AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Engineering

Lab Report

Experiment # 10

Title: Familiarization with the Raspberry Pi.

Date of Perform	9 April 2023	**Date of Submission**	30 April 2023
Course Title	MICROPROCESSOR AND EMBEDDED SYSTEMS		
Course Code	COE3104	**Section**	C
Semester	Spring 2022-23	**Degree Program**	BSc in CSE & EEE
Course Teacher	**RICHARD VICTOR BISWAS**		

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 # 01

Sl No	Name	ID	Department
1	Zaid Amin **Rawfin**	20-42459-1	CSE
2	Abzana Sultan **Ira**	20-41973-1	CSE
3	MD. Sanjid Bin Karim **Sezan**	19-41702-3	CSE
4	Abdur Rahman **Swapnil**	18-38950-3	CSE
5	Mst. Nurey Chomon **Atiya**	17-33371-1	EEE

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

Table of Contents

Experiment Title

Familiarization with the Raspberry Pi.

Abstract

This experiment aims to introduce the Raspberry Pi, a series of small single-board computers, and familiarize the user with its features and functions. Developed by the Raspberry Pi Foundation, the Raspberry Pi was created to promote basic computer science education in schools and developing countries. This experiment provides an overview of the Raspberry Pi and its various versions, highlighting its significance and potential uses. The methodology involves hands-on experience with the device, enabling users to gain practical knowledge and skills in using the Raspberry Pi.

Introduction

The Raspberry Pi is a series of small single-board computers that have gained widespread popularity in recent years due to their affordability and versatility. Developed by the Raspberry Pi Foundation in the UK, these computers are designed to promote basic computer science education in schools and developing countries. In addition to being used for educational purposes, the Raspberry Pi can also be utilized in a variety of other applications such as media centers, game consoles, and home automation systems. This experiment aims to introduce the user to the Raspberry Pi and provide a basic understanding of its features and functions. By gaining practical experience with the device, users can develop skills that are useful for a wide range of applications.

Theory and Methodology

The Raspberry Pi is a versatile and affordable single-board computer that has gained popularity for various applications in fields such as education, home automation, and robotics. In this experiment, the objective is to familiarize oneself with the Raspberry Pi, its hardware components, and its capabilities. The Raspberry Pi 3 Model B, which is equipped with a quad-core ARM Cortex-A53 processor, 1 GB LPDDR2 RAM, Wi-Fi, Bluetooth, Ethernet, USB, and GPIO pins, is used in this laboratory. The methodology involves setting up the Raspberry Pi, accessing and configuring the Linux operating system, and experimenting with the different hardware components and software libraries available for the Raspberry Pi.

Components List

- Activated Raspberry pi
- Resistor (220 Ω)
- LED lights (red, yellow, green)
- Breadboard
- Jumper wires
- Computer
- Monitor
- Display and Connectivity Cable (HDMI/DVI)
- Power Supply
- Breadboard

Task #1: LED Blinking

Circuit Diagram

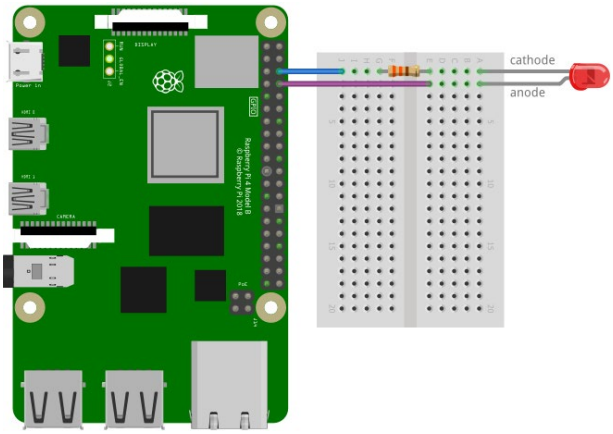

Figure: Setting up the circuit for the LED blinking program.

Hardware Output Results

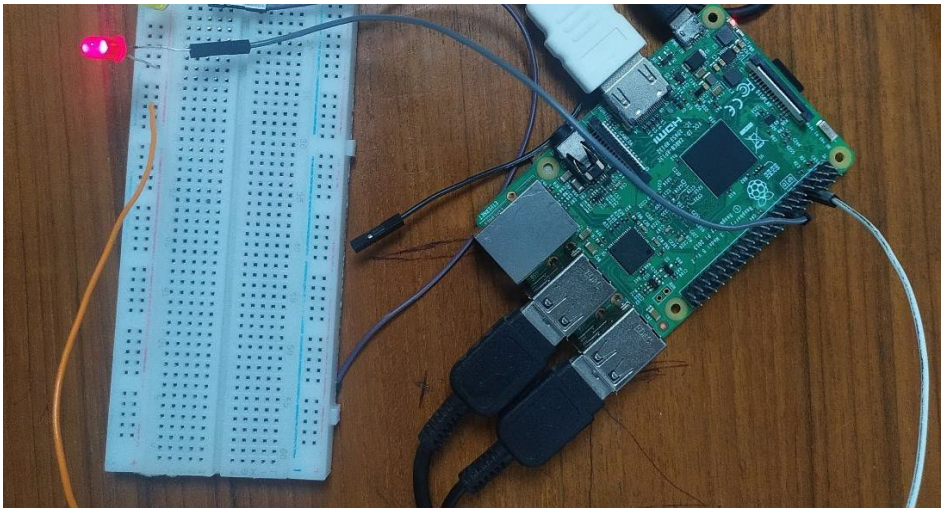

Figure: While LED is ON.

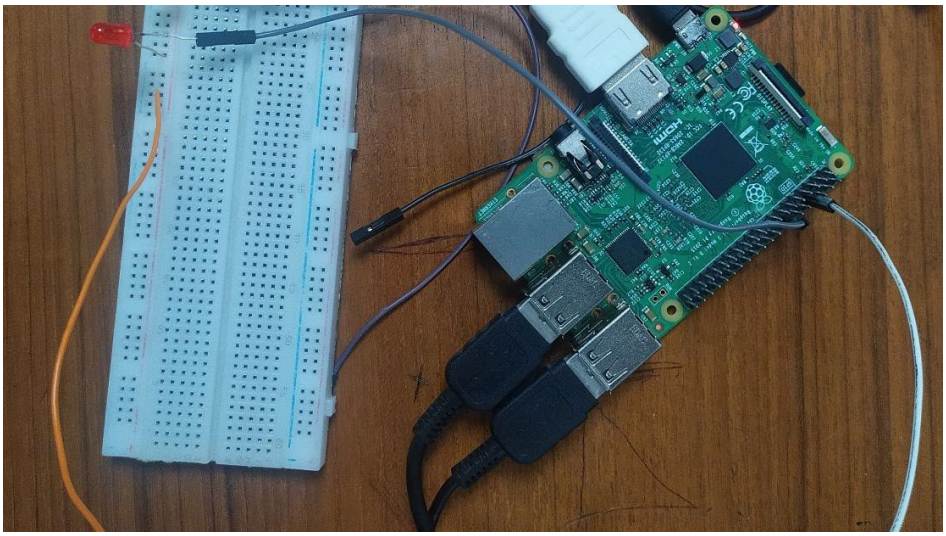

Figure: While LED is OFF.

Code and Explanation

```
import RPi.GPIO as GPIO      # Import RPi.GPIO library to control GPIO pins
import time                  # Import Time library contains the sleep()

GPIO.setmode(GPIO.BCM)       # BCM pin numbering is used
GPIO.setwarnings(False)      # to disable warni ngs
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
```

This code is written in Python and is designed to control the GPIO pins on a Raspberry Pi. The RPi.GPIO library is imported at the beginning of the code to allow access to these pins. The code first sets the pin numbering mode to BCM, which is the standard mode used in most Raspberry Pi projects. It then sets up GPIO14 as an output and turns it on by setting it to a high state. The code then prints a message to the terminal to indicate that the LED is on and sleeps for 2 seconds using the time.sleep() function. After the sleep period, the code sets GPIO14 to a low state to turn off the LED and prints another message to indicate that the LED is off. This code can be useful for beginners who are just starting to learn about controlling the GPIO pins on a Raspberry Pi.

Simulation Output Results and Flowchart

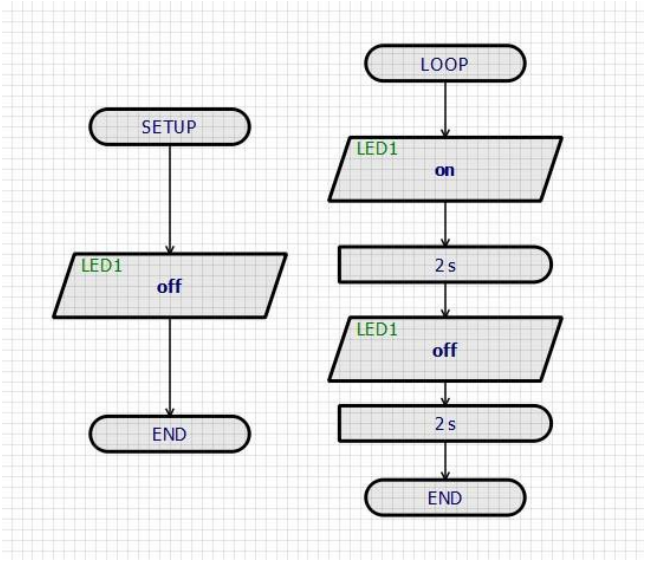

Figure: Proteus simulation flowchart for single LED

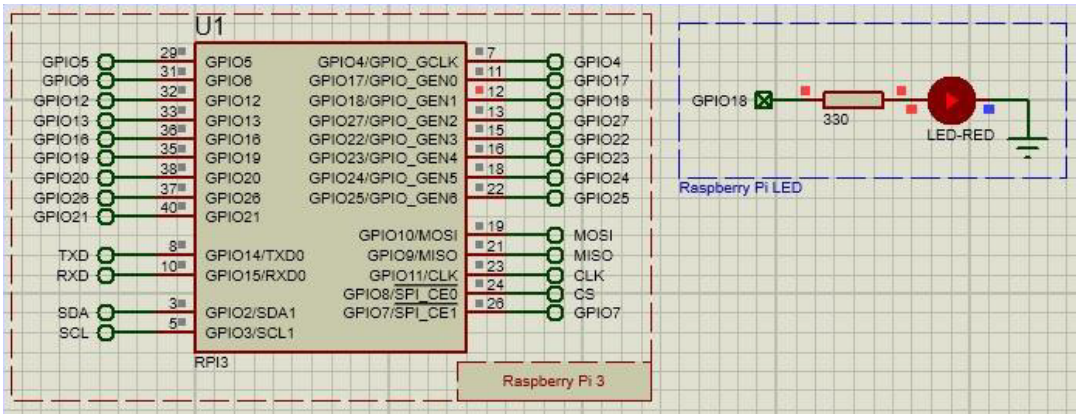

Figure: Proteus Simulation (While the LED is Turned On)

Task #2: LED Controlling With a Push Button Switch

Circuit Diagram

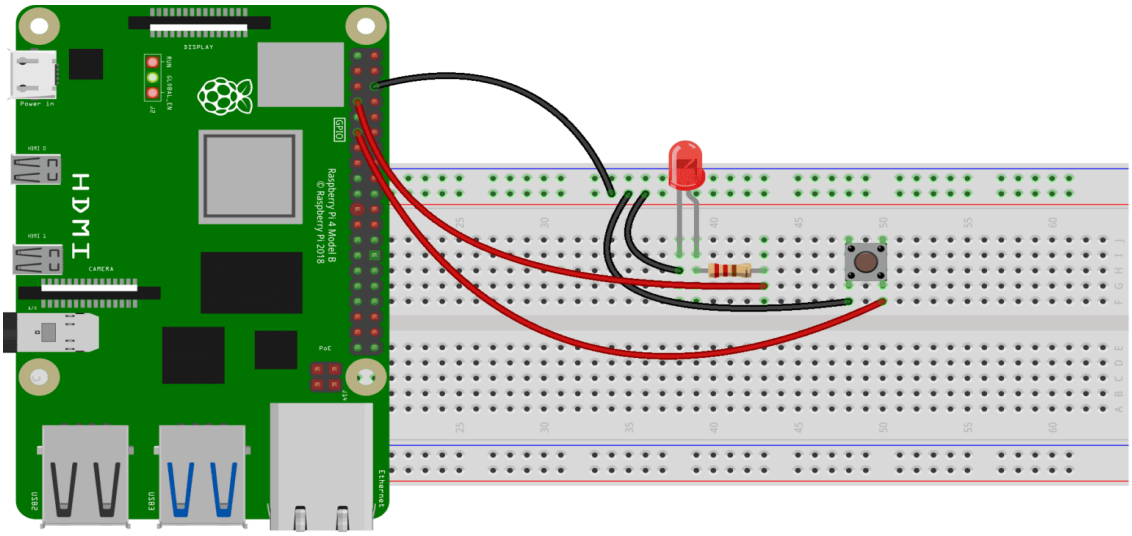

Figure: Setting up the circuit for the LED controlling experiment using a button switch.

Hardware Output Results

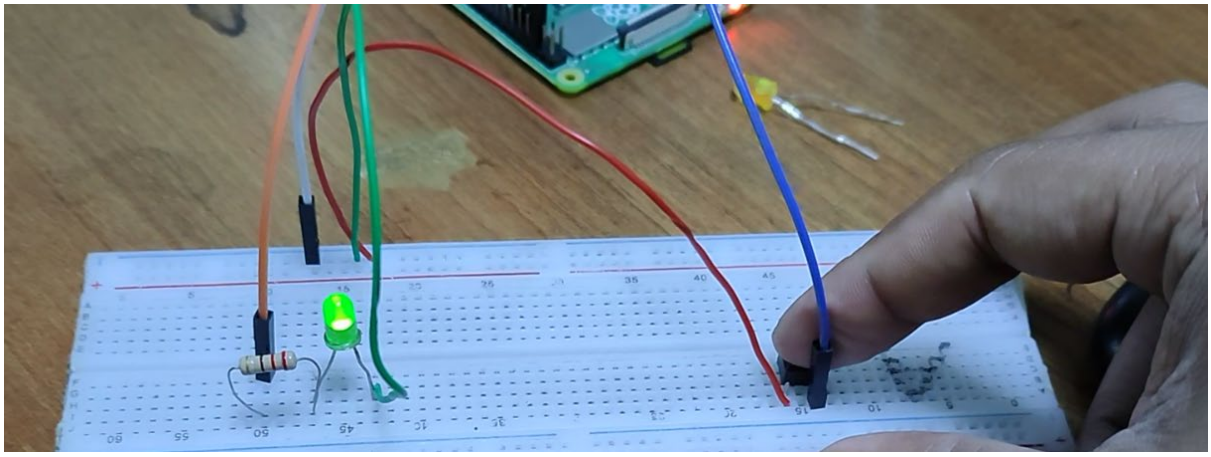

Figure: LED turned on when the button is pressed

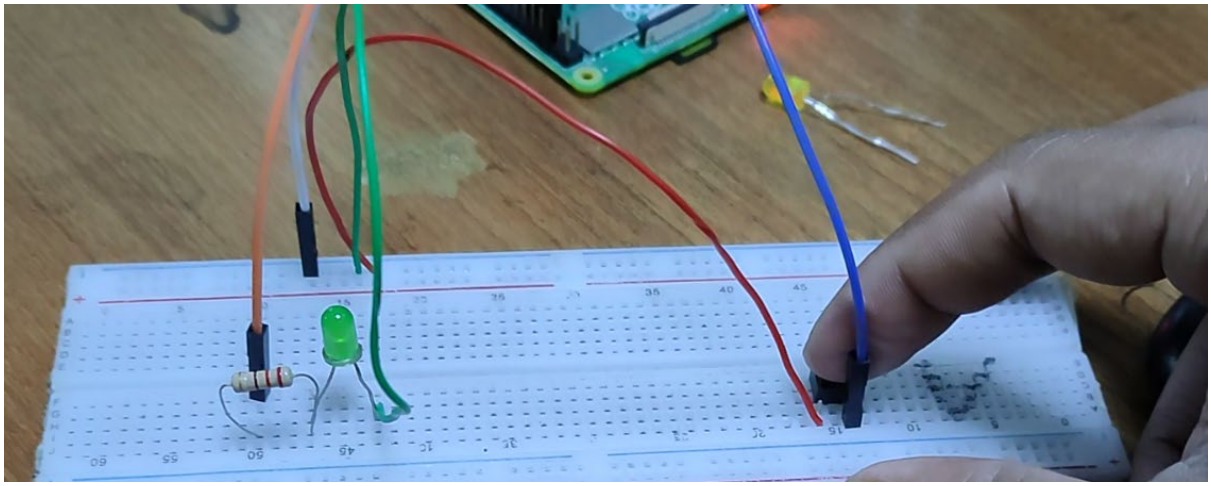

Figure: LED turned off when the button is released

Code and Explanation

```
import RPi.GPIO as GPIO          # Import RPi.GPIO library to control GPIO pins
import time                      # Import 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 for the LED
# to set GPIO15 as an input for the switch with pull-up resistor
GPIO.setup(15, GPIO.IN, pull_up_down=GPIO.PUD_UP)

while True:                              # Forever Loop
    if GPIO.input(15) == GPIO.LOW:       # if the switch is pressed
        GPIO.output(14, GPIO.HIGH)       # turn on the LED
        print("LED is ON")               # show message to Terminal
    else:                                # if the switch is released
        GPIO.output(14, GPIO.LOW)        # turn off the LED
        print("LED is OFF")              # show message to Terminal
    time.sleep(0.1)                      # wait for 100ms
```

The code uses the RPi.GPIO library to interact with the GPIO pins and the time library to control the timing of the LED blinking. In the code, the BCM numbering system is used to define the GPIO pins, and the setup() function is used to set up the input and output pins. The while loop runs continuously, checking the state of the push-button switch by calling the input() function on GPIO pin 15. If the switch is pressed, the code turns on the LED by setting GPIO pin 14 to HIGH and prints a message to the terminal saying that the LED is on. If the switch is released, the code turns off the LED by setting GPIO pin 14 to LOW and prints a message to the terminal saying that the LED is off. The sleep() function is used to pause the program for 100ms between each loop iteration.

Overall, this code demonstrates how to use GPIO pins and the RPi.GPIO library to control an LED with a push-button switch. It is a simple example that showcases the basic structure of a program using GPIO inputs and outputs. The code can be modified to control other electronic components and sensors connected to the Raspberry Pi's GPIO pins.

Simulation Output Results and Flowchart

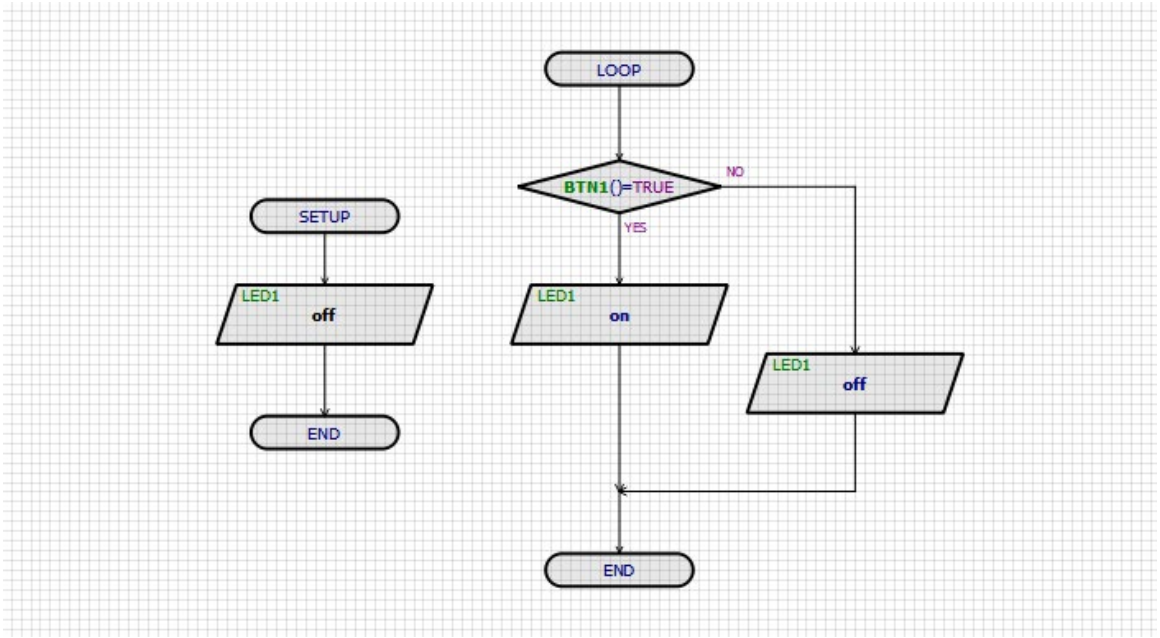

Figure: Proteus simulation flowchart

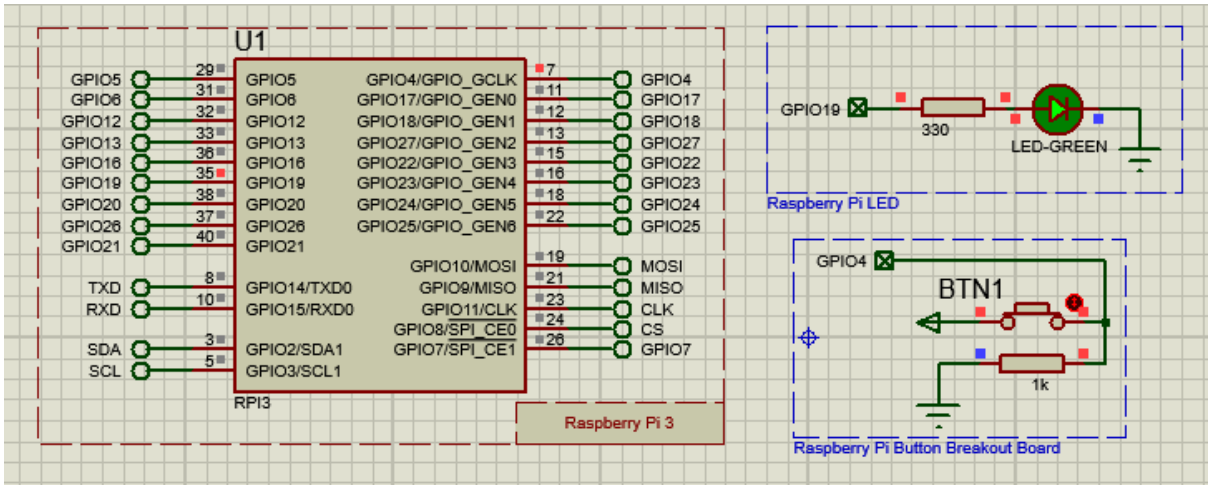

Figure: While the push button is pressed

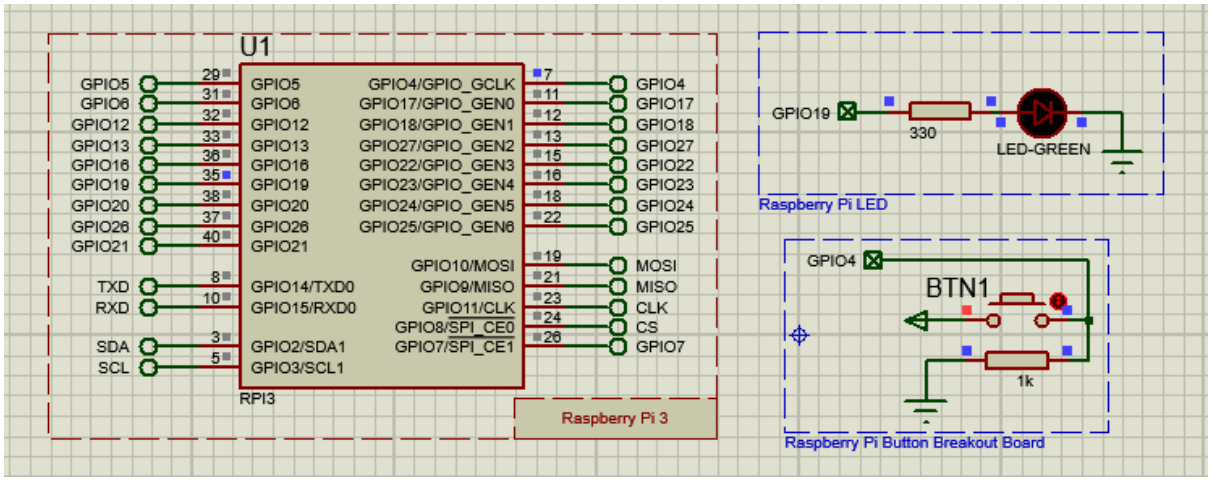

Figure: While the push button is relesed

Task #3: Simple Traffic Control System

Hardware Output Results

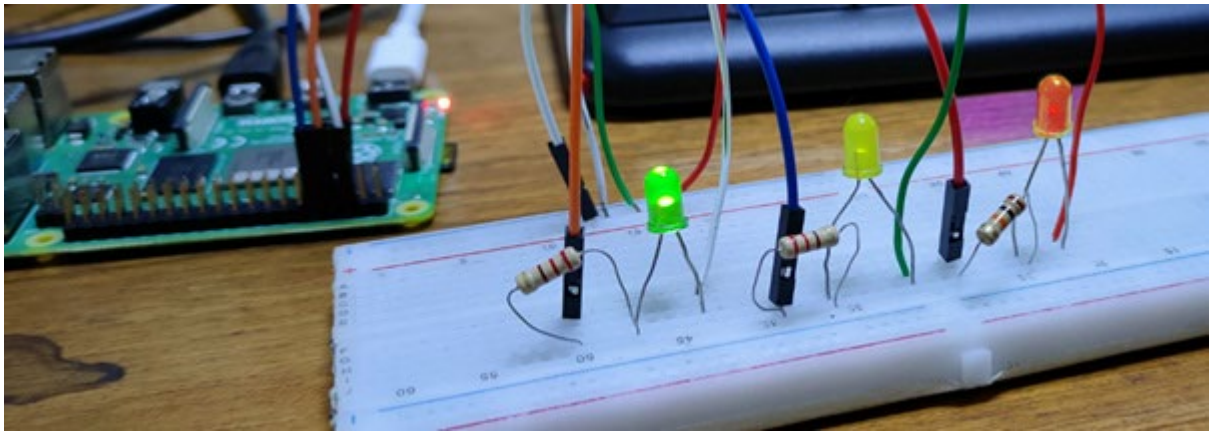

Figure: Circuit of the traffic light (when the green led is on)

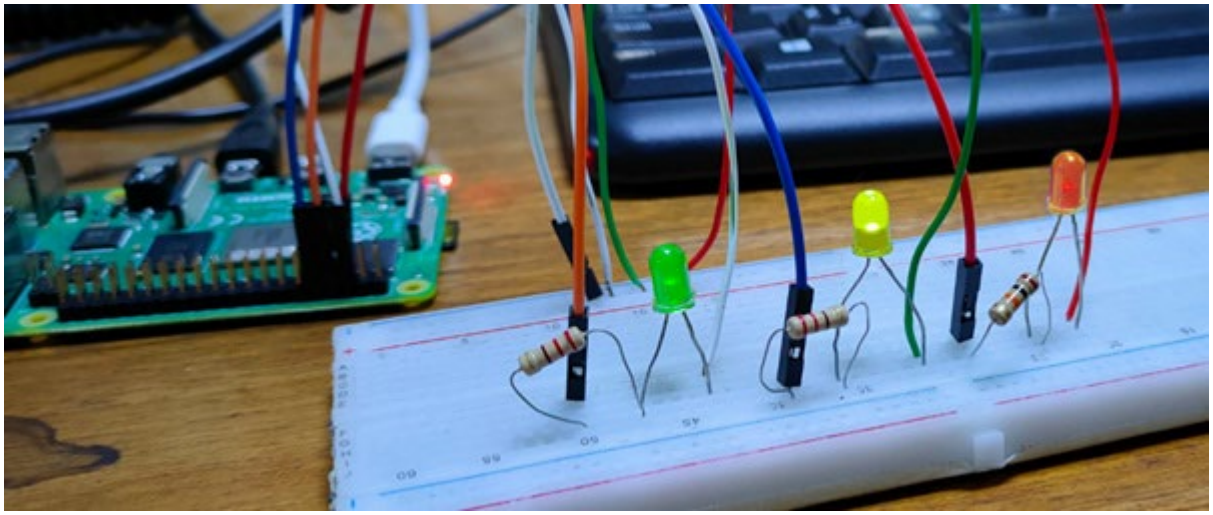

Figure: Circuit of the traffic light (when the yellow led is on)

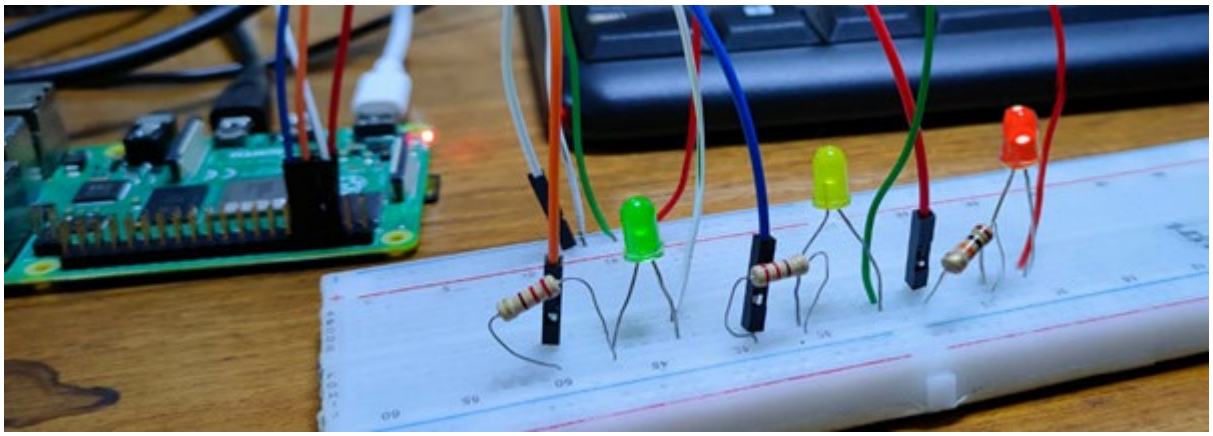

Figure: Circuit of the traffic light (when the red led is on)

Code and Explanation

```
import RPi.GPIO as GPIO     # Import RPi.GPIO library to control the GPIO pins
import time                 # Import Time library contains the sleep()

# Set up the Raspberry Pi pins
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 for Red LED
GPIO.setup(15, GPIO.OUT)            # To set GPIO15 as an output for Green LED
GPIO.setup(18, GPIO.OUT)            # To set GPIO18 as an output for Yellow LED

while True:                         # Forever loop
    # For Green LED
    GPIO.output(15, GPIO.HIGH)      # Green LED ON
    print("Green LED is ON")        # Print 'Green LED is ON' in the Terminal
    time.sleep(2)                   # Wait for 2 sec
    GPIO.output(15, GPIO.LOW)       # Green LED OFF
    print("Green LED is OFF")       # Print 'Green LED is OFF' in the Terminal

    # For Yellow LED
    for i in range(3):    # For loop will iterate from 0 to 2, three times
        GPIO.output(18, GPIO.HIGH) # Yellow LED ON
        # 'Green LED is ON of 1/2/3' will be printed in Terminal
        print("Yellow LED is ON of " + str(i+1) + " time")
        time.sleep(0.5)             # Wait for 0.5 sec
        GPIO.output(18, GPIO.LOW)  # Yellow LED OFF
        time.sleep(0.5)             # Wait for 0.5 sec
    print("Yellow LED is OFF")      # Print 'Yellow LED is OFF' in the Terminal

    # For Red LED
    GPIO.output(14, GPIO.HIGH)      # Red LED ON
    print("Red LED is ON")
    time.sleep(5)                   # Wait for 5 sec
    GPIO.output(14, GPIO.LOW)       # Red LED OFF
    print("Red LED is OFF")         # Print 'Red LED is OFF' in the Terminal
```

The code starts by importing the necessary libraries, RPi.GPIO and time. The GPIO pins are then set up using the BCM pin numbering system. The pins 14, 15, and 18 are designated as output pins for the red, green, and yellow LEDs, respectively.

The code then enters an infinite loop that will continue running until the program is terminated. Within this loop, there are three sections of code that correspond to each of the three LEDs. Each section turns on the LED, waits for a specified amount of time, turns off the LED, and then waits for another specified amount of time.

The first section controls the green LED. It turns on the LED for 2 seconds and then turns it off. The second section controls the yellow LED. It turns on the LED for 0.5 seconds and then turns it off, repeating this process three times with a delay between each cycle. And the third section controls the red LED. It turns on the LED for 5 seconds and then turns it off.

The code can be further modified to control the LED based on the input from a switch. This can be done by adding a new line of code that sets up a pin as an input pin, and another line that reads the input from that pin. Then an if statement can be added to the loop that will turn on or off the LED based on the value of the input from the switch.

Simulation Output Results and Flowchart

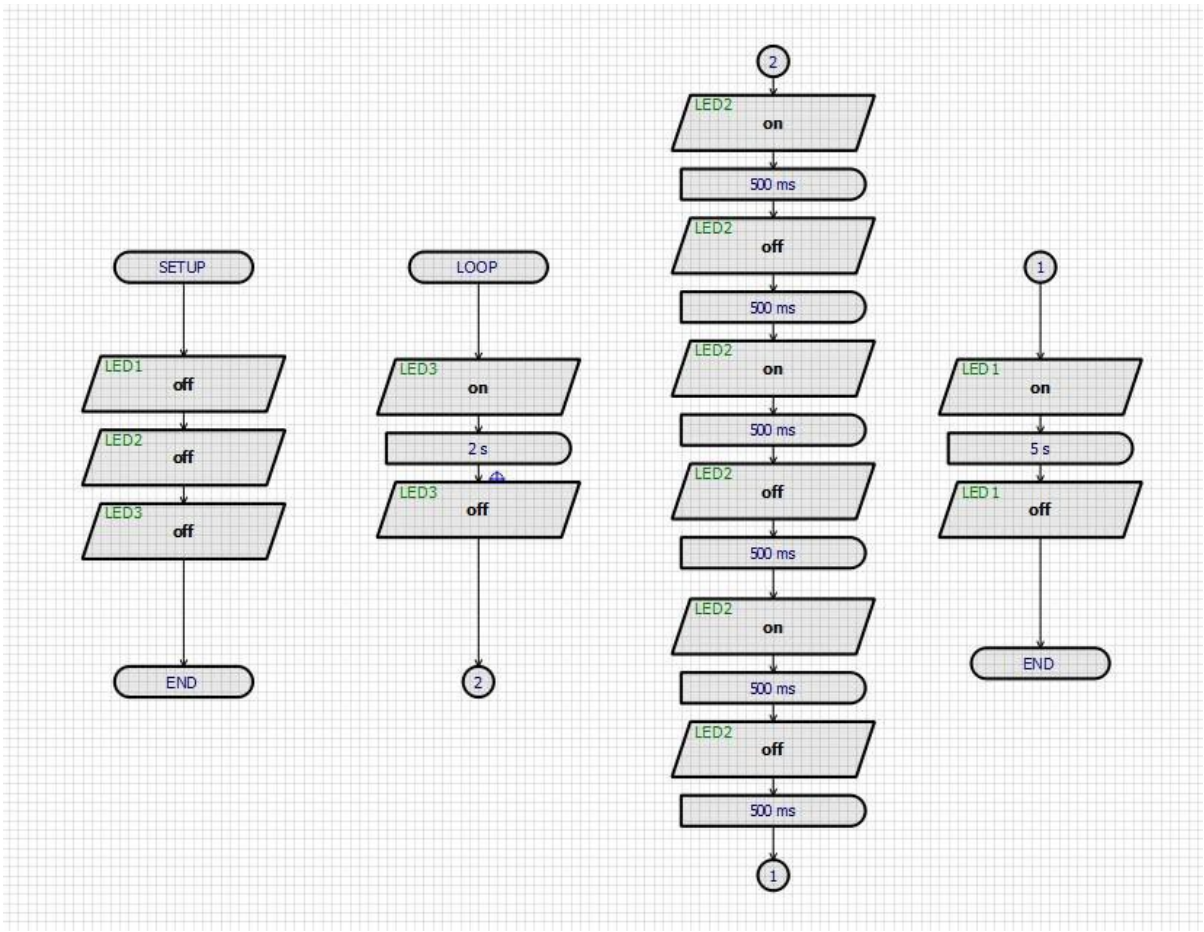

Figure: Proteus Simulation flowchart for the traffic controller system.

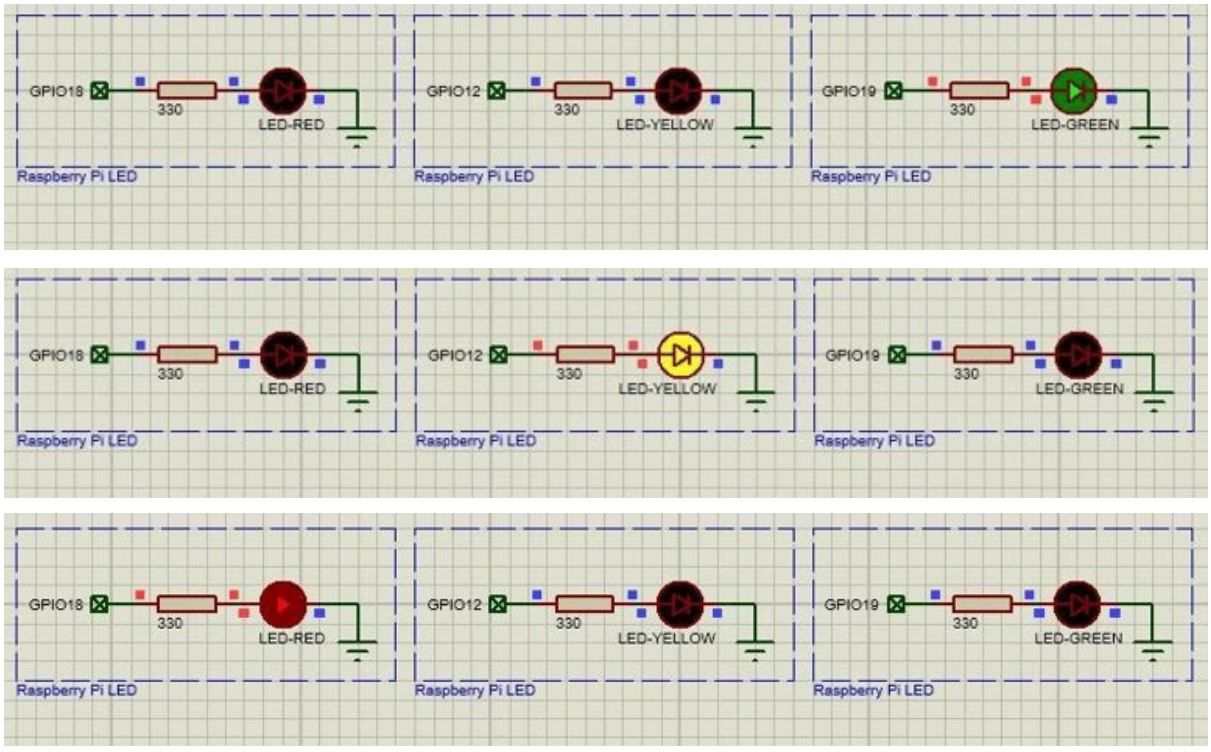

Figure: Proteus Simulation for the traffic controller system.

Discussion & Conclusion

This experiment provided an introduction to the Raspberry Pi, its hardware components, and its capabilities. The practical experience gained through the different tasks, including LED blinking, LED controlling with a push button switch, and simple traffic control system, allowed the user to develop skills that are useful for a wide range of applications. This experiment demonstrated the versatility and affordability of the Raspberry Pi and highlighted its potential uses in education, home automation, robotics, and more.

Overall, the Raspberry Pi is a powerful tool that can be utilized for a variety of purposes. Its small size, low cost, and ease of use make it an ideal choice for hobbyists, students, and professionals alike. By familiarizing oneself with the Raspberry Pi, one can gain valuable knowledge and skills that can be applied to a variety of applications, from simple LED projects to complex robotics projects. This experiment provided a great starting point for anyone interested in learning more about the Raspberry Pi and its capabilities.

References

- AIUB Lab Manual
- Raspberry pi datasheet
- https://www.raspberrypi.com/documentation/computers/getting-started.html