# Accessing GPIO pins of R-Pi

e-Yantra Team

June 9, 2016

## Contents

## 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.)

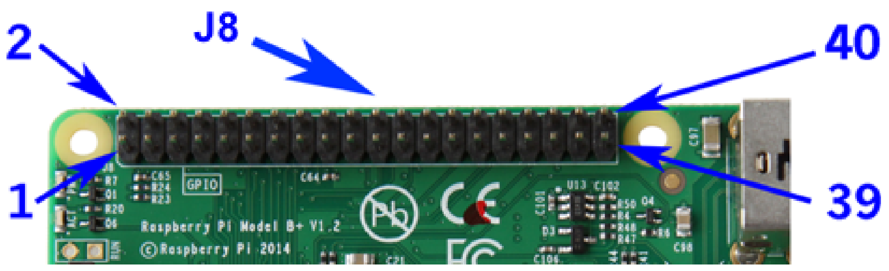


Figure 1: [3]

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

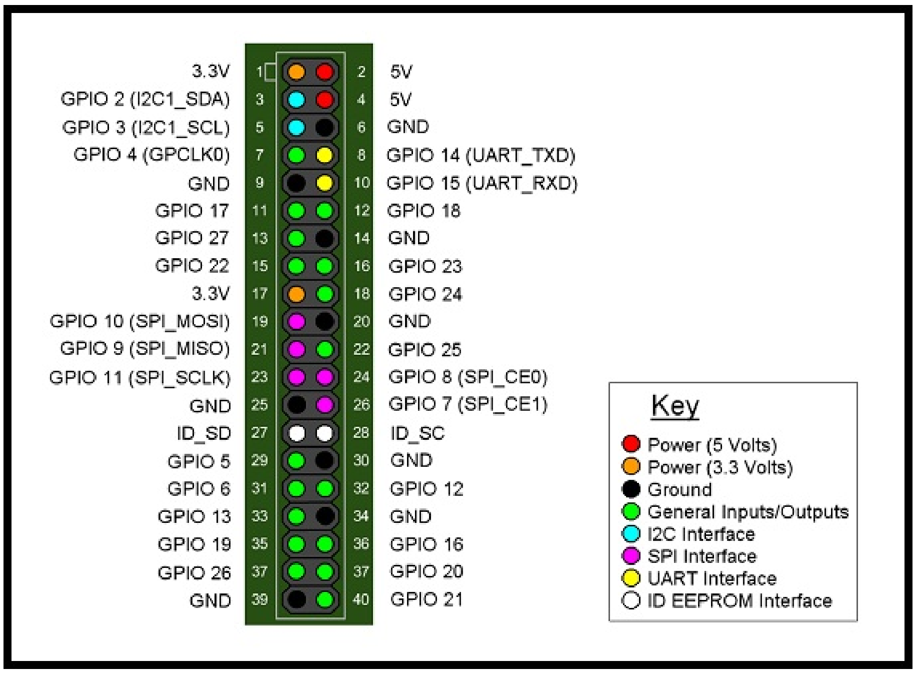


Figure 2: [4]

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:

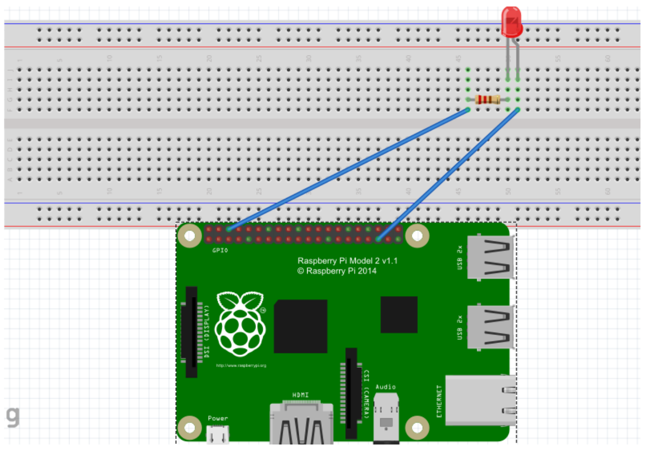
1. **BCM mode:** Referring the pins with the GPIO number
2. **Board mode:** Referring the pins using the IC pin numbers.

## 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(BCM mode)

#### Setting up the Hardware



As shown in the figure :

- Anode of the LED is connected to GPIO 19(IC pin 35).
- Cathode of the LED is connected to a resistor(330 ohms) which is in turn connected to GND pin on R-Pi 2.

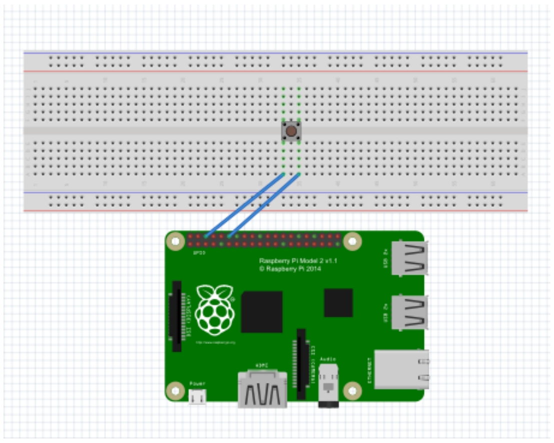
Note: Please refer the theory section for the pin description of R-Pi 2.

**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 BCM pin
GPIO.setmode(GPIO.BCM)

# setting the GPIO 19 as output(i.e IC pin 35) since we are using board i
# so refering the IC pin
GPIO.setup(19, GPIO.OUT)

# blink GPIO 19(i.e IC pin 35) 10 times
for i in range(0,10):
        blink(19) # call

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

## 6.2 Interfacing a Push button with R-Pi(Board Mode)

**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 IC pin no. 12

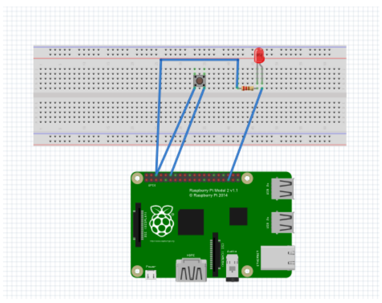
Note: Please refer the theory section for the pin description of R-Pi 2.
Also ensure that the push button pins you connect to R-Pi shoudlnt be shorted.

**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 IC pin(12)
# 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
```

## 6.3 Controlling an led using a push button.

**Setting up Hardware** As shown in figure:



- One pin of the push button is connected to Ground(Pin 9)
- The other pin of the push button is connected to IC pin no. 12
- The anode of led is connected to IC pin 35 of raspberry pi
- The cathode of led is connected to the the resistor of 300 ohms which is then connected to the ground.

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 should not be shorted.

**Code**

```
import RPi.GPIO as GPIO # module to control Pi GPIO channels
import time
# to use Raspberry Pi board pin numbers
#GPIO.cleanup()
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
GPIO.setup(35,GPIO.OUT)
i=0  # flag is set to zero
while True:       # COntinuous loop
    if GPIO.input(12)==False:
        if i==0:
            GPIO.output(35,GPIO.HIGH)
            if GPIO.input(12)==False:
                i=1
                time.sleep(0.5)
```

```
if GPIO.input(12)==False:
    if i==1:
        GPIO.output(35,GPIO.LOW)
        if GPIO.input(12)==False:
            i=0
            time.sleep(0.5)
```

## 7 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