Experiment Title: Familiarization with the Raspberry Pi.

Objectives:

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.

Theory and Methodology:

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries. The most important thing about different versions of Raspberry Pi is that it is a computer that costs $5 to $75.

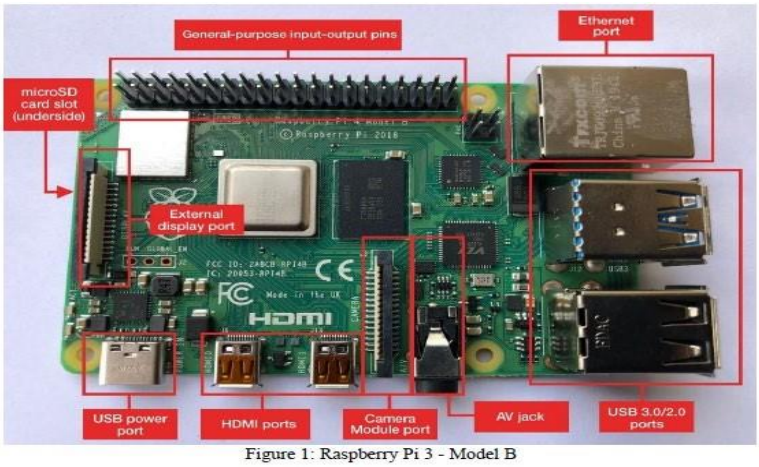

Figure 1: Raspberry Pi 3 - Model B

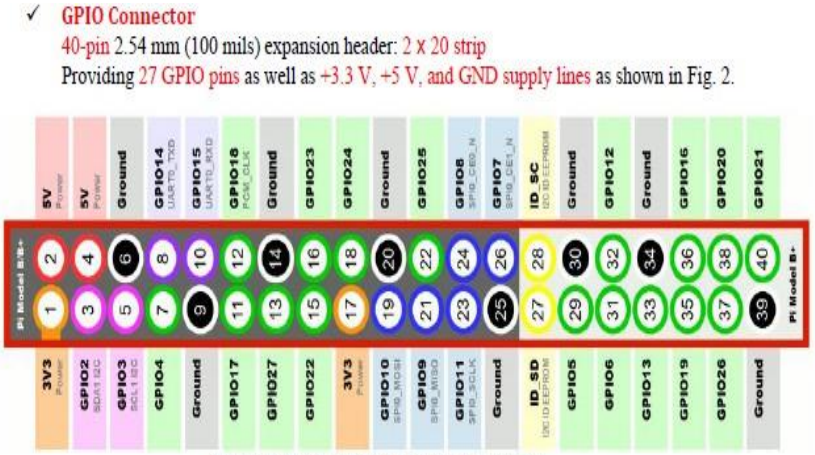

Figure 2: Raspberry Pi 3 - Model B GPIO pin

Equipment List:

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

Circuit Diagram:

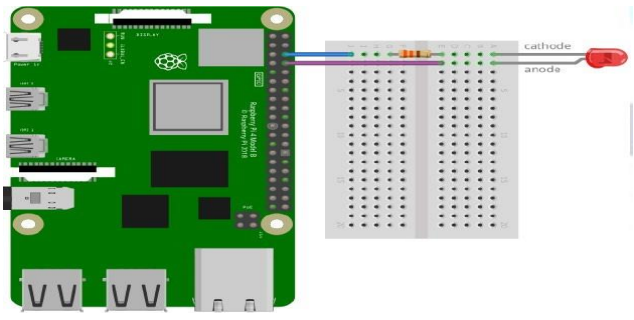

Fig.3 Experimental setup of the circuit for the LED blinking program using Raspberry Pi

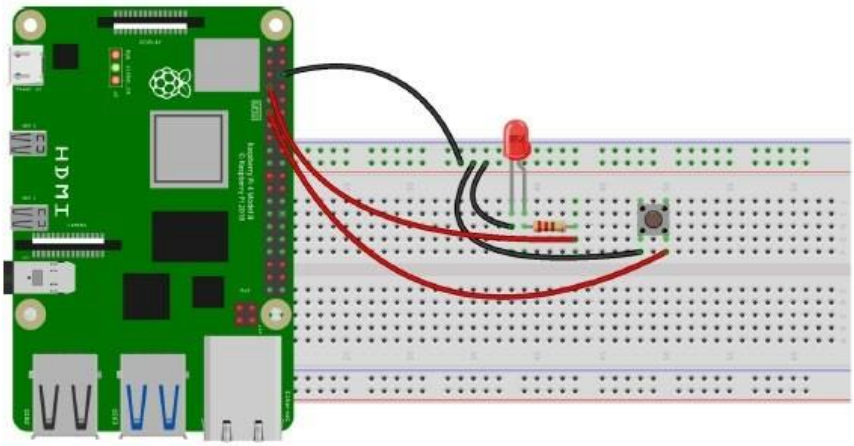

Fig.4 Experimental setup of the circuit for the LED blinking using button program using Raspberry Pi

Experimental Procedure:

To complete the experiment we used simulation software as we conducted the lab in online. So we build the circuit according to the figure 3 and figure 4. Then we wrote the code available in the lab manual and then check whether it was working or not. After successful completion of the experiment we now know the functionality of raspberry Pi.

Code/Program: (*Python 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:

Part 1: LED blink program without Switch:

$ 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.*

Part 2: LED blink program With Switch:

$ nano LED_withButton.py # *create a text file named LED_withButton.py writing at the command line*

from gpiozero import LED

imports LED functions from the gpiozero library from

gpiozero import Button

imports Button functions from the gpiozero library led

= LED(4)

declare the GPIO in pin 4 for LED output and store it in a variable named led. button

= Button(17)

declare the GPIO in pin 17 for Button input and store it in a variable named button. while

True:

initiate 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 the button to wait till release led.off()

turn off the LED

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:

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.

Part 1: LED blink program without Switch:

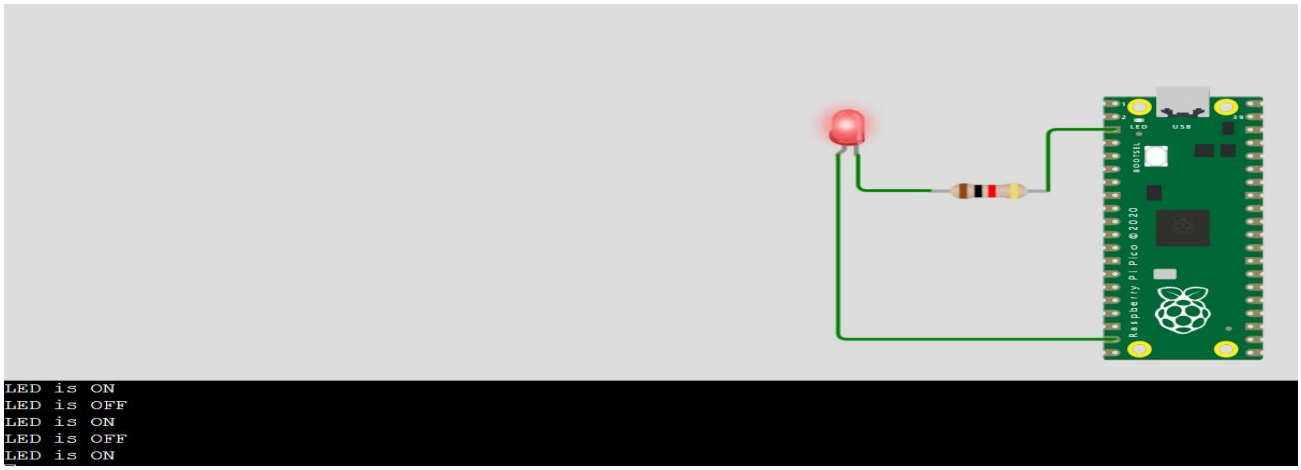

Fig.11 Led blink program without switch (LED is ON)

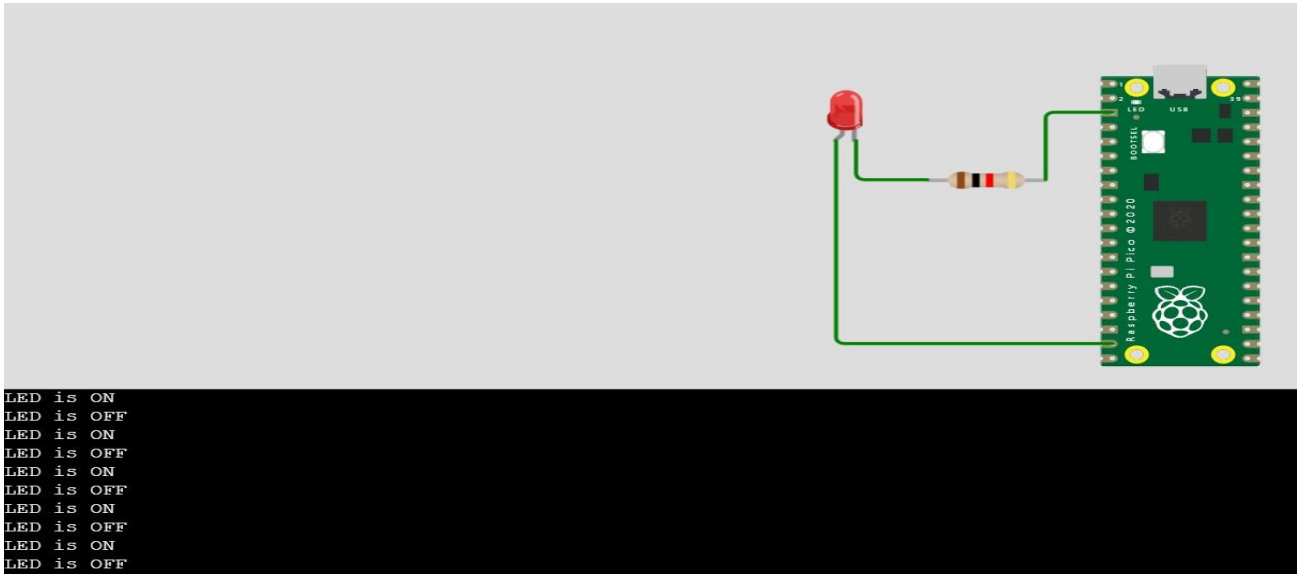

Fig.12 Led blink program without switch (LED is OFF)

Explanation For LED BLINK:

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.

Part 2: LED blink program with Switch:

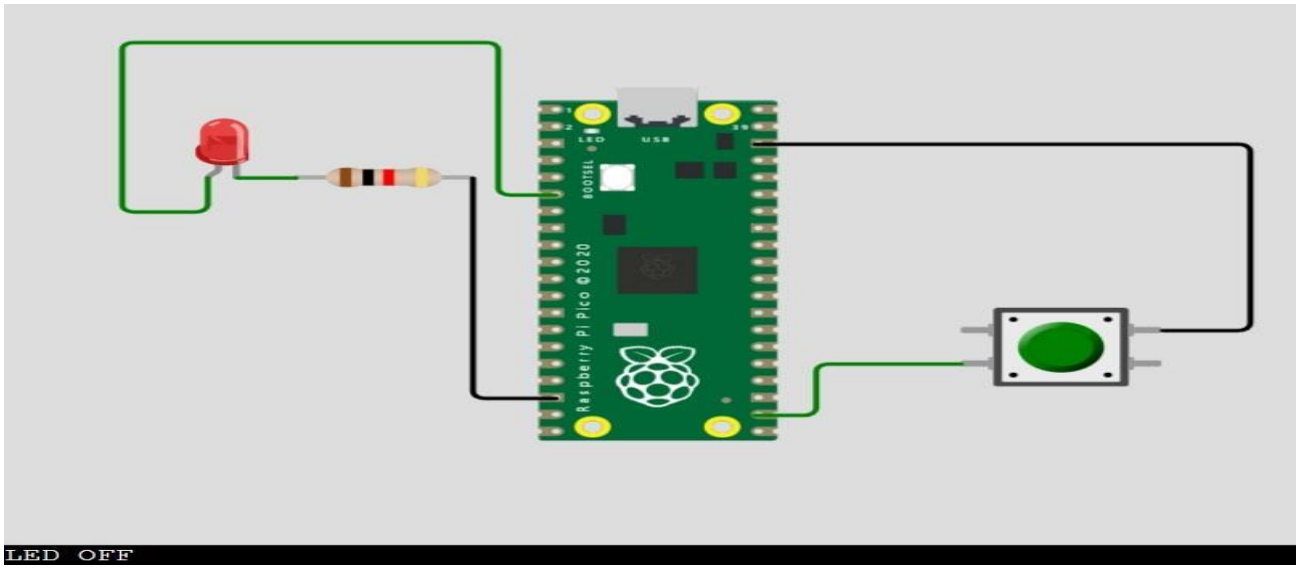

Fig. 8 Led blink program with switch (LED is OFF)

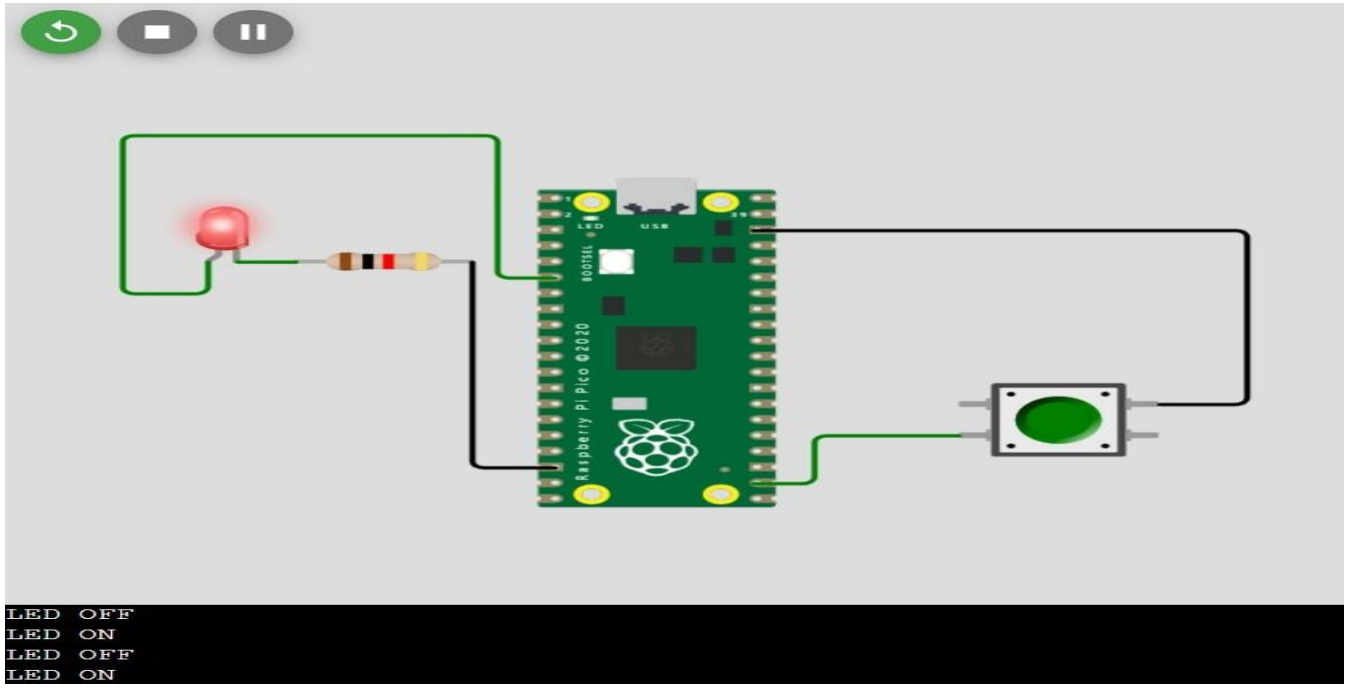

Fig.14 Led blink program with switch (LED is ON when pressed)

Explanation For LED-BUTTON:

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.

Discussion:

In this experiment, we delved into the practical utilization of Raspberry Pi for controlling LED lights, utilizing an online simulator as our experimental environment. Through this hands-on endeavor, our primary goal was to gain a comprehensive understanding of Raspberry Pi's functionality and its application in real-world scenarios.

First and foremost, we familiarized ourselves with the process of importing the GPIO (General Purpose Input/Output) library and employing it to suit our specific requirements within the simulator environment. This initial step not only acquainted us with the fundamentals of interacting with hardware using Raspberry Pi but also provided valuable insights into navigating the Linux environment, essential for executing commands and managing the system effectively.

By conducting the entirety of our task within an online simulator, we acquired the knowledge and proficiency necessary to replicate similar Raspberry Pi-based projects in virtual environments, thereby broadening our scope of experimentation without the constraints of physical hardware.

Looking ahead, the skills honed through this experiment serve as a solid foundation for tackling more complex projects, such as the development of smart irrigation systems or gas leakage detectors, leveraging the capabilities of Raspberry Pi to create innovative solutions for real-world problems.

Following the completion of our system setup, we meticulously operated the devised system, closely monitoring its functionality and recording the observed outcomes. These results, generated based on predefined formulas within the code, were systematically documented for further analysis and evaluation. Subsequently, to validate the effectiveness of our system implementation, we replicated a similar setup using simulation software like Proteus or online platforms such as wokwi.com. Discrepancies in LED blinking patterns and distance measurements between the physical and simulated environments were noted, potentially attributable to minor errors introduced during system configuration or inherent differences between virtual and physical conditions.

In conclusion, this experiment not only provided valuable hands-on experience with Raspberry Pi but also underscored the significance of comprehensive testing and evaluation in ensuring the reliability and functionality of electronic systems, setting the stage for future explorations and innovations in the realm of embedded computing.

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