

Accessing GPIO pins of R-Pi

e-Yantra Team

June 5, 2015

Contents

1	Objective	3
2	Prerequisites	3
3	Hardware Requirement	3
4	Software Requirement	3
5	Theory and Description	4
6	Experiment	12
6.1	Interfacing an LED with R-Pi	12
6.2	Interfacing a Push button with R-Pi	14
7	Exercise	15
8	References	15

1 Objective

In this tutorial we will learn how to write simple programs to access GPIO pins in an R-Pi.

2 Prerequisites

- Python programming skills
- Basic terminal commands

3 Hardware Requirement

1. Raspberry Pi (I will be using Version 2 Model B)
2. Power adapter
3. Connecting wires
4. LED
5. Push button
6. Resistor (330 ohms)
7. Bread board

4 Software Requirement

1. PyScripter (version 2.7 or above)
2. Mobaxterm (for windows users)

5 Theory and Description

The Raspberry Pi 2 Model B is the second generation Raspberry Pi. Compared to the Raspberry Pi 1 it has:

- A 900MHz quad-core ARM Cortex-A7 CPU
- 1GB RAM

Like the (Pi 1) Model B+, it also has:

- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot
- VideoCore IV 3D graphics core
- Because it has an ARMv7 processor, it can run the full range of ARM GNU/Linux distributions, including Snappy Ubuntu Core, as well as Microsoft Windows 10. [2]

Expansion Header

The Raspberry Pi 2 Model B board contains a single 40-pin expansion header labelled as 'J8' providing access to 26 GPIO pins. (Pins 1, 2, 39 and 40 are also labelled below.)

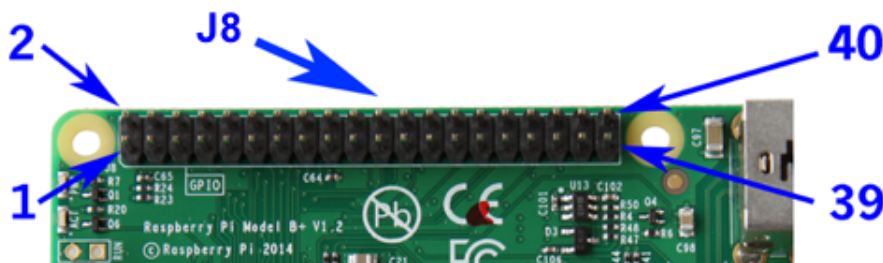


Figure 1: [3]

The diagram below illustrates the pin out diagram of Raspberry Pi 2:

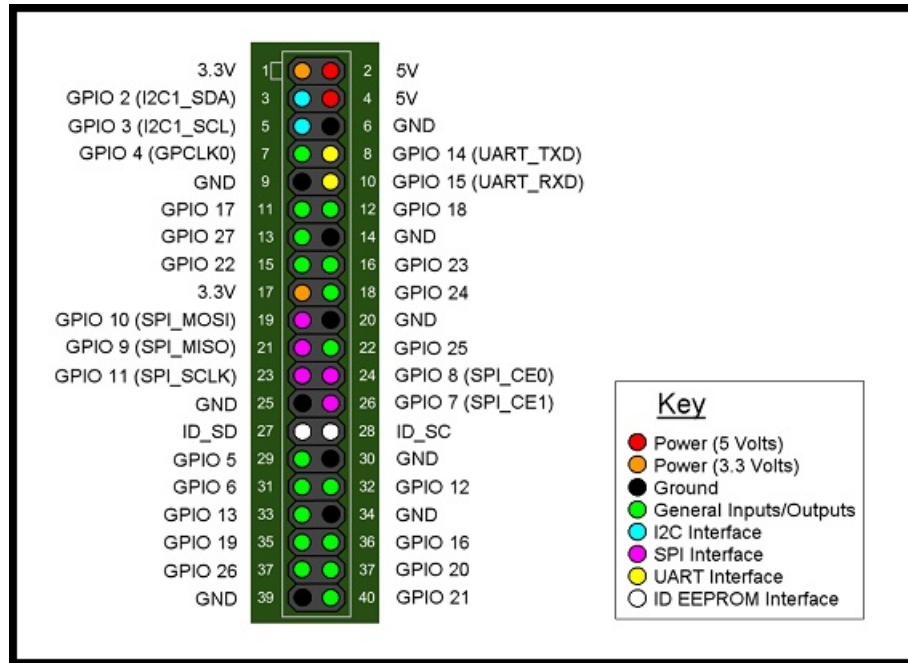


Figure 2: [4]

You must have noticed that the board contains pins named as GPIO (that are used for interfacing input and output devices) and hence in order to refer to the R-Pi pins there exists two modes:

1. **BCM mode:** Referring the pins with the GPIO number
2. **Board mode:** Referring the pins using the IC pin numbers.

Raspbian comes preloaded with Python, the official programming language of the Raspberry Pi and IDLE 3, a Python Integrated Development Environment. And hence we can directly program the Pi using Python. Although we can even program the Pi using C language (but i will be using Python language in this tutorial).

Different methods to program an R-Pi

Before we write a code to access GPIO pins of the Pi let's understand the different ways to program Pi.

1. GUI based programming using IDLE3. In order to do so follow these steps:
 - First, load up IDLE 3 by double-clicking the icon on your LXDE desktop(either on the monitor or using Mobaxterm as explained in the previous tutorial based on establishing a SSH connection).

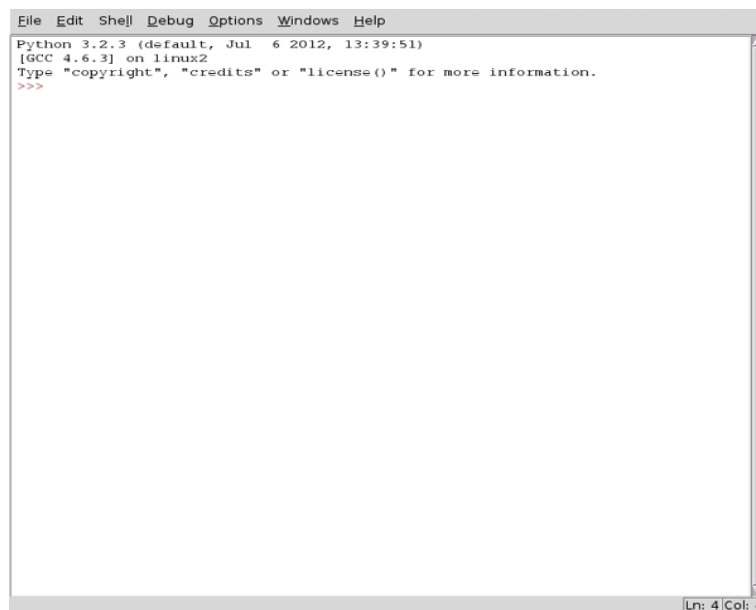
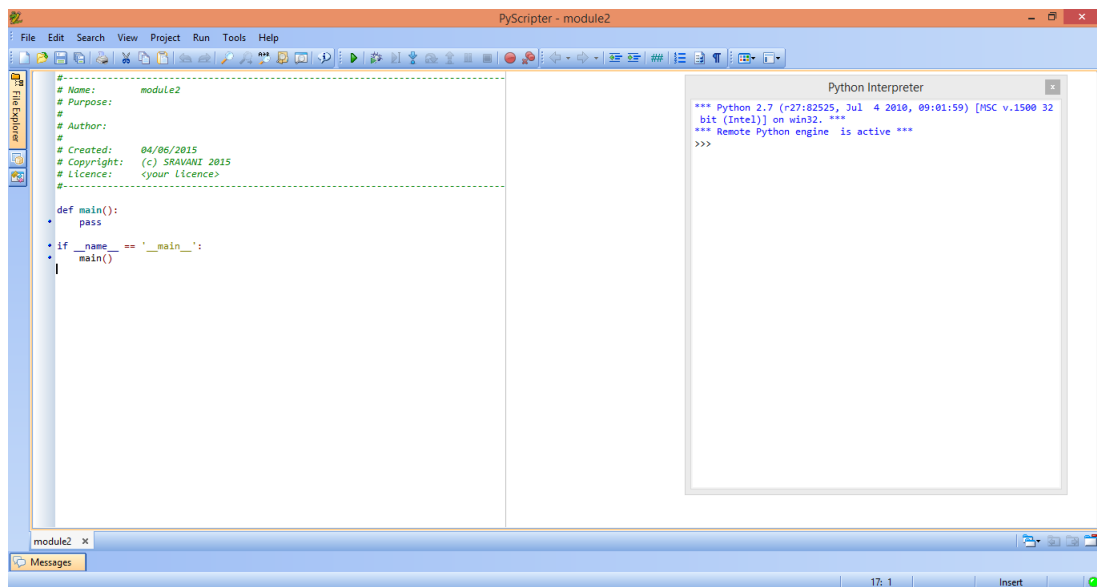
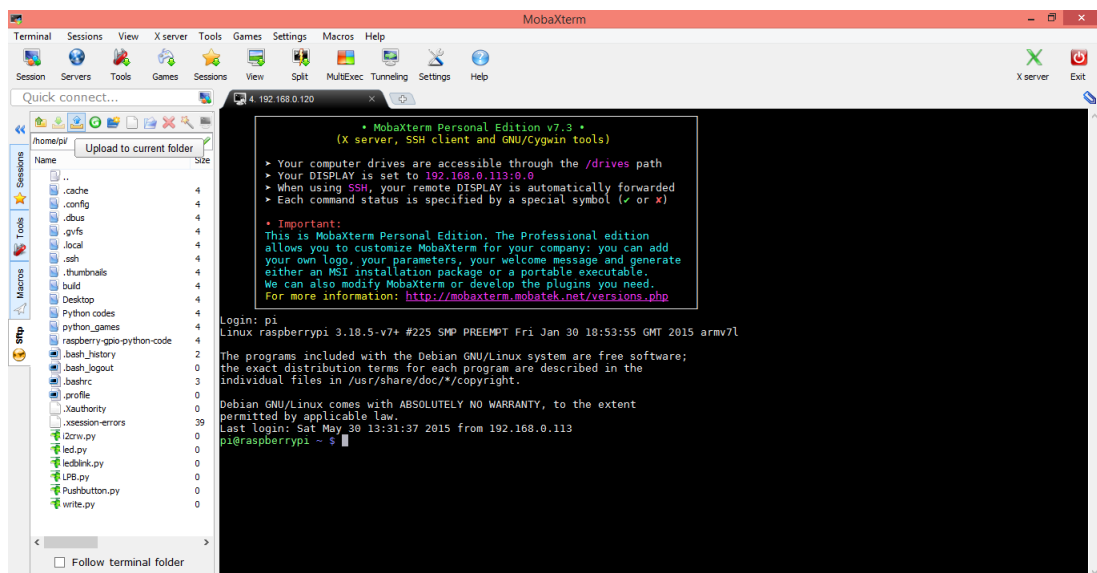


Figure 3: [1]

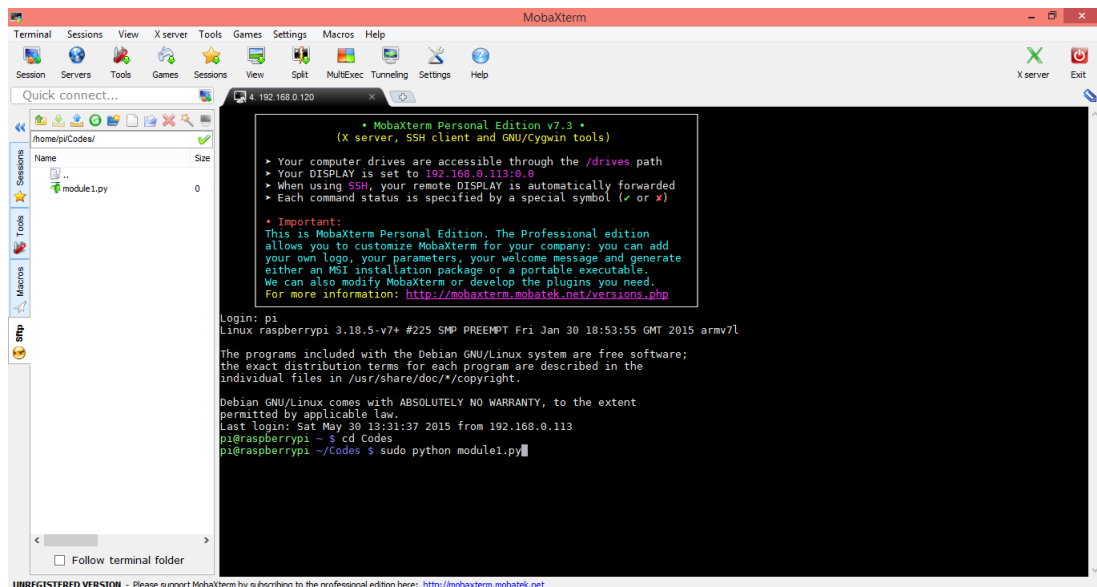
- Click File,New Window, which will then bring up a new blank window which you can type in.
 - Now click File,Save As and save your file in the desired folder.
 - Click Run, and then Run Module or simply press F5.
2. Terminal based programming:
 - Using PyScripter and MobaXterm: PyScripter is a free and open-source Python IDE used for programming in Python.In order to download PyScripter use the following link <https://code.google.com/p/pyscripter/> (I use version 2.5.3).



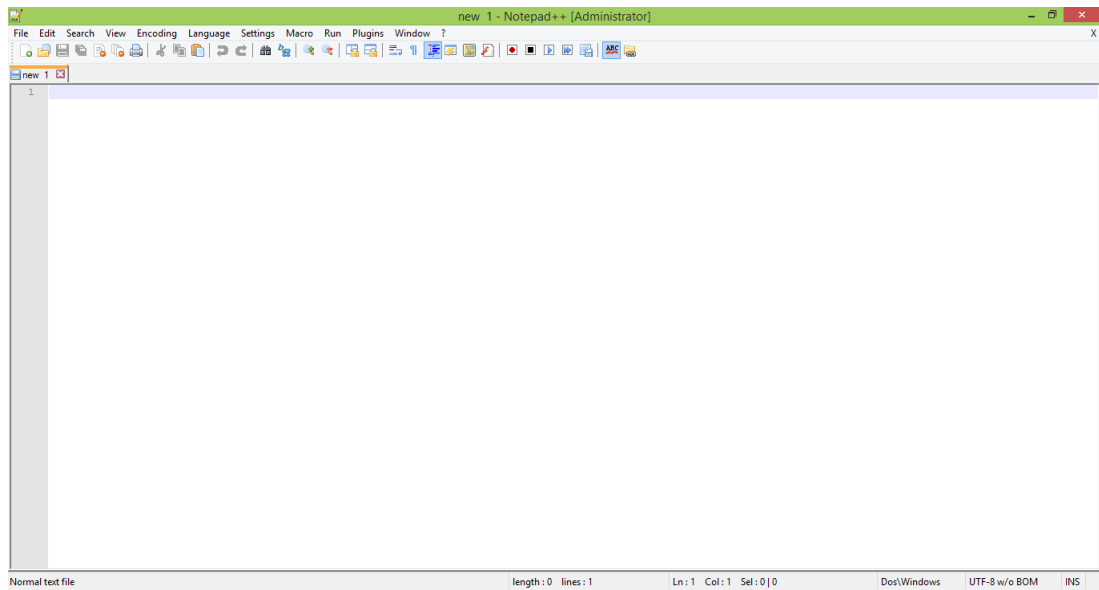
- Open the application. You will see a window like this:
- Delete the text already present and type your program. Once you finish typing the program goto File ↵ Save As and save the program.
- Now open MobaXterm. On the left side you can see the files in /home/pi/ directory and you also have some small icons on the toolbar. Click on the icon 'Upload to current folder' as shown:



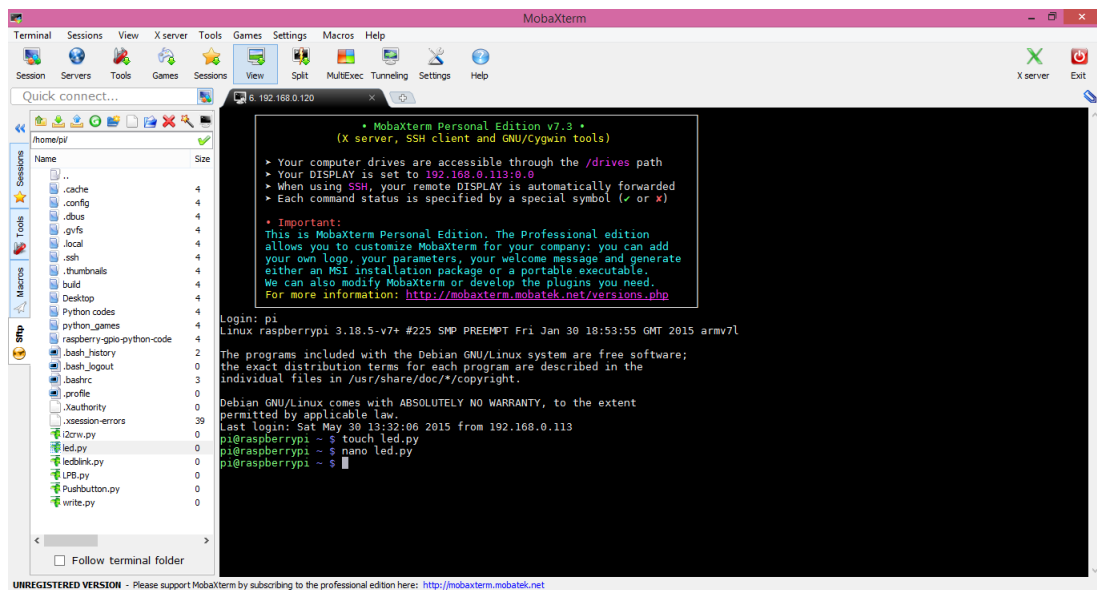
- (d) Upload that file wherein you have written your python program. Note: You can either upload your files to the home directory i.e. /home/pi/ or else you can create a new directory using the icons on the toolbar as illustrated before and upload your python file over there.
- (e) After you have uploaded the code then type the following command on the terminal to execute the code *python filename.py*. In case you have uploaded the file to a directory other than home directory then change the path by typing *cd directoryname* and then *sudo python filename.py* (Note: In order to execute files from any directory other than home directory dont forget to use *cd* command before your directory name and *sudo* python command before your file name that you want to execute.)



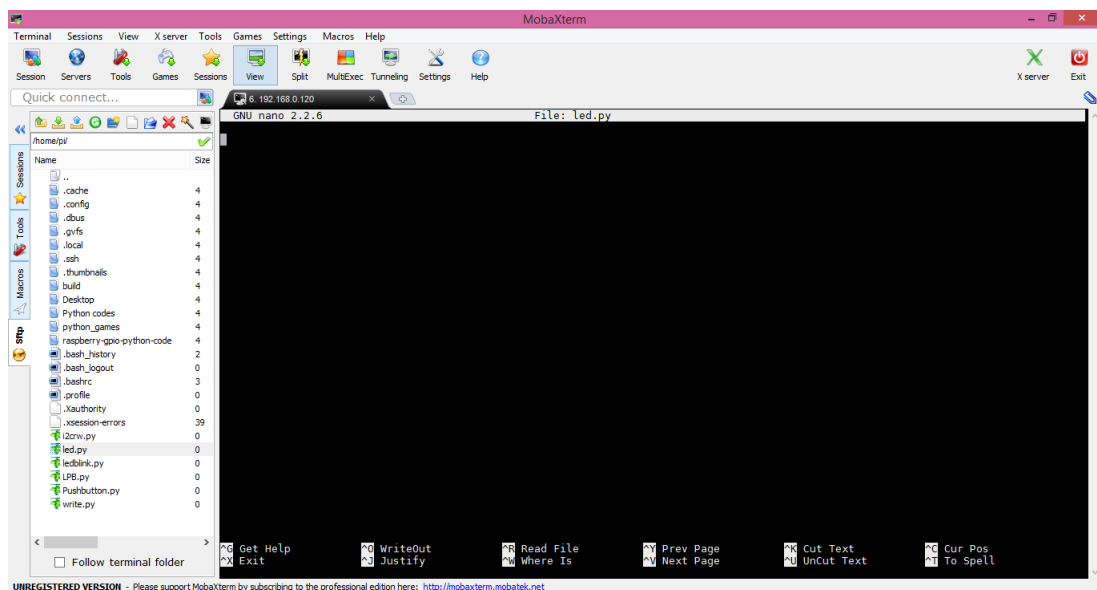
- Using Notepad++ and LXTerminal(or MobaXterm):
Notepad++ is a source code and a Windows text editor that is widely used by programmers.(It is more efficient than other text editors because it prompts you with indentation related errors that count in python programming)
- (a) Open the application and create a new file.



- (b) Type your python code and then save it as filename.py
 - (c) After that follow the steps(3-5) mentioned in the above method i.e using PScripter and MobaXterm.
- Using MobaXterm(for remote operation) or LXTerminal: In this method we can create a python file on the terminal window in the following way:
 - (a) Open the MobaXterm or LXTerminal.
 - (b) Type the command `touch filename.py` to create a file in a directory say home directory
 - (c) Once the file is created in order to edit it type the command `nano filename.py`



(d) A blank file opens as shown



- (e) Type the required code and save the contents by typing *Ctrl + X, Y* (to save the file type Y)
- (f) Then press enter to exit the editor onto terminal window.
- (g) You can now execute the file by using the command *sudo python filename.py* (Don't forget to change the directory if your file is not saved in the home directory. Please refer the steps mentioned before in the document)

Note: Just in case you want to view the contents of the file on the terminal window type *cat filename.py*

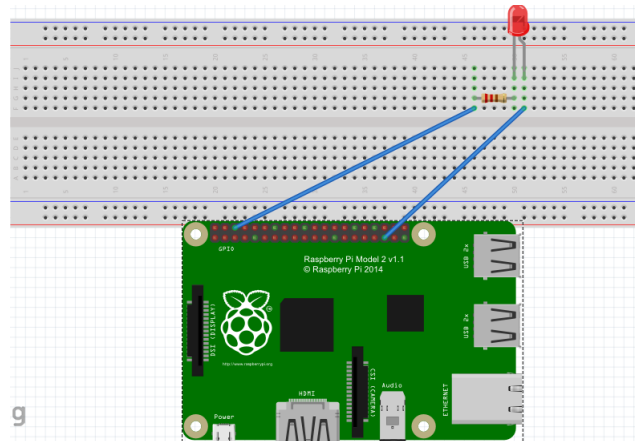
Now we are all set to write basic programs to access GPIO pins in your R-Pi

6 Experiment

In order to access GPIO pins we need to use the Rpi.GPIO package which is usually present in the Python libraries. (but if you are using an R-Pi 2 please ensure that the version of this package is greater than 0.5.10)

6.1 Interfacing an LED with R-Pi

Setting up the Hardware



As shown in the figure :

- Anode of the LED is connected to pin no. 18
- Cathode of the LED is connected to a resistor(330 ohms) which is in turn connected to GND pin on R-Pi 2.

Note: Please refer the theory section for the pin description of R-Pi 2.

Code

```
import RPi.GPIO as GPIO # module to control Pi GPIO channels
import time

# Function name : blink()
# Input : Pin number
# Output : Alternating high and low logic levels on the pin
# Example call: blink(pin)
def blink(pin):
    GPIO.output(pin,GPIO.HIGH)
    time.sleep(1) # to see the blinking effect clearly
                 # we give a delay
    GPIO.output(pin,GPIO.LOW)
    time.sleep(1)
    return

# to use Raspberry Pi board pin numbers
GPIO.setmode(GPIO.BOARD)

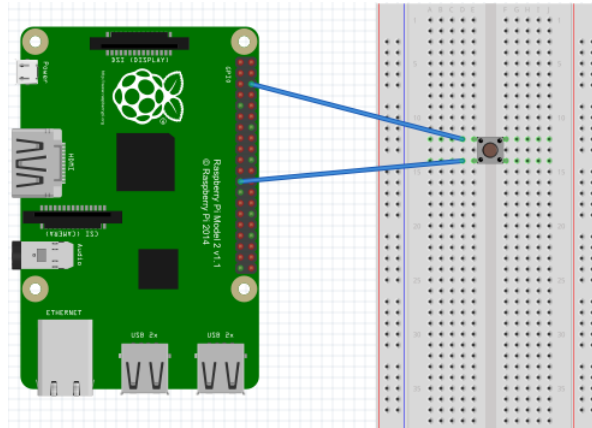
# set up GPIO output channel i.e. on pin 12
GPIO.setup(12, GPIO.OUT)

# blink GPIO18 10 times
for i in range(0,10):
    blink(12) # call

#to clean up all the ports used
GPIO.cleanup()
```

6.2 Interfacing a Push button with R-Pi

Setting up Hardware As shown in the figure :



- One pin of the push button is connected to Ground
- The other pin of the push button is connected to pin no. 12

Note: Please refer the theory section for the pin description of R-Pi 2.
Also ensure that the push button pins you connect to R-Pi shouldnt be shorted.

Code

```
import RPi.GPIO as GPIO # module to control Pi GPIO channels  
import time
```

```
# to use Raspberry Pi board pin numbers  
GPIO.setmode(GPIO.BOARD)
```

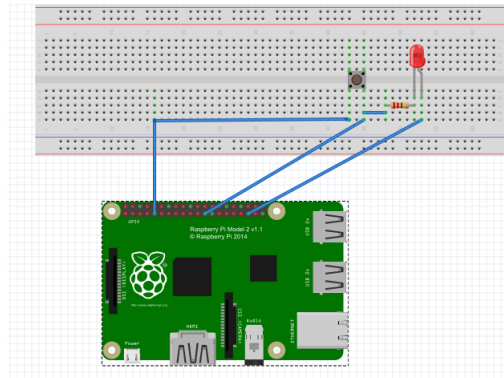
```
GPIO.setup(12, GPIO.IN, pull_up_down=GPIO.PUD_UP) # the input pin(12) is  
# normally pulled up to 3.3V therefore when we press the button a logic  
# low or false value is returned at this pin
```

```
while True:  
    input_state = GPIO.input(12) # a variable to measure the  
                                # logic state of the input pin  
    if input_state == False:  
        print('Button_Pressed')  
        time.sleep(0.2) # this is the min debouncing delay that  
                        # we give in order to ensure that the  
                        # switch is definitely pressed
```

7 Exercise

Now that you know how to write simple programs to access GPIO pins of an R-Pi, try your hand at the following problems

1. Controlling an led using a push button. The setup is shown below.



2. Generating a PWM signal on a GPIO pin to drive an LED

8 References

1. <http://www.engadget.com/2012/09/04/raspberry-pi-getting-started-guide-how-to/>
2. <https://www.raspberrypi.org/products/raspberry-pi-2-model-b/>
3. <http://pi4j.com/images/j8header-photo.png>
4. <http://data.designspark.info/uploads/images/53bc258dc6c0425cb44870b50ab30621>