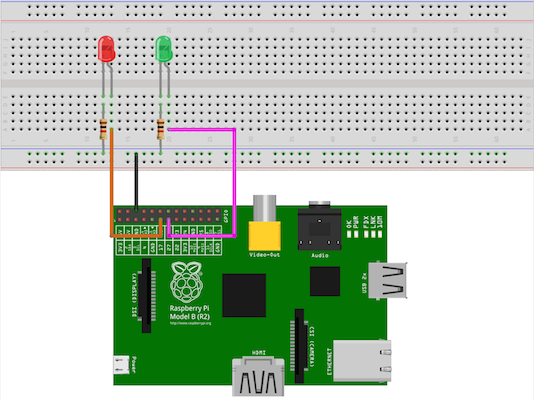
[**http://www.hardcopyworld.com/gnuboard5/bbs/board.php?bo\_table=lecture\_rpi&wr\_id=4**](http://www.hardcopyworld.com/gnuboard5/bbs/board.php?bo_table=lecture_rpi&wr_id=4)

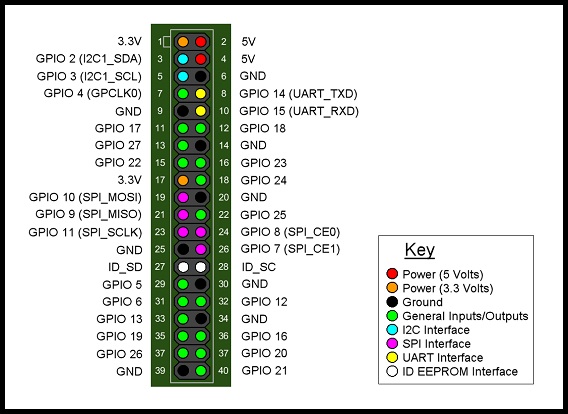
**라즈베리 파이 GPIO 제어 (CONSOLE, PYTHON)**

작성자 [**[http://www.hardcopyworld.com/gnuboard5/data/member/go/godstale.gif](http://www.hardcopyworld.com/gnuboard5/bbs/profile.php?mb_id=godstale) 하드카피**](http://www.hardcopyworld.com/gnuboard5/bbs/profile.php?mb_id=godstale) 15-04-09 18:33 조회4,292회 댓글1건

* [이전글](http://www.hardcopyworld.com/gnuboard5/bbs/board.php?bo_table=lecture_rpi&wr_id=5)
* [다음글](http://www.hardcopyworld.com/gnuboard5/bbs/board.php?bo_table=lecture_rpi&wr_id=3)
* [목록](http://www.hardcopyworld.com/gnuboard5/bbs/board.php?bo_table=lecture_rpi&page=)
* [답변](http://www.hardcopyworld.com/gnuboard5/bbs/write.php?w=r&bo_table=lecture_rpi&wr_id=4)
* [글쓰기](http://www.hardcopyworld.com/gnuboard5/bbs/write.php?bo_table=lecture_rpi)

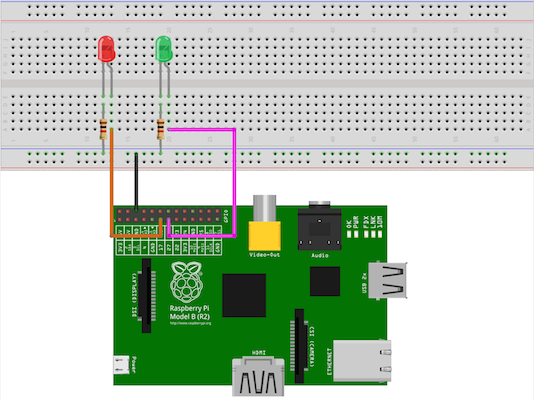
[](http://www.hardcopyworld.com/gnuboard5/bbs/view_image.php?bo_table=lecture_rpi&fn=1304495691_KEgXGRtO_LED_python.png)

라즈베리 파이의 GPIO 핀 구조는 아래와 같습니다.

[](http://www.hardcopyworld.com/ngine/aduino/wp-content/uploads/sites/3/2015/04/53bc258dc6c0425cb44870b50ab30621.jpg)

이 중 Digital 핀(GPIO xx)을 제어하는, 가장 기초에 해당하는 내용입니다.

테스트를 위해 아래와 같이 2개의 LED를 연결해 줍니다.

[](http://www.hardcopyworld.com/ngine/aduino/wp-content/uploads/sites/3/2015/04/LED_python.png)

**CONSOLE 에서 GPIO 제어**

먼저 해볼 방법은 console 에서 파일 다루듯이 제어하는 방법입니다. 왼쪽 빨간색 LED, GPIO 17번에 연결된 LED를 제어합니다.

cd 명령으로 /sys/class/gpio 폴더로 이동합니다. 그리고 폴더에 어떤 파일들이 있는지 한번 확인합니다.

* cd /sys/class/gpio
* ls -l

이 상태에서 GPIO17 핀을 초기화부터 해줍니다. 아래 명령으로... 그리고 다시 폴더에 어떤 파일들이 있는지 확인합니다.

* echo 17 > /sys/class/gpio/export
* ls -l

**gpio17** 심볼릭 링크가 생겼음을 알 수 있습니다. 이 링크를 통해 GPIO17을 제어합니다. 이제 GPIO17 을 input/output 어떤 모드로 사용할지 알려줘야 합니다.

* echo out > /sys/class/gpio/gpio17/direction

gpio17 폴더로 들어가보면 direction 파일이 생성되어 있고 거기엔 out 이라고 쓰여져 있을겁니다. GPIO17 핀이 out (3.3V 출력) 모드로 설정된 것입니다.

GPIO17 은 digital 핀이므로 여기에 0 또는 1 값을 넣어줄 수 있습니다. 예상하셨겠지만 0이 0V - LED off 상태를 의미하고, 1이 그 반대입니다. LED를 켜기 위해서는 아래와 같이 실행합니다.

* echo 1 > /sys/class/gpio/gpio17/value

다시 끄기 위해서는

* echo 0 > /sys/class/gpio/gpio17/value

테스트가 다 끝나면 이제 정리 작업을 해줍니다.

* cd /sys/class/gpio
* echo 17 > /sys/class/gpio/unexport
* ls -l

/sys/class/gpio 폴더의 내용을 확인해보면 앞서 생성된 gpio17 심볼릭 링크가 삭제되어 있음을 알 수 있습니다.

그런데 이렇게 제어하는건 좀 귀찮죠. 익숙하지도 않고... 다음은 파이썬으로 같은 작업을 해보겠습니다.

**파이썬으로 GPIO 제어 1 (RPI.GPIO)**

좀 더 우아하게 파이썬으로 GPIO 핀을 제어합니다. 먼저 RPi 라이브러리를 이용한 방법입니다. 제 경우에는 이 라이브러리가 이미 설치되어있던데;;; 다른 작업 하다가 설치한걸지도 모르겠네요. 아래와 같이 접속해보고 문제가 있는지 확인합니다.

* sudo python
* >>> import RPi.GPIO as GPIO
* >>> GPIO.setmode(GPIO>BCM)
* >>> quit()

문제가 발생한다면 라이브러리 설치합니다.

* wget 'http://downloads.sourceforge.net/project/raspberry-gpio-python/RPi.GPIO-0.5.4.tar.gz'
* tar zxvf RPi.GPIO-0.5.4.tar.gz
* cd RPi.GPIO-0.5.4/
* sudo apt-get install python-dev
* sudo python setup.py install

준비가 되었으면 테스트 파일을 만들어 봅니다. 위 회로도에 있는 2개의 LED (GPIO17, GPIO27)를 깜빡이는 예제입니다. 예제 파일을 만듭니다.

* nano rpitest.py

아래 내용을 모두 넣어줍니다.

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BCM)

GPIO.setup(17, GPIO.OUT) ## GPIO 17 ouput

GPIO.setup(27, GPIO.OUT) ## GPIO 27 output

def blink():

print "Starting blinking fever!"

iteracion = 0

while iteracion < 30: ## Blink for 30'

GPIO.output(17, True) ## On 17

GPIO.output(27, False) ## Off 27

time.sleep(1) ## Wait one second

GPIO.output(17, False) ## Off 17

GPIO.output(27, True) ## On 27

time.sleep(1) ## Wait one second

iteracion = iteracion + 2 ## Add 2 second, one for each blink

print "I'm done!"

GPIO.cleanup() ## Clean the GPIO

blink() ## Call the function

파이썬 파일을 실행시켜 봅니다. (root 권한 필요)

* sudo python rpitest.py

30초 간 깜빡이고 종료되면 성공입니다.

**파이썬으로 GPIO 제어 2 (WIRINGPI)**

필수 패키지부터 설치합니다.

* sudo apt-get install -y python-dev python-setuptools

WiringPi 다운로드, 빌드

* git clone git://git.drogon.net/wiringPi
* cd wiringPi
* sudo ./build
* cd ..

WiringPi2 다운로드, 인스톨

* git clone https://github.com/Gadgetoid/WiringPi2-Python.git
* cd WiringPi2-Python
* sudo python setup.py install

WiringPi 파이썬 라이브러리가 WiringPi2 입니다. WiringPi2-Python/examples 폴더에 가면 예제들이 있습니다. 이걸 이용해서 테스트 해봅니다.

* cd examples
* nano quick2wire-io.py

예제를 아래와 같이 수정합니다.

import wiringpi2

pin\_base = 65

i2c\_addr = 0x20

pins = [17, 27]

wiringpi2.wiringPiSetup()

#wiringpi2.mcp23017Setup(pin\_base,i2c\_addr)

for pin in pins:

wiringpi2.pinMode(pin,1)

wiringpi2.digitalWrite(pin,1)

wiringpi2.delay(100)

wiringpi2.digitalWrite(pin,0)

실행할 때는 루트 권한으로 합니다.

* sudo python quick2wire-io.py

문제없이 실행되면 끝!!

**참고자료**

* [라즈베리파이 GPIO 활용 강좌](http://www.rasplay.org/?p=4918)
* [WiringPi - GitHub](https://github.com/WiringPi/WiringPi2-Python)
* [How Raspberry Pi can ease your life](https://github.com/egabancho/cern_spring_campus/blob/master/How%20Raspberry%20Pi%20can%20ease%20your%20life.md)
* [Raspberry Pi: Control GPIO and your Pi Camera using your phone](https://rbnrpi.wordpress.com/raspberry-pi-control-gpio-and-your-pi-camera-using-your-phone/)

[**[http://www.hardcopyworld.com/gnuboard5/data/member/go/godstale.gif](http://www.hardcopyworld.com/gnuboard5/bbs/profile.php?mb_id=godstale) 하드카피**](http://www.hardcopyworld.com/gnuboard5/bbs/profile.php?mb_id=godstale)

모든 전통이 한 때는 오해를 면치 못했다. 마찬가지로 모든 아이디어는 한 때는 비웃음을 면치 못했다. -홀브룩 잭슨 ::::: 느닷없이 떠오르는 생각이 가장 귀중한 것이며, 보관해야할 가치가 있는 것이다. -베이컨 ::::: 미리 숙고하면 힘든 일도 그저 실행에 옮기는 순간 쉬워진다. - 로버트 M. 피어식

추천 **0**

* [페이스북으로 보내기](http://www.hardcopyworld.com/gnuboard5/bbs/sns_send.php?longurl=http%3A%2F%2Fwww.hardcopyworld.com%2Fgnuboard5%2Fbbs%2Fboard.php%3Fbo_table%3Dlecture_rpi%26wr_id%3D4&title=%EB%9D%BC%EC%A6%88%EB%B2%A0%EB%A6%AC+%ED%8C%8C%EC%9D%B4+GPIO+%EC%A0%9C%EC%96%B4+%28console%2C+python%29&sns=facebook)
* [트위터로 보내기](http://www.hardcopyworld.com/gnuboard5/bbs/sns_send.php?longurl=http%3A%2F%2Fwww.hardcopyworld.com%2Fgnuboard5%2Fbbs%2Fboard.php%3Fbo_table%3Dlecture_rpi%26wr_id%3D4&title=%EB%9D%BC%EC%A6%88%EB%B2%A0%EB%A6%AC+%ED%8C%8C%EC%9D%B4+GPIO+%EC%A0%9C%EC%96%B4+%28console%2C+python%29&sns=twitter)
* [구글플러스로 보내기](http://www.hardcopyworld.com/gnuboard5/bbs/sns_send.php?longurl=http%3A%2F%2Fwww.hardcopyworld.com%2Fgnuboard5%2Fbbs%2Fboard.php%3Fbo_table%3Dlecture_rpi%26wr_id%3D4&title=%EB%9D%BC%EC%A6%88%EB%B2%A0%EB%A6%AC+%ED%8C%8C%EC%9D%B4+GPIO+%EC%A0%9C%EC%96%B4+%28console%2C+python%29&sns=gplus)

**댓글목록**

[**[http://www.hardcopyworld.com/gnuboard5/data/member/ad/admin.gif](http://www.hardcopyworld.com/gnuboard5/bbs/profile.php?mb_id=admin) 최고관리자**](http://www.hardcopyworld.com/gnuboard5/bbs/profile.php?mb_id=admin) 작성일 15-10-30 14:24

RPi.GPIO 패키지 다운로드가 안되는 경우 아래 커맨드 사용해보세요.   
wget <https://pypi.python.org/packages/source/R/RPi.GPIO/RPi.GPIO-0.5.11.tar.gz>

# [라즈베리파이 GPIO 강좌 : 01. GPIO 소개 및 핀배치](http://www.rasplay.org/?p=2049)

# ****라즈베리파이 GPIO 강좌****

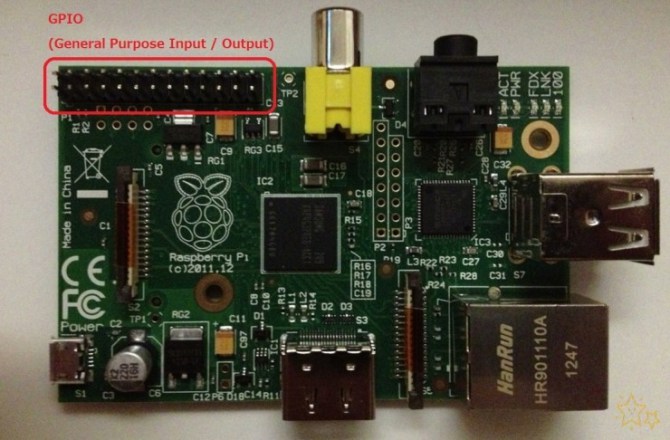
### 라즈베리파이 GPIO 강좌 : 01. GPIO 소개 및 핀배치

2013.03.01

##### **1. 라즈베리파이 GPIO 소개**

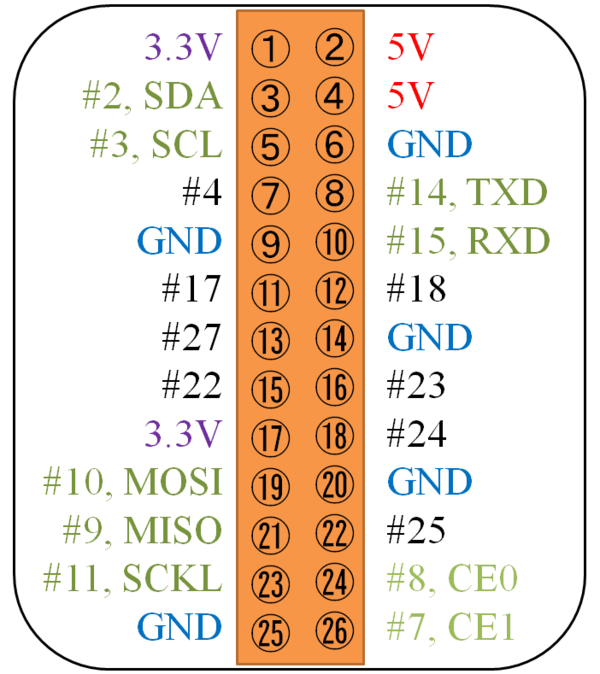
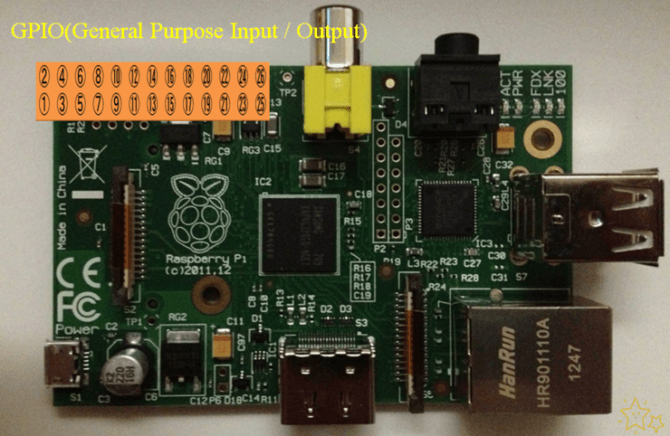
소형 PC 라즈베리파이는 USB, 이더넷, HDMI 출력, 영상출력, 사운드출력 등의 기능을 갖고 있는 하나의 작은 컴퓨터이다. 가격이 저렴하고, 작은 크기이지만 그 성능은 간단한 OS를 돌리고 동영상을 보고, 인터넷 서핑을 하는 목적에는 전혀 문제가 없는 완전한 컴퓨터인 것이다.

더욱이, 재미있는 점은 단일 마이크로세서와 같이 입출력 신호를 제어할 수 있는 포트가 있다는 것이다. 범용적인 목적으로 입/출력을 담당하는 GPIO (General Purpose Input / Output) 를 가지고 있으며, SPI 통신, I2C 통신, UART 통신 등도 갖추고 있다. 이는 마이크로프로세서를 다루는 사람들에게는 흥미로운 일이 아닐 수 없다. 왜냐하면 일반적으로 마이크로세서는 간단한 리얼타임OS(RTOS) 정도 올리던 것을 전부였는데 라즈베리파이는 데비안계열의 기능 축소판이긴 하지만 리눅스 OS를 올릴 수 있으며, 더욱이 자신이 원하는 포트 제어를 할 수 있기 때문이다. 물론, 제어전용 마이크로프로세서에 비해 리눅스를 기반으로 하고 있기에 시스템 부하에 따라 프로그램 실행 속도가 늦어지거나 실행 지점이 다르다는 등 실시간 제어가 안된다는 것이 문제 이기는 하지만 그런 제약이 없는 프로젝트라면 사용할 만 할 듯 싶다. 추가로 ADC, PWM, 인터럽트 기능이 없는 것도 조금 아쉽지만 제한된 범위에서의 제어에는 문제 없을 듯 싶다.

[](http://i1.wp.com/www.rasplay.org/wp-content/uploads/1.jpg)

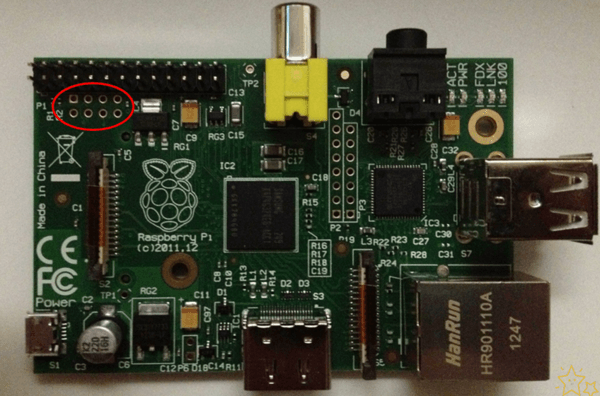
##### **2. 라즈베리파이 외부핀**

현재 라즈베리파이 모델B (버전2, 512RAM) 의 경우, 외부핀은 26핀으로 구성되어 있다. (별도로 P5 헤더 및 카메라핀 등 헤더핀이 삽입되어 있지는 않지만 공간을 할당해 놓은것은 제외한다.) 이 중, UART, SPI, I2C 관련핀들은 GPIO 목적으로도 사용되기도 하고 각 관련된 특수 기능을 수행하기도 한다. 아래는 각 핀을 배치에 따라 나열해 본 것이다.

[](http://i1.wp.com/www.rasplay.org/wp-content/uploads/2.png)[](http://i1.wp.com/www.rasplay.org/wp-content/uploads/Capture.png)

위 그림에서 알 수 있듯이 라즈베리파이의 P1 헤더핀에는 전원과 관련하여 9핀이 사용되고 있고, UART 관련 총2핀, I2C 관련 총2핀, SPI 관련 총5핀이 사용되고 있다. 또한, GPIO 전용으로 총 8핀으로 총 26개 핀으로 구성되어 있다. 물론, UART 관련 총2핀, I2C 관련 총2핀, SPI 관련 총5핀 는 각 주어진 기능을 수행하기도 하지만, GPIO로도 이용가능한 핀들이기에 GPIO는 총 17핀이다.

* 5v : 총 2핀
* 3.3v : 총 2핀
* GND : 총 5핀
* TXD, RXD : UART 관련 총2핀
* SDA, SCL : I2C 관련 총2핀
* MOSI, MISO, SCKL, CE0, CE1 : SPI 관련 총5핀
* GPIO 4, 17, 18, 22, 23, 24, 25, 27 : GPIO 전용 총 8핀

[](http://i1.wp.com/www.rasplay.org/wp-content/uploads/p5.png)

더욱이, 위의 그림의 빨간 동그라미의 부분은헤더핀이 삽입되어 있지는 않지만 헤더핀을 삽입하면 별도의 8개핀을 추가로 이용가능하다. 이는 핀번호 순서대로 5v,3.3v, GPIO28, 29, 30, 31, GND, GND 로 구성되어 있다. 본 강좌에서는 별도로 다룰 예정은 아니지만 필요한 사람은 헤더핀 납땜 후 사용하면 크기도 작도 간단히 사용하기 좋을 듯 싶다.

그 이외에도 리셋을 달 수 있는 P6 헤더, JTAG 과 관련된 P2, P3 헤더 등도 있지만 GPIO 강좌와는 관련이 없으니 설명은 생략하도록 한다.

마지막으로, 종합하면 라즈베리파이에서 사용할 수 있는 GPIO는 2, 3, 4, 7, 8, 9, 10, 11, 14, 15, 17, 18, 22, 23, 24, 25, 27, 28, 29, 30, 31 으로 총 21개의 핀이다. (라즈베리파이 모델B 버전2, 512RAM)

##### **3. 라즈베리파이 주의사항**

1) 전원구성  
: 라즈베리파이의 외부핀의 전원관련으로는 3.3v, 5v 의 두 개의 전원핀이 존재한다. 허나~ GPIO 관련하여 사용할 때는 3.3v 의 회로만을 구성해야한다. 이는 라즈베리파이의 동작 전원이 3.3v 이고 그 이상의 전원에 보호하는 기능이 없다. 즉, 5v는 단순히 microUSB 선으로 들어오는 5v 전원을 연결했을뿐 실제 라즈베리파이는 3.3v 범위안에서 작업을 해야한다. 예를들어 스위치, LED를 달더라도 3.3v 에 회로를 맞춰야 한다.

2) 전류사용  
: 3.3v 의 핀은 최대 30mA이 전류를 사용할 수 있다. 그 이상의 전류가 필요한 회로를 구성하게 되면 라즈베리파이의 메인 회로에 문제가 있을 수 있다.

3) 과전압, 과전류  
: 라즈베리파이는 각 외부핀에 대해 전원 보호 회로를 구성하고 있지 않다. 즉, GPIO 핀에 과전류, 과전압을 주게 되면 내부 라즈베리파이 회로에 영향을 주고 동작하지 않을 수도 있다.

4) 버전별 핀배열  
: 라즈베리파이 모델 B 는 출시된 시점에 따라 버전이 1, 2로 나뉜다. 이 두가지 버전은 P1 헤더핀 26개가 서로 다르게 설정되어 있어서 사용함에 있어서 주의해야 한다. 본 강좌에서 설명한 핀 배열은 라즈베리파이 모델B 버전2의 512RAM 버전이다.

[ 참고자료 ]

[1] “라즈베리파이 공식 사이트”, http://www.raspberrypi.org/  
[2] “라즈베리파이 위키미디어 설명”, http://en.wikipedia.org/wiki/Raspberry\_Pi  
[3] “Embedded Linux Wiki”, http://elinux.org/  
[4] “공식 판매점 Element14”, http://downloads.element14.com/raspberryPi1.html?COM=raspi-group  
[5] “공식 판매점 RS Componts”, http://uk.rs-online.com/web/generalDisplay.html?id=raspberrypi  
[6] “공식 판매점 Allied Electronics”, http://www.alliedelec.com/lp/120626raso/  
[7] “element14의 레즈베리파이 커뮤니티”, http://www.element14.com/community/groups/raspberry-pi  
[8] Embedded Linux 커뮤니티의 “RPi Tutorials”, http://elinux.org/RPi\_Tutorials  
[9] “RASPBERRY PI INFORMATION AND VIDEOS”, http://raspi.tv/  
[10] “Raspberry Pi Spy”, http://www.raspberrypi-spy.co.uk/  
[11] “RaspiHub”, http://www.raspihub.com/  
[12] “Adafruit Blog”, http://www.adafruit.com/blog/category/raspberry-pi/  
[13] “Raspberry Pi Pod”, http://www.recantha.co.uk/blog/  
[14] “doctormonk”, http://www.doctormonk.com/2013/02/raspberry-pi-and-breadboard-raspberry.html  
[15] “bcm2835”, http://www.open.com.au/mikem/bcm2835/

End.

written by Yoonseok Pyo  
<http://robotpilot.net/>  
<http://cafe.naver.com/openrt>

[[http://i1.wp.com/i.creativecommons.org/l/by-sa/3.0/88x31.png?w=670](http://creativecommons.org/licenses/by-sa/3.0/deed.ko)](http://creativecommons.org/licenses/by-sa/3.0/deed.ko)이 저작물은 [크리에이티브 커먼즈 저작자표시-동일조건변경허락 3.0 Unported 라이선스](http://creativecommons.org/licenses/by-sa/3.0/deed.ko) 에  
따라 이용할 수 있습니다.

### 관련

[](http://www.rasplay.org/?p=47)

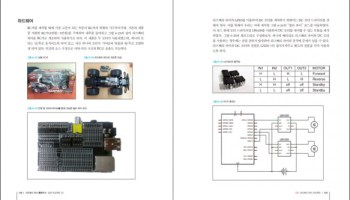
#### [산에는 가도없는 산딸기](http://www.rasplay.org/?p=47)

"기초 강좌"에서

[](http://www.rasplay.org/?p=3253)

#### [라즈베리파이 GPIO 강좌 : 05. Input 테스트 (스위치 입력, C언어)](http://www.rasplay.org/?p=3253)

"GPIO 기초강좌"에서

[](http://www.rasplay.org/?p=6524)

#### [번역X !!! 실전프로젝트 중심의 라즈베리파이 활용백서 출간 1초전 !!!!!](http://www.rasplay.org/?p=6524)

"산딸기 소식통"에서

09/04/2013 표윤석[GPIO 기초강좌](http://www.rasplay.org/?cat=291), [산딸기 GPIO](http://www.rasplay.org/?cat=256)라즈베리파이 GPIO 강좌 : 01. GPIO 소개 및 핀배치에 댓글 닫힘 [gpio](http://www.rasplay.org/?tag=gpio), [라즈베리파이](http://www.rasplay.org/?tag=%eb%9d%bc%ec%a6%88%eb%b2%a0%eb%a6%ac%ed%8c%8c%ec%9d%b4)

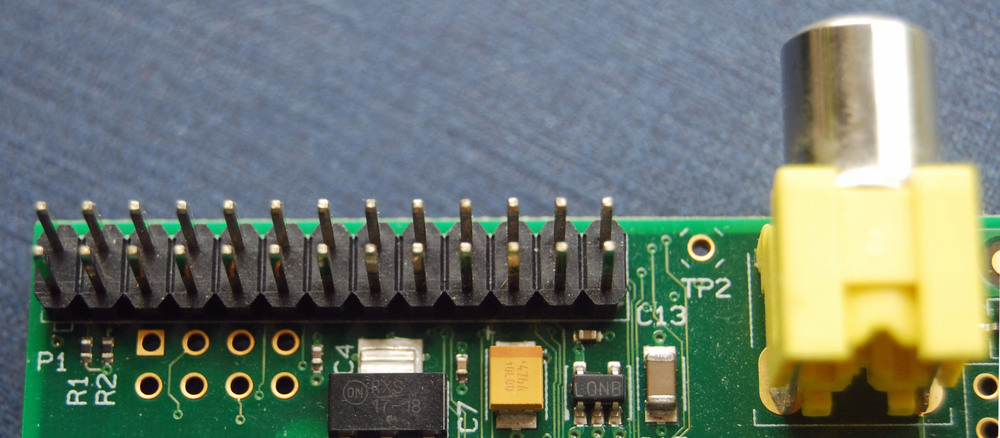
**«**[**Raspberry Pi Magic Mirror(OpenCV base facial recognition)**](http://www.rasplay.org/?p=2043)

[**라즈베리파이 활용 강좌 : 라즈베리파이 아이폰 테더링 무선 접속하기**](http://www.rasplay.org/?p=2045)**»**

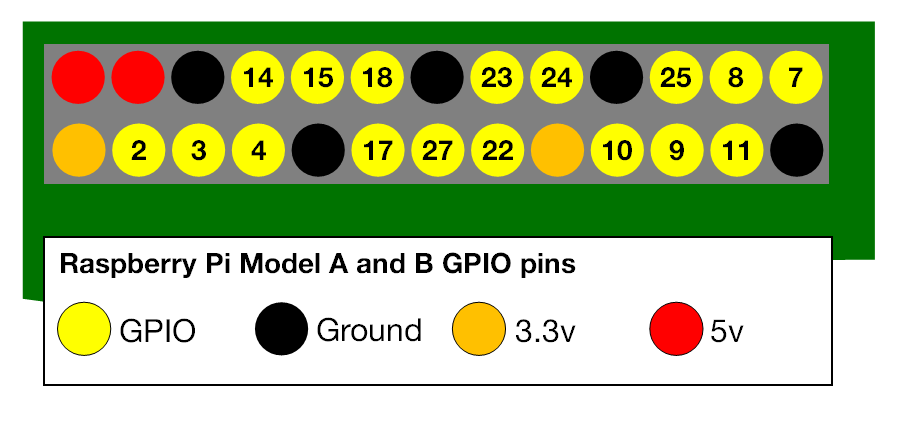
GPIO: RASPBERRY PI MODELS A AND B

AN INTRODUCTION TO GPIO AND PHYSICAL COMPUTING ON THE RASPBERRY PI

One powerful feature of the Raspberry Pi is the row of GPIO (general purpose input/output) pins along the edge of the board, next to the yellow video out socket.



These pins are a physical interface between the Pi and the outside world. At the simplest level, you can think of them as switches that you can turn on or off (input) or that the Pi can turn on or off (output). Seventeen of the 26 pins are GPIO pins; the others are power or ground pins.



WHAT ARE THEY FOR? WHAT CAN I DO WITH THEM?

You can program the pins to interact in amazing ways with the real world. Inputs don't have to come from a physical switch; it could be input from a sensor or a signal from another computer or device, for example. The output can also do anything, from turning on an LED to sending a signal or data to another device. If the Raspberry Pi is on a network, you can control devices that are attached to it from anywhere\*\* and those devices can send data back. Connectivity and control of physical devices over the internet is a powerful and exciting thing, and the Raspberry Pi is ideal for this. There are lots of brilliant examples of physical computing on [our blog](http://www.raspberrypi.org/blog/).

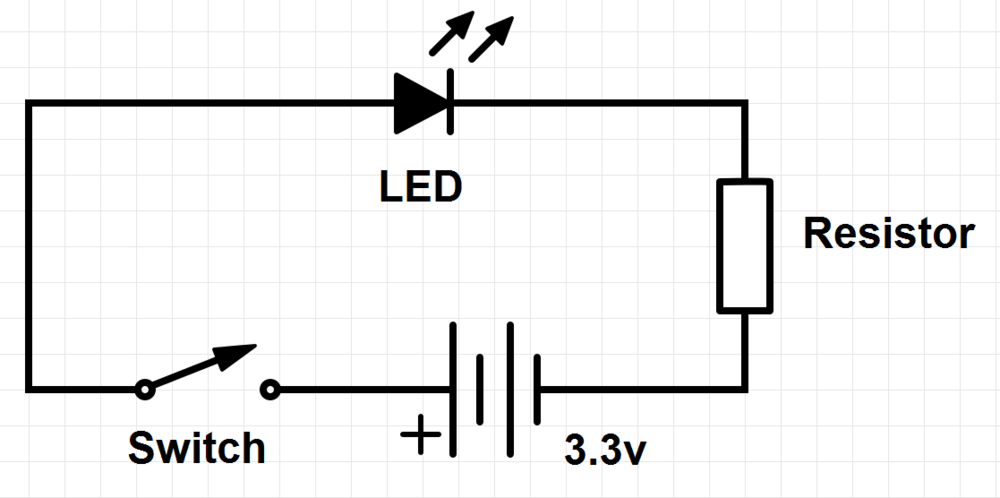
**Note**: Not **literally** anywhere, of course. You need things like access to the network, a network capable computing device, and electricity. Please do not write to us to point this out. :)

HOW THE GPIO PINS WORK

OUTPUT

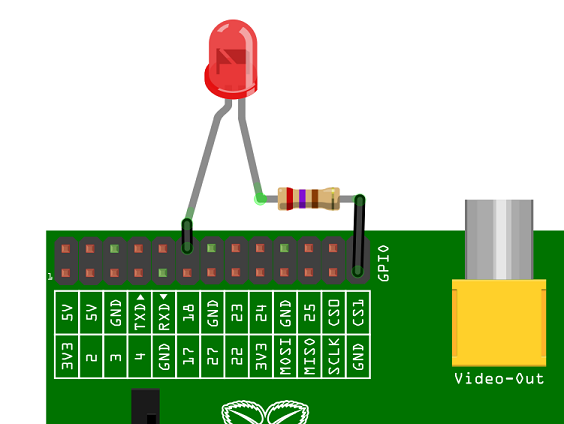
**WARNING**: If you follow the instructions, then messing about with the GPIO is safe and fun. Randomly plugging wires and power sources into your Pi, however, may kill it. Bad things can also happen if you try to connect things to your Pi that use a lot of power; LEDs are fine, motors are not. If you are worried about this, then you might want to consider using a breakout board such as the [Pibrella](http://pibrella.com/) until you are confident enough to use the GPIO directly.

Ignoring the Pi for a moment, one of the simplest electrical circuits that you can build is a battery connected to a light source and a switch (the resistor is there to protect the LED):



When we use a GPIO pin as an output, the Raspberry Pi replaces **both the switch and the battery** in the above diagram. Each pin can turn on or off,or go HIGH or LOW in computing terms. When the pin is HIGH it outputs 3.3 volts (3v3); when the pin is LOW it is off.

Here's the same circuit using the Raspberry Pi. The LED is connected to a GPIO pin (which can output +3v3) and a ground pin (which is 0v and acts like the negative terminal of the battery):



The next step is to write a program to tell the pin to go HIGH or LOW. Here's an example using [Python](http://www.raspberrypi.org/learning/quick-reaction-game/) (see Step 2), and here's how to do it in [Scratch](http://www.raspberrypi.org/learning/robot-antenna/).

INPUT

GPIO **outputs** are easy; they are on or off, HIGH or LOW, 3v3 or 0v. **Inputs** are a bit trickier because of the way that digital devices work. Although it might seem reasonable just to connect a button across an input pin and a ground pin, the Pi can get confused as to whether the button is on or off. It might work properly, it might not. It's a bit like floating about in deep space; without a reference it would be hard to tell if you were going up or down, or even what up or down meant!

This is why you will see phrases like "pull up" and "pull down" in Raspberry Pi GPIO tutorials. It's a way of giving the input pin a reference so it knows for certain when an input is received.

If you'd like to have a go at using the GPIO as an input then have a look at our[screaming jelly baby](http://www.raspberrypi.org/learning/screaming-jellybaby/) and [quick reaction game](http://www.raspberrypi.org/learning/quick-reaction-game/) tutorials for Python, or a [reaction game](http://www.raspberrypi.org/learning/reaction-game/) for Scratch.

THE END OF THE GUIDE. THE START OF SOMETHING AMAZING

We hope that this has encouraged you to have a go at physical computing using the Pi's GPIO; it's really not as daunting as it looks. It all starts with a simple LED, but it can take you to incredible places. Do not underestimate the fun, creativity and sense of achievement you can get from a little computer and a bunch of pins. Have fun! And if you do make something cool please let us know. :)

GLOSSARY

GPIO

General purpose input/output; in this specific case the pins on the Raspberry Pi and what you can do with them. So called because you can use them for all sorts of purposes; most can be used as either inputs or outputs, depending on your program.

LED

Light-emitting diode- a small, low-power light source used widely in electronics. Ideal as an introduction to physical computing on the Pi.

PHYSICAL COMPUTING

Computing that involves tangible things connected to a computer, beyond standard input and output devices like keyboards and monitors. Think buttons, lights, robots, alarms, sensors, home automation, teddy bears called Babbage in near space and so on. We love physical computing because as well as being lots of fun, it's such a powerful teaching and learning tool and encourages creativity, problem solving, and collaboration. Computing **beyond the screen** engages children of all ages, and you can make very cool stuff!

APPENDIX 1. A NOTE ON PIN NUMBERING

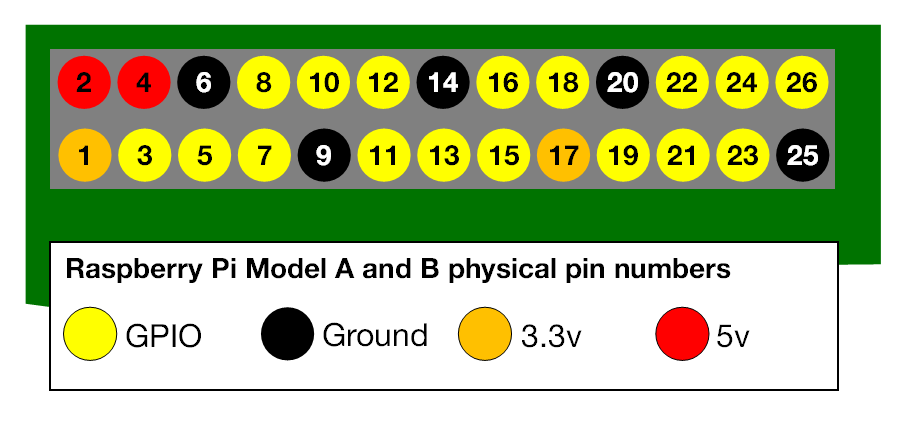
When programming the GPIO pins there are two different ways to refer to them: GPIO numbering and physical numbering.

GPIO NUMBERING

These are the GPIO pins as the computer sees them. The numbers don't make any sense to humans, they jump about all over the place, so there is no easy way to remember them. You will need a printed reference or a reference board that fits over the pins.

PHYSICAL NUMBERING

The other way to refer to the pins is by simply counting across and down from pin 1 at the top left (nearest to the SD card). This is 'physical numbering' and it looks like this:



WHICH SYSTEM SHOULD I USE?

Beginners and young children may find the physical numbering system simpler -- you simply count the pins. You'll still need a diagram like the one above to know which are GPIO pins, which are ground and which are power though.

Generally we recommend using the GPIO numbering. It's good practice and most resources use this system. Take your pick though -- as long as you use the same system within a program then all will be well. Note that pin numbering can also depend on what programming language you are using: Scratch GPIO, for example, uses physical pin numbers whereas in Python you can choose which to use.

For more details on the advanced capabilities of the GPIO pins see gadgetoid's[interactive pinout diagram](http://pi.gadgetoid.com/pinout).