EXPERIMENT-1

OBJECTIVE

Introduction: Objective, scope and outcome of the course.

THEORY

The Internet of Things (IoT) describes the network of physical objects— "things"—that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet. These devices range from ordinary household objects to sophisticated industrial tools. With more than 7 billion connected IoT devices today.

Industrial IoT

Industrial IoT (IIoT) refers to the application of IoT technology in industrial settings, especially with respect to instrumentation and control of sensors and devices that engage cloud technologies. Refer to this for a good example of IIoT

SCOPE

The advancements in Artificial Intelligence and Machine Learning have made the automation of IoT devices easy. Basically, AI and ML programs are combined with IoT devices to give them proper automation. Due to this, IoT has also expanded its area of application in various sectors.

OUTCOMES

1. Explain the definition and usage of the term "Internet of Things" in different contexts
2. Understand the key components that make up an IoT system
3. Differentiate between the levels of the IoT stack and be familiar with the key technologies and protocols employed at each layer of the stack
4. Apply the knowledge and skills acquired during the course to build and test a complete, working IoT system involving prototyping, programming and data analysis
5. Understand where the IoT concept fits within the broader ICT industry and possible future trends
6. Appreciate the role of big data, cloud computing and data analytics in a typical IoT system

EXPERIMENT-2

OBJECTIVE

Start Raspberry Pi and try various Linix commands in command terminal window:

ls, cd, touch, mv, rm, man, mkdir, rmdir, tar, gzip, cat, more, less, ps, sudo, cron, chown, chgrp, ping etc

COMMANDS: -

1) touch: Create a new file or update its timestamp.

- **Syntax**: touch [OPTION]…[FILE]
- **Example**: Create empty files called 'file1' and 'file2'
  - $ touch file1 file2

2) cat: Concatenate files and print to stdout.

- **Syntax**: cat [OPTION]…[FILE]
- **Example**: Create file1 with entered content
  - $ cat > file1
  - Hello
  - ^D

3) cp: Copy files

- **Syntax**: cp [OPTION]source destination
- **Example**: Copies the contents from file1 to file2 and contents of file1 is retained
  - $ cp file1 file2

4) mv: Move files or rename files

- **Syntax**: mv [OPTION]source destination
- **Example**: Create empty files called 'file1' and 'file2'
  - $ mv file1 file2

5) rm: Remove files and directories

- **Syntax**: rm [OPTION]…[FILE]
- **Example**: Delete file1
  - $ rm file1

6) mkdir: Make directory

- **Syntax**: mkdir [OPTION] directory
- **Example**: Create directory called dir1
  - $ mkdir dir1

7) rmdir: Remove a directory

- **Syntax**: rmdir [OPTION] directory
- **Example**: Create empty files called 'file1' and 'file2'
  - $ rmdir dir1

8) cd: Change directory

- **Syntax**: cd [OPTION] directory
- **Example**: Change working directory to dir1
  - $ cd dir1

9) pwd: Print the present working directory

- **Syntax**: pwd [OPTION]
- **Example**: Print 'dir1' if a current working directory is dir1
  - $ pwd

10) ls or list

- **Syntax: $ ls**
- **Example**: The ls command prints out the contents of a directory

11) more

- more is one of the oldest terminal pagers in the UNIX ecosystem. Originally, more could only scroll down, but now we can use it to scroll up one screen-full at a time, and scroll down either one line or one screen-full:
- **more filename.txt**

12) chown

- chown [OPTIONS] USER [: GROUP] FILE(s)

- USER is the user name or the user ID (UID) of the new owner. GROUP is the name of the new group or the group ID (GID). FILE(s) is the name of one or more files, directories or links. Numeric IDs should be prefixed with the + symbol.

EXPERIMENT-3

OBJECTIVE

Run some python programs on Pi like:

(a)Read your name and print Hello message with name

PROGRAM

```
person = input ('Enter your name: ')
print ('Hello', person)
```

OUTPUT

```
Enter your name: RAM

Hello RAM
```

(b) Read two numbers and print their sum, difference, product and division.

PROGRAM

```
 # Python program to perform Addition Subtraction Multiplication
# and Division of two numbers

num1 = int(input("Enter First Number: "))
num2 = int(input("Enter Second Number: "))

print("Enter which operation would you like to perform?")
ch = input("Enter any of these char for specific operation +,-,*,/: ")

result = 0
if ch == '+':
   result = num1 + num2
elif ch == '-':
   result = num1 - num2
elif ch == '*':
   result = num1 * num2
elif ch == '/':
   result = num1 / num2
else:
   print("Input character is not recognized!")

  print(num1, ch , num2, ":", result)
```

OUTPUT 1: **ADDITION**

Enter First Number: 100
Enter Second Number: 5
Enter which operation would you like to perform?
Enter any of these char for specific operation +,-,*,/: +
100 + 5 : 105

OUTPUT 2: DIVISION

Enter First Number: 20
Enter Second Number: 5
Enter which operation would you like to perform?
Enter any of these char for specific operation +,-,*,/: /
20 / 5 : 4.0

OUTPUT 3: SUBTRACTION

Enter First Number: 8
Enter Second Number: 7
Enter which operation would you like to perform?
Enter any of these char for specific operation +,-,*,/: -
8 - 7 : 1

OUTPUT 4: MULTIPLICATION

Enter First Number: 6
Enter Second Number: 8
Enter which operation would you like to perform?
Enter any of these char for specific operation +,-,*,/: *
6 * 8 : 48

(c) Word and character count of a given string

PROGRAM

```
def char_frequency(str1):
    dict = {}
    for n in str1:
        keys = dict.keys()
        if n in keys:
            dict[n] += 1
        else:
            dict[n] = 1
    return dict
print(char_frequency('google.com'))
```

OUTPUT:

```
{'o': 3, '.': 1, 'g': 2, 'l': 1, 'e': 1, 'c': 1, 'm': 1}
```

(d) Area of a given shape (rectangle, triangle and circle) reading shape and appropriate Values.

PROGRAM

```
from standard input
# Python Program to find the area of triangle
a = 5
b = 6
c = 7
# Uncomment below to take inputs from the user
# a = float(input('Enter first side: '))
# b = float(input('Enter second side: '))
# c = float(input('Enter third side: '))
# calculate the semi-perimeter
s = (a + b + c) / 2
# calculate the area
area = (s*(s-a)*(s-b)*(s-c)) ** 0.5
print('The area of the triangle is %0.2f' %area)
```

OUTPUT

```
The area of the triangle is 14.70
```

EXPERIMENT-4

OBJECTIVE

Run some python programs on Pi like:

a) Print a name 'n' times, where name and n are read from standard input, using for and while loops.

PROGRAM

```
ef string_print(n):
print("THE STRING IS 'Money Heist'")
print("The string will be printed", n ,"times")
for i in range(n):
print("Money Heist")
#input function
string_print(5)
```

OUTPUT

```
THE STRING IS 'Money Heist'
The string will be printed 5 times
Money Heist
Money Heist
Money Heist
Money Heist
Money Heist
```

b) Handle Divided by Zero Exception

PROGRAM

```
n=int(input("Enter the value of n:"))
d=int(input("Enter the value of d:"))
c=int(input("Enter the value of c:"))
try:
```

```
q=n/(d-c)
print("Quotient:",q)
except ZeroDivisionError:
print("Division by Zero!")
```

OUTPUT

```
Enter the value of n:2
Enter the value of d:5
Enter the value of c:5
Division by Zero!
```

(C) Print current time for 10 times to the current time.

PROGRAM

```
Import datetime
x= datetime.datetime.now()
y = x + datetime.timedelta(0,10)
print(x.time())
print(y.time())
```

OUTPUT

```
17:23:46.918031
17:23:56.918031
```

(d) Read a file line by line and print the word count of each line.

PROGRAM

```
L = ["Geeks\n", "for\n", "Geeks\n"]
# writing to file
file1 = open('myfile.txt', 'w')
file1.writelines(L)
```

```
file1.close()

# Using readlines()
file1 = open('myfile.txt', 'r')
Lines = file1.readlines()
 count = 0
# Strips the newline character
for line in Lines:
 count += 1
 print("Line{}: {}".format(count, line.strip()))
```

OUTPUT

Line1: Geeks

Line2: for

Line3: Geeks

EXPERIMENT-5

OBJECTIVE

(a) Light an LED through Python program.

SOLUTION

One of the biggest selling points of the Raspberry Pi is its GPIO, or General-Purpose Input/output ports. They are the little pins sticking out of the circuit board and allow you to plug various devices into your Raspberry Pi. With a little programming, you can then control them or detect what they are doing.

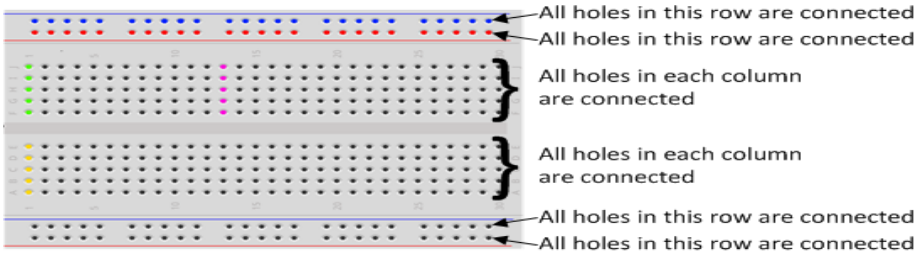
In this tutorial I am going to show you how to light an LED. In addition to your Raspberry Pi running Raspbian, what you will need is:

- **A Breadboard**
- **An LED**
- **A 330 ohm resistor**

- **The Breadboard:**

The breadboard is a way of connecting electronic components to each other without having to solder them together. They are often used to test a circuit design before creating a Printed Circuit Board (PCB).

- The holes on the breadboard are connected in a pattern.



With the breadboard in the CamJam EduKit, the top row of holes are all connected together – marked with red dots. And so are the second row of holes – marked with blue dots. The same goes for the two rows of holes at the bottom of the breadboard.

In the middle, the columns of wires are connected together with a break in the middle. So, for example, all the green holes marked are connected together, but they are not connected to the

yellow holes, nor the purple ones. Therefore, any wire you poke into the green holes will be connected to other wires poked into the other green holes.

The LED

When you pick up the LED, you will notice that one leg is longer than the other. The longer leg (known as the 'anode'), is always connected to the positive supply of the circuit. The shorter leg (known as the 'cathode') is connected to the negative side of the power supply, known as 'ground'.

LEDs will only work if power is supplied the correct way round (i.e. if the 'polarity' is correct). You will not break the LEDs if you connect them the wrong way round – they will just not light. If you find that they do not light in your circuit, it may be because they have been connected the wrong way around.

LED stands for Light Emitting Diode, and glows when electricity is passed through it.

The Resistor

You must ALWAYS use resistors to connect LEDs up to the GPIO pins of the Raspberry Pi. The Raspberry Pi can only supply a small current (about 60mA). The LEDs will want to draw more, and if allowed to they will burn out the Raspberry Pi. Therefore, putting the resistors in the circuit will ensure that only this small current will flow and the Raspberry Pi will not be damaged.

Resistors are a way of limiting the amount of electricity going through a circuit; specifically, they limit the amount of 'current' that is allowed to flow. The measure of resistance is called the Ohm (Ω), and the larger the resistance, the more it limits the current. The value of a resistor is marked with coloured bands along the length of the resistor body.

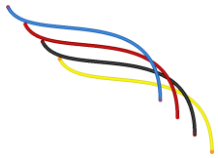
You will be using a 330Ω resistor. You can identify the 330Ω resistors by the colour bands along the body. The colour coding will depend on how many bands are on the resistors supplied:

- If there are four colour bands, they will be Orange, Orange, Brown, and then Gold.
- If there are five bands, then the colours will be Orange, Orange, Black, Black, Brown.

It does not matter which way round you connect the resistors. Current flows in both ways through them.

Jumper Wires:

Jumper wires are used on breadboards to 'jump' from one connection to another. The ones you will be using in this circuit have different connectors on each end. The end with the 'pin' will go into the Breadboard. The end with the piece of plastic with a hole in it will go onto the Raspberry Pi's GPIO pins.



The Raspberry Pi's GPIO Pins :

GPIO stands for **General Purpose Input Output**. It is a way the Raspberry Pi can control and monitor the outside world by being connected to electronic circuits. The Raspberry Pi is able to control LEDs, turning them on or off, or motors, or many other things. It is also able to detect whether a switch has been pressed, or temperature, or light. In the CamJam EduKit you will learn to control LEDs and a buzzer, and detect when a button has been pressed. The diagram below left shows the pin layout for a Raspberry Pi Models A and B (Rev 2 - the original Rev 1 Pi is slightly different), looking at the Raspberry Pi with the pins in the top right corner. The new 40 pin Raspberry Pi's shares exactly the same layout of pins for the top 13 rows of GPIO pins.

Explanation:

So, what is happening in the code? Let's go through it a line at a time:

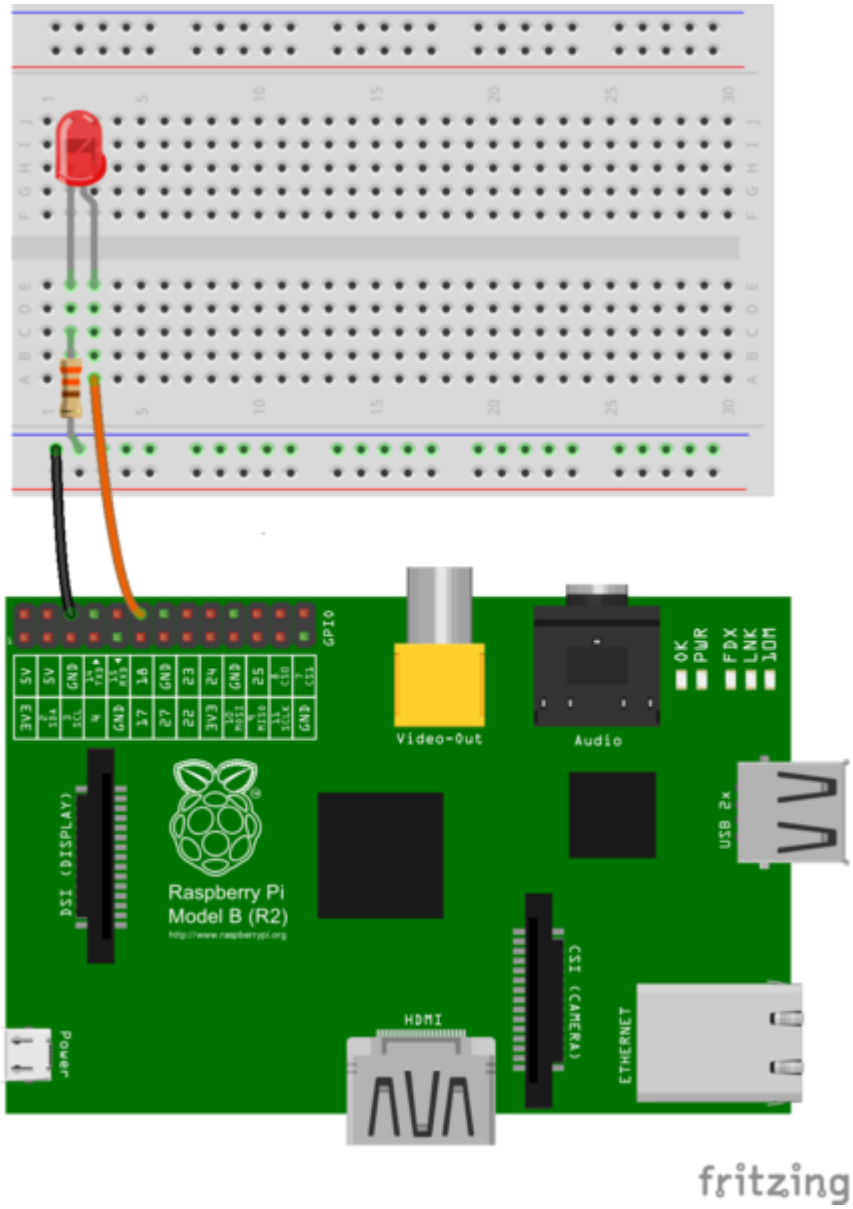
import RPi.GPIO as GPIO	The first line tells the Python interpreter (the thing that runs the Python code) that it will be using a 'library' that will tell it how to work with the Raspberry Pi's GPIO pins. A 'library' gives a programming language extra commands that can be used to do something different that it previously did not know how to do. This is like adding a new channel

	to your TV so you can watch something different.
import time	Imports the Time library so that we can pause the script later on.
GPIO.setmode(GPIO.BCM)	Each pin on the Raspberry Pi has several different names, so you need to tell the program which naming convention is to be used.
GPIO.setwarnings(False)	This tells Python not to print GPIO warning messages to the screen.
GPIO.setup(18,GPIO.OUT)	This line tells the Python interpreter that pin 18 is going to be used for outputting information, which means you are going to be able to turn the pin 'on' and 'off'.
print "LED on"	This line prints some information to the terminal.
GPIO.output(18,GPIO.HIGH)	This turns the GPIO pin 'on'. What this actually means is that the pin is made to provide power of 3.3volts. This is enough to turn the LED in our circuit on.
time.sleep(1)	Pauses the Python program for 1 second
print "LED off"	This line prints some information to the terminal.
GPIO.output(18,GPIO.LOW)	This turns the GPIO pin 'off', meaning that the pin is no longer supplying any power.

Building the Circuit:

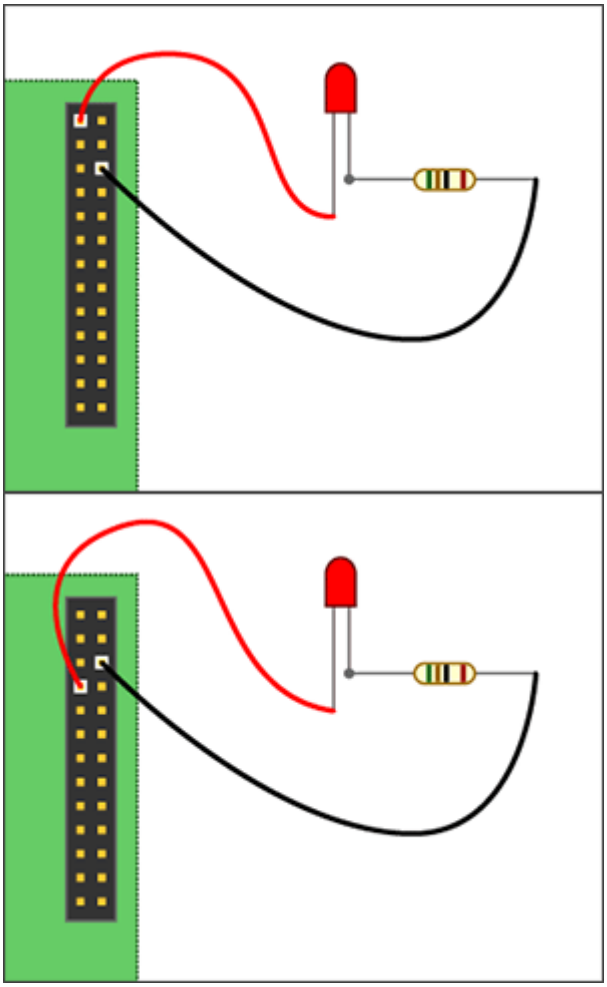
The circuit consists of a power supply (the Raspberry Pi), an LED that lights when the power is applied, and a resistor to limit the current that can flow through the circuit.

You will be using one of the 'ground' (GND) pins to act like the 'negative' or 0 volt ends of a battery. The 'positive' end of the battery will be provided by a GPIO pin. Here we will be using pin 18. When they are 'taken high', which means it outputs 3.3 volts, the LED will light. Now take a look at the circuit diagram below.



You should turn your Raspberry Pi off for the next bit, just in case you accidentally short something out.

- Use one of the jumper wires to connect a ground pin to the rail, marked with blue, on the breadboard. The female end goes on the Raspberry Pi's pin, and the male end goes into a hole on the breadboard.
- Then connect the resistor from the same row on the breadboard to a column on the breadboard, as shown above.
- Next, push the LEDs legs into the breadboard, with the long leg (with the kink) on the right.
- Lastly, complete the circuit by connecting pin 18 to the right hand leg of the LED. This is shown here with the orange wire.



The Code:

You are now ready to write some code to switch the LED on. Turn on your Raspberry Pi and open the terminal window.

Create a new text file "LED.py" by typing the following:

nano LED.py

Type in the following code:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(18,GPIO.OUT)
print "LED on"
GPIO.output(18,GPIO.HIGH)
time.sleep(1)
print "LED off"
GPIO.output(18,GPIO.LOW)
```

Once you have typed all the code and checked it, save and exit the text editor with "Ctrl + x" then "y" then "enter".

Running the Code:

To run this code type:

sudo python LED.py

You will see the LED turn on for a second and then turn off.

If your code does not run and an error is reported, edit the code again using nano LED.py.

Explanation:

So, what is happening in the code? Let's go through it a line at a time:

import RPi.GPIO as GPIO	The first line tells the Python interpreter (the thing that runs the Python code) that it will be using a 'library' that will tell it how to work with the Raspberry Pi's GPIO pins. A 'library' gives a programming language extra commands that can be used to do something different that it previously did not know how to do. This is like adding a new channel to your TV so you can watch something different.
import time	Imports the Time library so that we can pause the script later on.
GPIO.setmode(GPIO.BCM)	Each pin on the Raspberry Pi has several different names, so you need to tell the program which naming convention is to be used.
GPIO.setwarnings(False)	This tells Python not to print GPIO warning messages to the screen.
GPIO.setup(18,GPIO.OUT)	This line tells the Python interpreter that pin 18 is going to be used for outputting information, which means you are going to be able to turn the pin 'on' and 'off'.
print "LED on"	This line prints some information to the terminal.
GPIO.output(18,GPIO.HIGH)	This turns the GPIO pin 'on'. What this actually means is that the pin is made to provide power of 3.3volts. This is enough

	to turn the LED in our circuit on.
time.sleep(1)	Pauses the Python program for 1 second
print "LED off"	This line prints some information to the terminal.
GPIO.output(18,GPIO.LOW)	This turns the GPIO pin 'off', meaning that the pin is no longer supplying any power.

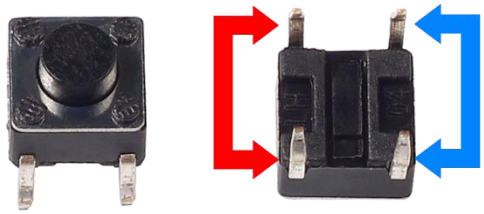
(b) Get input from two switches and switch on corresponding LEDs Controlling an LED by a Button.

SOLUTION:

Introduction:
In this lesson, we will learn how to turn an LED on or off by a button.

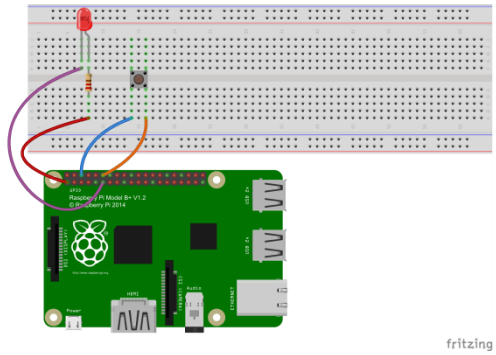
Components:
- 1* Raspberry Pi
- 1* Breadboard
- 1* LED
- 1* Button
- 1* Resistor (220Ω)
- Jumper wires

Principle:
Button
Buttons are a common component used to control electronic devices. They are usually used as switches to connect or disconnect circuits. Although buttons come in a variety of sizes and shapes, the one used here is a6mmmini-button as shown in the following pictures. Pins pointed out by the arrows of same color are meant to be connected.



When the button is pressed, the pins pointed by the blue arrow will connect to the pins pointed by the red arrow (see the above figure), thus closing the circuit, as shown in the following diagrams.

Experimental Procedures:
Step 1: Build the circuit



Step 2: Change directory
cd /home/pi/Sunfounder_SuperKit_Python_code_for_RaspberryPi/
and write the program in Python

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import time

LedPin = 11    # pin11 --- led
BtnPin = 12    # pin12 --- button

Led_status = 1

def setup():
        GPIO.setmode(GPIO.BOARD)       # Numbers GPIOs by physical location
        GPIO.setup(LedPin, GPIO.OUT)   # Set LedPin's mode is output
        GPIO.setup(BtnPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)    # Set BtnPin's
mode is input, and pull up to high level(3.3V)
        GPIO.output(LedPin, GPIO.HIGH) # Set LedPin high(+3.3V) to off led

def swLed(ev=None):
        global Led_status
        Led_status = not Led_status
        GPIO.output(LedPin, Led_status)  # switch led status(on-->off; off-->on)
        if Led_status == 1:
                print 'led off...'
        else:
                print '...led on'

def loop():
        GPIO.add_event_detect(BtnPin, GPIO.FALLING, callback=swLed,
bouncetime=200) # wait for falling and set bouncetime to prevent the callback function from
being called multiple times when the button is pressed
        while True:
                time.sleep(1)   # Don't do anything

def destroy():
        GPIO.output(LedPin, GPIO.HIGH)     # led off
```

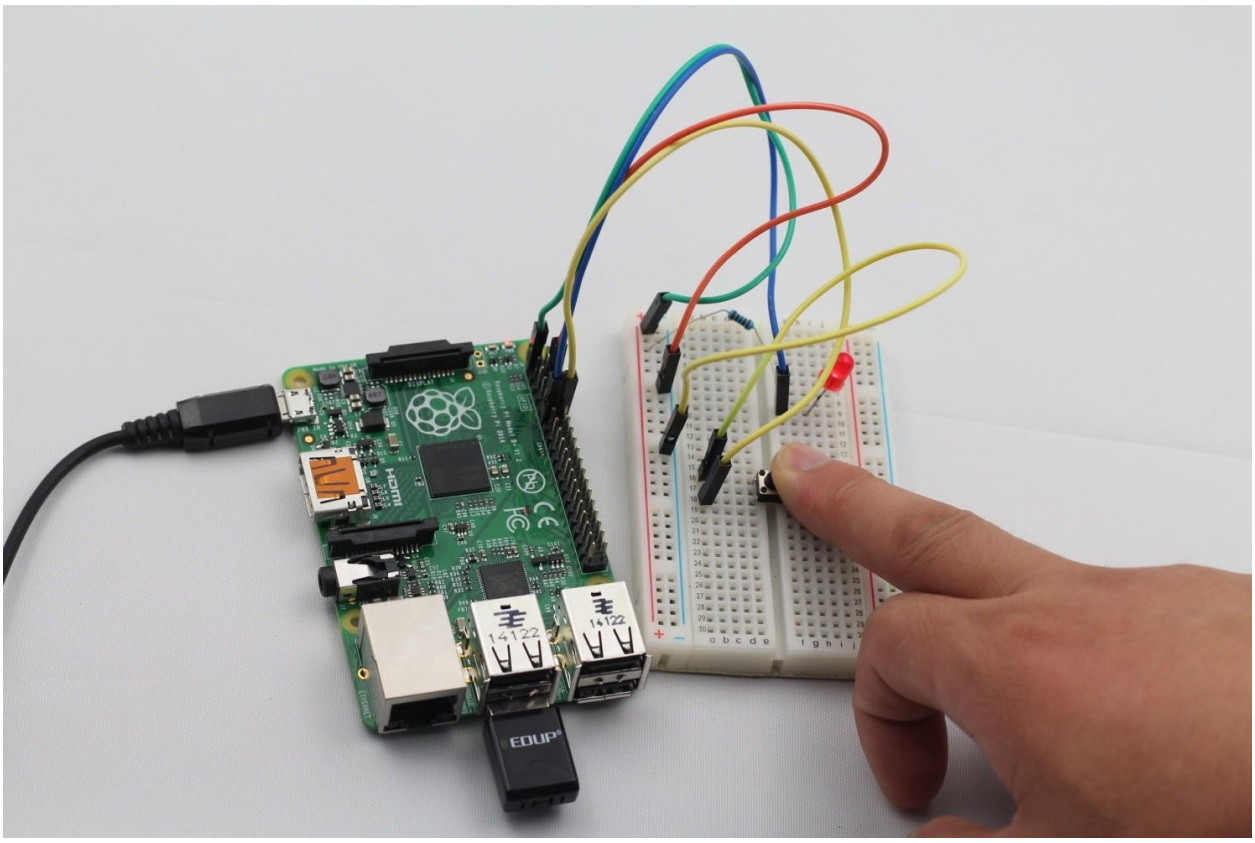
```
        GPIO.cleanup()              # Release resource

if __name__ == '__main__':     # Program start from here
        setup()
        try:
        loop()
except KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the child program destroy() will be executed.
                    destroy()
```

Step 3: Run

sudo python 02_btnAndLed.py

Now, press the button, and the LED will light up; press the button again,and the LED will go out. At the same time, the state of the LED will be printed on the screen.



(c) Flash an LED at a given on time and off time cycle, where the two times are taken from a file.

SOLUTION:

Components required

- One led
- 100 ohm resistor
- Jumper cables

Raspberry Pi GPIO Specifications:

- Output Voltage : 3.3V
- Maximum Output Current : 16mA per pin with total current from all pins not exceeding 50mA

For controlling a Led using Raspberry Pi, both python and the GPIO library is needed.

Installing Python GPIO Library:

Note: Python and GPIO library are preinstalled if you are using Raspbian.

- Make sure your Rasperry Pi is connected to the internet using either a LAN cable or a WiFi adapter.
- Open the terminal by double clicking the LXTerminal icon
- Type the following command to download the GPIO library as a tarball

  wget http://raspberry-gpio-python.googlecode.com/files/RPi.GPIO-0.4.1a.tar.gz

- Unzip the tarball

  tar zxvf RPi.GPIO-0.4.1a.tar.gz

- Change the directory to unzipped location

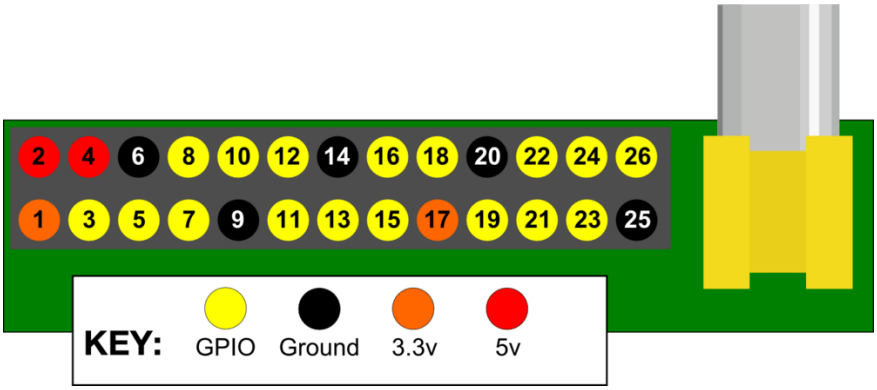
  cd RPi.GPIO-0.4.1a

- Install the GPIO library in python

  sudo python setup.py install

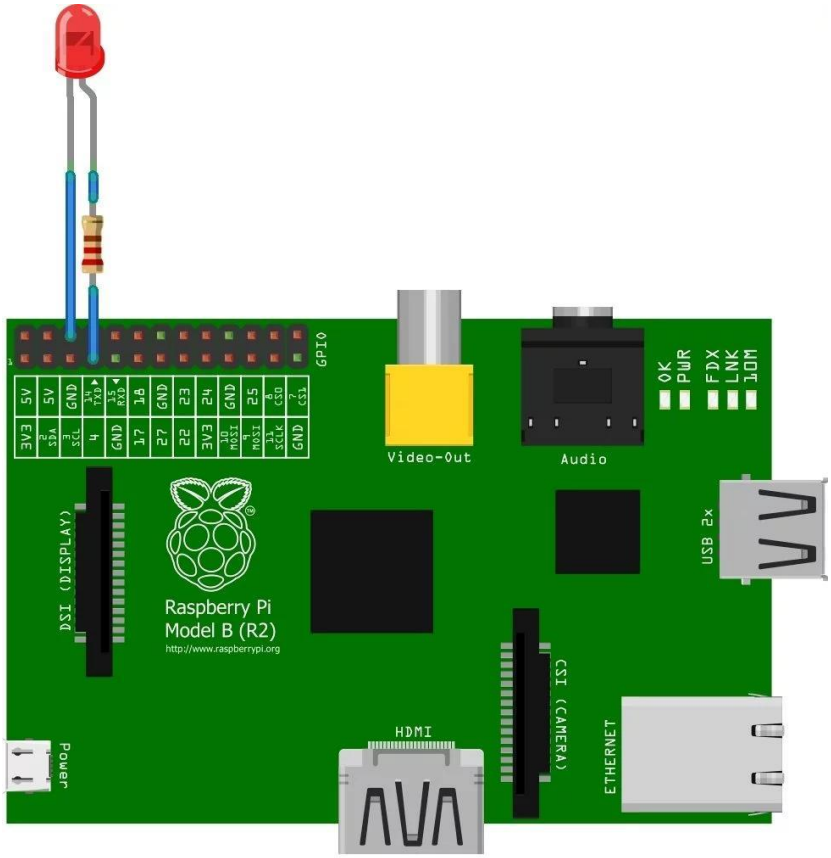
Raspberry Pi GPIO Pin Out:

Raspberry Pi has 17 GPIO pins out of 26 pins



Circuit Diagram:

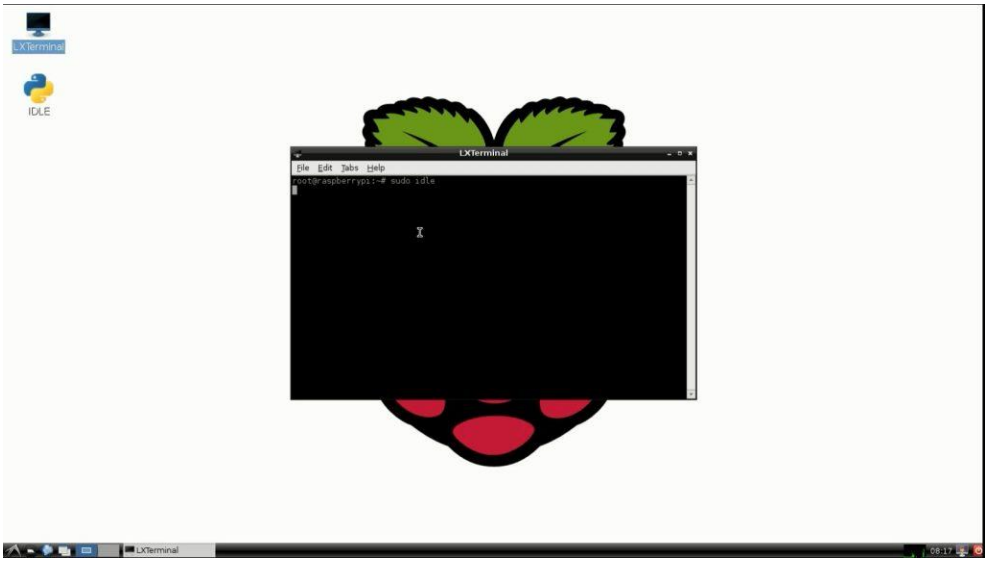
- Connect the Led to 6 (ground) and 11 (gpio) with a 100Ω resistor in series



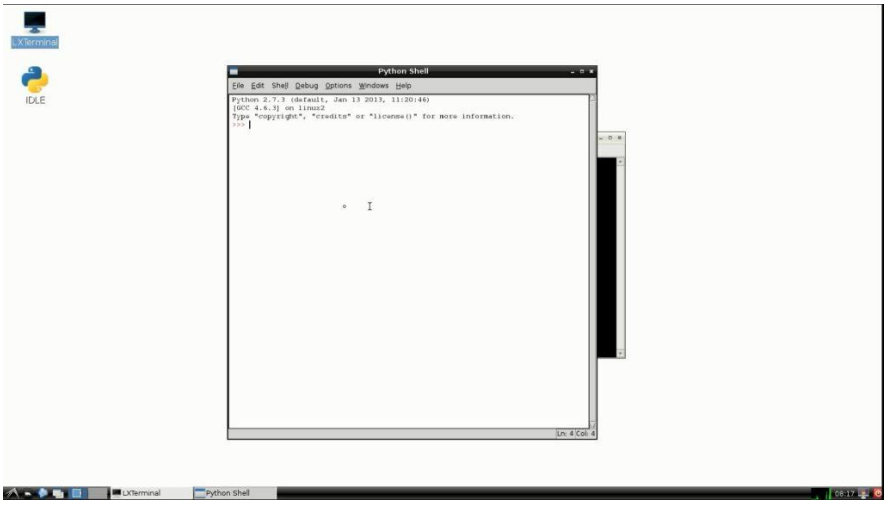
Python Programming:

- Open Terminal
- Launch IDLE IDE by typing

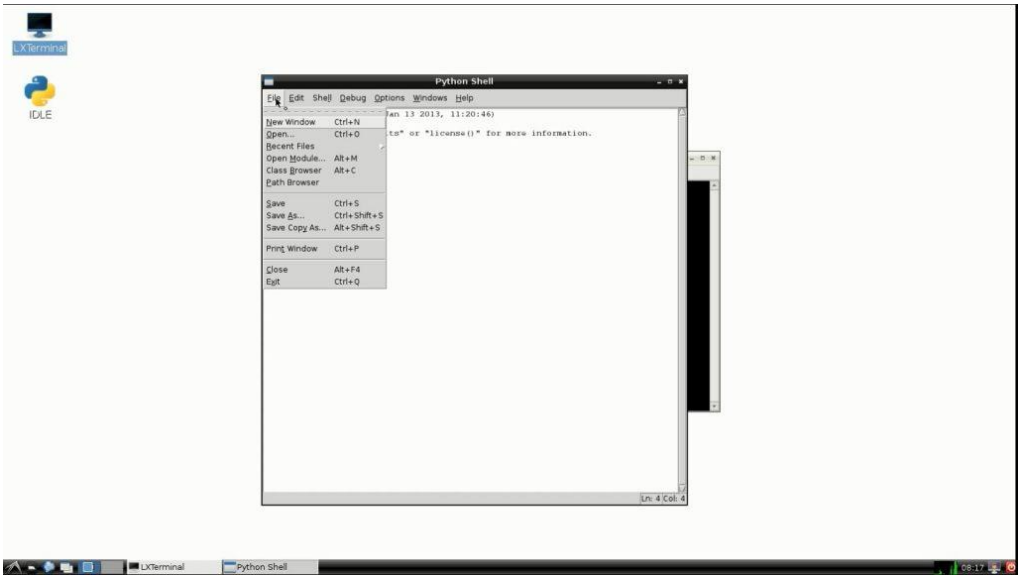
sudo idle



This launches IDLE with superuser privileges which is necessary execute scripts for controlling the GPIO pins

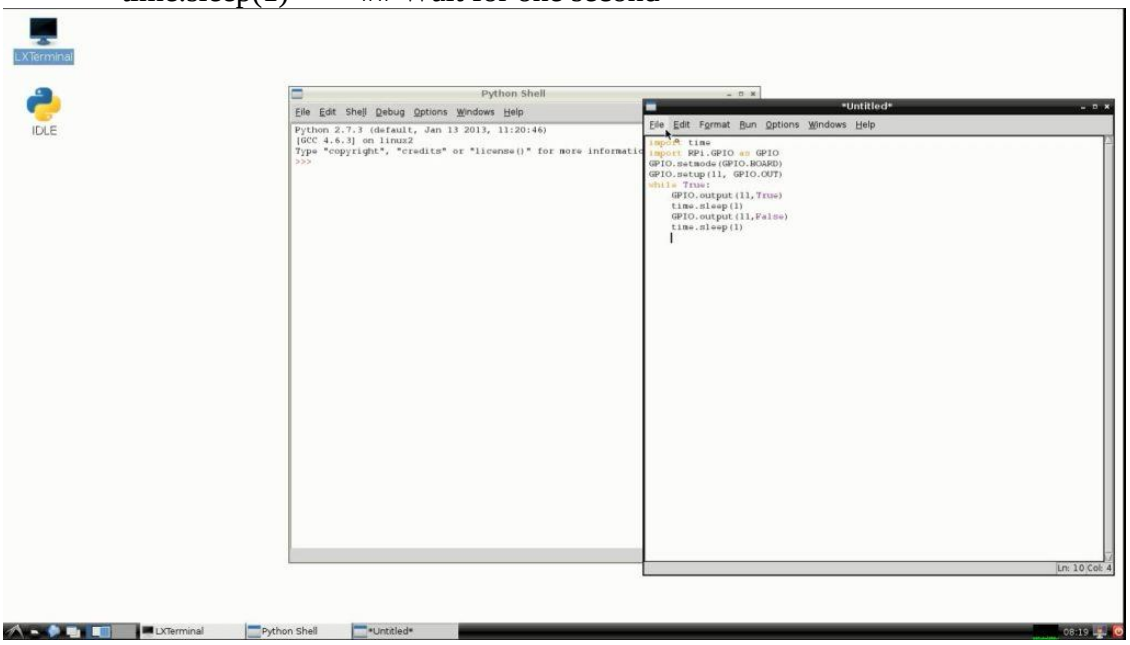


- After the IDLE launches, open a new window by FILE>OPEN or Ctrl+N

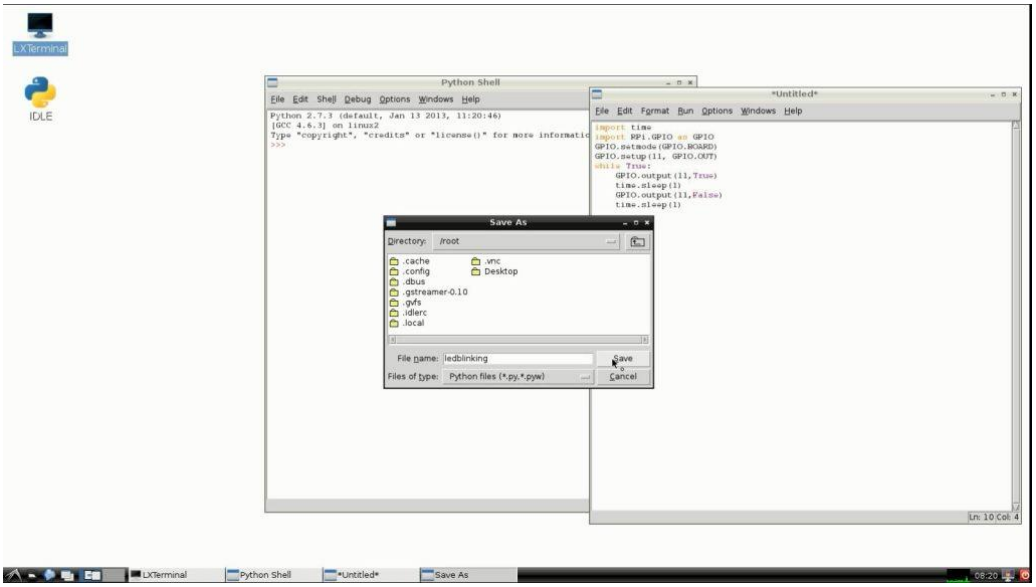


- Type the code below in the window

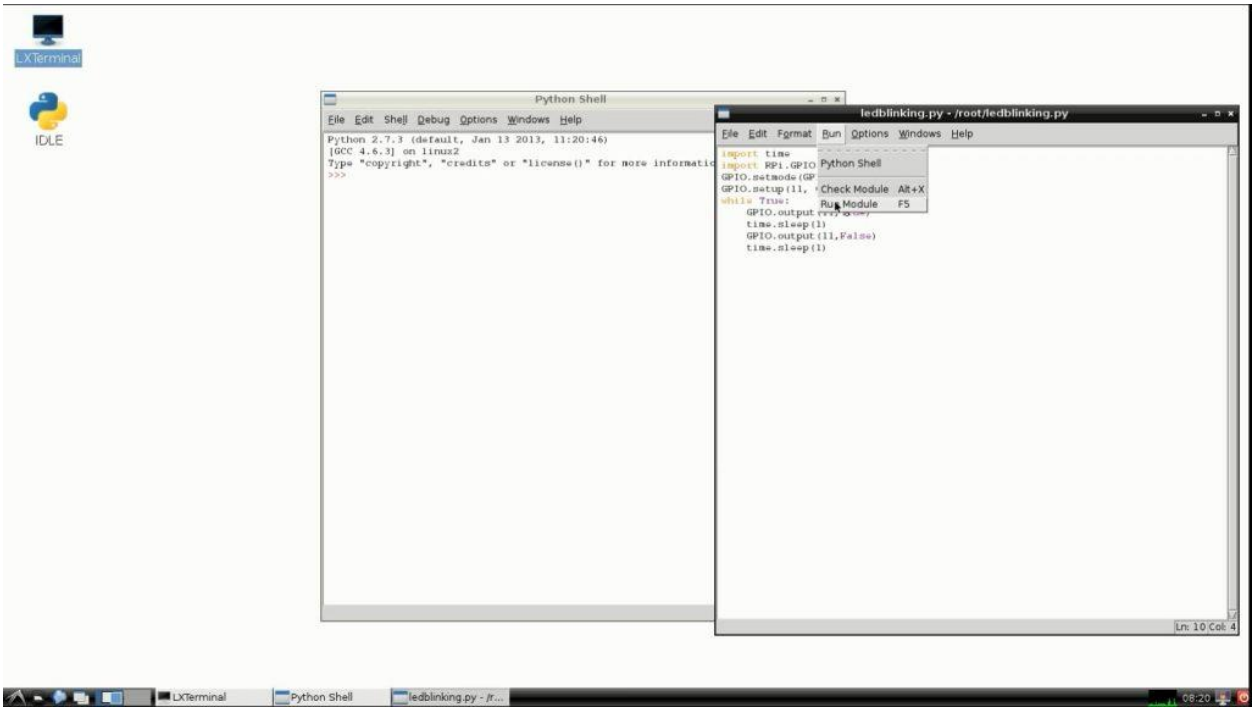
```
import time
import RPi.GPIO as GPIO       ## Import GPIO library
GPIO.setmode(GPIO.BOARD)      ## Use board pin numbering
GPIO.setup(11, GPIO.OUT)      ## Setup GPIO Pin 11 to OUT
while True:
        GPIO.output(11,True)  ## Turn on Led
        time.sleep(1)         ## Wait for one second
        GPIO.output(11,False) ## Turn off Led
        time.sleep(1)         ## Wait for one second
```



- Save the code by FILE>SAVE or Ctrl+S



- To run your code RUN>RUN or Ctrl+F5



- The Led will start blinking now.