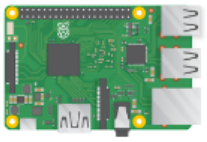
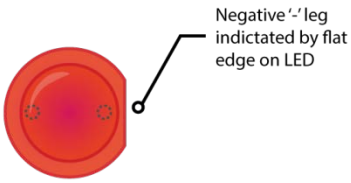
Ex. No: 3 Interface LED/Buzzer with Raspberry Pi, write a program to turn ON LED for 1 sec after every 2 seconds

To Interface LED/Buzzer with Raspberry Pi, and turn ON LED for 1 sec after every 2 seconds, need to build a circuit out of these components:

Breadboard	Male-to-female jumper wires	LED	Resistor	Power component
				

LED contains two legs in which one leg is longer than the other. The long leg is the positive leg, and also called the anode. It should always be connected to the positive side of a circuit. The short leg is the negative leg, called the cathode. It needs to be connected to the negative side.

Some LEDs have legs of the same length. In that case, the positive leg is the leg where the plastic edge of the LED is round. Where the negative leg is, the edge will be flattened, like in the image below



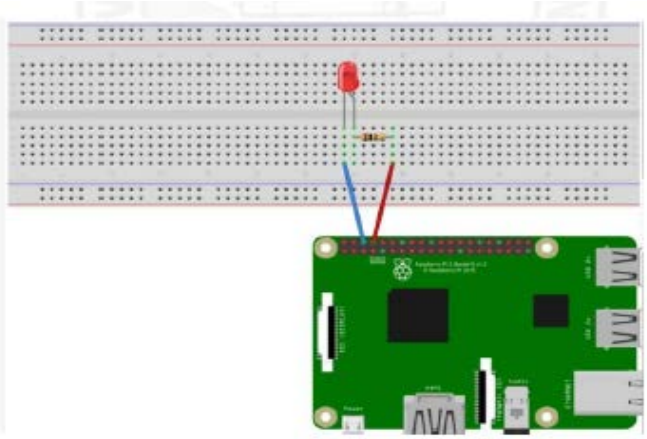
.

Circuit Creation:

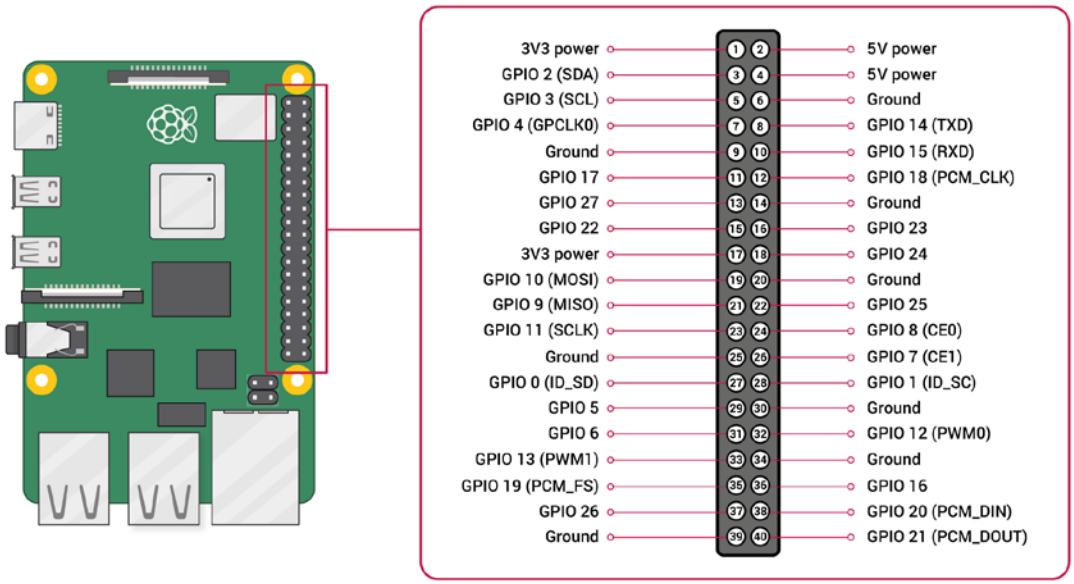
1 Push the positive leg of the LED into row 1 of your breadboard, close to the left side of the ravine. Place the negative leg into row 1 on the other side of the ravine.

2 A resistor is a non-polarised component, so it doesn't matter which way around to put it into the breadboard. Push one leg into the same row that the negative LED leg is in, so it connects to the LED. Push the other resistor leg into any other free row on the right side of the ravine.

3 Now take a male-to-female jumper wire and push the male end into the same row as the LED, on the left side of the ravine near the LED's positive leg. Push the female end onto the 3V3 GPIO pin.



GPIO PIN DIGRAM



- Now, connect components to the ground (GND) GPIO pin:
- Make sure that Raspberry Pi is powered on.
- Take another male-to-female jumper wire and push the male end into the same row as the resistor's second leg, on the same side of the ravine.
- Then push the female end onto your GND pin. Now LED is light up.

If the LED doesn't light, try the following:

1) Check Raspberry Pi is on
2) Check all components are pushed firmly into the breadboard
3) Check LED is the right way around
4) Make sure the legs of the components are on the right side of the ravine
5) Try another LED

Python Code:

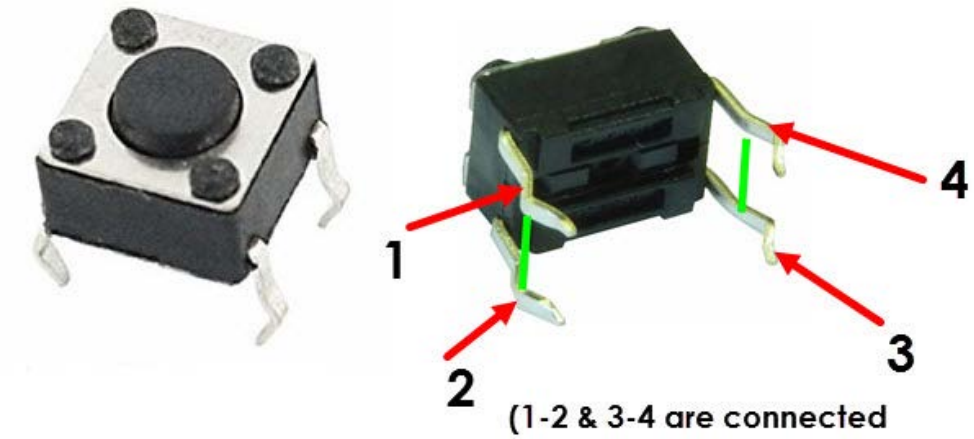
```
import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

GPIO.setup(8,GPIO.OUT)

while True:

        GPIO.output(8, True)

        print("LED is on")

        time.sleep(1)

        GPIO.output(8, False)

        print("LED is off")

        time.sleep(2)
```

Ex. No: 4 Interface Push button/Digital sensors (IR/LDR) with Raspberry Pi, write a program to turn ON LED when push button is pressed or at sensor detection.

- The GPIO pins of the Raspberry Pi is an important feature as they enable the Raspberry Pi to interface with external Physical Components like LEDs, Motors, Buttons, etc.
- The GPIO Pins or General Purpose Input Output Pins, as the name suggests, can be configured as either Output Pin or an Input Pin.
- If it is setup as an Output Pin, the GPIO Pin drives an Output Device like an LED. On the contrary, if the GPIO Pin is configured as an Input Pin, it will read the incoming data from an external device, like a Button, in this scenario.
- From the above statements, it is clear that if the Raspberry Pi wants to read the value from an external device, the corresponding GPIO pin must be declared as an Input Pin.
- But when a GPIO Pin of the Raspberry Pi is declared as Input, it must be 'tied' to High or Low or else it is called as a Floating Input Pin. A Floating Input is a pin which is defined as input and left as it is.
- Any Digital Input Pin is very sensitive and catches even the slightest changes and will pick up the stray capacitances from your finger, breadboard, air etc.
- In order to avoid this, a Digital Input Pin must be tied to VCC or GND with the help of Pull-up or Pull-down resistors.

Basics of Push Button

- Push Button is simplest of devices and it is the basic input device that can be connected to any controller or processor like Arduino or Raspberry Pi.
- A push button in its simplest form consists of four terminals. In that, terminals 1 and 2 are internally connected with each other and so are terminals 3 and 4.

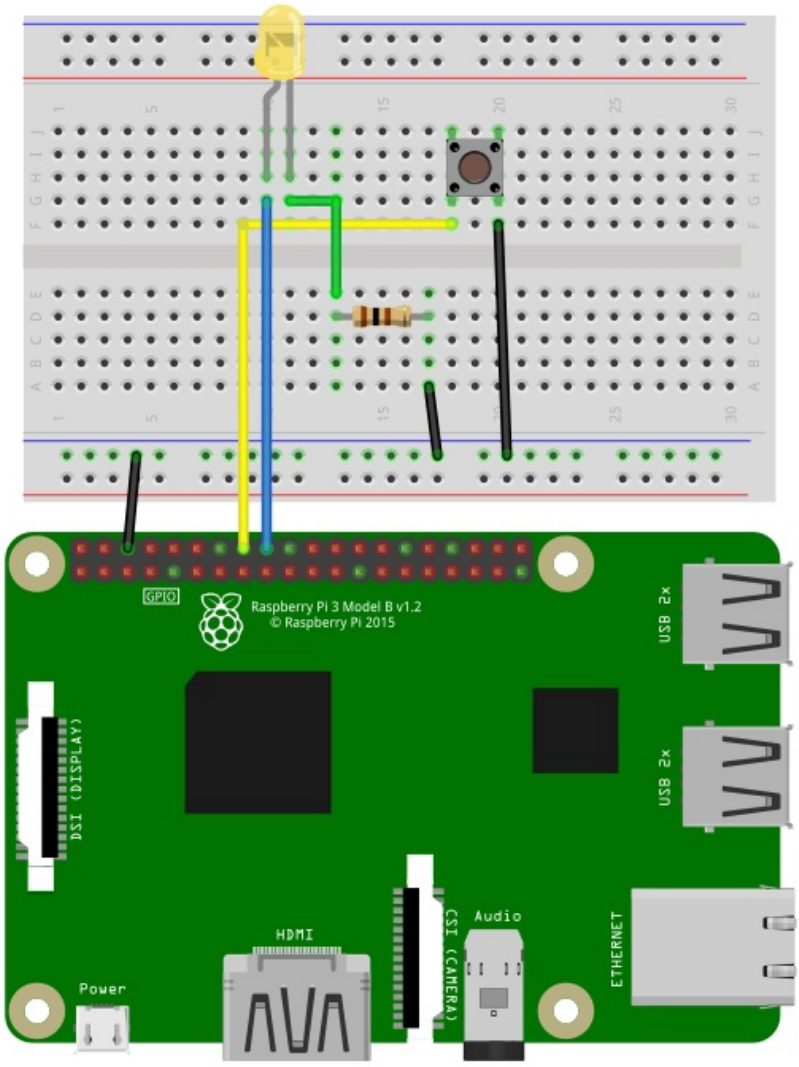


Interfacing a Push Button with Raspberry Pi

- As mentioned in the "GPIO as Input" section, when a GPIO pin is declared as Input, it must be connected to VCC or GND with the help of either a Pull-up resistor or a Pull-down resistor.

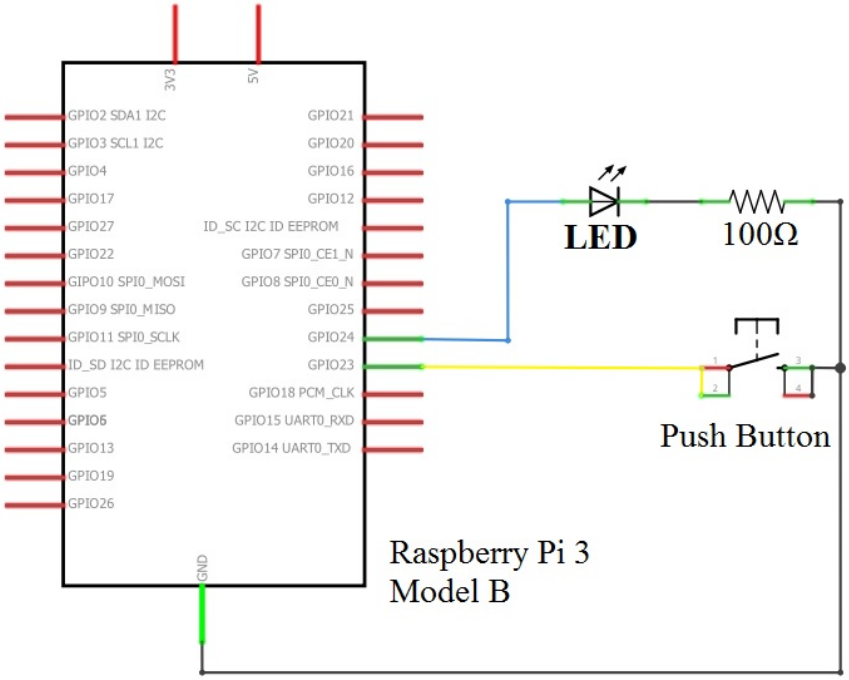
- But, modern boards like Arduino and Raspberry Pi have a feature of Internal Pull-up or Internal Pull-down. Using this feature, the Pin will be pulled-High or Low from the inside of the chip.
- While defining a GPIO pin of the Raspberry Pi as Input, add an additional statement in the program to activate the internal pull-up or pull-down.

Circuit Diagram



Components Required

- Raspberry Pi
- Push Button
- 5mm LED
- 100Ω Resistor (1/4 Watt)
- Mini Breadboard
- Connecting Wires
- Power Supply



Python Code:

```
import RPi.GPIO as GPIO
import time

button = 16
led    = 18

def setup():
      GPIO.setmode(GPIO.BOARD)
      GPIO.setup(button, GPIO.IN, pull_up_down=GPIO.PUD_UP)
      GPIO.setup(led, GPIO.OUT)
def loop():
       while True:
            button_state = GPIO.input(button)
            if  button_state == False:
               GPIO.output(led, True)
               print('Button Pressed...')
               while GPIO.input(button) == False:
                 time.sleep(0.2)
            else:
               GPIO.output(led, False)
def endprogram():
       GPIO.output(led, False)
       GPIO.cleanup()


if __name__ == '__main__':

        setup()

        try:
             loop()

        except KeyboardInterrupt:
             print 'keyboard interrupt detected'
             endprogram()
```