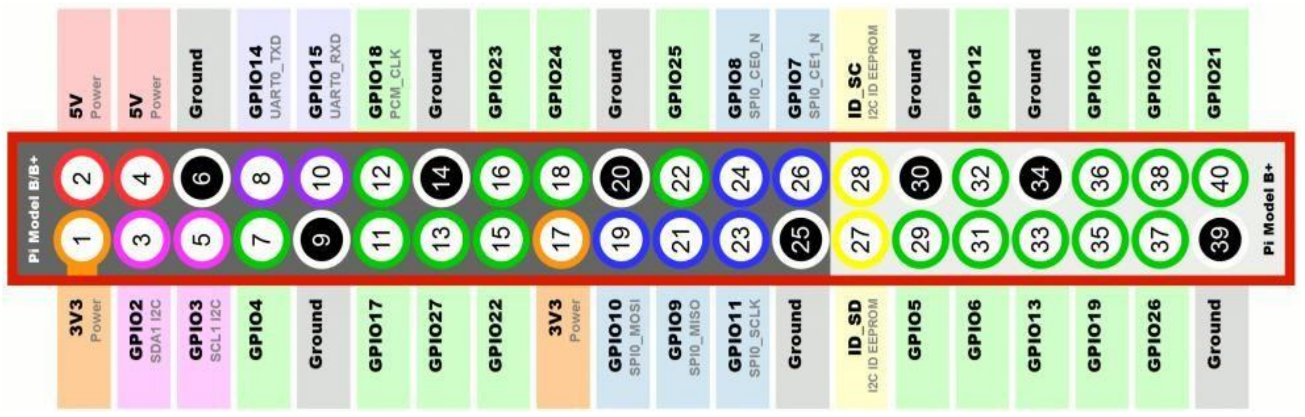


Figure 1: Raspberry Pi 3 - Model B GPIO pin

<u>Lab Task 1: LED Blinking circuit diagram:</u>

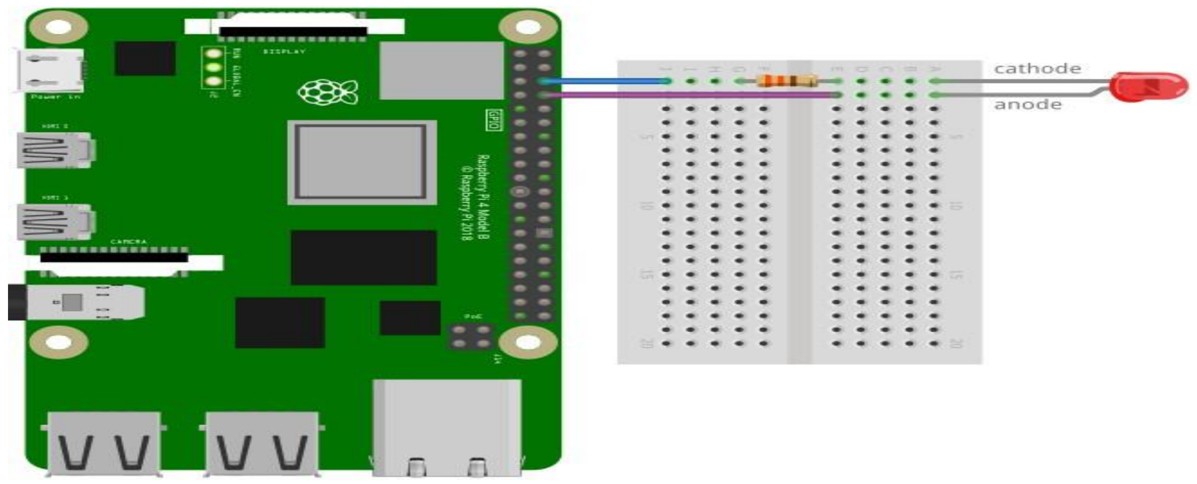


Figure 2: Setting up the circuit for the LED blinking program.

Lab Task 2: LED controlling with a push button switch circuit diagram

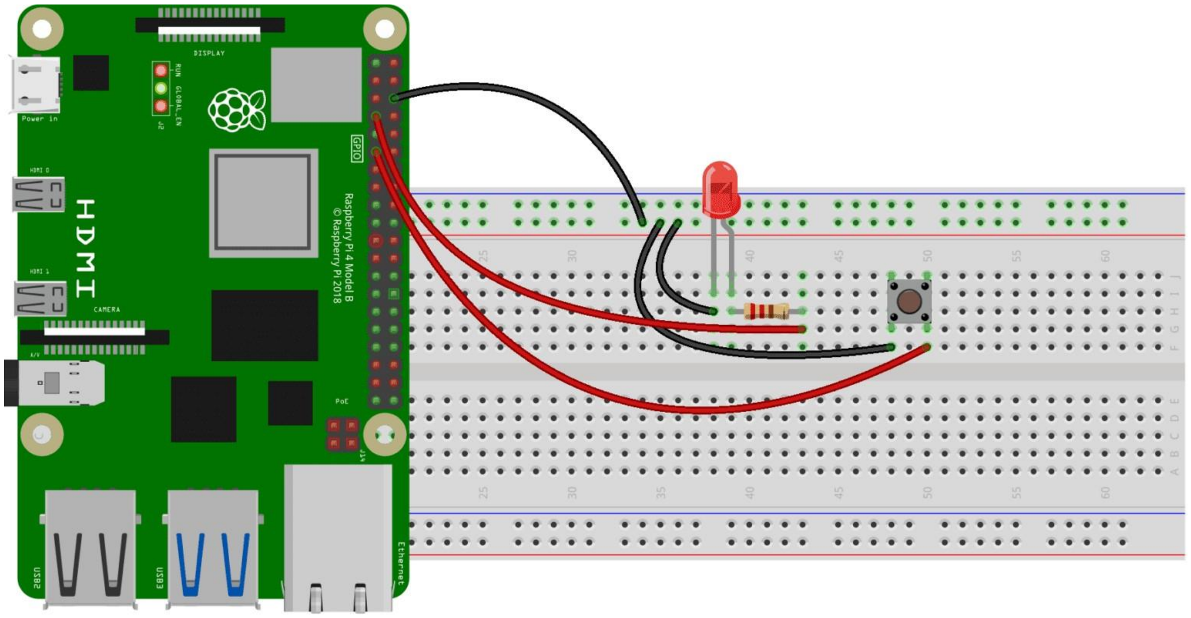


Figure 3: Setting up the circuit for the LED controlling experiment using a button switch.

Lab Task # 3: Simple Traffic Control System

Design a traffic control system using RED, YELLOW, and GREEN LEDs.

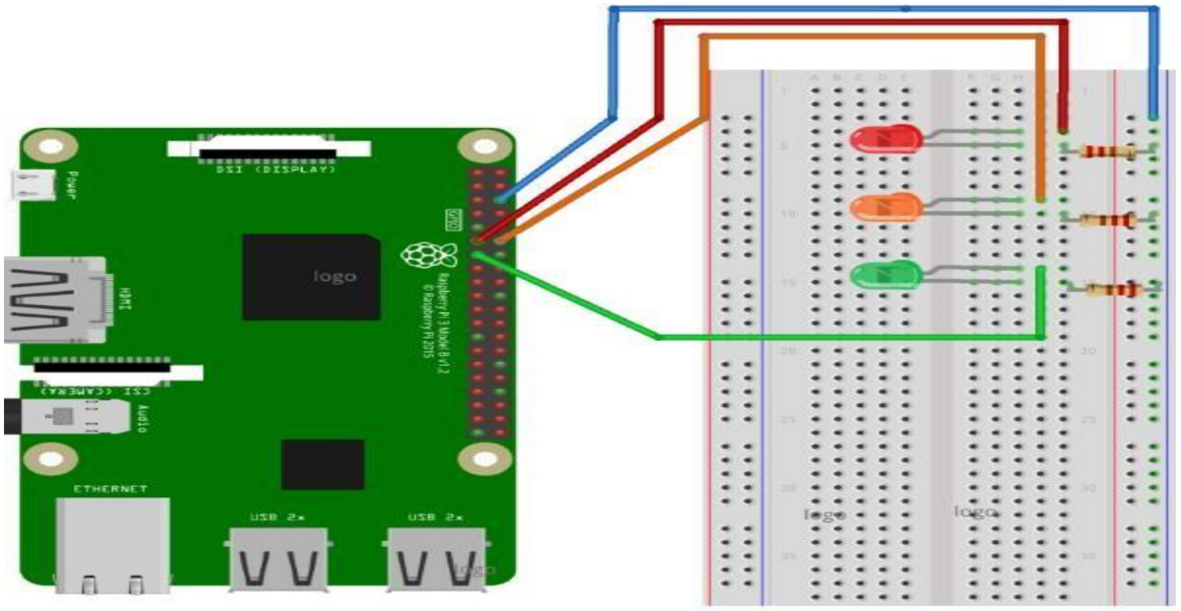


Figure 4: Setting up the circuit for the traffic control system using RED, YELLOW, and GREEN LEDs.

Experimental Output Results:

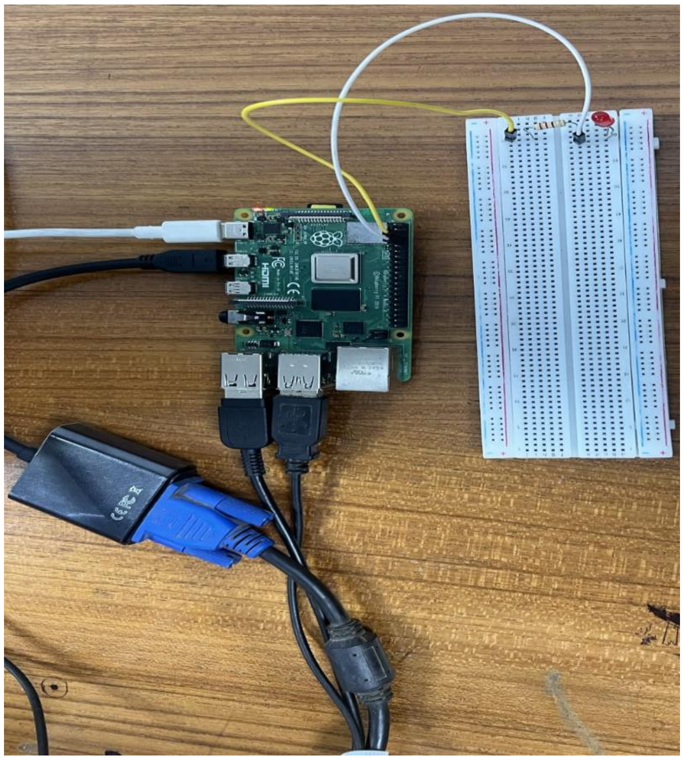


Figure 5: LED Blinking circuit diagram LED OFF

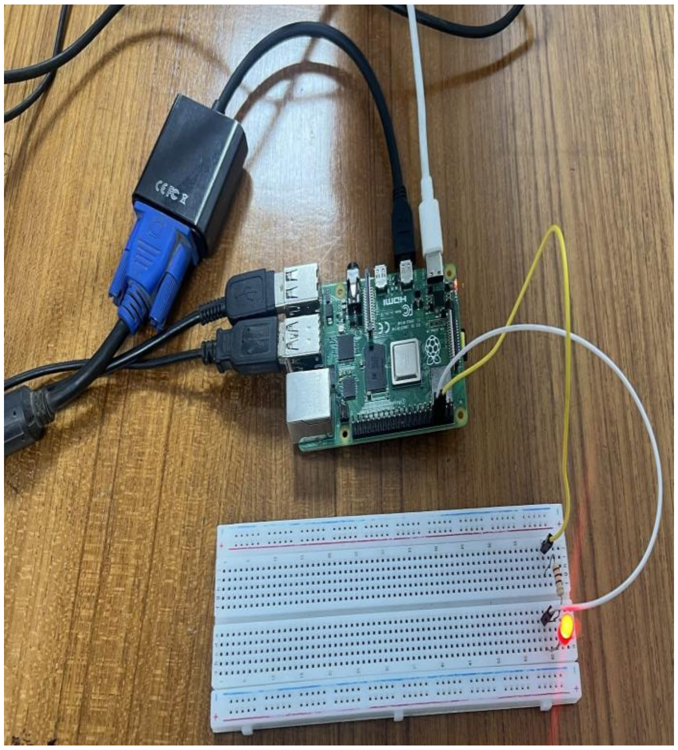


Figure 6: LED Blinking circuit diagram LED ON

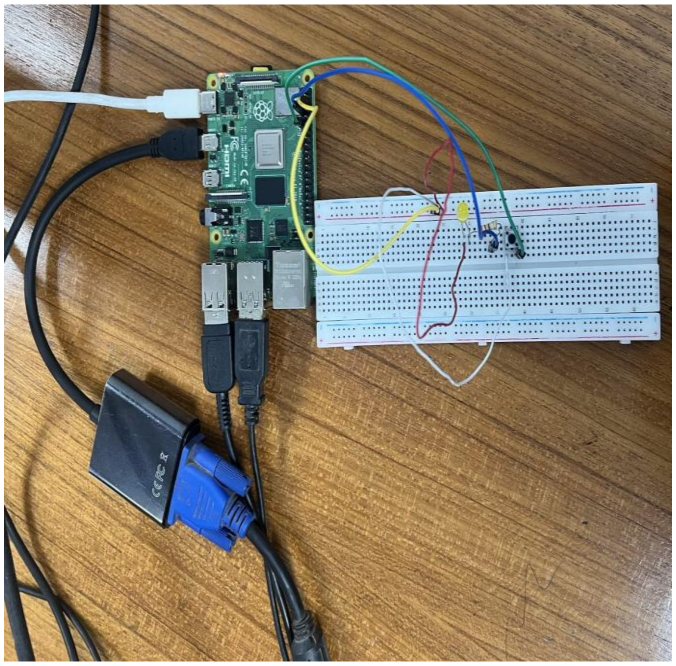


Figure 7: LED controlling with a push button

switches circuit diagram LED OFF

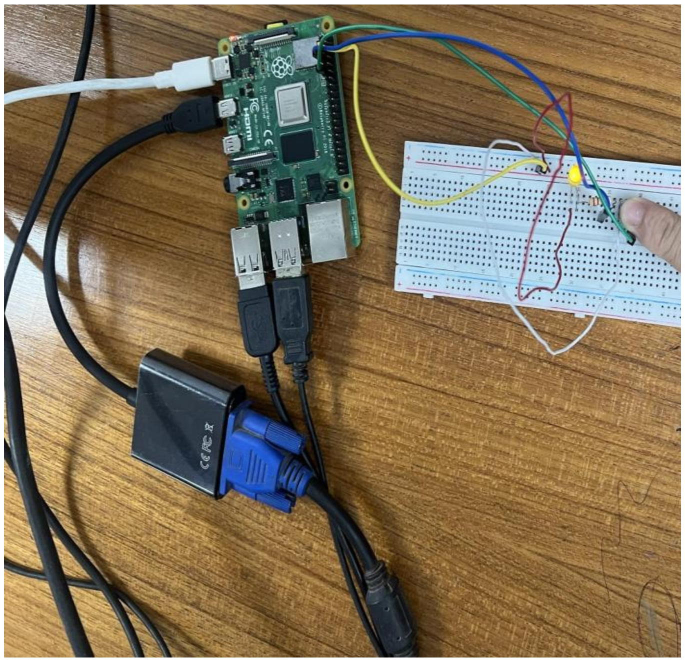


Figure 8: LED controlling with a push button switches circuit diagram LED ON

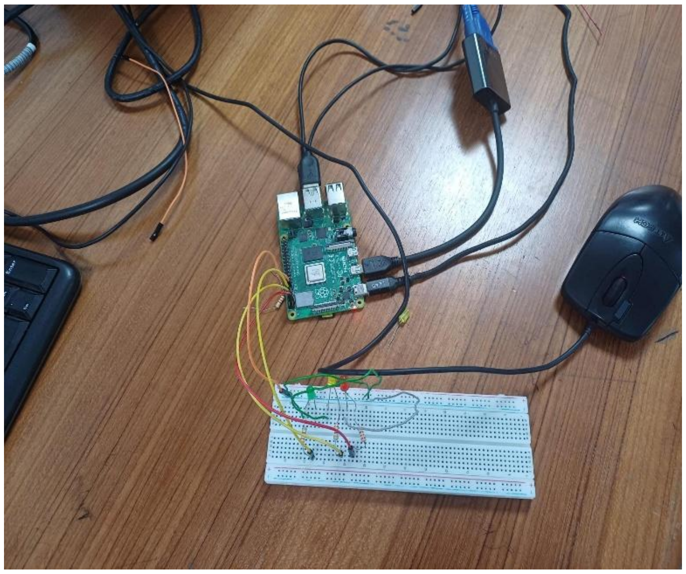


Figure 9: traffic control system where RED, YELLOW, and GREEN LEDs. is off

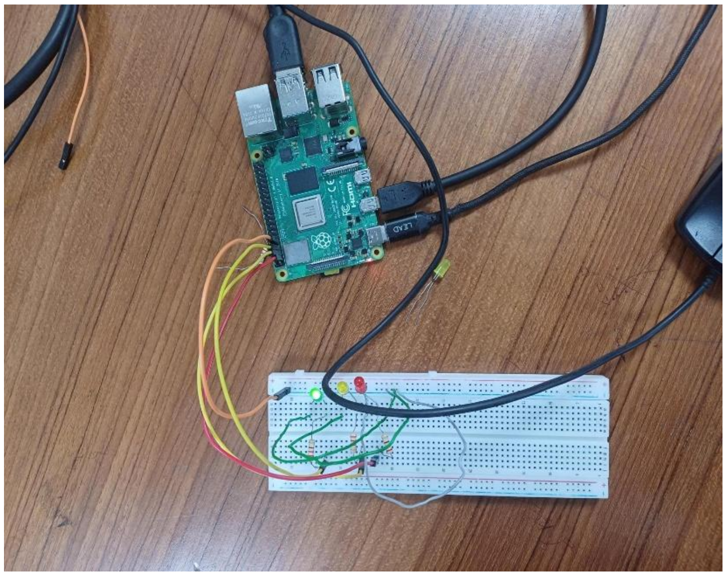


Figure 10: traffic control system where GREEN LEDs. is on

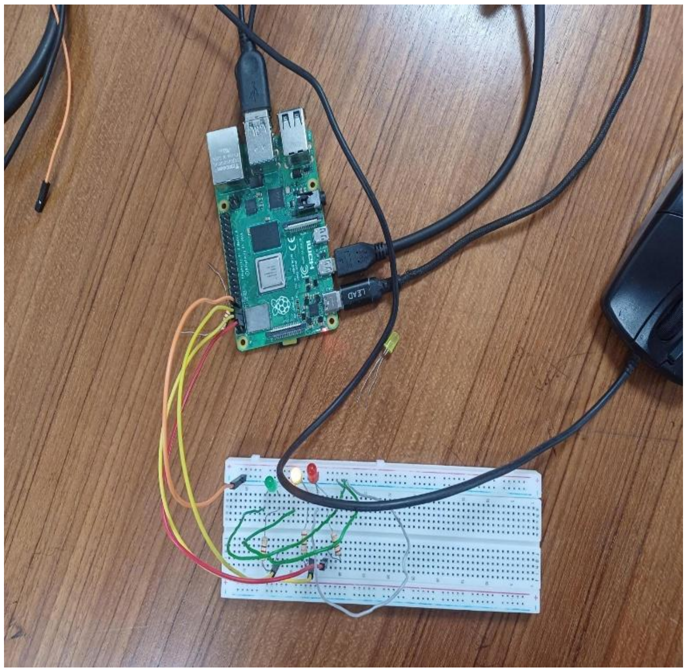


Figure 11: traffic control system

where YELLOW is on

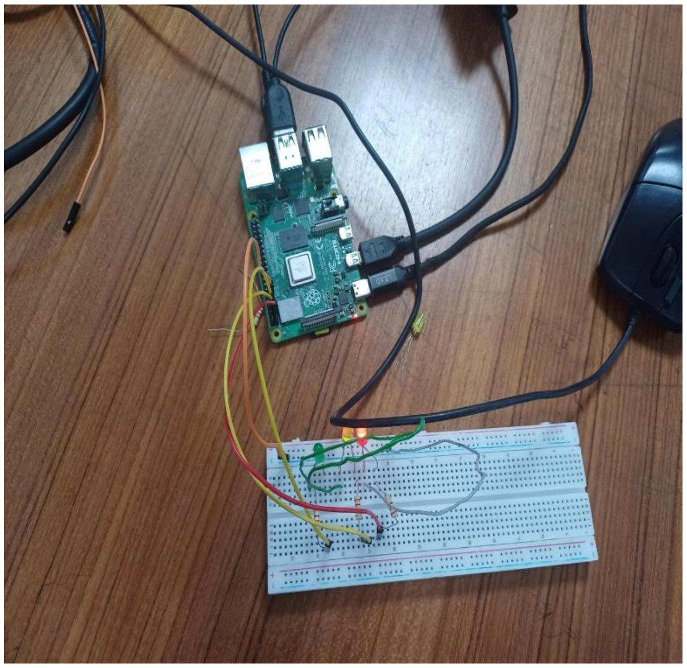
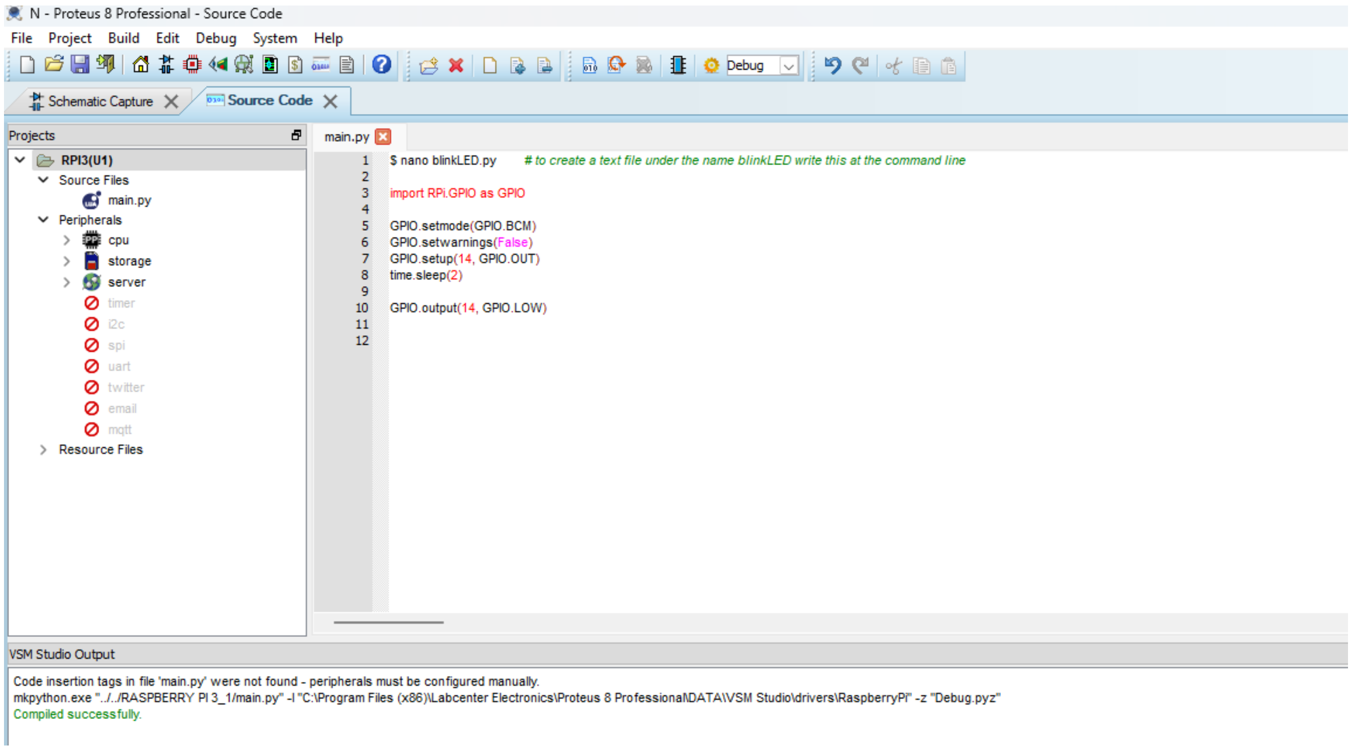


Figure 12: traffic control system where RED is on

Setup Procedure:

- The Raspberry Pi was connected to a power supply, HDMI monitor, keyboard, and mouse.
- A breadboard was placed next to the Raspberry Pi for circuit building.
- An LED and a resistor were inserted into the breadboard.
- A jumper wire was used to connect the anode (long leg) of the LED to a GPIO pin on the Raspberry Pi.
- The cathode (short leg) of the LED was connected to a resistor, and the other end of the resistor was connected to the ground rail of the breadboard.
- A jumper wire was used to connect the ground rail of the breadboard to a GND pin on the Raspberry Pi.
- A push button was inserted into the breadboard with its terminals placed across the middle gap.
- One terminal of the push button was connected to another GPIO pin using a jumper wire.
- The opposite terminal of the button was connected to the ground rail to complete the circuit.
- All GPIO pin connections were double-checked to ensure correct placement.

Sketch Explanation:



```
$ nano blinkLED.py       # to create a text file under the name blinkLED write this at the command line

import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(14, GPIO.OUT)
time.sleep(2)

GPIO.output(14, GPIO.LOW)
```

```
Code insertion tags in file 'main.py' were not found - peripherals must be configured manually.
mkpython.exe "../../RASPBERRY PI 3_1/main.py" -I "C:\Program Files (x86)\Labcenter Electronics\Proteus 8 Professional\DATA\VSM Studio\drivers\RaspberryPi" -z "Debug.pyz"
Compiled successfully.
```

Figure 13: LED Blinking Code

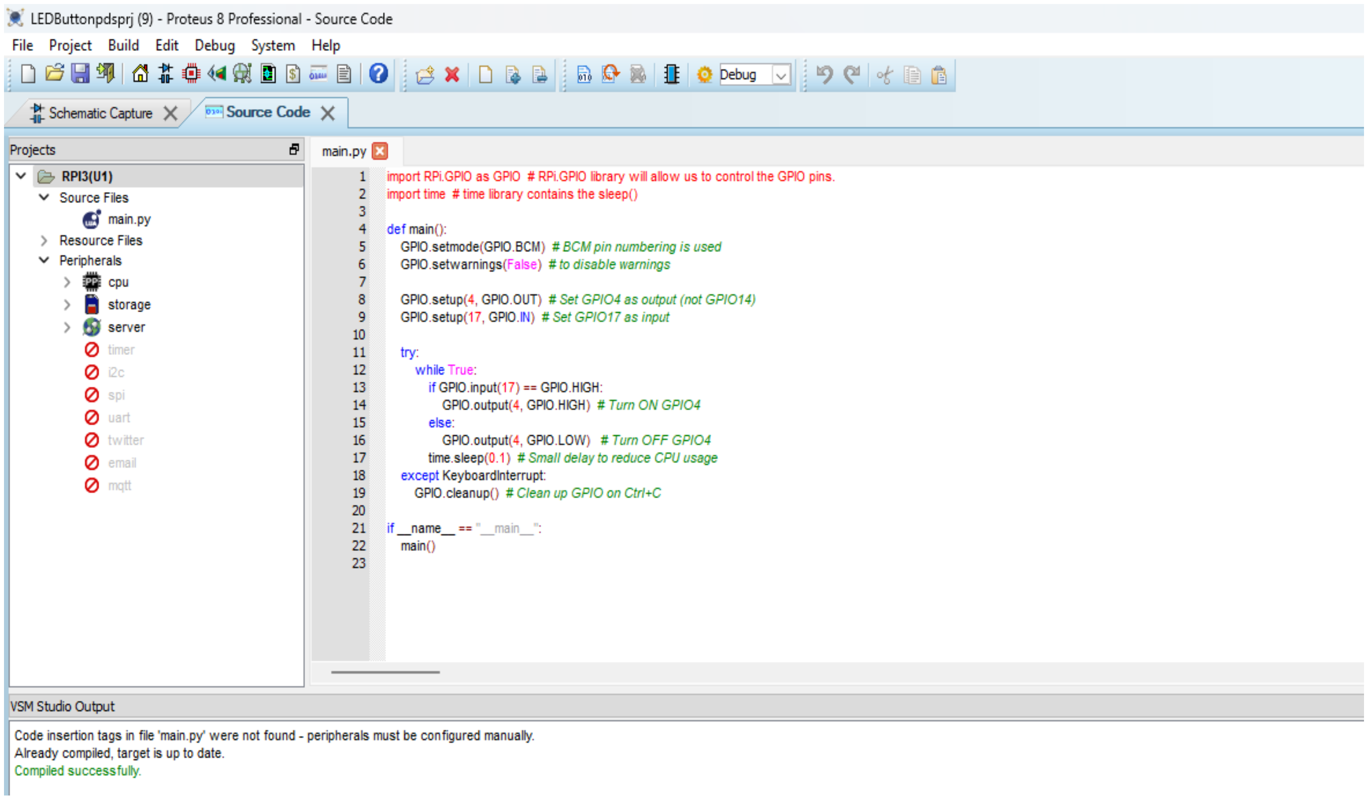
1. Opened Nano Editor:

   Command: $ nano blinkLED02.py, Opened a text editor to create a new Python file named blinkLED.py.

2. RPi.GPIO is used to control the GPIO pins on the Raspberry Pi.
3. time is used for adding delays with sleep ().
4. Used BCM (Broadcom**)** numbering to refer to GPIO14 pins by their chip number.
5. Prevented warning messages from displaying if GPIO was already configured earlier.
6. Configured pin 14 to send output signals (used to control the LED).
7. Sent a HIGH signal to GPIO14, which powered the LED and turned it on.
8. Displayed the text “LED is ON” in the terminal to indicate the LED is on.

9. Paused the program for 2 seconds**,** keeping the LED on.
10. Sent a LOW signal to GPIO14, which turned the LED off.
11. Displayed “LED is OFF” in the terminal to show that the LED has been turned off.
12. Saved the File in Nano: “Ctrl+X” then “Y” then “enter”.

☐ Pressed Enter to finalize the file name (nano BlinkLED01.py)



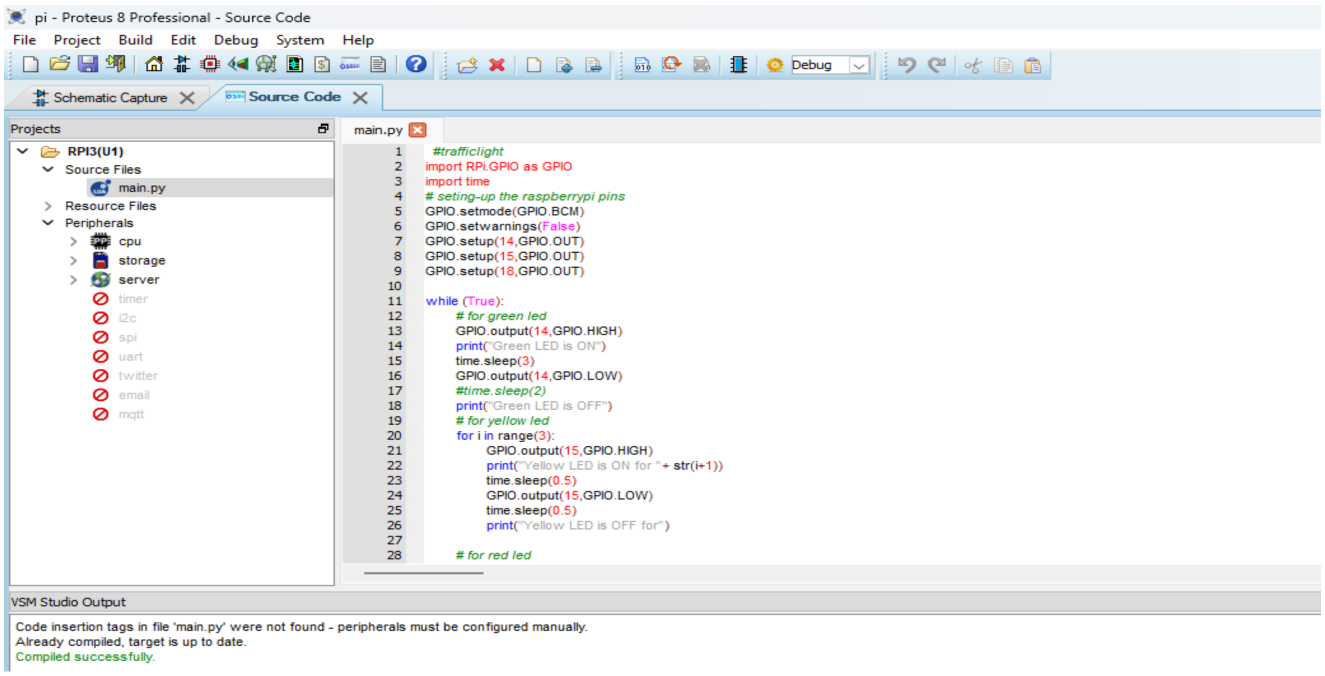
```python
import RPi.GPIO as GPIO  # RPi.GPIO library will allow us to control the GPIO pins.
import time  # time library contains the sleep()

def main():
   GPIO.setmode(GPIO.BCM)  # BCM pin numbering is used
   GPIO.setwarnings(False)  # to disable warnings

   GPIO.setup(4, GPIO.OUT)  # Set GPIO4 as output (not GPIO14)
   GPIO.setup(17, GPIO.IN)  # Set GPIO17 as input

   try:
      while True:
         if GPIO.input(17) == GPIO.HIGH:
            GPIO.output(4, GPIO.HIGH)  # Turn ON GPIO4
         else:
            GPIO.output(4, GPIO.LOW)   # Turn OFF GPIO4
         time.sleep(0.1)  # Small delay to reduce CPU usage
   except KeyboardInterrupt:
      GPIO.cleanup()  # Clean up GPIO on Ctrl+C

if __name__ == "__main__":
   main()
```

```
Code insertion tags in file 'main.py' were not found - peripherals must be configured manually.
Already compiled, target is up to date.
Compiled successfully.
```

Figure 14: LED controlling with a push button switch

1. Required libraries. RPi.GPIO lets you control the GPIO pins. time is used for the delay.
2. Defines the main() function.
3. BCM mode refers to the GPIO number rather than physical board pin numbers.
4. Disables warning messages for GPIO re-use.
5. GPIO4 is the output connected to the LED.
6. GPIO17 is the input connected to the push button.
7. An infinite loop (while True) keeps checking the button status.
8. If button (GPIO17) is pressed (reads HIGH), turn ON the LED.
9. Else, turn OFF the LED.
10. A short delay helps reduce CPU usage and avoids bouncing issues.
11. Ensures GPIO pins are reset to a safe state if the user manually stops the program with Ctrl+C.

1. This runs the main() function if the script is executed directly.



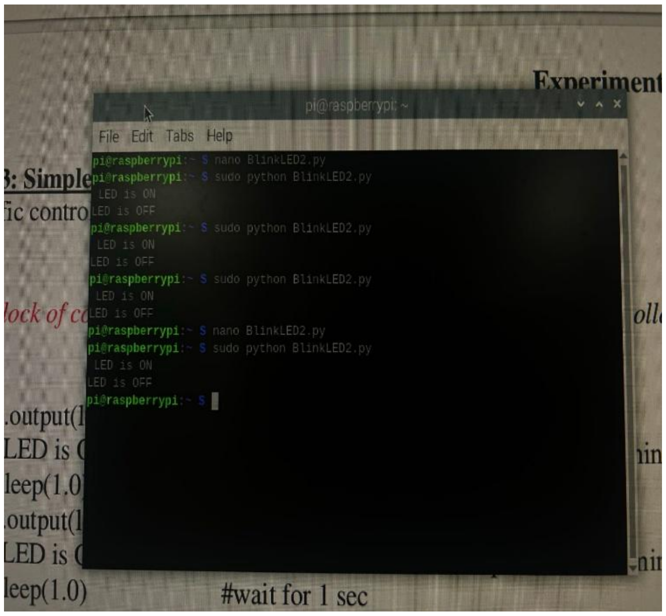
```
 #trafficlight
import RPi.GPIO as GPIO
import time
# seting-up the raspberrypi pins
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(14,GPIO.OUT)
GPIO.setup(15,GPIO.OUT)
GPIO.setup(18,GPIO.OUT)

while (True):
        # for green led
        GPIO.output(14,GPIO.HIGH)
        print("Green LED is ON")
        time.sleep(3)
        GPIO.output(14,GPIO.LOW)
        #time.sleep(2)
        print("Green LED is OFF")
        # for yellow led
        for i in range(3):
                GPIO.output(15,GPIO.HIGH)
                print("Yellow LED is ON for "+ str(i+1))
                time.sleep(0.5)
                GPIO.output(15,GPIO.LOW)
                time.sleep(0.5)
                print("Yellow LED is OFF for")

        # for red led
```

```
Code insertion tags in file 'main.py' were not found - peripherals must be configured manually.
Already compiled, target is up to date.
Compiled successfully.
```

Figure 15: Simple Traffic Control System.

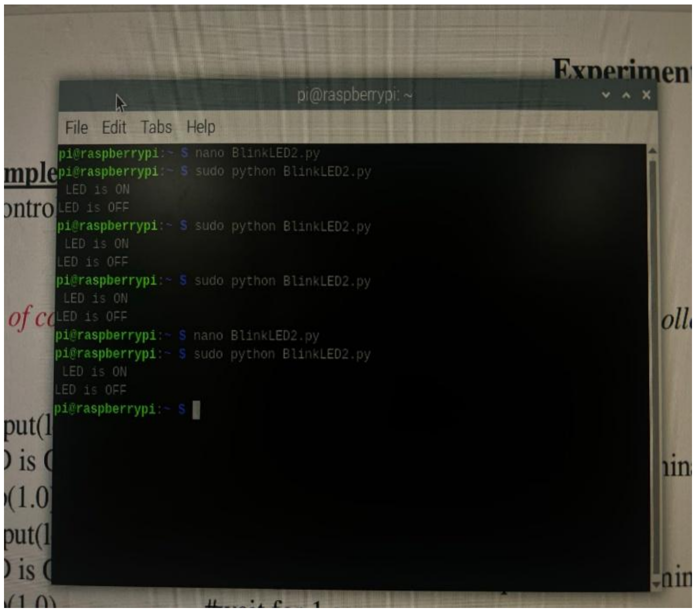
1. Infinite Loop with while(True):
   The while(True): statement creates an infinite loop. This is often used when we want a program (like blinking an LED) to run continuously until manually stopped.
2. LED ON with GPIO.output(ledpinnumber, GPIO.HIGH)
   This line sends a HIGH signal (3.3V) to the specified GPIO pin, turning the LED ON. The ledpinnumber should be previously defined to reference the correct GPIO pin.
3. Status Message – "LED is ON"
   The line print 'LED is ON' (Python 2 syntax) outputs a message to the terminal so the user knows the LED is turned on.
4. Pause using time.sleep(1.0)
   Introduces a delay of 1 second. This keeps the LED ON for a visible duration before switching it OFF.
5. LED OFF with GPIO.output(ledpinnumber, GPIO.LOW)
   Sends a LOW signal (0V) to the GPIO pin, turning the LED OFF.
6. Status Message – "LED is OFF"
   The print 'LED is OFF' message notifies the user that the LED has been turned off.
7. Loop Control with break
   A break statement inside an if condition can be used to exit the infinite loop.
8. This prints "Raspberry Pi" 20 times before breaking out of the loop.

Hardware Output from Raspberry Pi 3 - Model B:



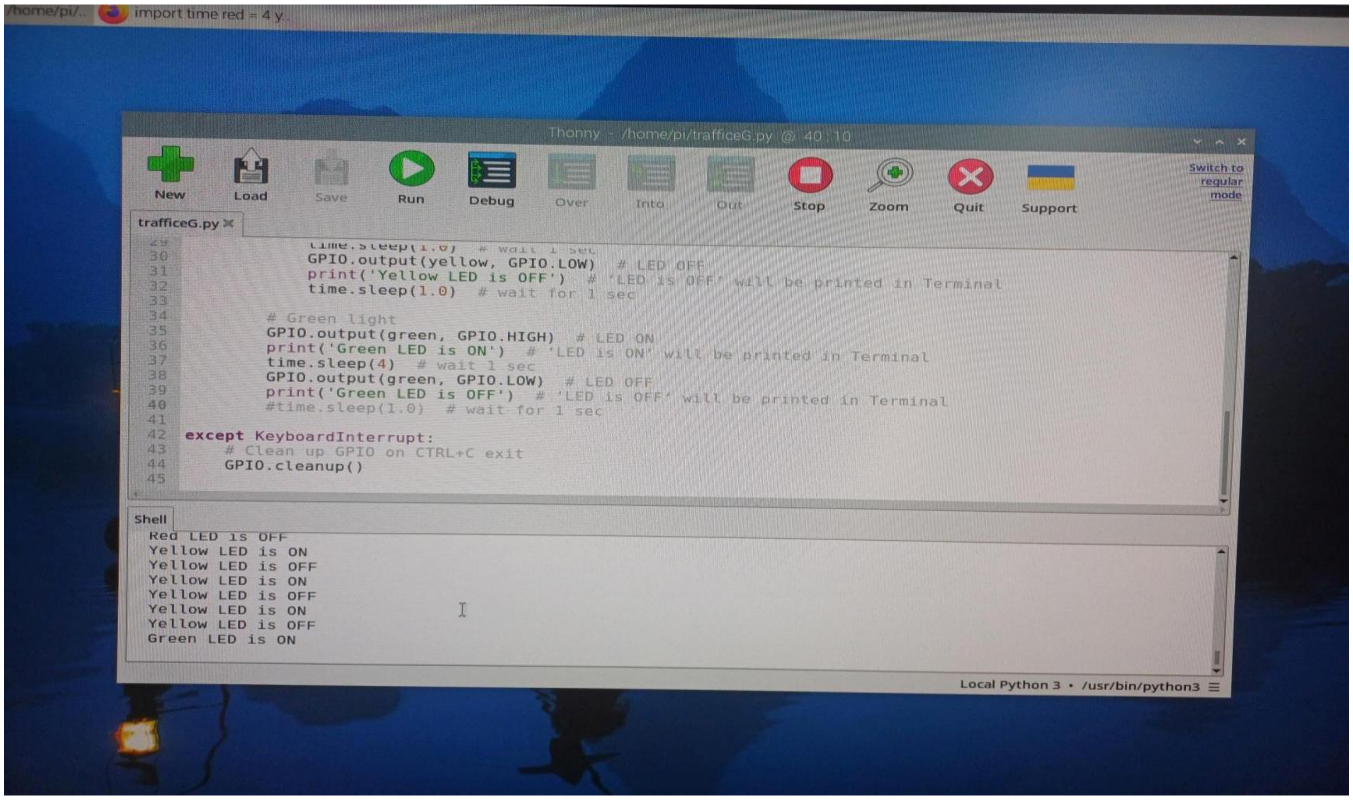
```
pi@raspberrypi:~ $ nano BlinkLED2.py
pi@raspberrypi:~ $ sudo python BlinkLED2.py
 LED is ON
LED is OFF
pi@raspberrypi:~ $ sudo python BlinkLED2.py
 LED is ON
LED is OFF
pi@raspberrypi:~ $ sudo python BlinkLED2.py
 LED is ON
LED is OFF
pi@raspberrypi:~ $ nano BlinkLED2.py
pi@raspberrypi:~ $ sudo python BlinkLED2.py
 LED is ON
LED is OFF
pi@raspberrypi:~ $
```

Figure 16: LED Blinking Output



```
pi@raspberrypi:~ $ nano BlinkLED2.py
pi@raspberrypi:~ $ sudo python BlinkLED2.py
 LED is ON
LED is OFF
pi@raspberrypi:~ $ sudo python BlinkLED2.py
 LED is ON
LED is OFF
pi@raspberrypi:~ $ sudo python BlinkLED2.py
 LED is ON
LED is OFF
pi@raspberrypi:~ $ nano BlinkLED2.py
pi@raspberrypi:~ $ sudo python BlinkLED2.py
 LED is ON
LED is OFF
pi@raspberrypi:~ $
```

Figure 17: LED controlling with a push button



```
            time.sleep(1.0)  # wait 1 sec
            GPIO.output(yellow, GPIO.LOW)  # LED OFF
            print('Yellow LED is OFF')  # 'LED is OFF' will be printed in Terminal
            time.sleep(1.0)  # wait for 1 sec

        # Green light
        GPIO.output(green, GPIO.HIGH)  # LED ON
        print('Green LED is ON')  # 'LED is ON' will be printed in Terminal
        time.sleep(4)  # wait 1 sec
        GPIO.output(green, GPIO.LOW)  # LED OFF
        print('Green LED is OFF')  # 'LED is OFF' will be printed in Terminal
        #time.sleep(1.0)  # wait for 1 sec

except KeyboardInterrupt:
    # Clean up GPIO on CTRL+C exit
    GPIO.cleanup()
```

```
Red LED is OFF
Yellow LED is ON
Yellow LED is OFF
Yellow LED is ON
Yellow LED is OFF
Yellow LED is ON
Yellow LED is OFF
Green LED is ON
```

Figure 18: LED controlling traffic control system using RED, YELLOW, and GREEN LEDs.

<u>Simulation Output Results:</u>

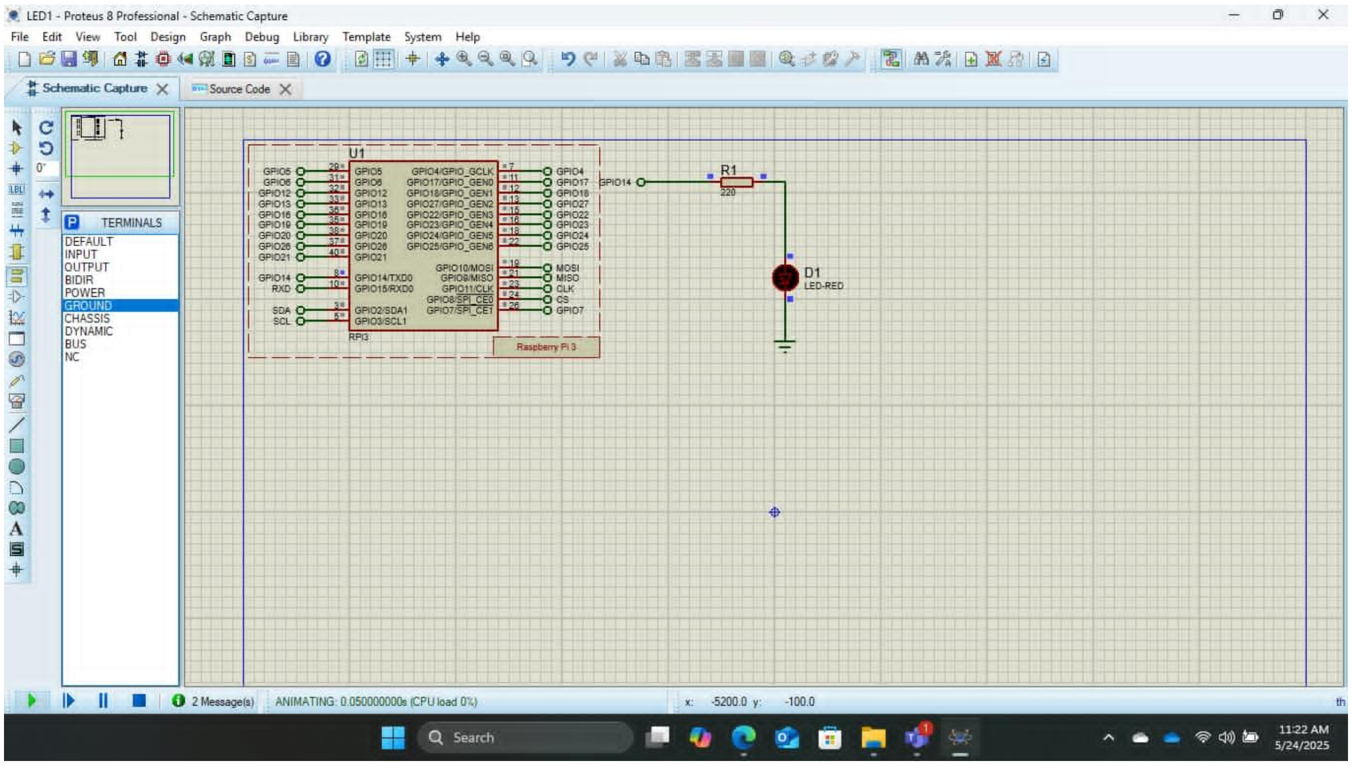


Figure 19: LED Blinking Simulation LED is OFF

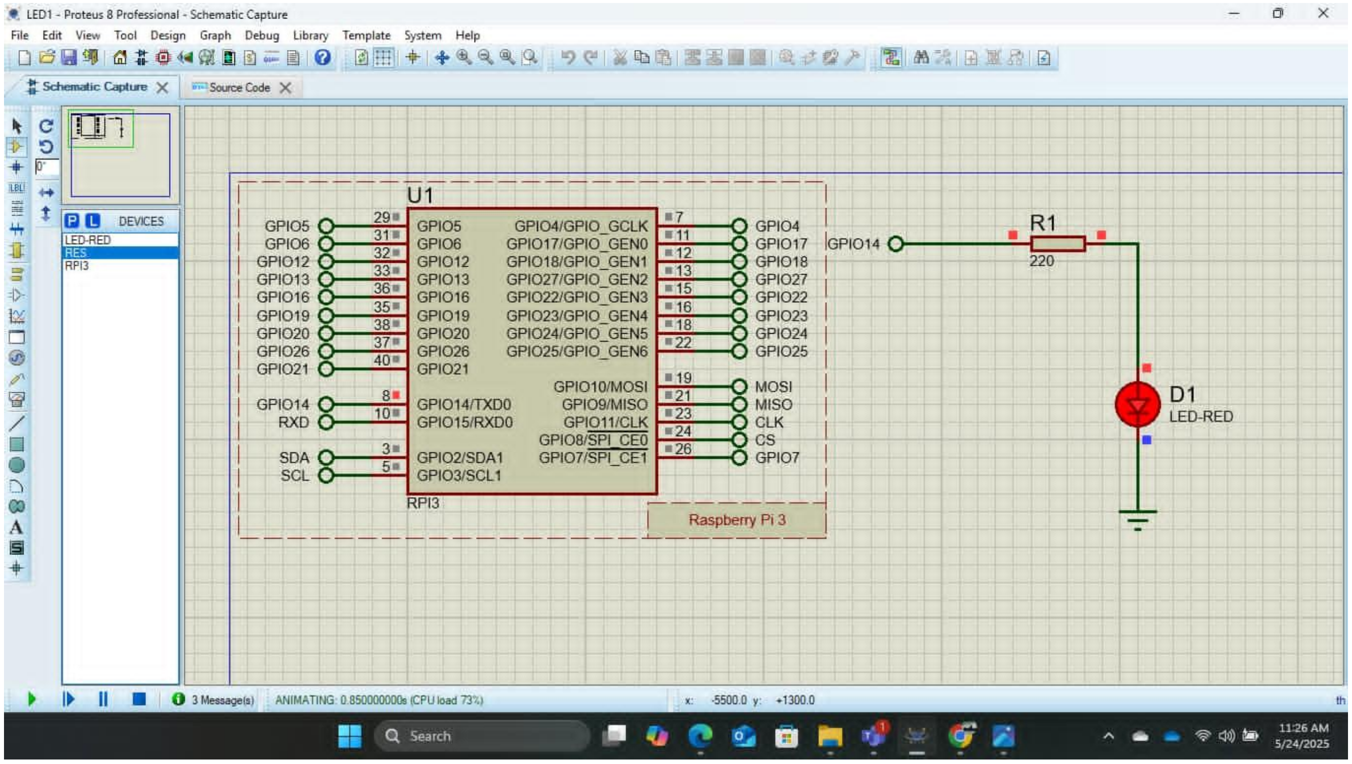


Figure 20: LED Blinking Simulation LED is ON

Simulation procedure of LED Blinking: Simulation methodologies

1. Open Proteus 8 Professional.
2. Start a New Project and Go to File → New Project and follow the wizard to set up a blank schematic.
3. Add Components

   Click on the P (Pick Devices) button and add the following:

   Raspberry Pi 3 (model: RPI3)

   Green LED

   Resistor (value: 220Ω)

   Ground terminal

4. Connect the Circuit

   GPIO14 (Pin 8) of Raspberry Pi → Resistor (220Ω)

   Resistor → Anode of LED

   Cathode of LED → Ground

5. Write Python Code

   Go to the Source Code tab.

   Write the following code in main.py of Lab Manual

6. Check the bottom Output window for "Compiled successfully".
7. Start the Simulation

   Press the Play button.

   Observe:

   The LED will turn ON for 2 seconds.

   Then turn OFF for 2 seconds.

   Repeats in a loop.

Simulation procedure of LED controlling with a push button switch:

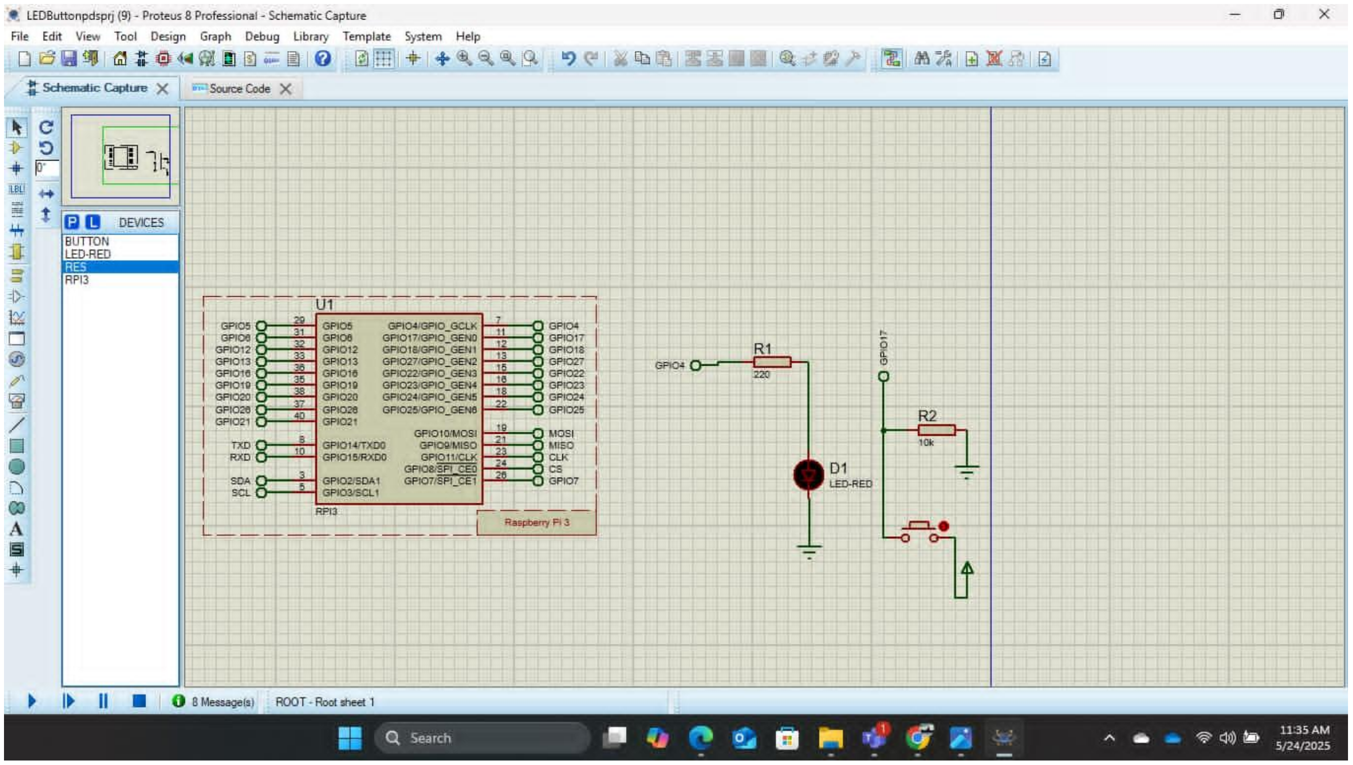


Figure 21: LED controlling with a push button switch Simulation LED is OFF

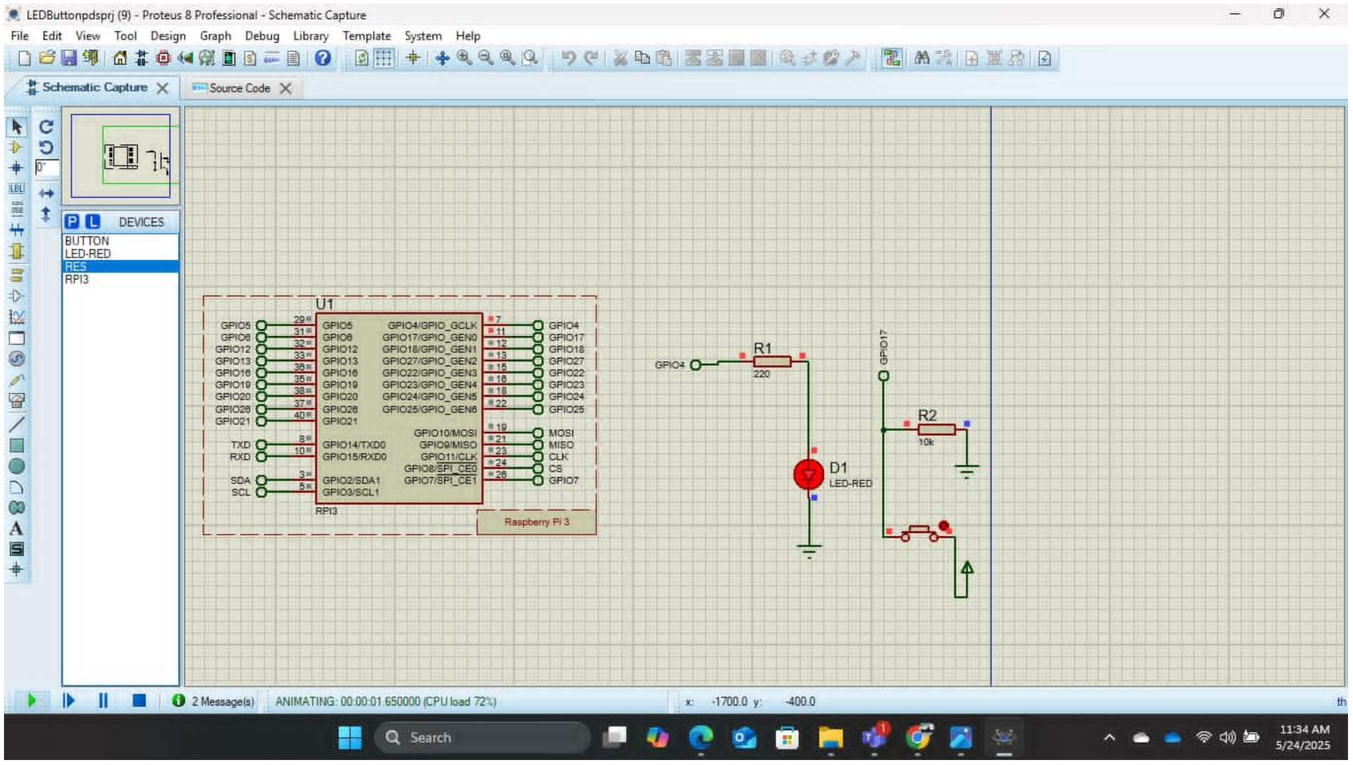

LEDButtonpdsprj (9) - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Schematic Capture
Source Code
DEVICES
BUTTON
LED-RED
RES
RPI3
U1
GPIO4
R1
220
D1
LED-RED
GPIO17
R2
10k
Raspberry Pi 3
RPI3
2 Message(s)
ANIMATING: 00:00:01.650000 (CPU load 72%)

Figure 22: LED controlling with a push button switch Simulation LED is ON

Simulation methodologies

1. Open a new Project as like First Simulation
2. Add Components

   Click on the P (Pick Devices) button and add the following:

   Raspberry Pi 3 (model: RPI3)

   Red LED

   Resistor (value: 220Ω and 10k)

   Button and 5V Power

   Ground terminal and Default as GPIO4 for LED and GPIO17 for Button

3. Connect the Components than open Source Code Write the code helping Google Because lab manual code did not run.
4. Run the Simulation of play button and Press the Button than LED turn on.

Simulation procedure of Simple Traffic Control System:

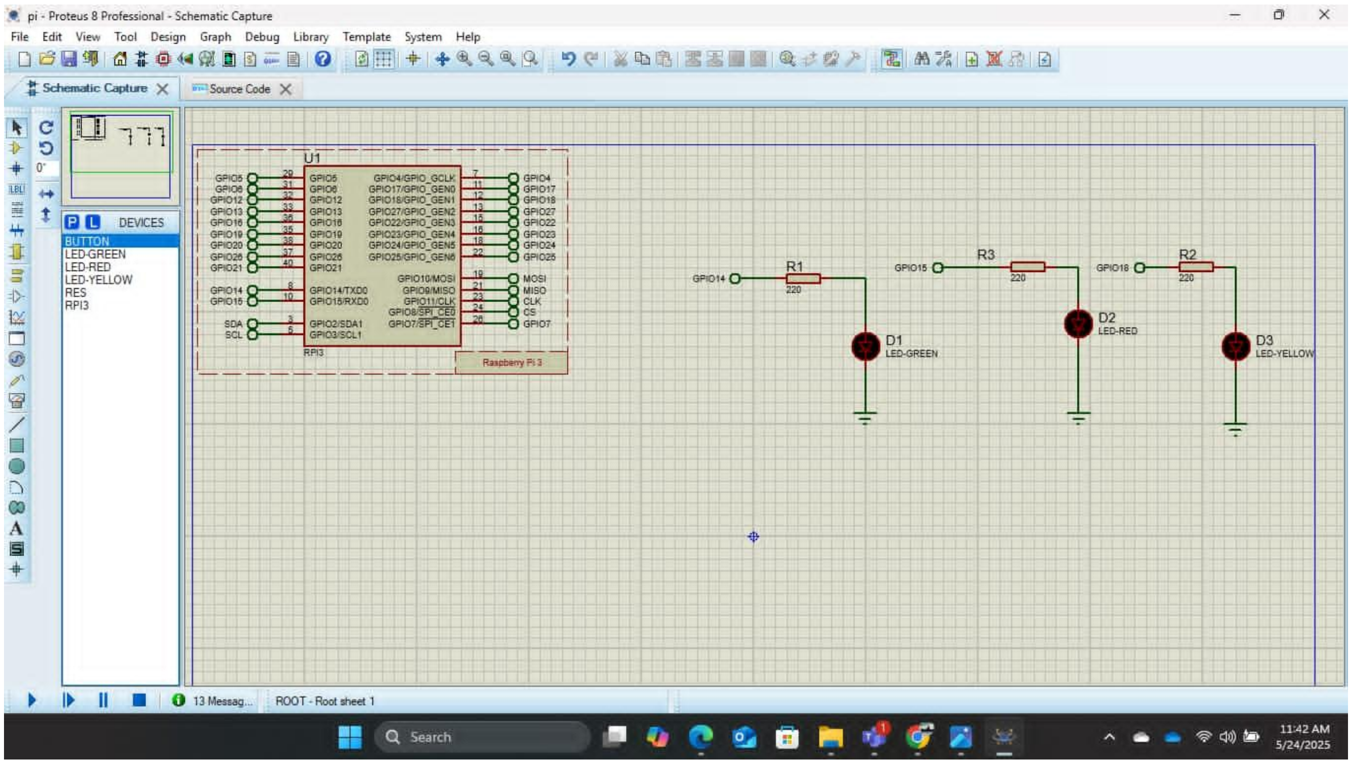


Figure 23: Simple Traffic Control System Simulation LEDs are OFF

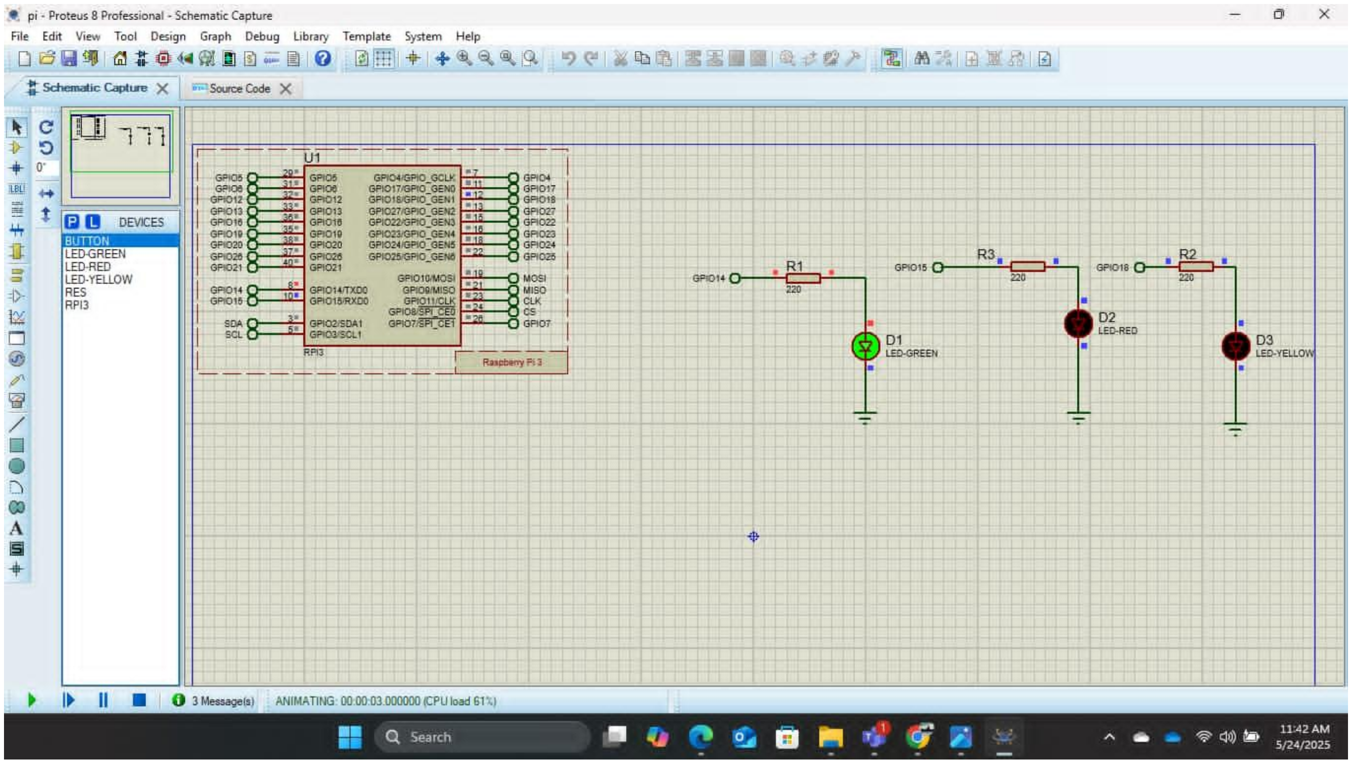


Figure 24: Simple Traffic Control System Simulation Green LED is ON

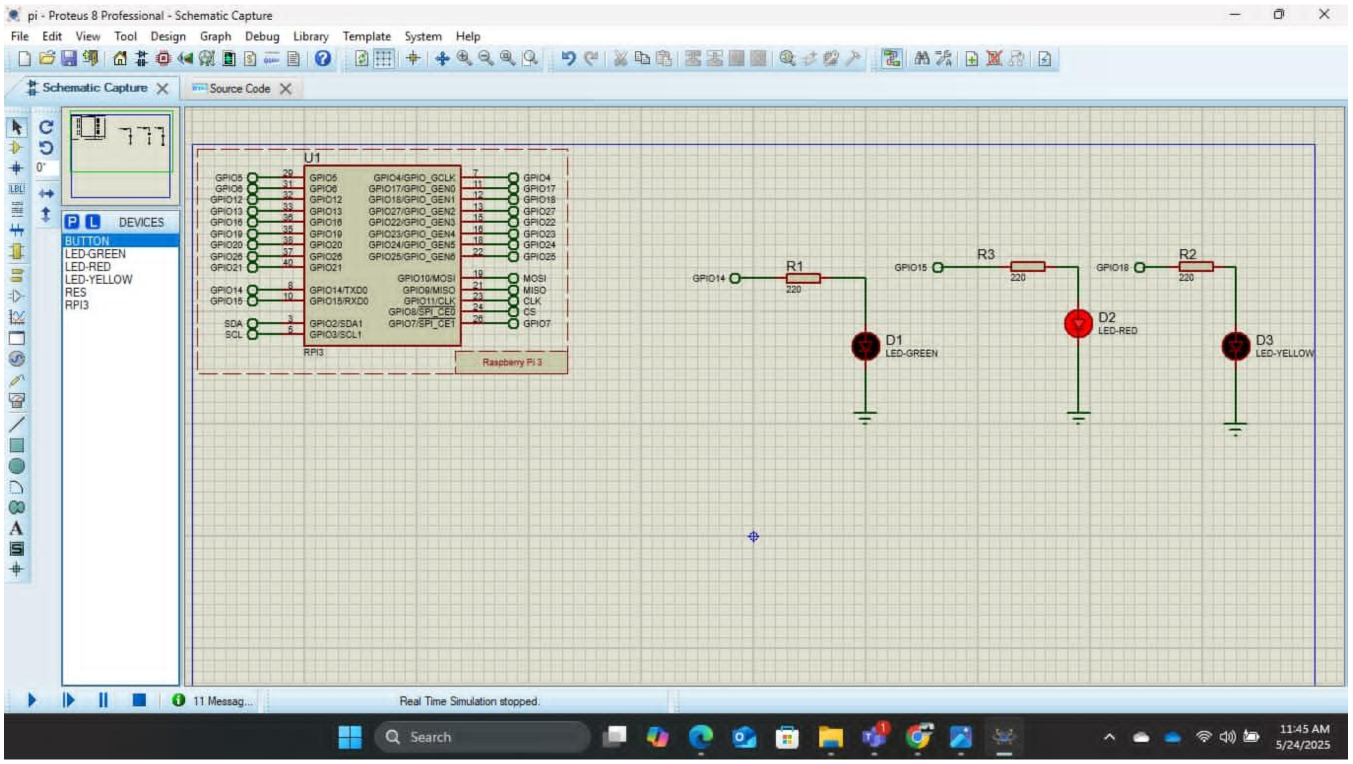


Figure 25: Simple Traffic Control System Simulation Red LED is ON

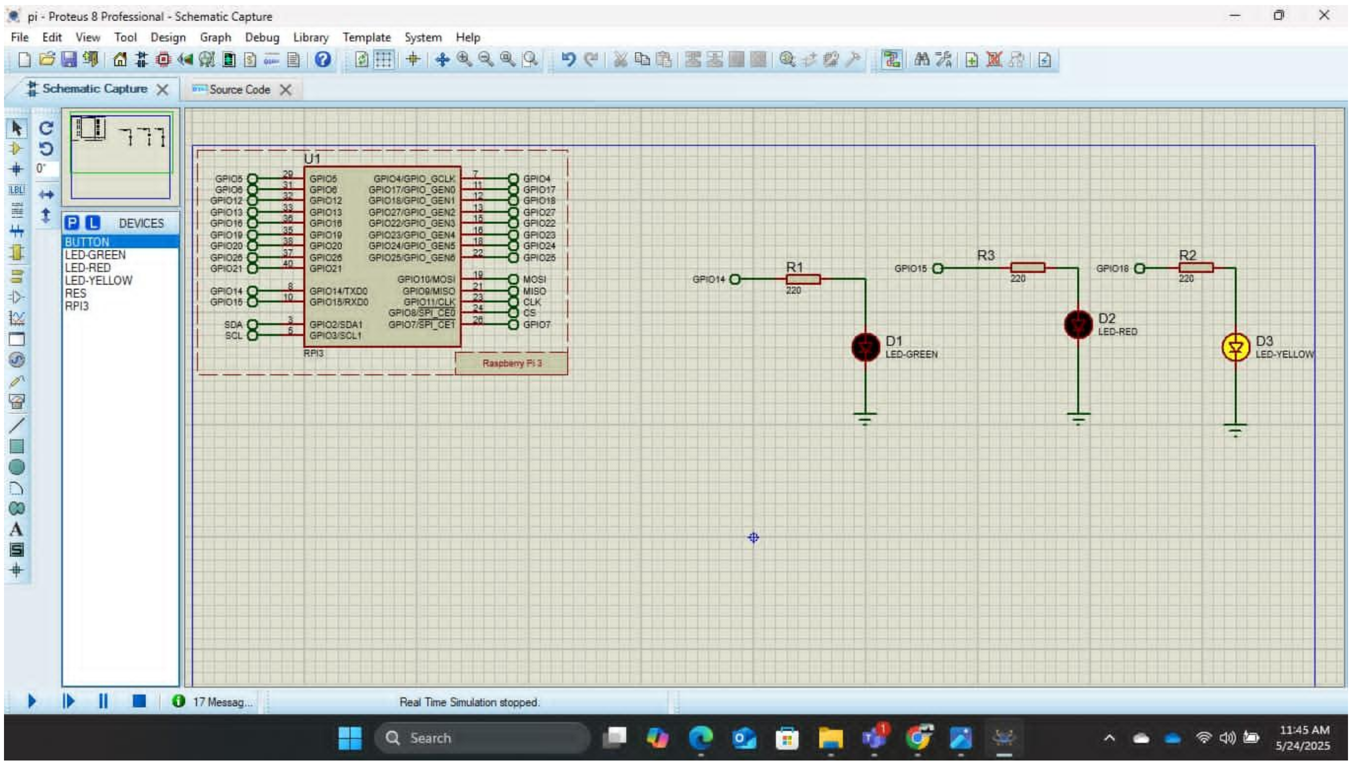


Figure 26: Simple Traffic Control System Simulation Yellow LED is ON

Simulation methodologies

1. Open a new Project as like First Simulation
2. Add Components

   Click on the P (Pick Devices) button and add the following:

   Raspberry Pi 3 (model: RPI3)

   Red LED, Green LED, Yellow LED

   3 Resistor (value: 220Ω)

   Ground terminal and Default GPIO14, GPIO15, GPIO18

3. Connect the Components than open Source Code Write the code from lab manual.
4. Run the Simulation of play button and Press the Button than LED turn on.

Discussion:

1. The experiment was successful.

2. All cables & wires must be connected properly according to Figure

3. The LEDs & Switches used in the experiment must be connected properly according to their operating terminals

5. Virtual Terminals in the simulation should be connected properly.

6. The sketches used in the experiment must be error free and should be for the correct microcontroller board.

7. The equipment used in the experiments is relatively fragile and must he handled carefully.

Reference(s):

[1] https://www.raspberrypi.org/learn/, accessed on 2nd July 2023.