'ODROID-N2' on this page refers to the ODROID-N2 series (**N2**, **N2+**, **N2L**).

RPi.GPIO for ODROID

- Tested on **C1+/C2/N2/XU4 + Shifter Shield**.
- Thanks to @jfath from our forum for porting this useful library.
  - You can visit its Github repository.
- You can use **WiringPi** and **WiringPi-Python** alternatively.
  - Visit the wiki page: {EACH_PRODUCT} - application_note - gpio - wiringpi.

RPi.GPIO is a library for using GPIO on a Raspberry Pi with Python. It implements basic GPIO functions and it's easy to use.

Install

Clone the repository.

target

```
$ sudo apt-get install git python-dev
$ git clone https://github.com/hardkernel/RPi.GPIO-Odroid
```

Install by typing this command.

target

```
$ cd RPi.GPIO-Odroid
$ sudo python setup.py build install
```

Installation does automatically.

Example Code

Let's do an example.
Before you proceed, make sure **you've connected a jump cable** from physical pin

- C1+/C2/N2/XU4 with Shifter Shield: **#13 to #31**.

- XU4 without Shifter Shield: **#13 to #19**.

The Github repository has this example file in RPi.GPIO-Odroid/test/simplerw.py.
You can run it by enter this command.

target

```
$ sudo python simplerw.py
# results
To read output correctly, jumper pin 13 (bcm27) to pin 31 (bcm6)
Press Ctrl-C to exit
('*****Input pin state (Output HIGH) ', 1, '*****\n')
('*****Input pin state (Output LOW) ', 0, '*****\n')
('*****Input pin state (Output HIGH) ', 1, '*****\n')
('*****Input pin state (Output LOW) ', 0, '*****\n')
('*****Input pin state (Output HIGH) ', 1, '*****\n')
```

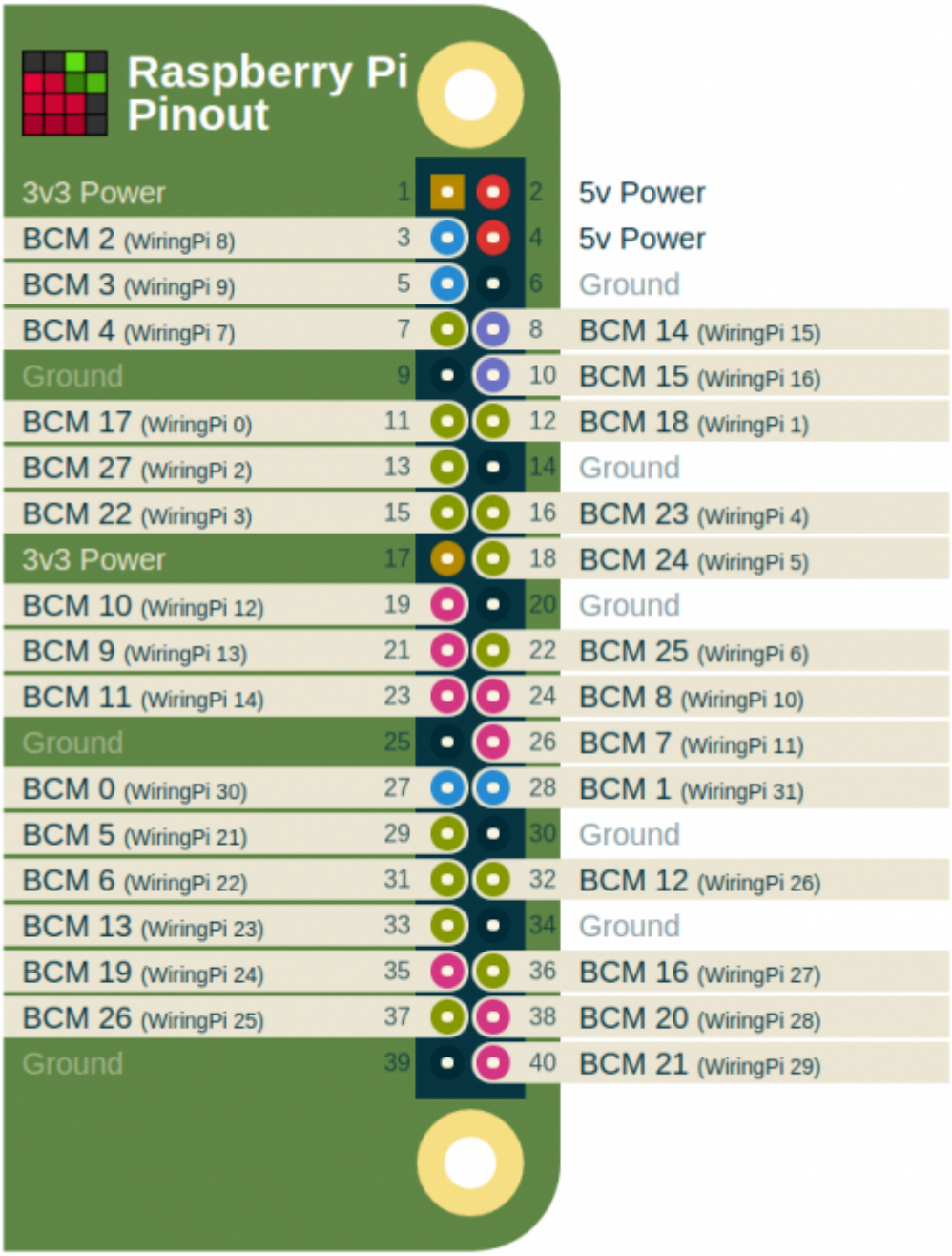
```
import RPi.GPIO as GPIO
import time

LedPinW = 27    # pin13, bcm27
LedPinR = 6    # pin31, bcm6

def setup():
  GPIO.setmode(GPIO.BCM)       # Numbers GPIOs by chip numbering scheme
  GPIO.setup(LedPinR, GPIO.IN, pull_up_down=GPIO.PUD_UP)   # Set LedPin's
mode is input
  GPIO.setup(LedPinW, GPIO.OUT)   # Set LedPin's mode is output
  GPIO.output(LedPinW, GPIO.HIGH) # Set LedPin high(+3.3V) to turn on led

def blink():
  while True:
    GPIO.output(LedPinW, GPIO.HIGH)  # led on
    time.sleep(2)
    pstate=GPIO.input(LedPinR)
    print("*****Input pin state (Output HIGH) ", pstate, "*****\n")
    time.sleep(2)
    GPIO.output(LedPinW, GPIO.LOW) # led off
    time.sleep(2)
    pstate=GPIO.input(LedPinR)
    print("*****Input pin state (Output LOW) ", pstate, "*****\n")
    time.sleep(2)

def destroy():
  GPIO.output(LedPinW, GPIO.LOW)   # led off
  GPIO.setup(LedPinW, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)   # Set LedPin's
mode is input
  GPIO.cleanup()                 # Release resource

if __name__ == '__main__':     # Program start from here
```

```
  print('To read output correctly, jumper pin 13 (bcm27) to pin 31 (bcm6)')
  print('Press Ctrl-C to exit')
  setup()
  try:
    blink()
  except KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the child program
destroy() will be  executed.
    destroy()
```

As you can see, this example code confirms operation of pin **output/input** and controlling **pull up/down mode**.

About BCM numbering

Actually BCM numbering for GPIO pin map is for Raspberry Pi products. So some people might not be familiar with this.
But fortunately, there's a helpful website offers different kind of pin maps of Raspberry Pi, and of course we can refer to.



- Screenshot captured on pinout.xyz

ODROID's 40 pin structure is equal to Raspberry Pi has, at least in vout and ground pins. Futher, this page provides with WiringPi numbering, and our each product provides WiringPi numbering pin map (as well as WiringPi library) so that you can write a source code with this.

References

[1] https://forum.odroid.com/viewtopic.php?f=97&t=30577
[2] https://github.com/jfath/RPi.GPIO-Odroid
[3] https://github.com/joshua-yang/RPi.GPIO-Odroid
[3] https://sourceforge.net/p/raspberry-gpio-python/wiki/Home/
[4] https://pinout.xyz/pinout/wiringpi