AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Engineering

Lab Report

Experiment # 10
Experiment Title: Familiarization with the Raspberry Pi

Date of Perform:	Date Month 2023	**Date of Submission:**	7 August 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**		

Group #

Sl No	Name	ID	PROGRAM	SIGNATURE

Faculty use only

FACULTY COMMENTS	**Marks Obtained**	
	Total Marks	

Table of Contents

Experiment Title: Familiarization with the Raspberry Pi

Objectives:

The objectives of this experiment are to-

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

Equipment List:

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

Circuit Diagram:

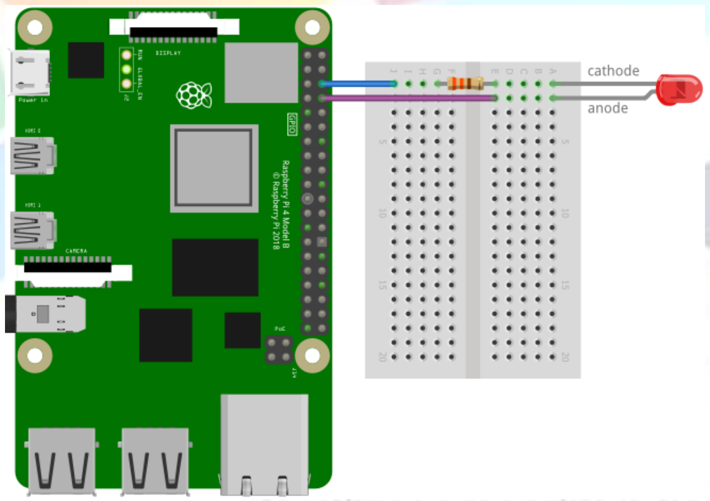

Fig.1 Experimental setup of the circuit for the LED blinking program using Raspberry Pi

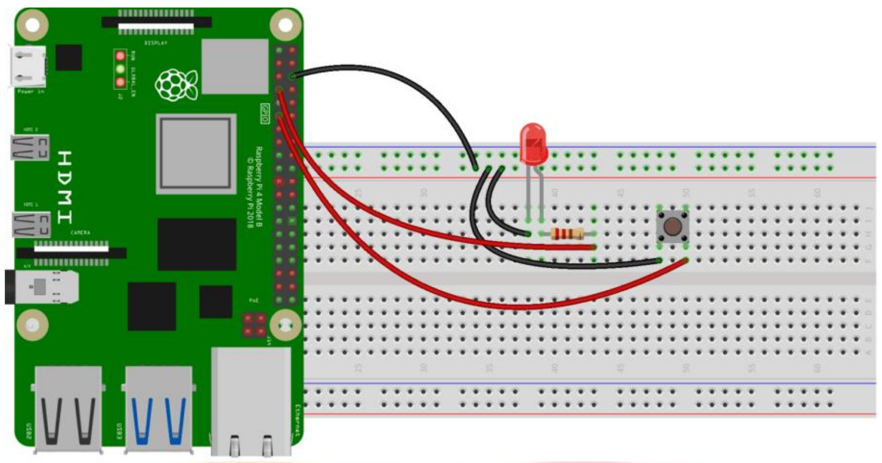
Fig. 2 Experimental setup of the circuit for the LED controlling program with a button switch using Raspberry Pi

Code/Program:
The following is the code for the implementation of the LED blinking program with switch and without switch using Raspberry Pi with the necessary code explanation:

<u>Part 1: LED blink program without Switch:</u>

```
$ nano blinkLED.py              # to create a text file under the name blinkLED write this at the command
line
import RPi.GPIO as GPIO      # RPi.GPIO library will allow us to control the GPIO pins.
import time                              # time library contains the sleep ()
GPIO.setmode(GPIO.BCM)   # BCM pin numbering is used
GPIO.setwarnings(False)      # to disable warnings
GPIO.setup(14, GPIO.OUT) # to set GPIO14 as an output.
GPIO.output(14, GPIO.HIGH) # to specify the GPIO 14 as HIGH
print "LED is ON"                  # show message to Terminal.
time.sleep(2)                          # for two seconds
GPIO.output(14, GPIO.LOW) # to specify the GPIO 14 as LOW
print "LED is OFF"             # show message to Terminal.
```

<u>Part 2: LED blink program With Switch:</u>

```
from gpiozero import LED
#imports LED functions from gpiozero library
from gpiozero import Button
#imports Button functions from gpiozero library

led = LED(4)
#declare the GPIO pin 4 for LED output and store it in led variable
button = Button(17)
#declare the GPIO pin 17 for Button output and store it in button variable
```

```
while True:
#initiated an infinite while loop()
	Button.wait_for_press()
#use the built-in function of the button to wait till press
	led.on()
#turn on the led
	Button.wait_for_release()
#use the built-in function of button to wait till release
	led.off()
#turn off the led
```

Part 3: Simple Traffic Control System:

```
# Modules
from goto import *
import time
import var
import pio
import resource
import RPi.GPIO as GPIO

# Peripheral Configuration Code (do not edit)
#---CONFIG_BEGIN---
import cpu
import FileStore
import VFP

def peripheral_setup () :
    # Peripheral Constructors
    pio.cpu = cpu.CPU()
    pio.storage = FileStore.FileStore()
    pio.server = VFP.VfpServer()
    pio.storage.begin()
    pio.server.begin(0)

    # GPIO Setup
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    GPIO.setup(12, GPIO.OUT)  # Use GPIO 12 for RED LED
    GPIO.setup(16, GPIO.OUT)  # Use GPIO 16 for YELLOW LED
    GPIO.setup(20, GPIO.OUT)  # Use GPIO 20 for GREEN LED

# Install interrupt handlers
def peripheral_loop () :
    pio.server.poll()
    # RED LED ON
    GPIO.output(12, GPIO.HIGH)
    print('RED LED is ON')
    time.sleep(3.0)  # Wait for 3 seconds

    # RED LED OFF, YELLOW LED ON
    GPIO.output(12, GPIO.LOW)
    GPIO.output(16, GPIO.HIGH)
    print('RED LED is OFF, YELLOW LED is ON')
    time.sleep(1.0)  # Wait for 1 seconds

    # YELLOW LED OFF, GREEN LED ON
    GPIO.output(16, GPIO.LOW)
    GPIO.output(20, GPIO.HIGH)
    print('YELLOW LED is OFF, GREEN LED is ON')
    time.sleep(3.0)  # Wait for 3 seconds

    # GREEN LED OFF
    GPIO.output(20, GPIO.LOW)
    print('GREEN LED is OFF')
    time.sleep(1.0)  # Wait for 1 seconds

#---CONFIG_END---
# Main function
def main () :
    # Setup
    peripheral_setup()

    # Infinite loop for repeating the traffic light simulation
    while True:
        peripheral_loop()

# Command line execution
if __name__ == '__main__' :
    main()
```

Hardware Output Results:

Here is the hardware implementation of the LED blinking program with switch and without switch and the necessary explanation of the implementation:

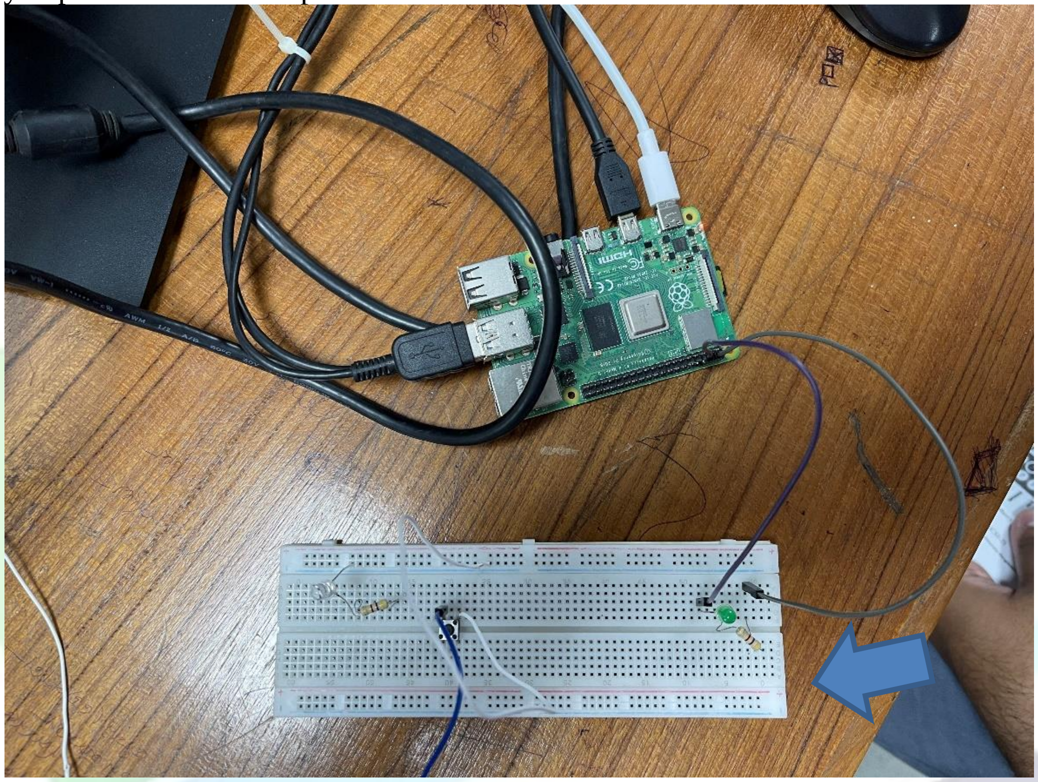
Fig. 3 Hardware implementation of LED blinking without switch

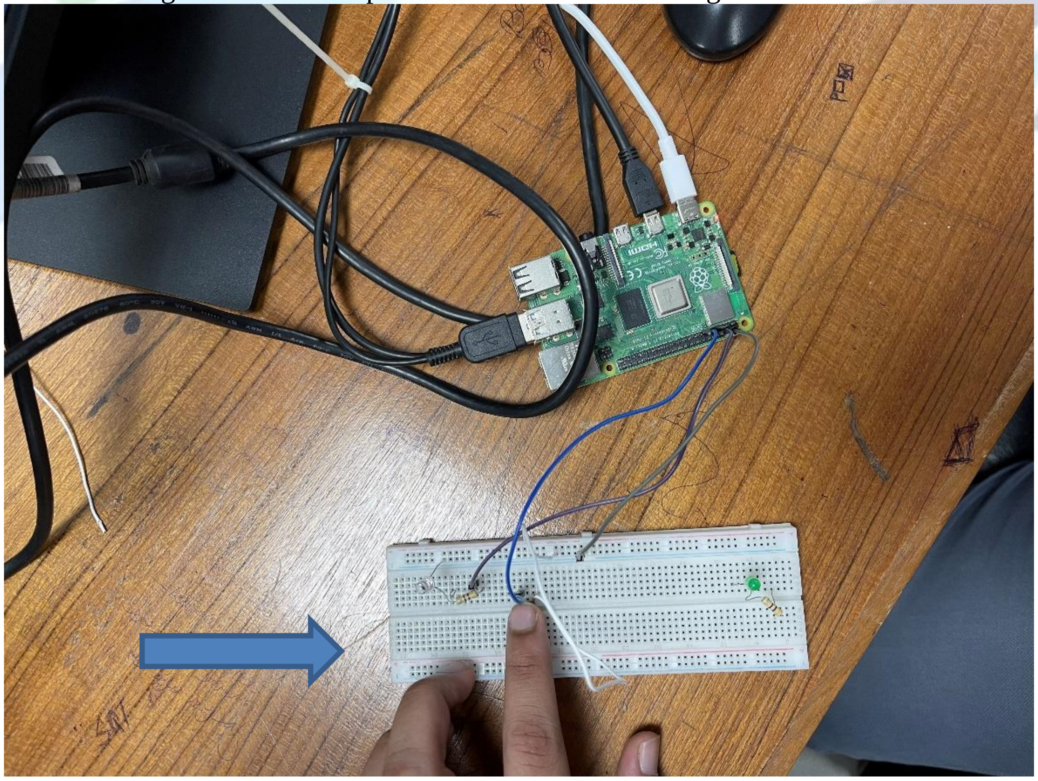
Fig. 4 Hardware implementation of LED blinking with switch

Explanation:
In this experiment, a LED and a register were connected with the Raspberry Pi at GPIO4 pin and to ground the connection it connected to GND pin of the Raspberry Pi. For making a LED test with button, the button was connected with the GPIO17 pin and the LED was connected with GPIO4 pin. Both of them were connected with the GND pin of the Raspberry Pi.

Experimental Output Results:
Here is the hardware implementation of the LED blinking program without a switch and the necessary explanation of the implementation:

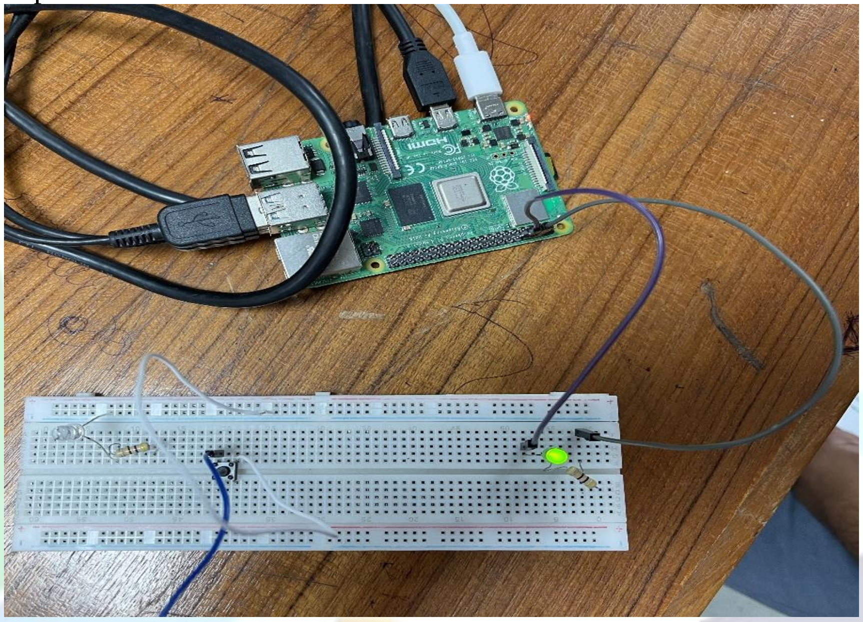
Fig.5 LED ON during the Led blink program without switch on Raspberry Pi

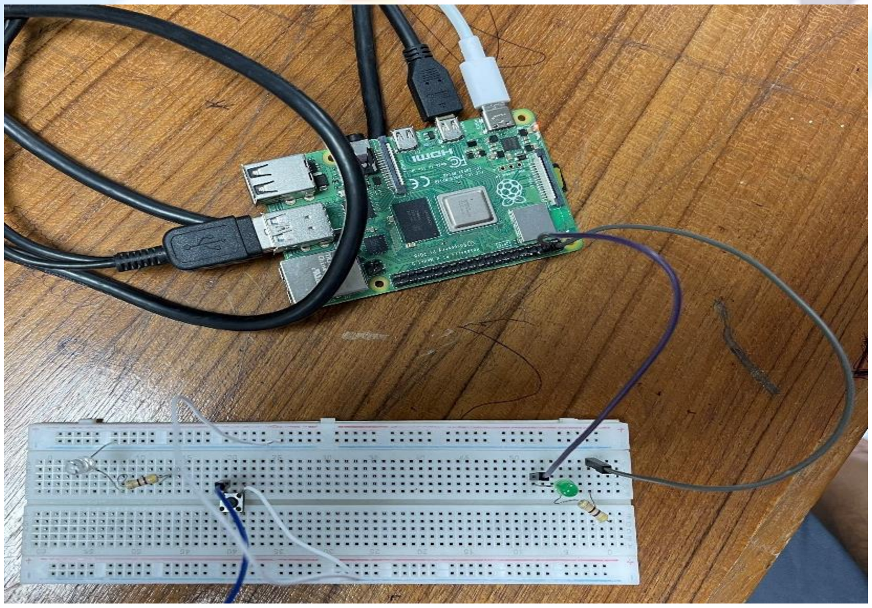
Fig.6 LED ON during Led blink program without switch on Raspberry Pi

Explanation: In this experiment, when the circuit was built accordingly and code was ingested in the

Raspberry Pi, the LED blinked after each 1 second. This meant that the GPIO4 pin was high for 1 second and low for 1 second. This duration was set using a delay function which can increase or decrease the delay of the time of blinking.

Here is the hardware implementation of the LED blinking program with switch and the necessary explanation of the implementation:

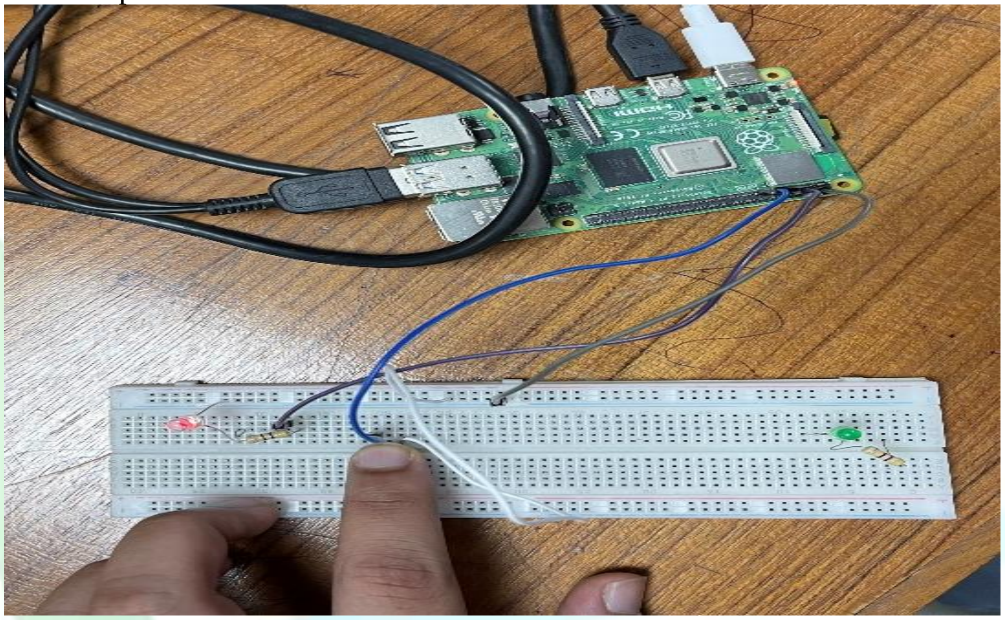

Fig.7 LED ON when the button is pressed on Led blink program with switch on Raspberry Pi

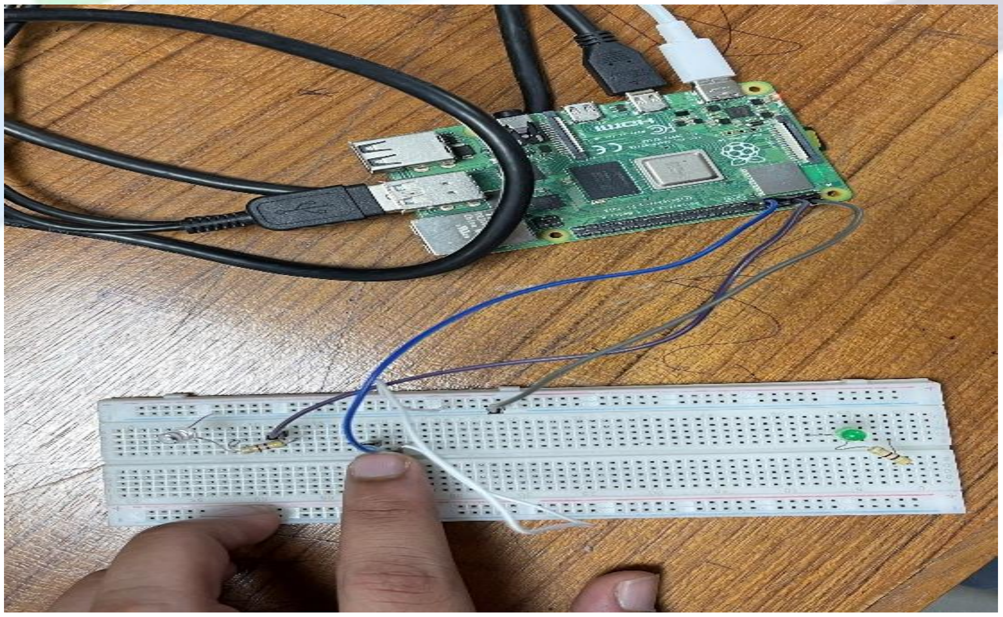

Fig.8 LED OFF when the button is not pressed on Led blink program with switch on Raspberry Pi

Explanation: In this experiment, when the button was pressed, the GPIO 14 pin received a HIGH input which caused the program to let the Raspberry Pi to provide a HIGH output on the GPIO 4 pin that made the LED turn on. When the button was not pressed, the GPIO 14 pin was LOW and caused the GPIO 4 pin to provide a LOW output causing the LED to be OFF.

Simulation Output Results:
Here is the simulation implementation of the LED blinking program with switch, without switch and the simple traffic control system and the necessary explanation of the implementation:

Part 1: LED blink program without Switch:

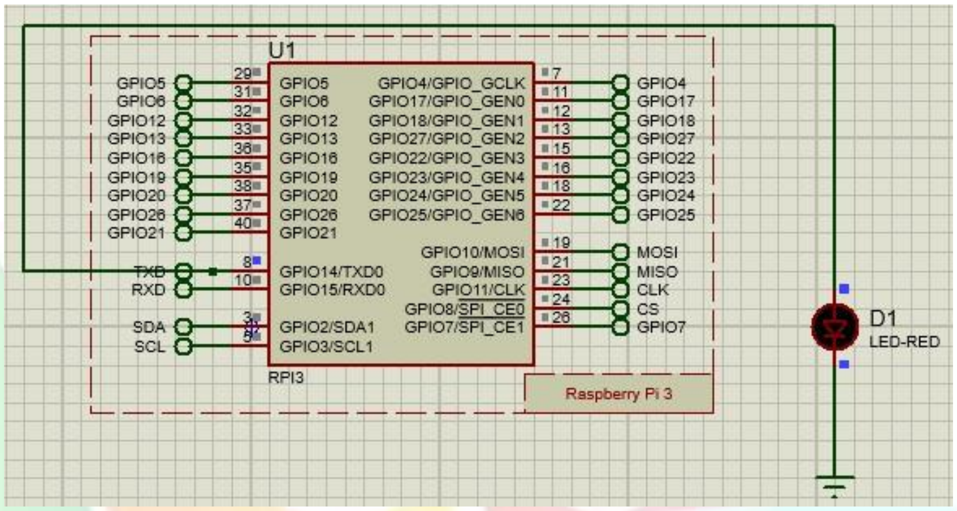

Fig.9 Led blink program without switch (LED is OFF)

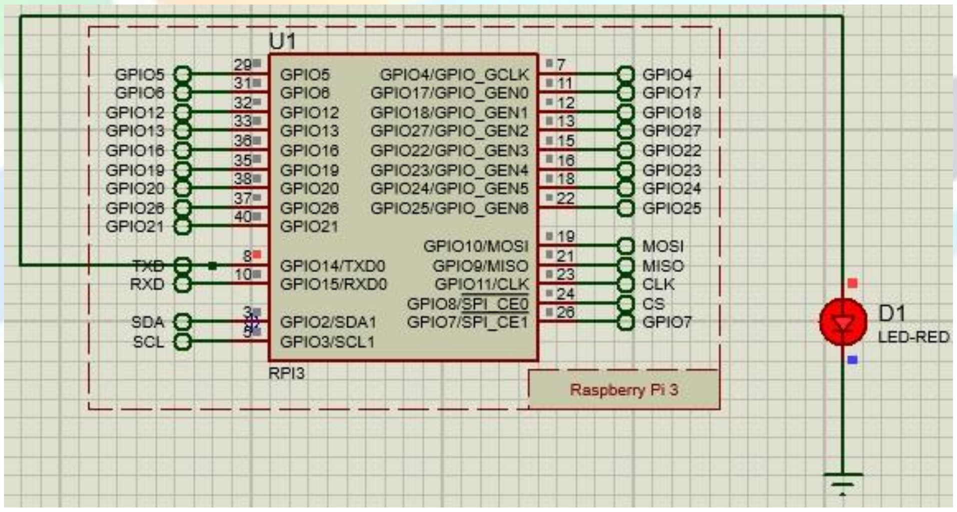

Fig.10 Led blink program without switch (LED is ON)

Part 2: LED blink program without Switch:

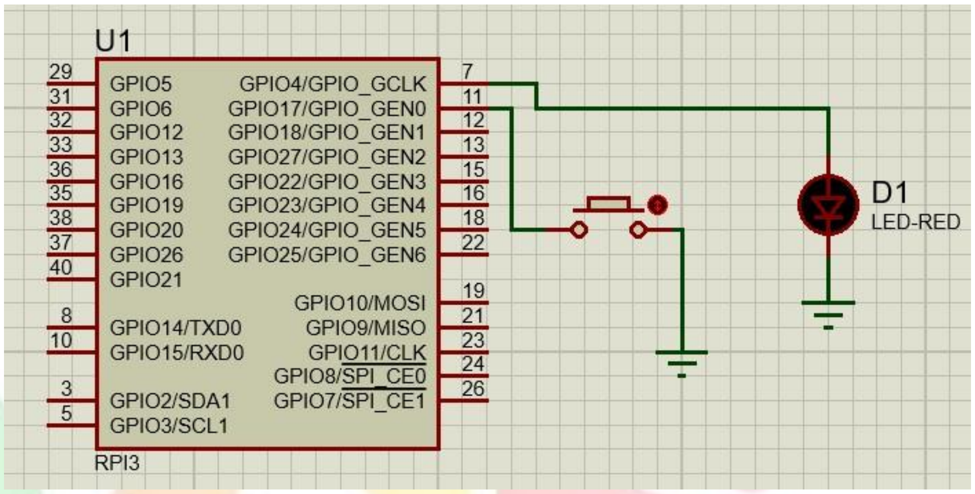

Fig.11 Led blink program with switch (LED is OFF)

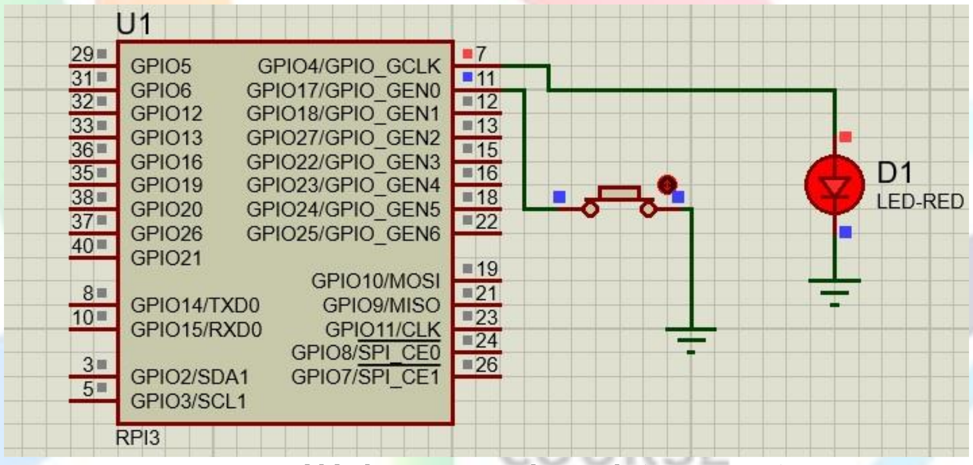

Fig.12 Led blink program with switch (LED is ON)

Part 3: Simple Traffic Control System:

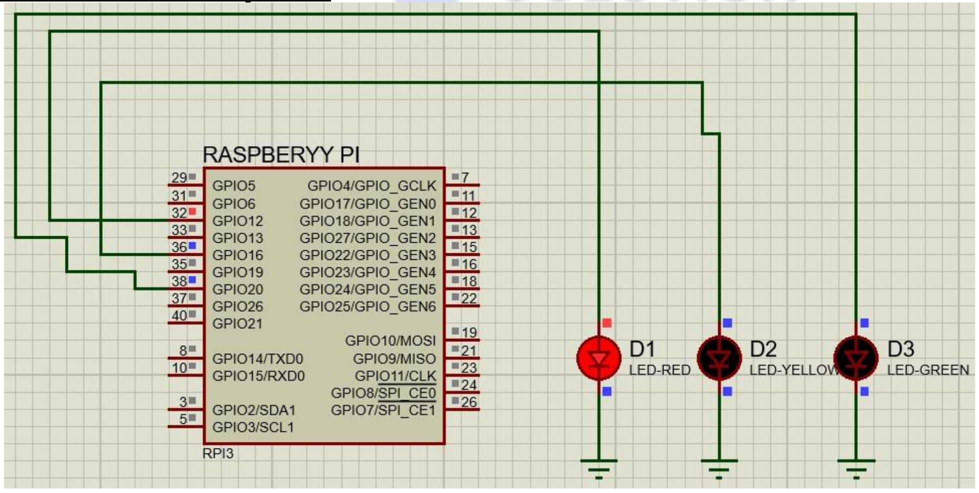

Fig.13 Simple Traffic Control System (Red LED is ON)

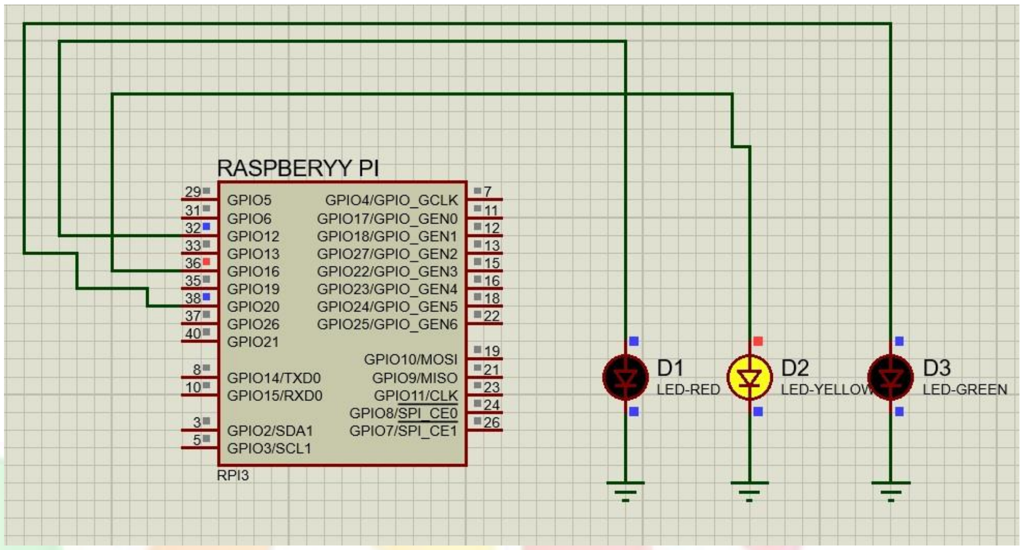

Fig.14 Simple Traffic Control System (Yellow LED is ON)

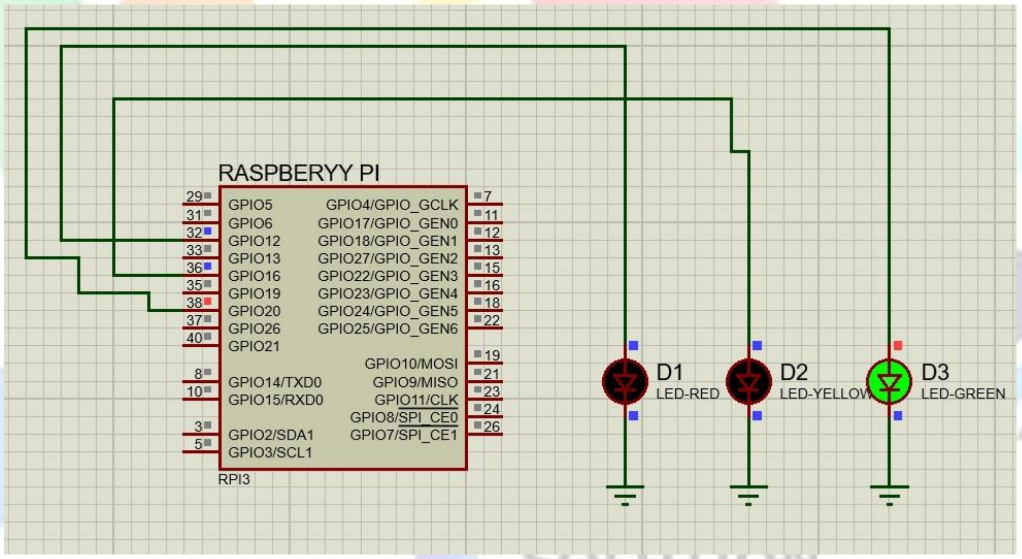

Fig.15 Simple Traffic Control System (Green LED is ON)

Explanation:

In this experiment, the circuits were developed on the Proteus Simulation software. All the components were loaded onto the developing canvas and the circuits were made accordingly. Afterward, the codes were written in Python. Then the Python scripts were loaded onto the Raspberry Pi. Then the simulations were conducted and the results were observed accordingly.

Discussion:

In this experiment, we learn how to use Raspberry Pi to create system which can control LED light and build a traffic management system. Through this experiment, we managed to have better understanding of Raspberry pi. We learn how we can import GPIO library and use it for our need in the simulator. We got familiar with Linux environment. The whole task was performed in an online simulator, which gives us the knowledge to run a similar Raspberry Pi task on online simulator. In future, this experiment will help us to build similar systems such as smart irrigation systems, gas leakage detector etc. using Raspberry pi. After that, the system was operated, and the systems operations were observed. The results were shown accordingly based on the formulas that were set on the code. All the results that were observed were carefully noted down for further evaluation. A similar system was developed on the simulation softwares like Proteus. The results that were obtained on the physical operation were evaluated with simulated outcomes. There were some Led Blinking discrepancies that were observed. The distance that was generated on the simulation's serial monitor were a bit different compared to the ones that were observed on the physical testing. This might be caused due to minor system and human errors. This caused the inconsistencies in the values of the serial monitor. Moreover, the detecting rate of the system in the physical environment and simulation virtual environment were a bit different. This was ruled as normal as human error was general in the physical world. From the observation it can be said that after both hardware and software implementation showed the expected outcomes, and the experimental objectives was achieved.

References:

1) Raspberry pi datasheet.
2) https://www.raspberrypi.org/documentation/linux/

3) *https://www.raspberrypi.org/documentation/remote-access/ssh/*
4) AIUB Microprocessor and Embedded Systems Lab Manual 10