

American International University- Bangladesh

Department of Electrical and Electronic Engineering

EEE 4103: Microprocessor and Embedded Systems Laboratory

Title: Familiarization with the Raspberry Pi.

Introduction:

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.

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.

Hardware

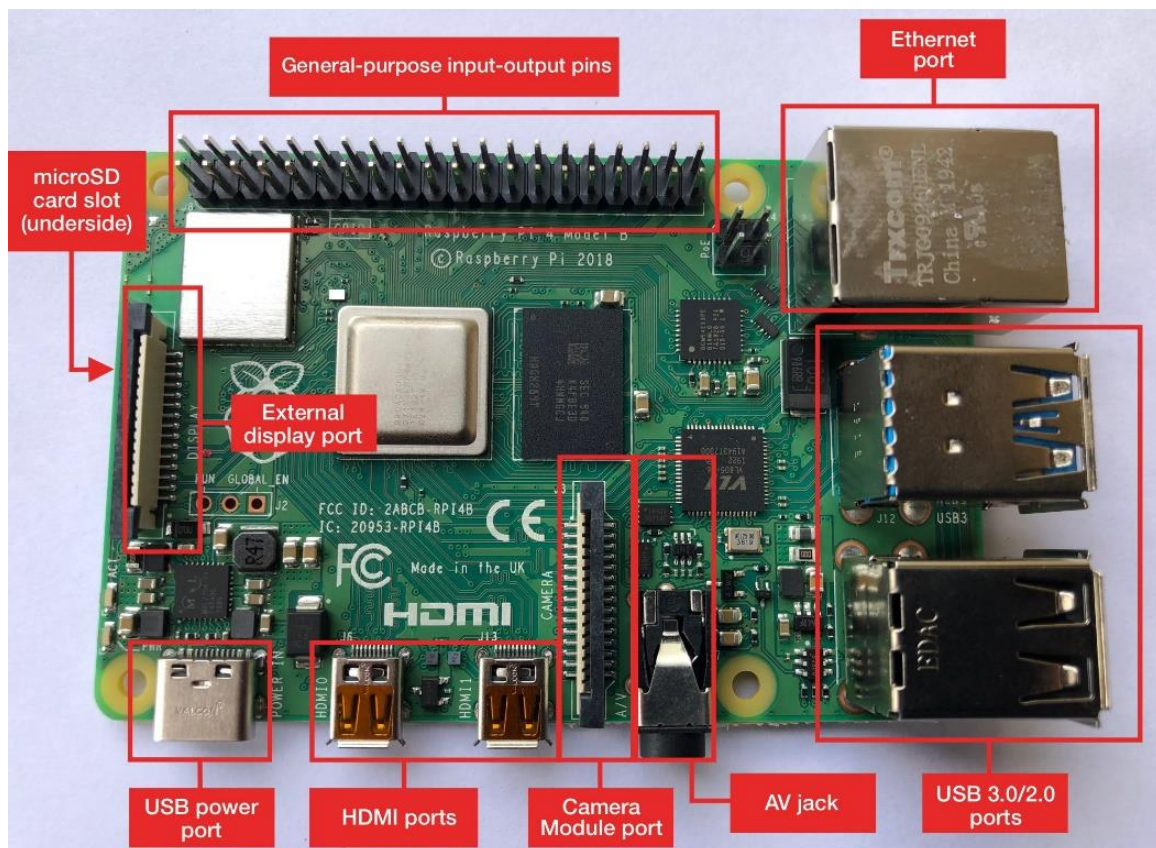


Figure 1: Raspberry Pi 3 - Model B

In this laboratory version, **Raspberry Pi 3 - Model B** will be used. Thus, only the hardware specifications of Raspberry Pi 3 – Model B are discussed.

Technical Specifications

1. Processor

- ✓ Broadcom BCM2387 chipset.
- ✓ 64-bit 1.2 GHz Quad-Core ARM Cortex-A53.

2. 802.11 b/g/n Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE)

- ✓ **IEEE 802.11 b / g / n Wi-Fi. Protocol:** WEP, WPA WPA2, algorithms AES-CCMP (maximum key length of 256 bits), the **maximum range of 100 meters**.
- ✓ **IEEE 802.15 Bluetooth**, symmetric encryption algorithm Advanced Encryption Standard (AES) with a 128-bit key, a **maximum range of 50 meters**.

3. GPU

- ✓ Dual Core Video Core IV® Multimedia Co-Processor. Provides Open GL ES 2.0, hardware-accelerated Open VG, and 1080p30 H.264 high-profile decode.
- ✓ Capable of 1 Gpixel/s, 1.5 Gtexel/s, or 24 GFLOPs with texture filtering and DMA infrastructure.

4. Memory

- ✓ **1 GB** LPDDR2.

5. Operating System

- ✓ **Boots from a Micro SD card**, running a version of the Linux operating system or Windows 10 IoT.

6. Dimensions

- ✓ 85 x 56 x 17mm

7. Power

- ✓ **Micro USB socket 5 V, 2.5 A**

8. Ethernet

- ✓ **10/100 Base T Ethernet socket.**

9. Video Output

- ✓ **HDMI** (rev 1.3 & 1.4)
- ✓ Composite RCA (PAL and NTSC)

10. Audio Output

- ✓ Audio Output 3.5 mm jack
- ✓ HDMI
- ✓ **USB 4.0 x USB 2.0 Connector**

11. GPIO Connector

- ✓ **40-pin** 2.54 mm (100 mils) expansion header: **2 x 20 strip**
- ✓ Providing **27 GPIO pins** as well as **+3.3 V, +5 V, and GND supply lines** as shown in Fig. 2.

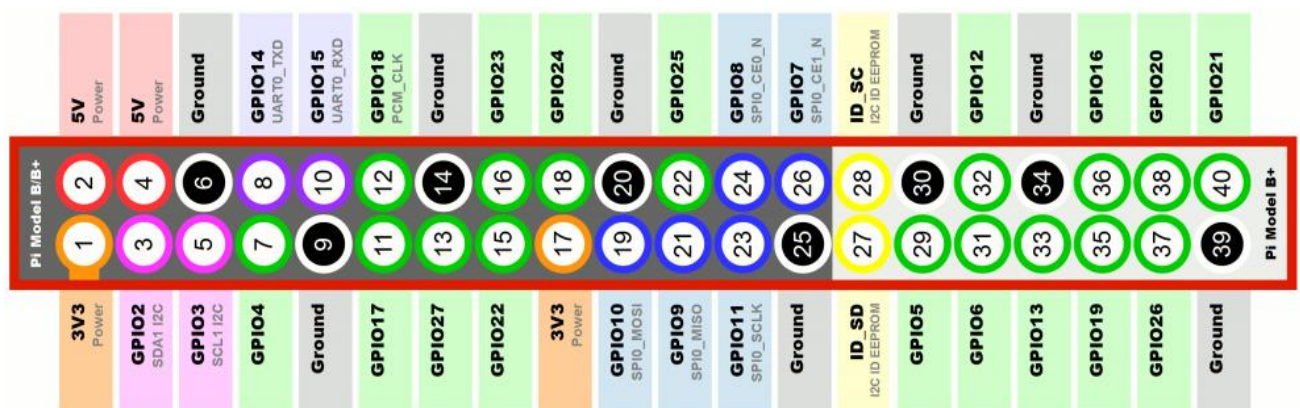


Figure 2: Raspberry Pi 3 - Model B GPIO pin

12. Camera Connector

- ✓ **15-pin MIPI Camera Serial Interface (CSI-2)**

13. Display Connector

- ✓ Display Serial Interface (DSI) 15-way flat flex cable connector with two data lanes and a clock lane.

14. USB

- ✓ Four built-in USB ports provide enough connectivity for a mouse, keyboard, or anything else that you feel the RPi needs.

15. Antenna

- ✓ There's no need to connect an external antenna to the Raspberry Pi 3. Its radios are connected to this chip antenna soldered directly to the board.

16. HDMI connector:

- ✓ The HDMI port provides digital video and audio output. 14 different video resolutions are supported, and the HDMI signal can be converted to DVI (used by many monitors), composite (analog video signal usually carried over a yellow RCA connector), or SCART (a European standard for connecting audio-visual equipment) with external adapters.

17. Status LED

There are **five status LEDs** on the corner of the board.

Table 1: Status of LEDs

| | | |
|-----|--------|---|
| ACT | Green | Light is ON when the SD card is accessed/used |
| PWR | Red | Steady ON when Pi is connected to 3.3 V power |
| FDX | Green | ON if the network adapter is a full duplex |
| LNK | Green | Network activity light ON when Ethernet is connected |
| 100 | Yellow | ON if the network connection is 100 Mbps |

The status LEDs give information about the operating condition and any problems of the board.

Table 2: Troubleshooting from the LED status

| Status LED | Possible Problem |
|--|--|
| The red power LED does not light, and nothing is on display | The power is not properly connected |
| The red power LED light is blinking | The red power LED should never blink. A blinking red LED (PWR) means the 5 V power supply is dropping out. Use a different power supply. |
| The red power LED is ON, and the green LED is glowing faintly and steadily | The power supply is OK. But faint and steady green light (ACT) means the SD card has some problem in starting the operating system (no boot code). |

Setting Up the Raspberry Pi

You need the following items-

- [1] Raspberry Pi Board
- [2] Monitor
- [3] Display and Connectivity Cable (HDMI/DVI)
- [4] Keyboard and Mouse
- [5] Power Supply (good-quality power supply that can supply **at least 2 A at 5 V for the Model 3B**)
- [6] SD Card (minimum **4 GB**)
- [7] Ethernet (network) cable (Optional)
- [8] Audio Lead (without an HDMI Cable an audio lead is necessary to produce sound)

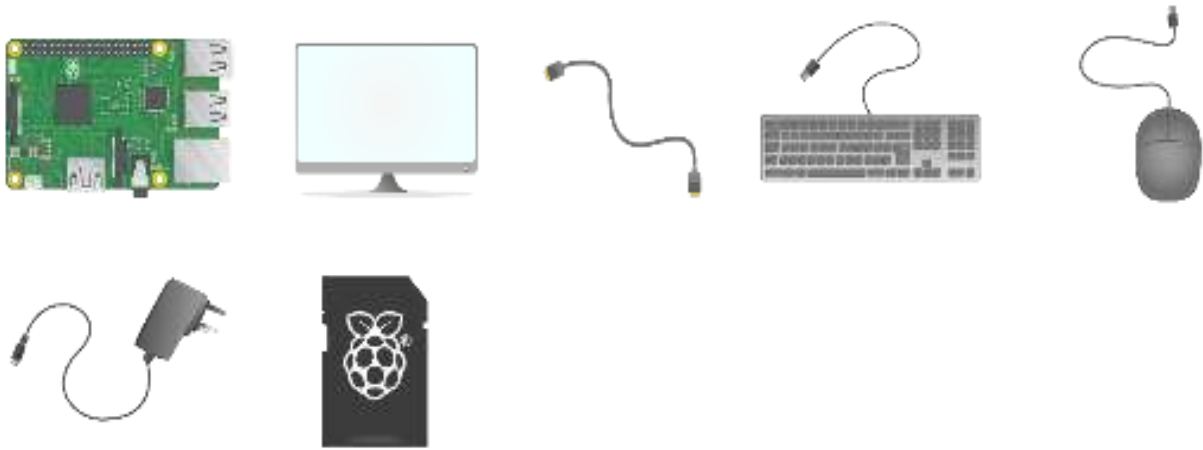


Figure 3: Apparatus for Setting up Raspberry Pi

Operating Systems

When setting up Raspberry Pi, always keep in mind that you are setting up a small PC. Like a PC set up Pi must be set up with Operating System (OS). **Raspbian** is the official operating system for all models of the Raspberry Pi. However, there are **third-party operating systems**, like Ubuntu Mate, Snappy Ubuntu Core, Windows 10 IoT core, OSMC, Librelec, Pinet, and RISC OS. Logos of different OS for Pi are shown in following Fig. 4.



Figure 4: Logos of Different Operating Systems

Getting an operating system

The **recommended operating system** for use with the Raspberry Pi is called **Raspbian**. Raspbian is a version of GNU/Linux, designed specifically to work well with the Raspberry Pi.

Download and image Raspbian directly: This is a faster process and is great if you need to image multiple cards for a workshop or class. The steps are given below:

- [1] Using a computer with an SD card reader, visit the official Raspberry Pi Download page. Link: <https://www.raspberrypi.org/downloads/>
- [2] Go to the following link to know the step-by-step process to install the Raspberry Pi imager first: <https://www.youtube.com/watch?v=ntaXWS8Lk34>
- [3] The following link will help to know step by step process to configure the Raspberry Pi: <https://www.youtube.com/watch?v=wjWZhV1v3Pk>
- [4] The following link will help to know step by step process to update the Raspberry Pi: <https://www.youtube.com/watch?v=y9KILxtMTOA>

Power on your Raspberry Pi for the first time

Remember that after booting the Pi, the user credentials like the 'username' and 'password' will be asked. Raspberry Pi comes with a default username and password and so always use it whenever it is being asked. The login credentials are:

login: pi
password: raspberry

Introduction:

In this experiment, an LED will be controlled by using Raspberry Pi. **Python** will be used to blink an LED. This experiment will give a basic idea of **Python language** as well as import GPIO pins of Raspberry Pi. This experiment will be done from a **Linux environment** thus enabling to use of terminal and shell scripting. As Raspberry Pi runs on a Linux environment, it is always advised to use **text editors**, like **Gvim, Nano Editor, Emacs Editor, and Pico Editor**. However, when you installed your Raspbian it comes with Integrated Development Environment (IDE) for Python.

Apparatus:

- [1] Activated Raspberry pi
- [2] LED
- [3] Push switch
- [4] Resistor (220 Ω)
- [5] Breadboard
- [6] Jumper wires

Experimental Procedures:

Lab Task # 1: LED Blinking

- 1) The first step of this lab task is to set up the circuit as shown in Fig. 5.

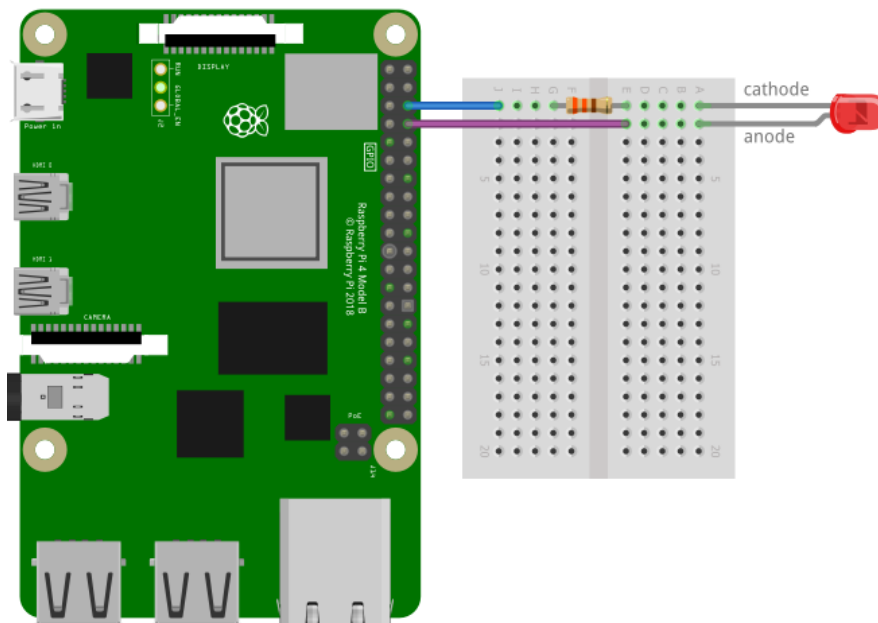


Figure 5: Setting up the circuit for the LED blinking program.

- 2) After setting up the circuit, Raspberry Pi should be powered on and interfaced with Monitor, Mouse, and Keyboard.
- 3) Open the terminal.
- 4) Write your code according to the following code example in the terminal.

Python Program:

\$ 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.
```

To save the program, press “Ctrl+x” then “Y” then “enter”.

To simply run the program type:

```
sudo python blinkLED.py
```

The LED will turn on for two seconds and then turn off.

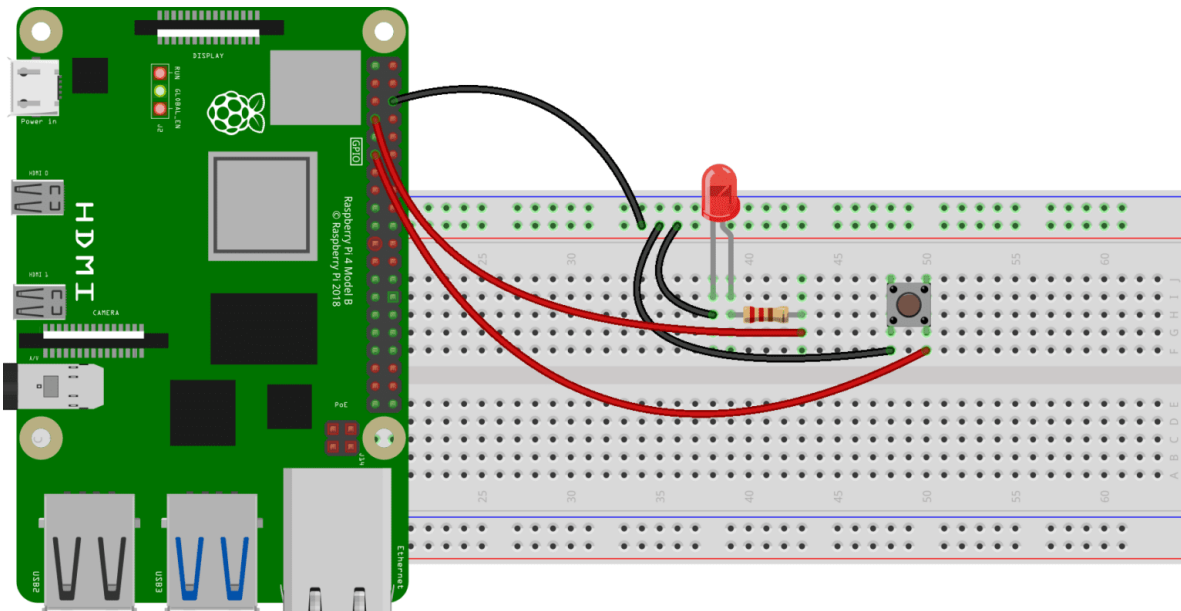
Lab Task # 2: LED controlling with a push button switch

Figure 6: Setting up the circuit for the LED controlling experiment using a button switch.

Python code for controlling LED with the button on Raspberry Pi 4

The Python code for controlling the LED with the button on Raspberry Pi 4 is simple, we will first make a file with the name “LED_withButton.py” and open it with the nano editor as follows:

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

Type the following python code in the file to control the LED with the button:

```
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
```

On the command line text editor, the program will look like as given below:

```
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
```

Explanation of the Code:

In the above code, we simply import the libraries of LED and Button from the gpiozero. Then we used the two variables led and button to which we assigned GPIO pin 4 for the LED and GPIO pin 17 for the button. After declaring these variables, in an infinite while loop, we have turned the LED ON by pressing the button, and on releasing the button, the LED is turned off.

Save the nano editor's file by pressing CTRL+S and exit the editor by using the shortcut key CTRL+X. To execute the code file of LED_withButton.py, use the command:

```
$ pytho LED_withButton.py
```

Lab Task # 3: Simple Traffic Control System

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

Hint:

To *repeat a block of code infinitely*, use the *while statement*. These are controlled by a conditional expression.

```
while (True):
    GPIO.output(ledpinnumber, GPIO.HIGH)    # Forever Loop
    print 'LED is ON'                        # LED ON
    time.sleep(1.0)                          # 'LED is ON' will be printed in Terminal
    GPIO.output(ledpinnumber, GPIO.LOW)     #wait 1 sec
    print 'LED is OFF'                      #LED OFF
    time.sleep(1.0)                         # 'LED is OFF' will be printed in Terminal
                                           #wait for 1 sec
```

Sometimes you may want a *loop to stop executing*. To do so, a *'break' statement* may be enclosed within an if statement. For example:

```
x = 0;
while (True):
    print("Raspberry Pi")
    x += 1
    if x > 20:
        break
# text string "Raspberry Pi" will be printed 20 times on the terminal.
```

The *for statement* is used when there is a *block of code* that needs *to be repeated a number of times*. For example, here the word 'Raspberry Pi' is printed five times:

```
for i in range (0, 5):
    print("Raspberry Pi")
```

Questions for Report Writing:

- 1) Include all codes and scripts in the lab report following the lab report writing template.
- 2) Show the output/results in the form of images. Give their captions and descriptions.
- 3) Include the **Proteus simulation** of the same program of this manual. Explain the simulation methodologies of both simulators.

Reference(s):

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