Raspberry Pi GPIO
with Python

Rev. R610

이세우 (dltpdn@gmail.com)

세부목차

1. **Introduction**
2. OS(Rasbian) Installation
3. Development Environment
4. GPIO
5. Digital Output
6. Digital Input
7. Analog Output - PWM
8. Analog Input - RC Circuit
9. 고수준 Sensor Modules

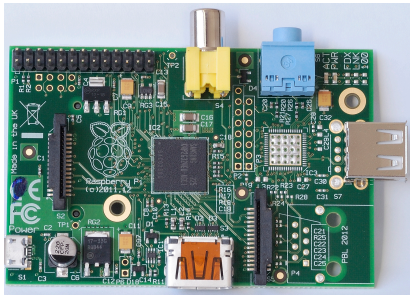
Introduction

- **Raspberry-Pi**
  - https://en.wikipedia.org/wiki/Raspberry_Pi
  - 영국 라즈베리파이 재단, Eben Upton
  - 학교와 개발도상국 컴퓨터 과학 교육 증진 목표
  - 2011 알파 보드
  - 2012 첫 판매
  - 브로드컴 BCM2835 SoC
  - PC와 동일하게 사용

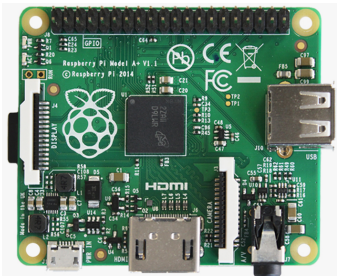
Introduction

❖ Raspberry Pi Boards

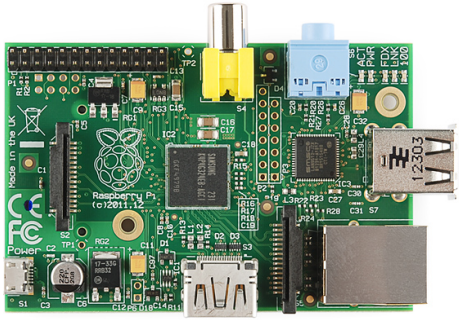
- https://www.raspberrypi.org/products/



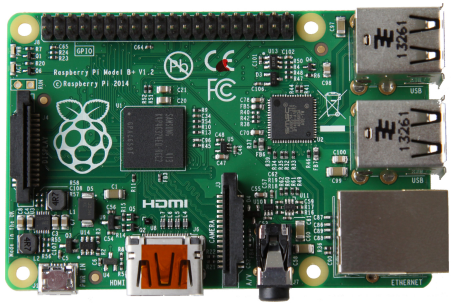
Model A



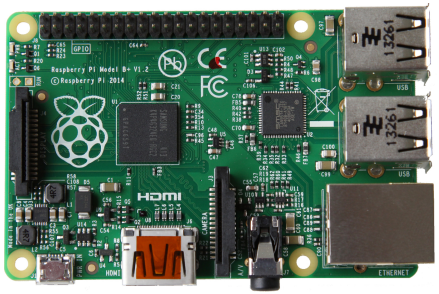
Model A+



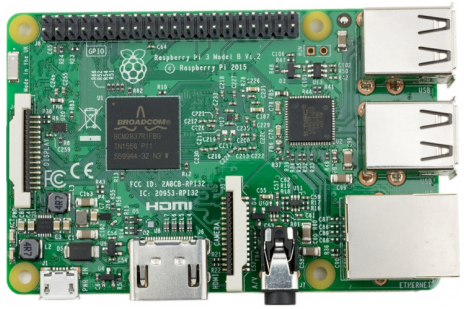
Model B



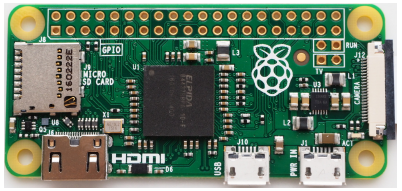
Model B+



Raspberry Pi 2



Raspberry Pi 3

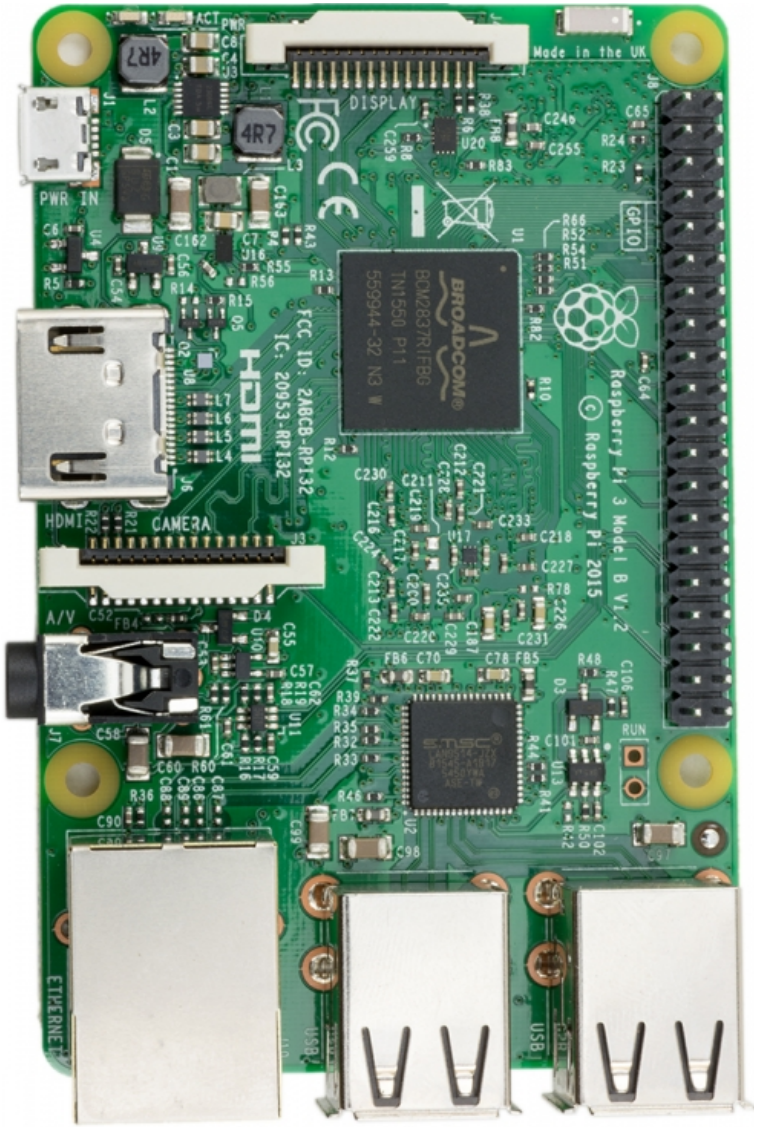


Raspberry Pi Zero

Introduction

❖ Raspberry Pi 3

- ARMv8 CPU 1.2GHz 64bit quad-core
- 802.11n Wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy(BLE)
- 1GB RAM
- 4 USB ports
- 40 GPIO Pins
- Full HDMI port
- Ethernet port
- 3.5mm audio
- Camera interface(CSI)
- Display interface(DSI)
- Micro SD card slot
- VideoCore IV 3D graphics core



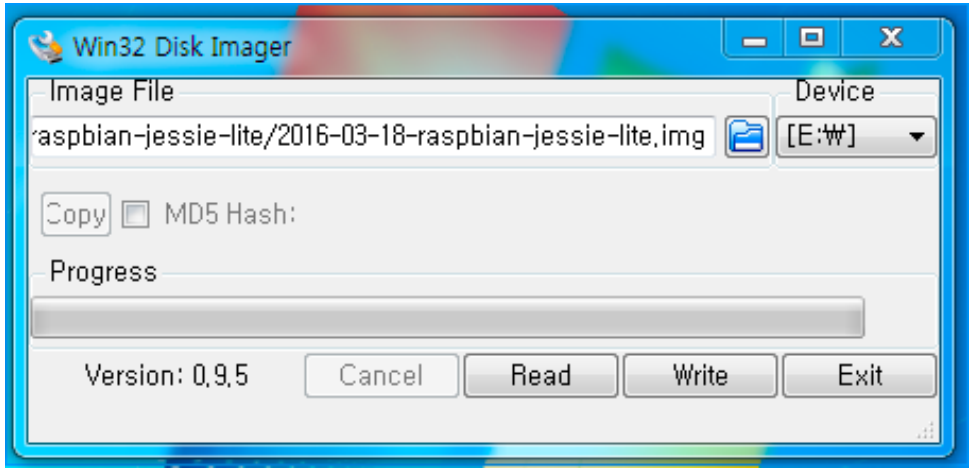
세부목차

1. Introduction
2. **OS(Rasbian) Installation**
3. Development Environment
4. GPIO
5. Digital Output
6. Digital Input
7. Analog Output - PWM
8. Analog Input - RC Circuit
9. 고수준 Sensor Modules

Rasbian Installation

❖ Rasbian Installation

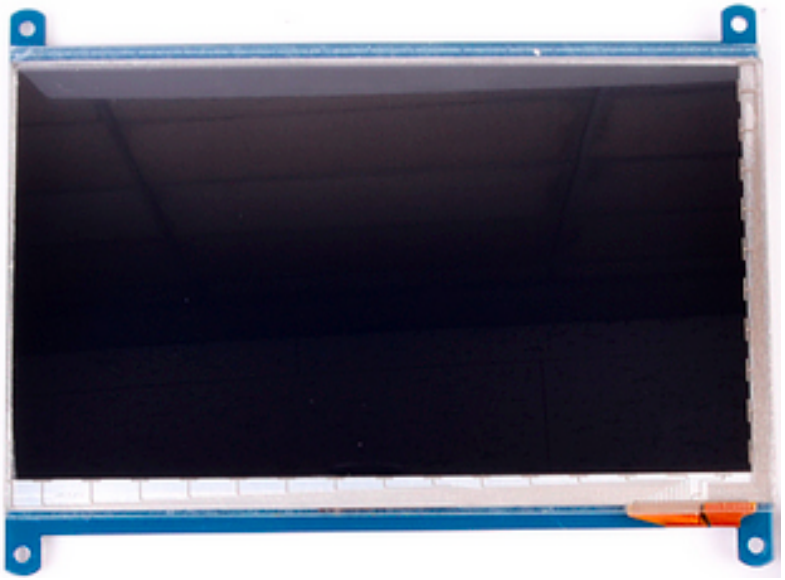
- OS image file download
  - https://www.raspberrypi.org/downloads/raspbian/
  - Raspbian Jessie, Jessi Lite
- Image file을 SD카드에 복사
  - http://sourceforge.net/projects/win32diskimager/
  - Win32 Disk Imager
- USB-Serial console 연결 할 경우
  - Serial console tty 활성화
  - Config.txt 파일에 추가
    - enable_uart=1
    - Core_req=250
- Direct Lan Cable 연결 할 경우
  - Cmdline.txt 파일 rootwait 뒤에 추가
    - Ip=192.168.0.2
- SD카드를 보드에 꼽고 전원 인가
- Putty로 Serial 또는 IP로 로그인
  - ID : pi
  - PWD : raspberry



Rasbian Installation

❖ LCD Monitor setup

- http://elinux.org/index.php?title=RPiconfig
- /boot/config.txt
  - PC의 바이오스 역할
- Max_usb_current
  - Usb 출력 제한 변경
  - 600mA →1200mA
- Hdmi_group
  - 1 : CEA, 2: DMT
- Hdmi_mode=87
  - Custom mode
- Hdmi_cvt
  - width, height, framerate, aspect, margins, interlace, rb

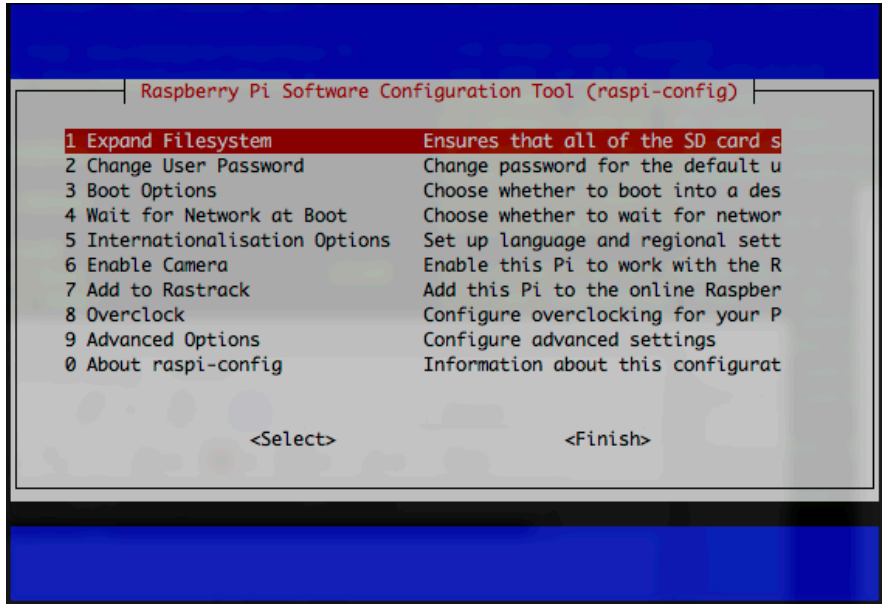


```
max_usb_current=1
hdmi_group=2
hdmi_mode=87
hdmi_cvt=1024  600 60 6 0 0 0
```

Rasbian Installation

❖ 기본설정

- Sudo raspi-config
  - 1 Expand Filesystem
  - 5 Internationalisation options
    - Time Zone > Asia > Seoul
    - Change Locale
      - en_US.UTF-8
      - ko_KR.EUC-KR
      - ko_KR.UTF-8
      - Default : en_US.UTF-8
- sudo apt-get update
- sudo apt-get upgrade



Rasbian Installation

❖ Static IP setup

- sudo vi /etc/dhcpcd.conf (백업 필수)

```
interface eth0
 static ip_address=192.168.0.xx
 static routers=192.168.0.254
 static domain_name_servers=8.8.8.8
```

- sudo reboot
- Ifconfig eth0

❖ Wifi setup

- sudo iwlist wlan0 scan
- sudo vi /etc/wpa_supplicatn/wpa_supplicant.conf

```
Network={
      ssid="my_ap"
      psk="my_password"
}
```

- Sudo ifdown wlan0
- Sudo ifup wlan0
- Ifconfig wlan0

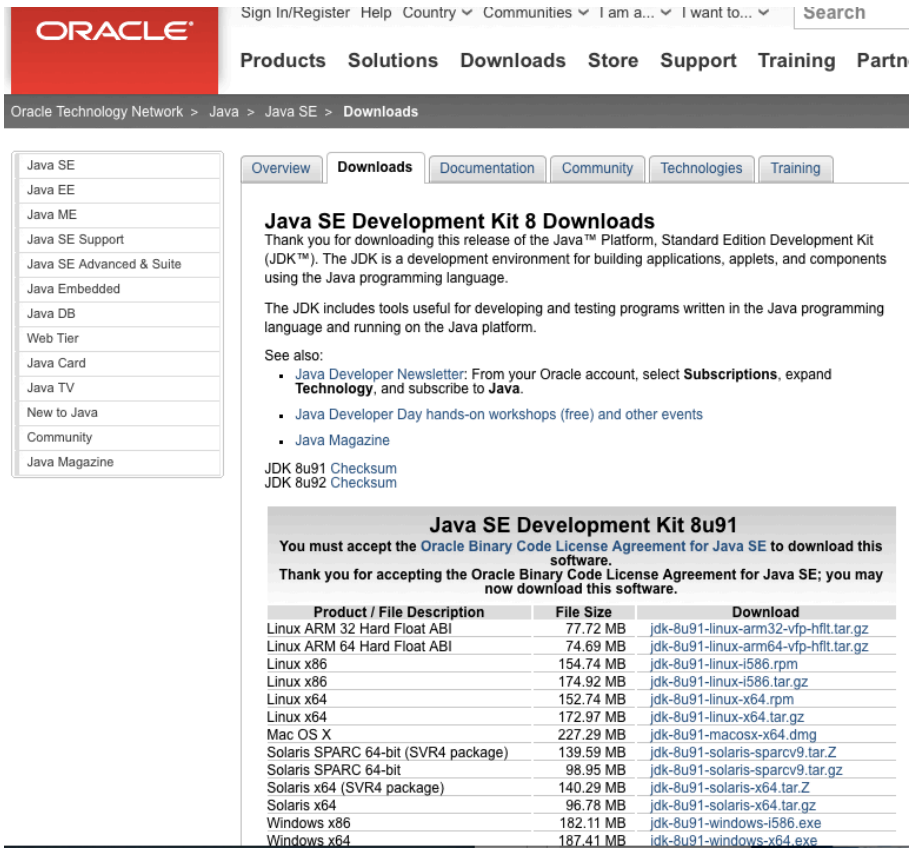
세부목차

1. Introduction
2. OS(Rasbian) Installation
3. **Development Environment**
4. GPIO
5. Digital Output
6. Digital Input
7. Analog Output - PWM
8. Analog Input - RC Circuit
9. 고수준 Sensor Modules

개발 환경

❖ JDK 설치

- Eclipse 동작을 위해서 필요
  - http://java.sun.com
  - http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html



개발 환경

❖ Eclipse 설치

- http://www.eclipse.org/downloads/
- Eclipse IDE for JavaEE Developer

개발 환경

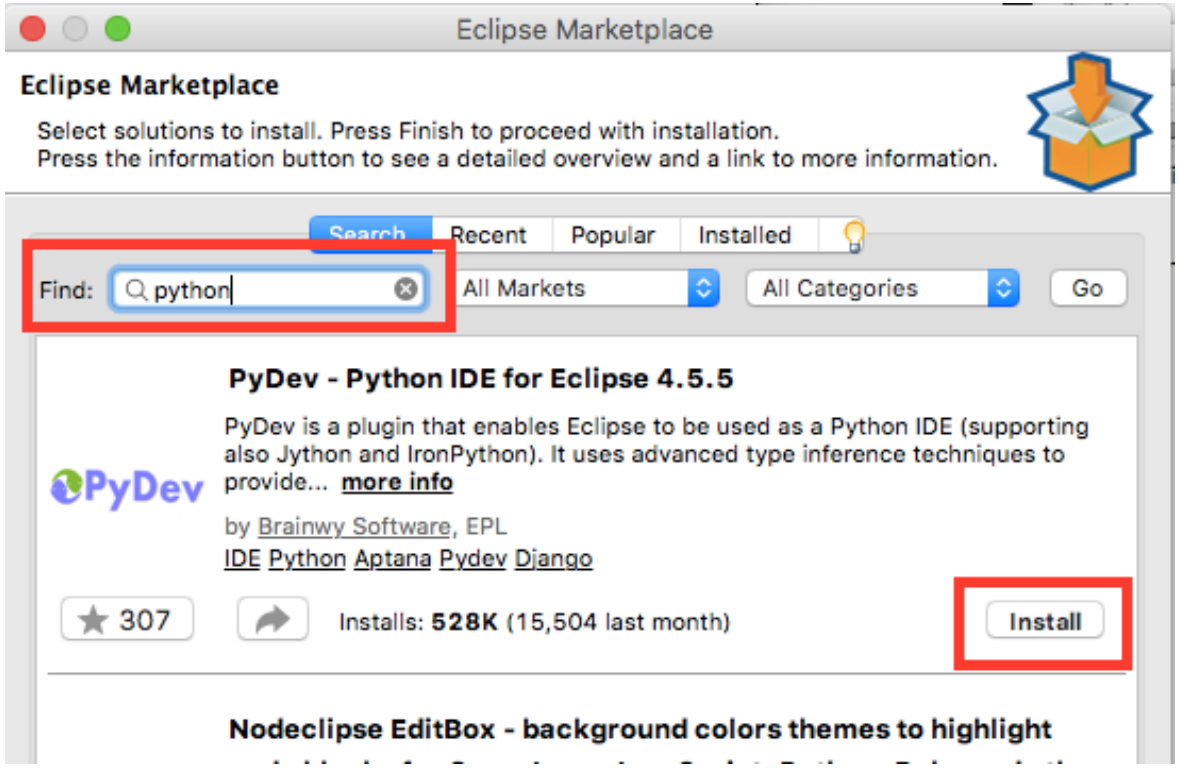
❖ Python 설치

- https://www.python.org/downloads/release/python-2711/
- 2.7 버전 설치

개발 환경

❖ PyDev 설치

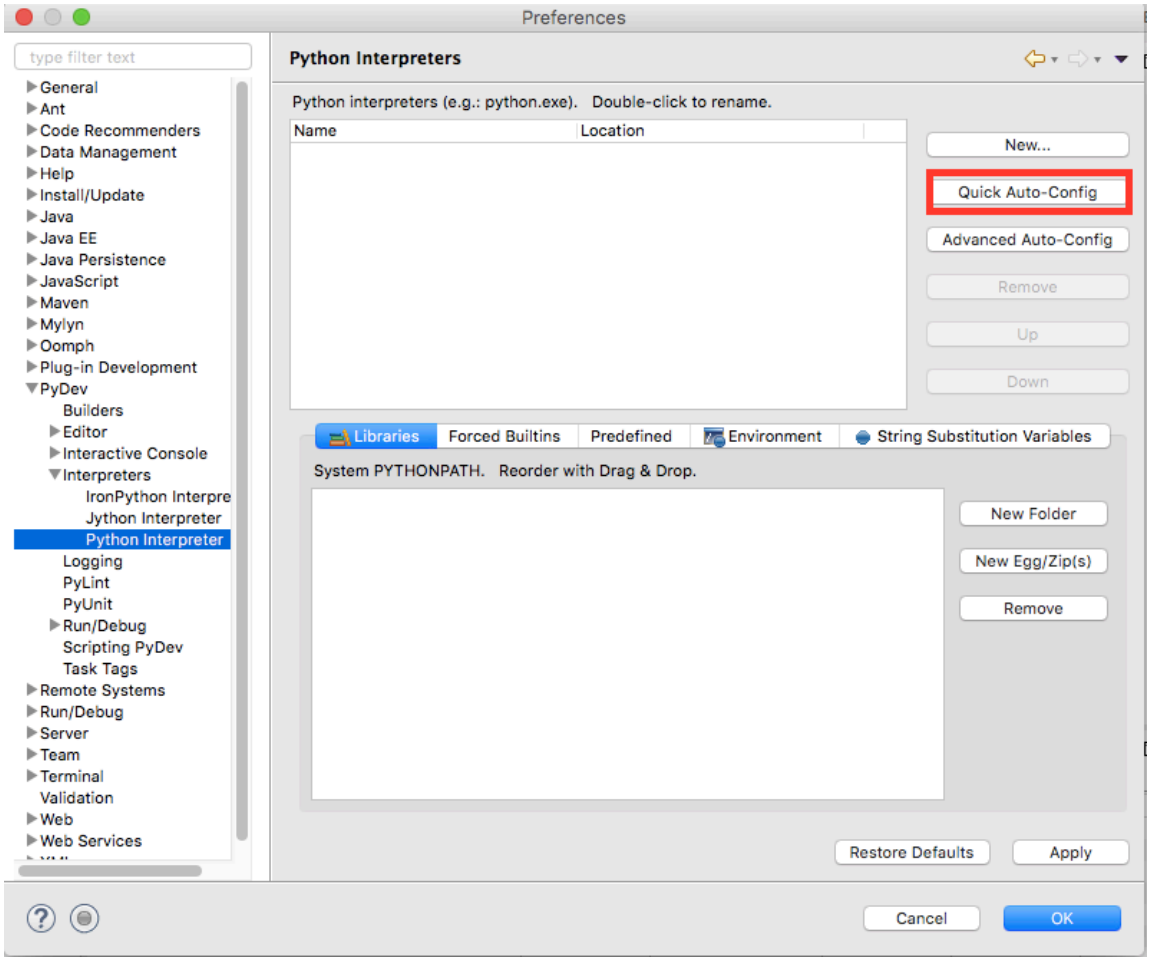
- Eclipse > Help > Eclipse Marketplace
- Python 검색



개발 환경

❖ PyDev setup

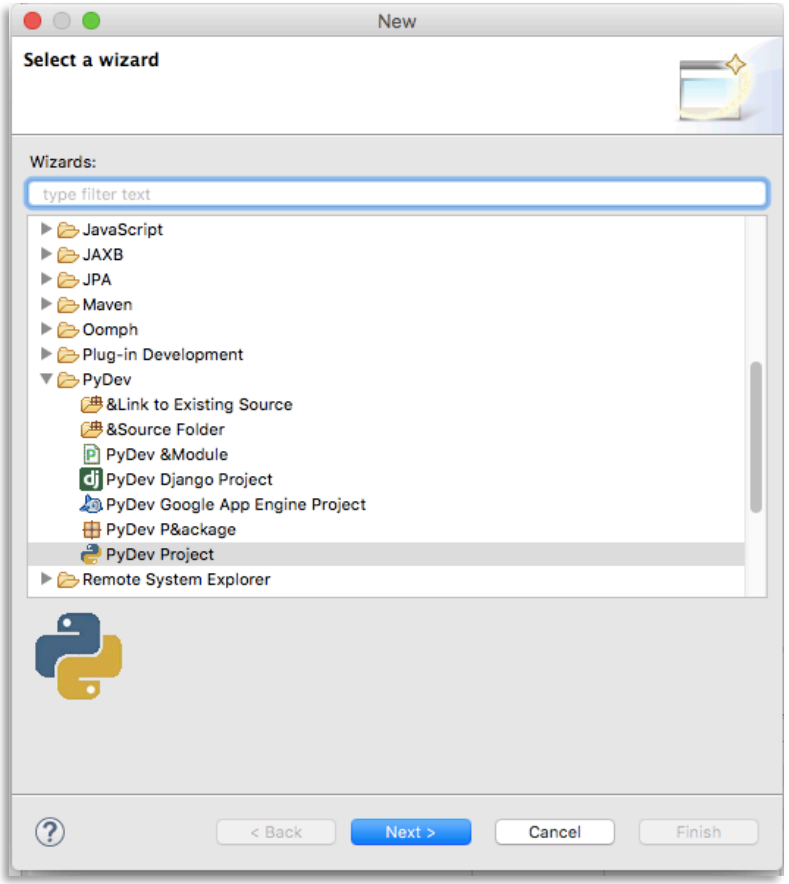
- Eclipse Preferences
- Pydev > Interpreters > Python Interpreter
- Quick Auto-Config

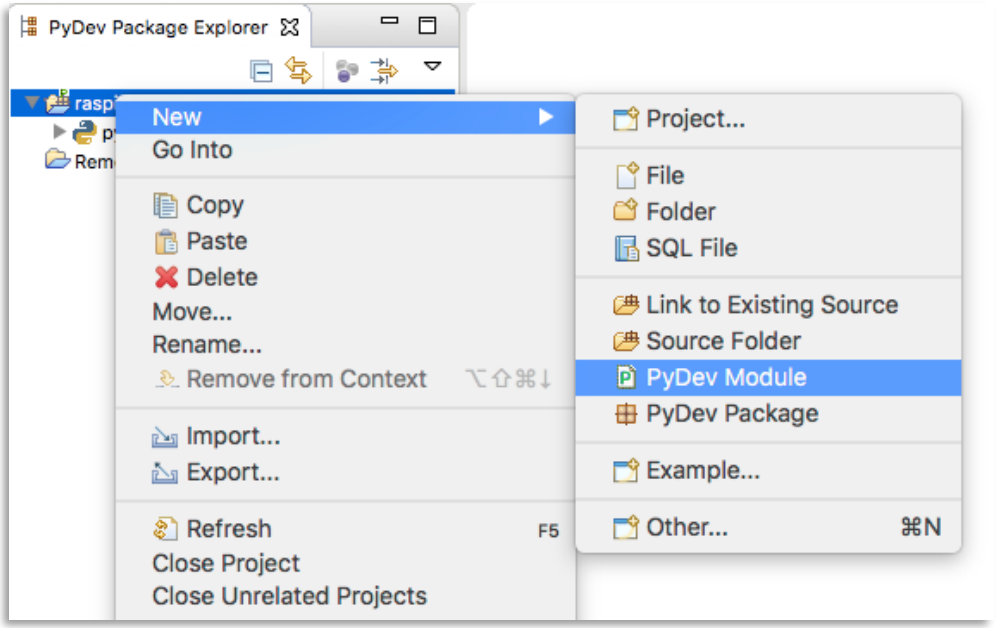


개발 환경

❖ 프로젝트 생성

- New > Project
- New > PyDev Module



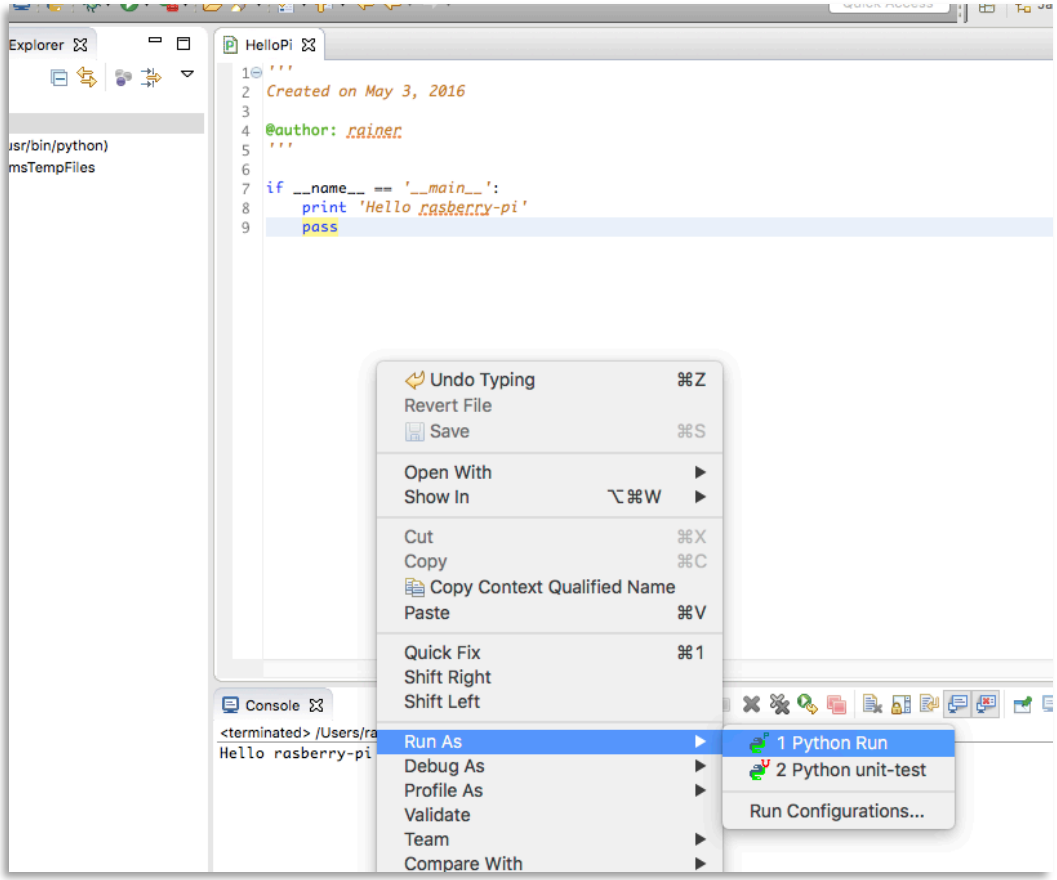


개발 환경

❖ 코드 작성 및 실행

- 우클릭 > Run As > Python Run

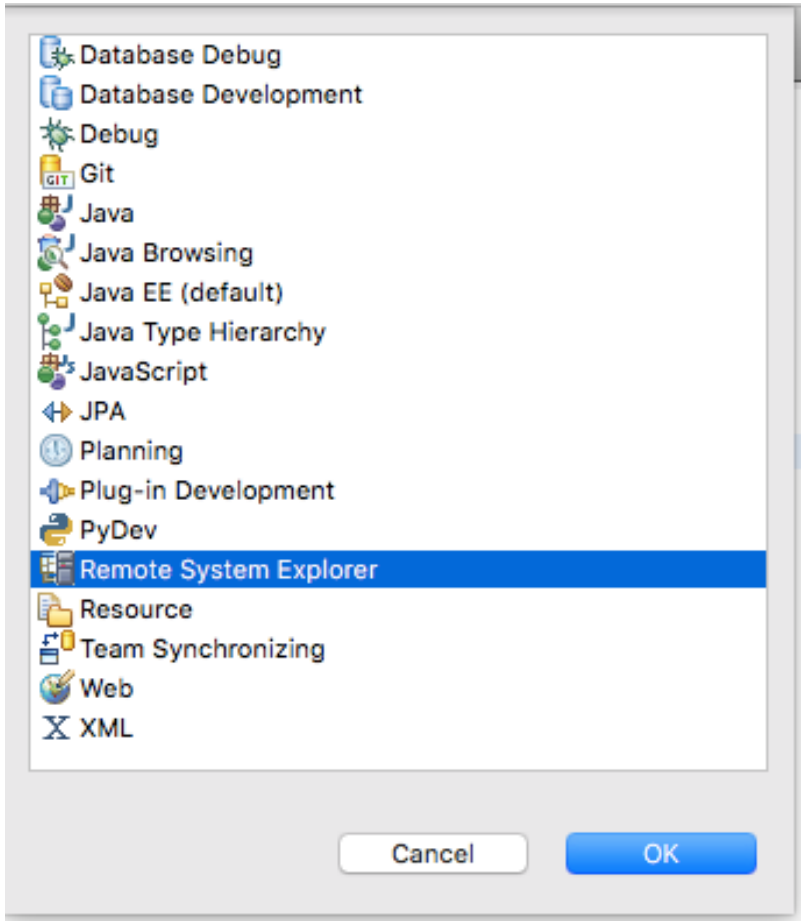
```
print “Hello raspberry Pi”
```



개발 환경

❖ RemoteSystem 설정

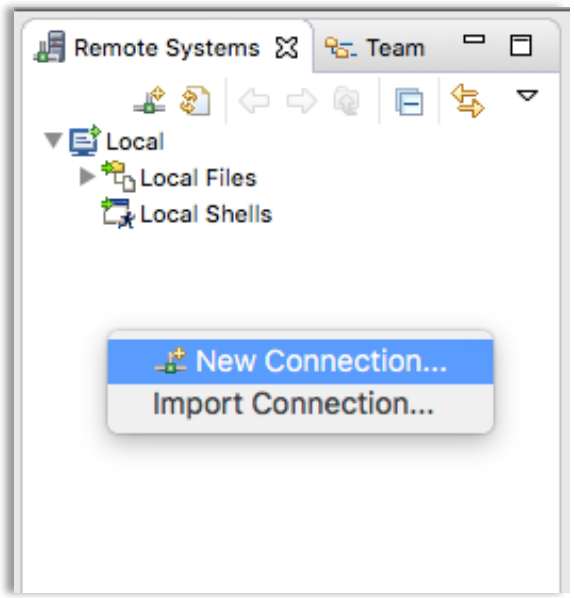
- Eclipse > Window > Perspective > Open Perspective > Other

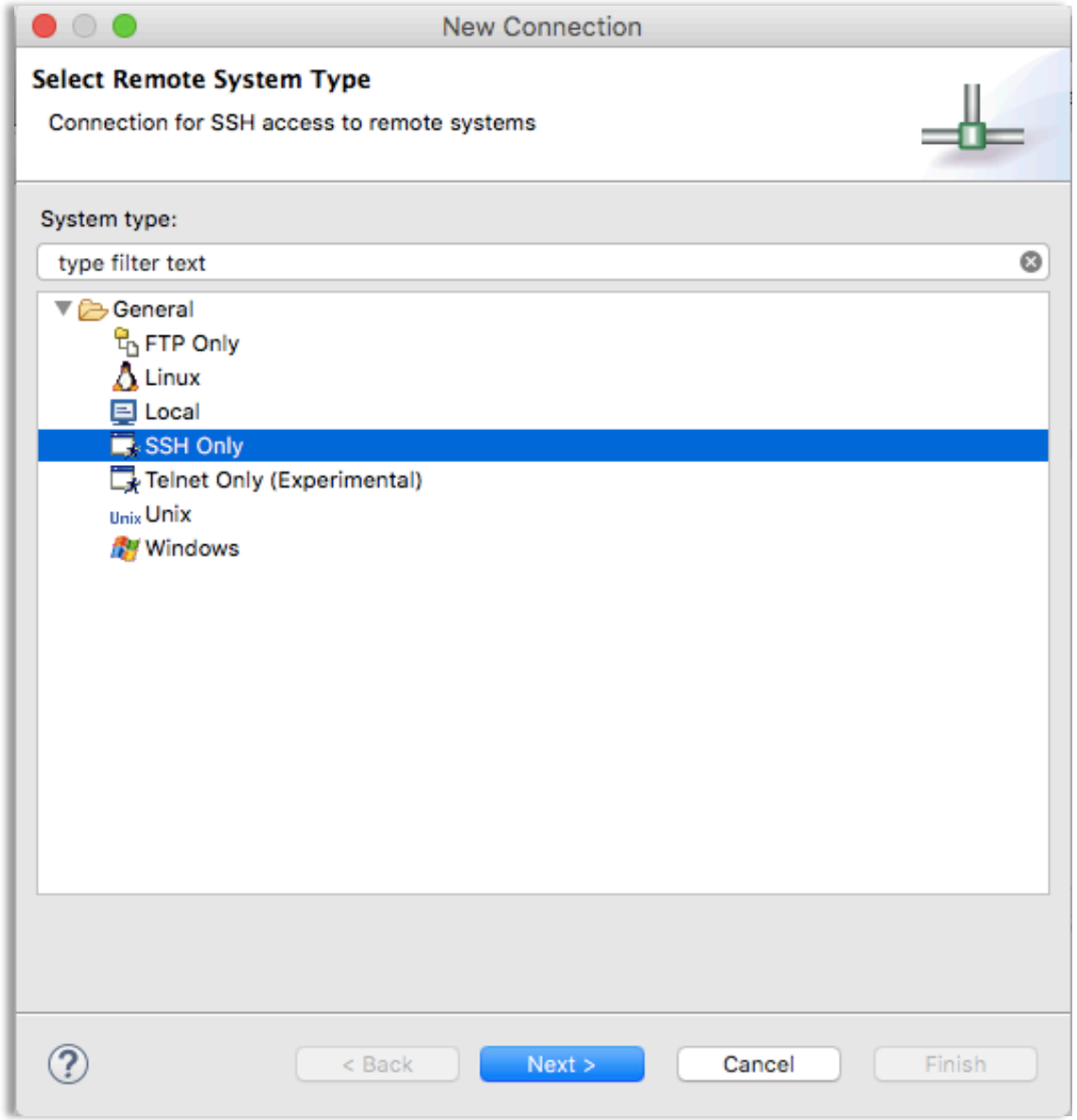


개발 환경

❖ RemoteSystem 설정

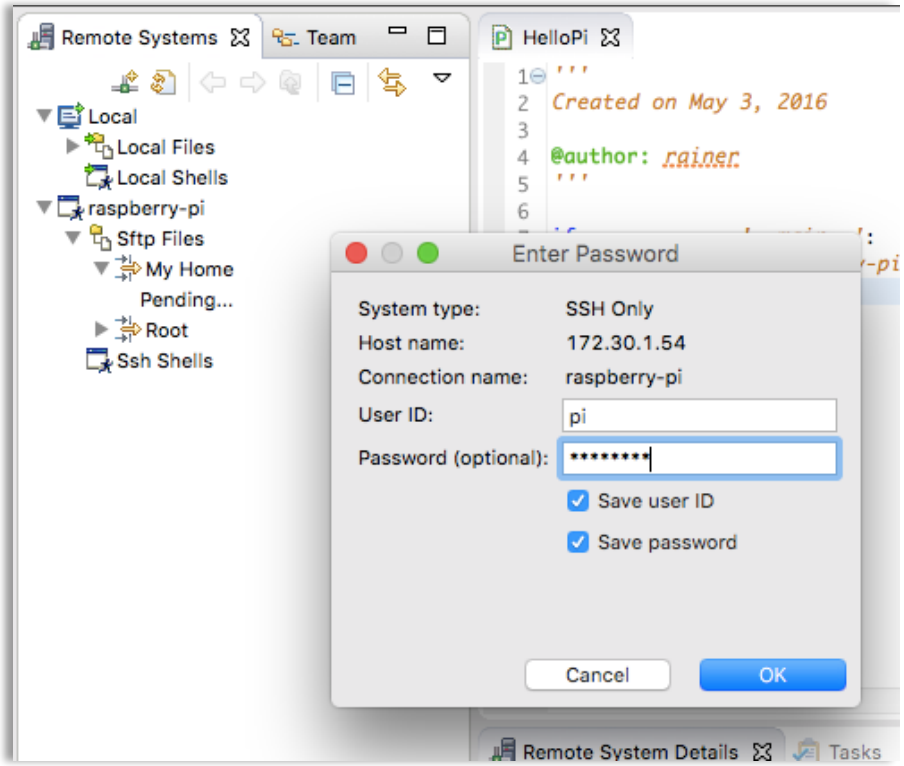
- Remote System > 우 클릭

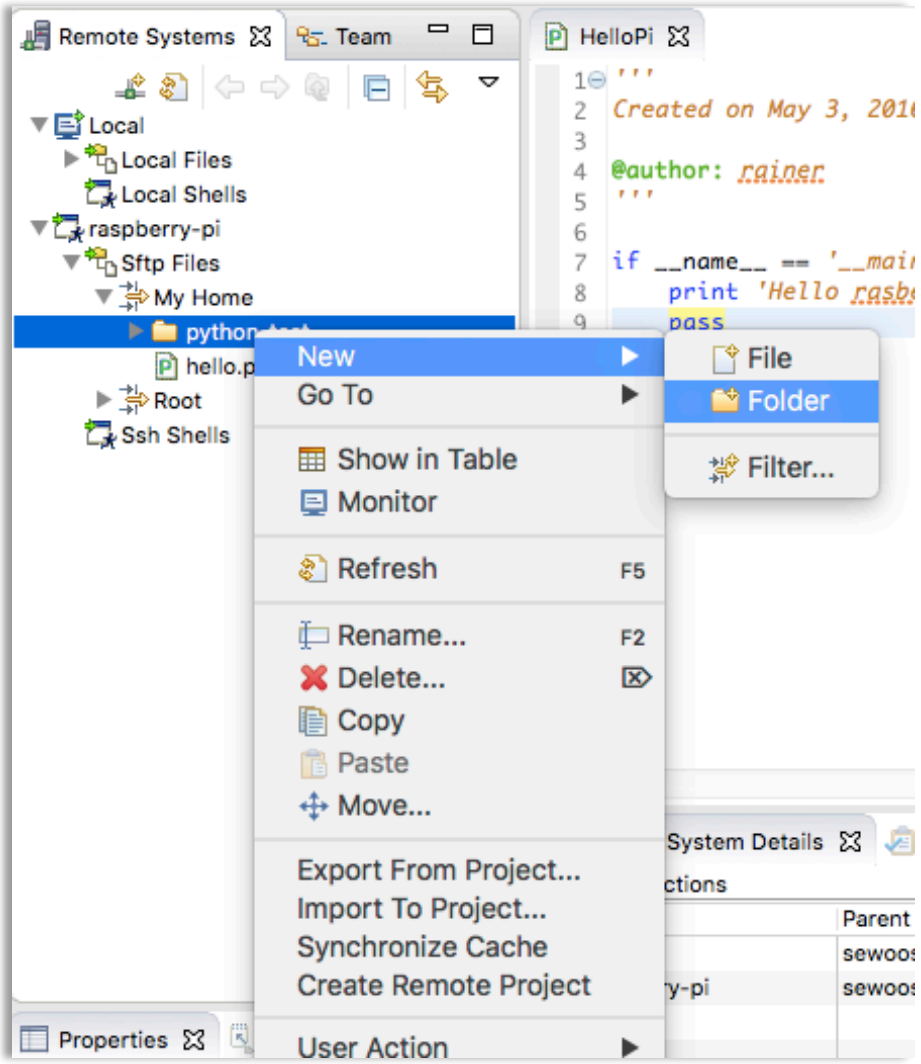




개발 환경

❖ 로그인

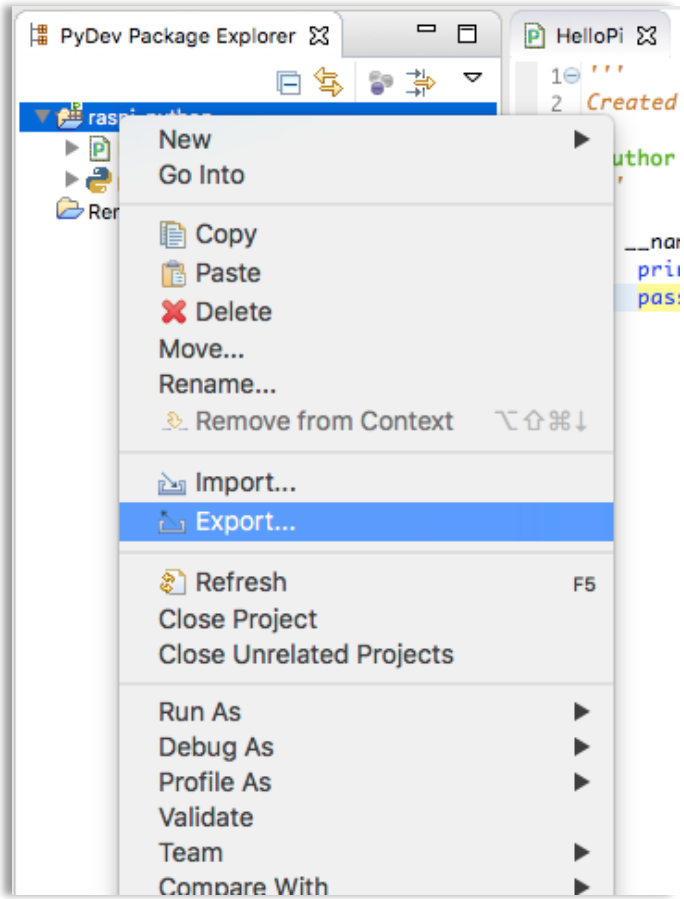

Remote Systems
Team
HelloPi
Local
Local Files
Local Shells
raspberry-pi
Sftp Files
My Home
Pending...
Root
Ssh Shells
Enter Password
System type: SSH Only
Host name: 172.30.1.54
Connection name: raspberry-pi
User ID: pi
Password (optional): ********
Save user ID
Save password
Cancel
OK
Remote System Details
Tasks

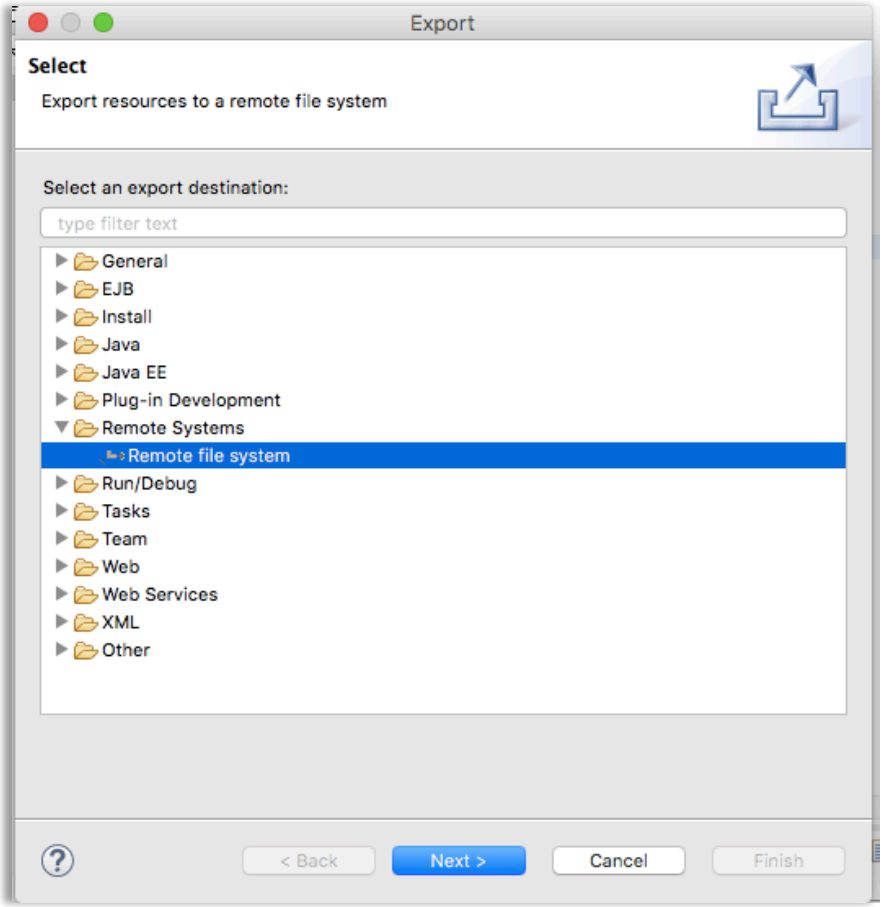

Remote Systems
Team
HelloPi
Local
Local Files
Local Shells
raspberry-pi
Sftp Files
My Home
Root
Ssh Shells
New
Go To
Show in Table
Monitor
Refresh F5
Rename... F2
Delete...
Copy
Paste
Move...
Export From Project...
Import To Project...
Synchronize Cache
Create Remote Project
User Action
File
Folder
Filter...
Properties
System Details
Parent

개발 환경

❖ 원격 적용

- PyDev Perspective에서 생성한 프로젝트 Export
- Remote Systems > Remote File System

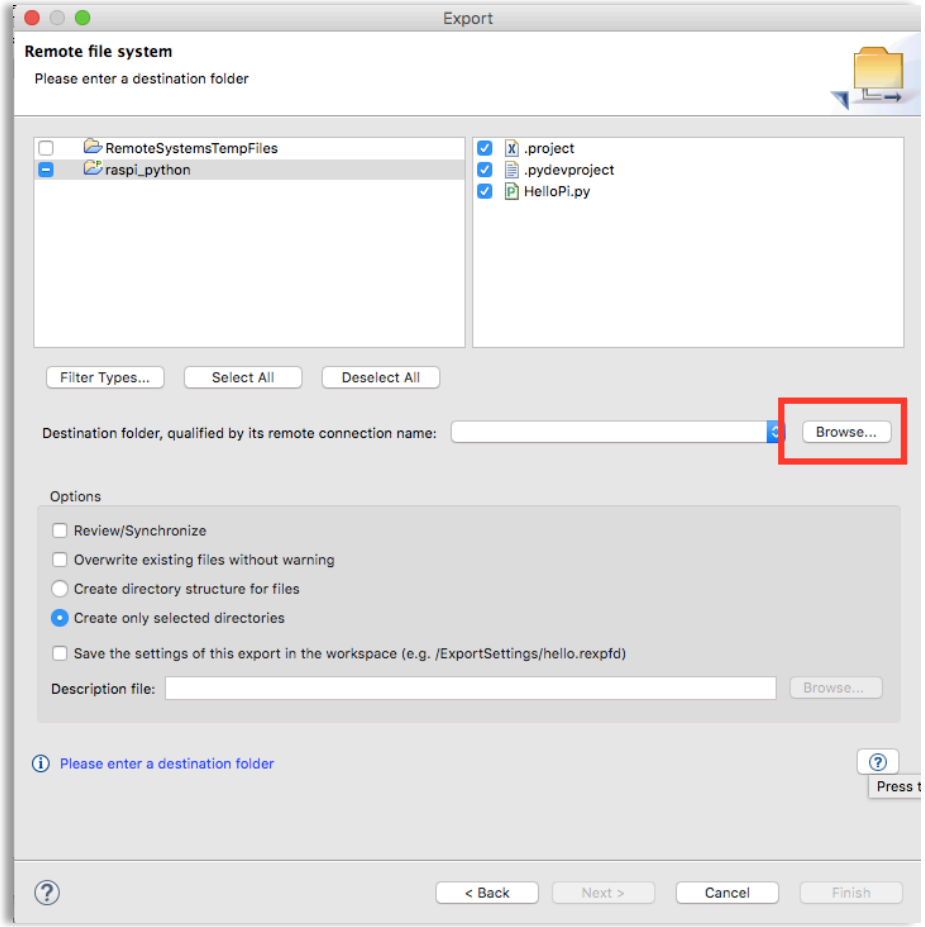


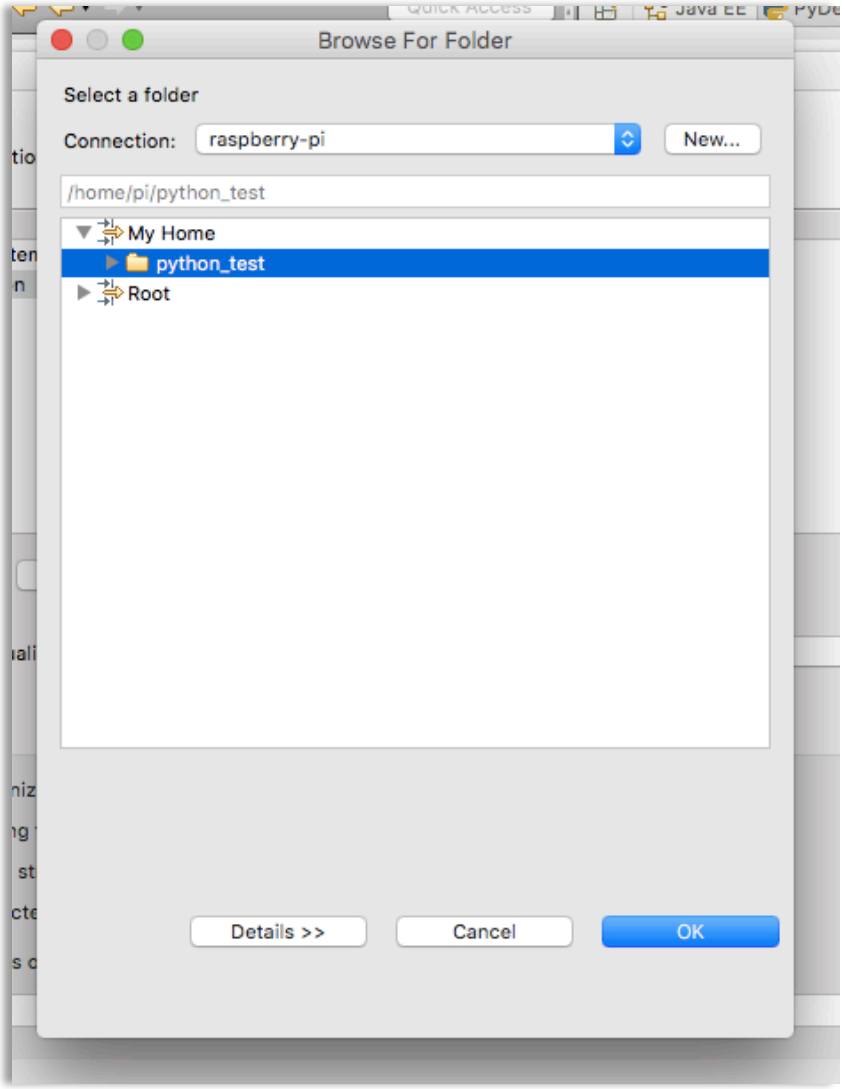


개발 환경

❖ 원격 적용

- Raspberry-pi 디렉토리 선택

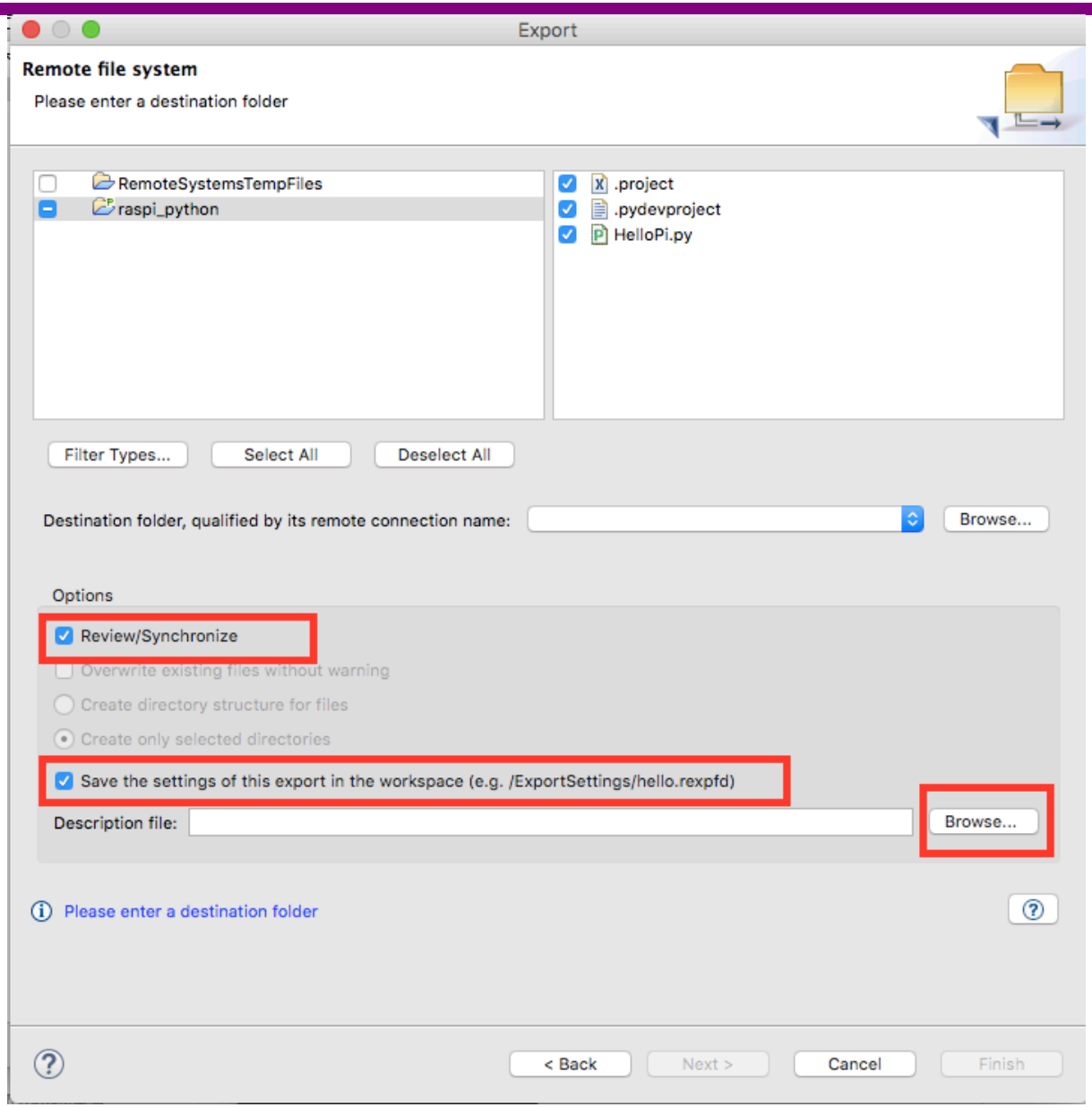




개발 환경

❖ 원격 적용

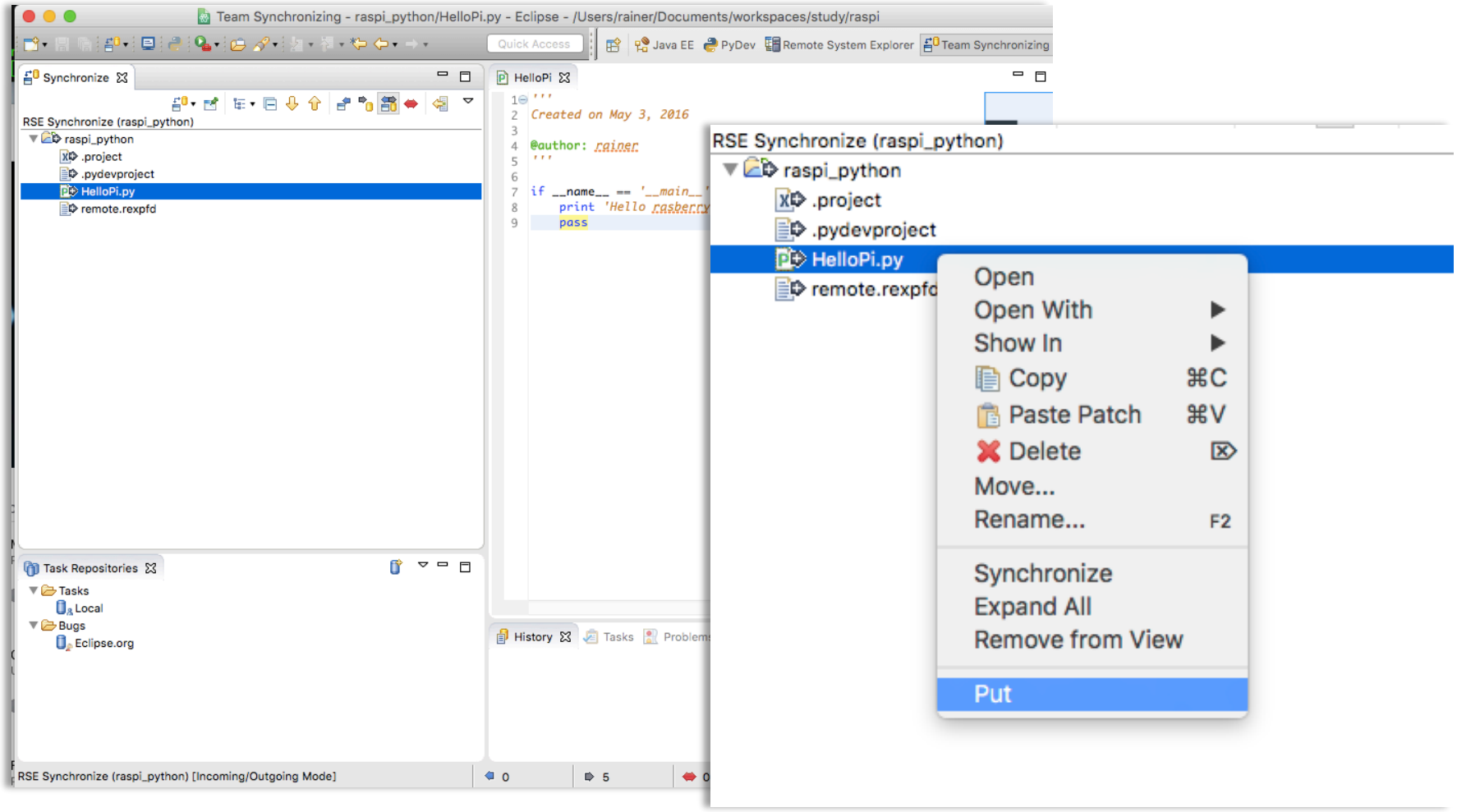
- Description file 저장



개발 환경

❖ 원격 적용

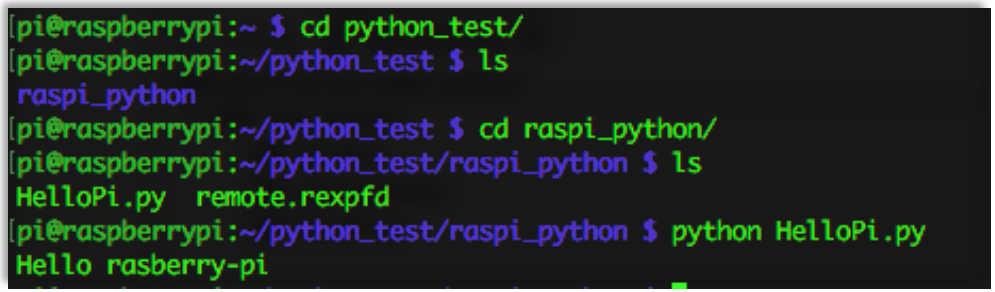
- Synchronize View로 원격 적용



개발 환경

❖ 원격 적용

- 터미널로 접속해서 실행

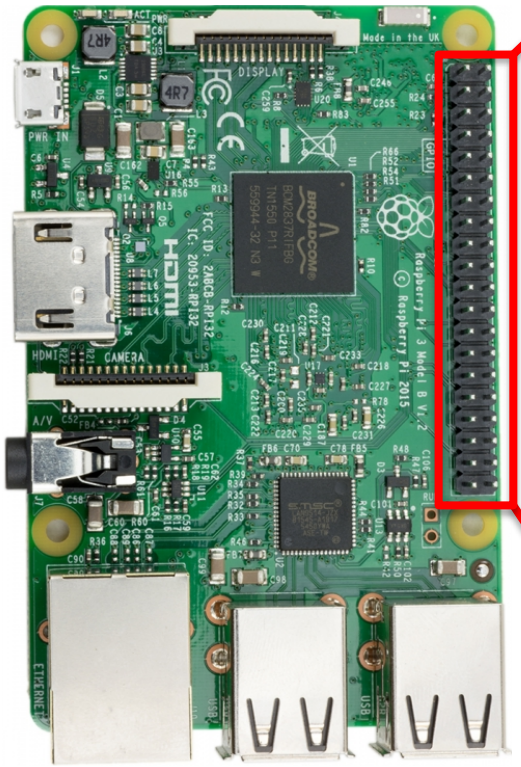


세부목차

1. Introduction
2. OS(Rasbian) Installation
3. Development Environment
4. **GPIO**
5. Digital Output
6. Digital Input
7. Analog Output - PWM
8. Analog Input - RC Circuit
9. 고수준 Sensor Modules

GPIO

❖ GPIO Pin Header



Raspberry Pi 3 GPIO Header

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I²C)		DC Power 5v	04
05	GPIO03 (SCL1 , I²C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)		(I²C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Rev. 2
29/02/2016

www.element14.com/RaspberryPi

GPIO

❖ GPIO 제어

- Sysfs
  - 리눅스 커널 2.6+
  - 특수 파일 시스템
  - GPIO 포트 제어를 위한 드라이버 포함
  - 터미널 직접 조작
  - System call을 이용한 C 프로그래밍
- 저수준 C 프로그래밍
  - BCM 28xx 레지스터 접근
  - Soc 데이타 시트 참조
- C 라이브러리
  - Wiring-pi
    - http://wiringpi.com/ (Gordon)
  - BCM2835
    - http://www.airspayce.com/mikem/bcm2835/

GPIO

❖ Python GPIO Modules

- Rpi.GPIO
  - https://pypi.python.org/pypi/RPi.GPIO
  - https://sourceforge.net/projects/raspberry-gpio-python/ (Ben Croston)
  - Raspberry Pi 기본 설치
  - 사용 용이
  - GPIO와 Software PWM만 지원
  - 실시간성 어플리케이션에 부적합
- WiringPi-Python
  - https://github.com/WiringPi/WiringPi-Python
  - C언어로 구현된 WringPi의 wrapped version
  - Arduino style의 코드를 지원
  - GPIO, Serial, SPI, I2C, Hardware PWM 등 하드웨어 기능 모두 사용
  - WriginPi Library 종속성
  - Raspberry-Pi build/install 필요

❖ Sysfs를 이용한 콘솔 명령

```
$ echo “18” > /sys/class/gpio/export
$ ls /sys/class/gpio/

$ echo “out” > /sys/class/gpio/gpio18/direction
$ echo “1” > /sys/class/gpio/gpio18/value
$ echo “0” > /sys/class/gpio/gpio18/value

$ echo “18” > /sys/class/gpio/unexport
$ ls /sys/class/gpio/
```

❖ Sysfs를 이용한 C Program - gpio_led.c

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

int main( int argc, char **argv) {
        int gpio_pin = 18;
        int fd;
        char buff[BUFSIZ];

       fd = open("/sys/class/gpio/export", O_WRONLY);
        if (fd == -1) {
                perror("fail to open export." );
                return 1;
       }
        sprintf(buff, "%d" , gpio_pin);
        if (write (fd, buff, sizeof(buff)) == -1) {
                perror("fail to export GPIO" );
                return 1;
       }
        close(fd);
```

```
 sprintf(buff, "/sys/class/gpio/gpio%d/direction" ,
gpio_pin);
       fd = open(buff, O_WRONLY);
        if (fd == -1) {
                perror("fail to open GPIO Direction  File" );
                return 1;
       }
        if (write (fd, "out" , 4) == -1) {
                perror("fail to set direction\n" );
                return 1;
       }
        close(fd);
```

❖ Sysfs를 이용한 C Program <계속>

```
sprintf(buff, "/sys/class/gpio/gpio%d/value" , gpio_pin);
      fd = open(buff, O_WRONLY);
       int i=0;
       for(i=0; i<10; i++) {
            if (write (fd, "1" , sizeof("1")) == -1) {
                 perror("fail to set value:1" );
                 return 1;
            }
            printf("set gpio %d value as HIGH\n", gpio_pin);
            sleep(1);
            if (write (fd, "0" , sizeof("0")) == -1) {
                 perror("fail to set value:0" );
                 return 1;
            }
            printf("set gpio %d value as LOW\n", gpio_pin);
            sleep(1);
      }
       close(fd);
```

```
fd = open("/sys/class/gpio/unexport" , O_WRONLY);
      sprintf(buff, "%d" , gpio_pin);
      write(fd, buff, strlen(buff));
      close(fd);

      return 0;
}
```

```
$ gcc –o gpioled gpioled.c
$ sudo ./gpioled
```

GPIO

❖ Wiring-Pi

- 설치
  - sudo apt-get install git-core
  - Git clone git://git.drogon.net/wiringPi
  - Cd wiringPi
  - ./build
- Gpio 명령
  - wiringPi를 설치하면 gpio 명령을 사용할 수 있다.

```
$ gpio -g mode 18 out
$ gpio -g write 18 1
$ gpio -g write 18 0
```

LED Blink Wiring-Pi Code

- wiringpi_led.c

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>

#define LED 1 // BCM18
int main( void) {
      puts("!!!Hello wiringPi!!!" ); /* prints !!!Hello World!!! */
      int i;

      wiringPiSetup();
      pinMode(LED, OUTPUT);
      for(i=0; i<10; i++){
            digitalWrite(LED, HIGH);
            printf("pin %d HIGH\n" , LED);
            delay(1000);
            digitalWrite(LED, LOW);
            printf("pin %d LOW\n" , LED);
            delay(1000);
      }
      return EXIT_SUCCESS;
}
```

```
$ gcc –o wiringpi_led wiringpi_led.c –lwiringPi
$ sudo ./wiringpi_led
```

GPIO

❖ Rpi.GPIO 기본 함수

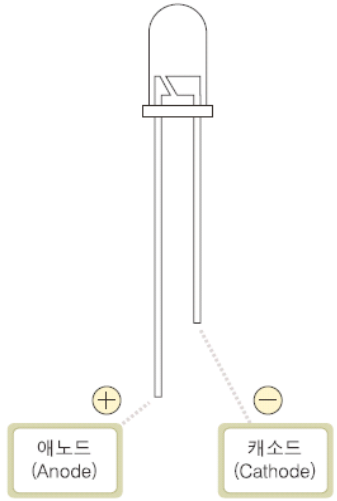
- import Rpi.GPIO as GPIO
  - 모듈 import
- GPIO.setmode(GPIO.BCM or GPIO.BOARD)
  - GPIO.BCM : Broadcom Soc에서 정의한 번호
  - GPIO.BOARD : 보드에 핀 순번
- GPIO.setup(channel, direction[, *intial*=state])
  - channel : GPIO Pin 번호
  - direction : GPIO.IN, GPIO.OUT
  - state : GPIO.HIGH, GPIO.LOW
- GPIO.setup(channel, GPIO.IN [, pull_up_down=pud])
  - pud = GPIO.PUD_UP, GPIO.PUD_DOWN
- GPIO.input(channel)
- GPIO.output(channel, state)
- GPIO.cleanup()
  - 종료전에 자원 반납
- https://sourceforge.net/p/raspberry-gpio-python/wiki/Examples/

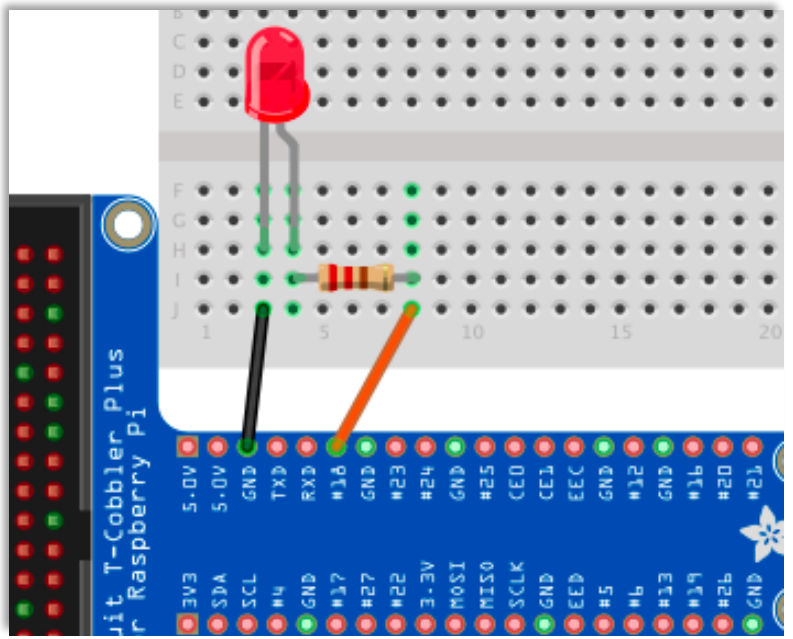
세부목차

1. Introduction
2. OS(Rasbian) Installation
3. Development Environment
4. GPIO
5. **Digital Output**
6. Digital Input
7. Analog Output - PWM
8. Analog Input - RC Circuit
9. 고수준 Sensor Modules

Digital Output

- **LED Blink**
  - LED 깜박이기
  - 부품
    - LED(Light Emitting Diode)
      - 짧은 다리 : Cathod(캐소드), 음극(-)
      - 긴 다리 : Anode(애노드), 양극(+)
      - 캐소드 쪽 머리 테두리가 깍여 있다.
    - 저항(Register)
      - 전류의 흐름을 방해
      - 전압,전류 저하
      - 전극이 없음
      - 단위 : Ω(ohm, 옴)
  - 회로 연결
    - LED 긴 다리 : GPIO18 + 저항
    - LED 짧은 다리 : GND





Digital Output

❖ 저항 값 구하기

- LED 전압, 전류 확인
  - Data sheet
  - 5mm 적색 기준
  - 전류 20mA
  - 전압 약 1.8V~2.2V
- 옴의 법칙
  - V = I * R
  - R = V / I
  - (공급전압 - LED전압) / 전류
  - (5 - 2) / 0.02 = 150
  - 150에 근사한 값의 저항 사용 : 220Ω

Absolute Maximum Ratings: (Ta=25℃) .

ITEMS	Symbol	Absolute Maximum Rating	Unit
Forward Current	I_F	20	mA
Peak Forward Current	I_{FP}	30	mA
Suggestion Using Current	I_{su}	16-18	mA
Reverse Voltage (V_R=5V)	I_R	10	uA
Power Dissipation	P_D	105	mW
Operation Temperature	T_{OPR}	-40 ~ 85	℃
Storage Temperature	T_{STG}	-40 ~ 100	℃
Lead Soldering Temperature	T_{SOL}	Max. 260℃ for 3 Sec. Max. (3mm from the base of the epoxy bulb)	

Absolute Maximum Ratings: (Ta=25℃)

ITEMS	Symbol	Test condition	Min.	Typ.	Max.	Unit
Forward Voltage	V_F	I_F=20mA	1.8	---	2.2	V
Wavelenength (nm) or TC(k)	Δλ	I_F=20mA	620	---	625	nm
*Luminous intensity	I_V	I_F=20mA	150	---	200	mcd
50% Viewing Angle	2θ1/2	I_F=20mA	40	---	60	deg

Forward Current	I_F	20				mA
Forward Voltage	V_F	I_F=20mA	1.8	---	2.2	V

Digital Output

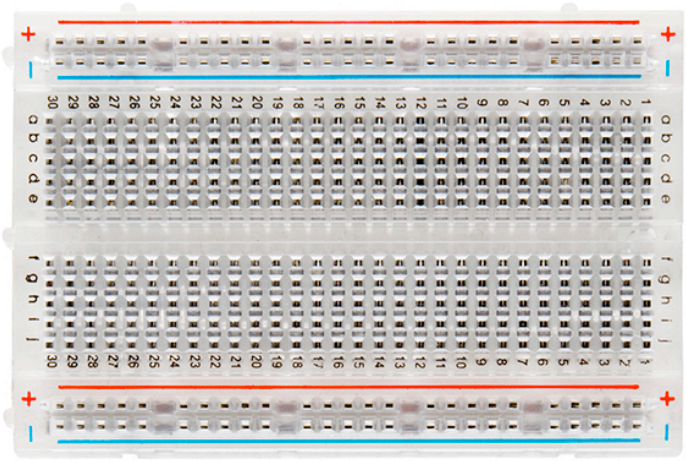
❖ **저항값 읽기**

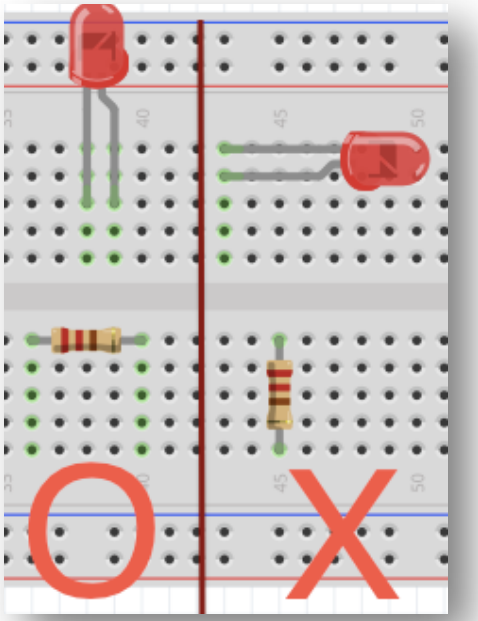
- 과거 숫자 인쇄 기술 부족
- 4개 또는 5개의 색갈 띠
- 금색 또는 은색을 오른쪽에
- 4색인 경우
  - 3번째는 승수(0의 갯수)
  - 4번째는 오차범위
- 5색인 경우
  - 4번째는 승수(0의 갯수)
  - 5번째는 오차범위
- 예시
  - 갈색(1),검정(0),오렌지(10^3), 금색
    - 10,000Ω = 10KΩ, ±5%
  - 빨강(2),빨강(2),갈색(10^1),금색
    - 220Ω, ±5%
  - 오렌지(3),오렌지(3),갈색(10^1), 금색
    - 330Ω, ±5%
  - 갈색(1), 검정(0), 검정(0), 노랑(10^4), 갈색
    - 1,000,000Ω = 1MΩ ,±1%

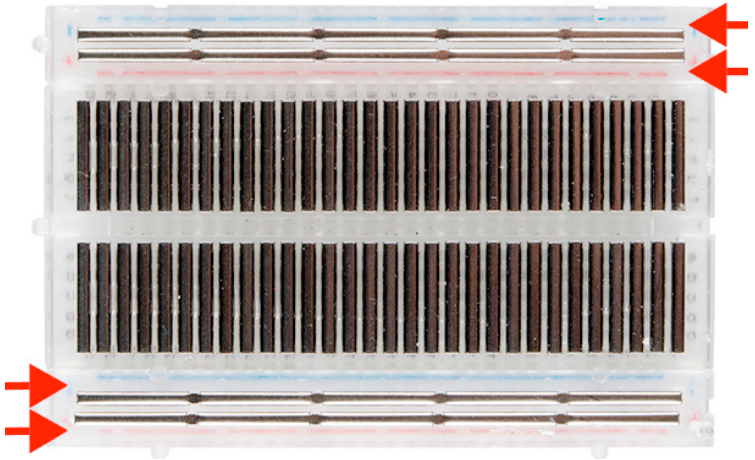
색명 (COLOR)		색띠			승수 (MULTIPLIER)	허용차 (TOLERANCE)
		1ST BAND	2ND BAND	3RD BAND		
흑색 BLACK		0	0	0	0	
갈색 BROWN		1	1	1	1	±1%(F)
빨강 RED		2	2	2	2	±2%(G)
등색 ORANGE		3	3	3	3	
황색 YELLOW		4	4	4	4	
녹색 GREEN		5	5	5	5	±0.5%(D)
청색 BLUE		6	6	6	6	±0.25%(C)
자색 GVIOLET		7	7	7	7	±0.1%(B)
회색 GRAY		8	8	8	8	±0.05%(A)
백색 WHITE		9	9	9	9	
금색 GOLD					-1	±5%(J)
은색 SILVER					-2	±10%(K)

Digital Output

- Bread Board (빵판)
  - PCB 보드를 만들기 전 프로토타입
  - 납땜할 필요 없음
  - 분리 및 재조립 가능
  - 같은 열끼리 연결
  - 좌우 세로 줄 전원 연결(버스영역)
  - 중앙 5칸씩 , 부품 연결(IC 영역)

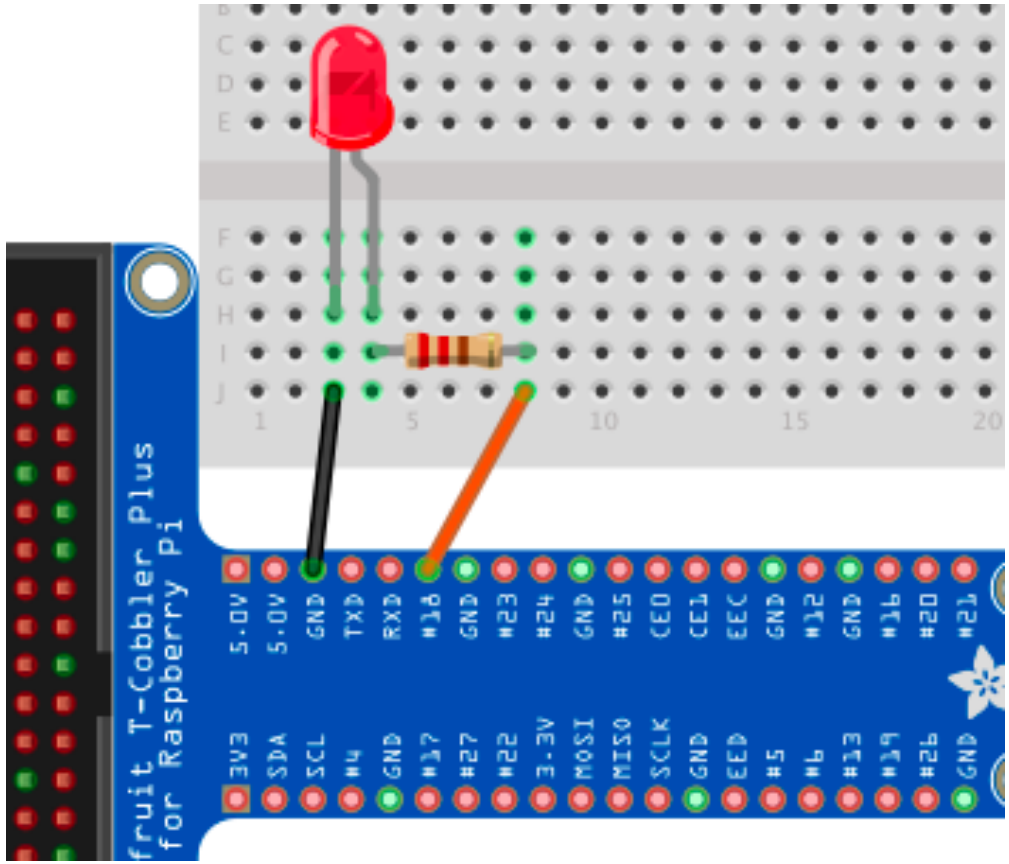






Digital Output

❖ **LED Blink 회로구성**



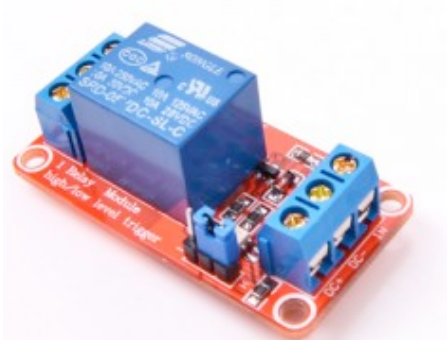
Digital Output

❖ LED Blink Python Code

```
import RPi.GPIO as GPIO
import time


led_pin = 18

try:
   GPIO.setmode(GPIO.BCM)
   GPIO.setup(led_pin, GPIO.OUT)

   while True:
      GPIO.output(led_pin, True)
      time.sleep(0.5)
      GPIO.output(led_pin, False)
      time.sleep(0.5)
finally:
   print 'clean up'
   GPIO.cleanup()
```

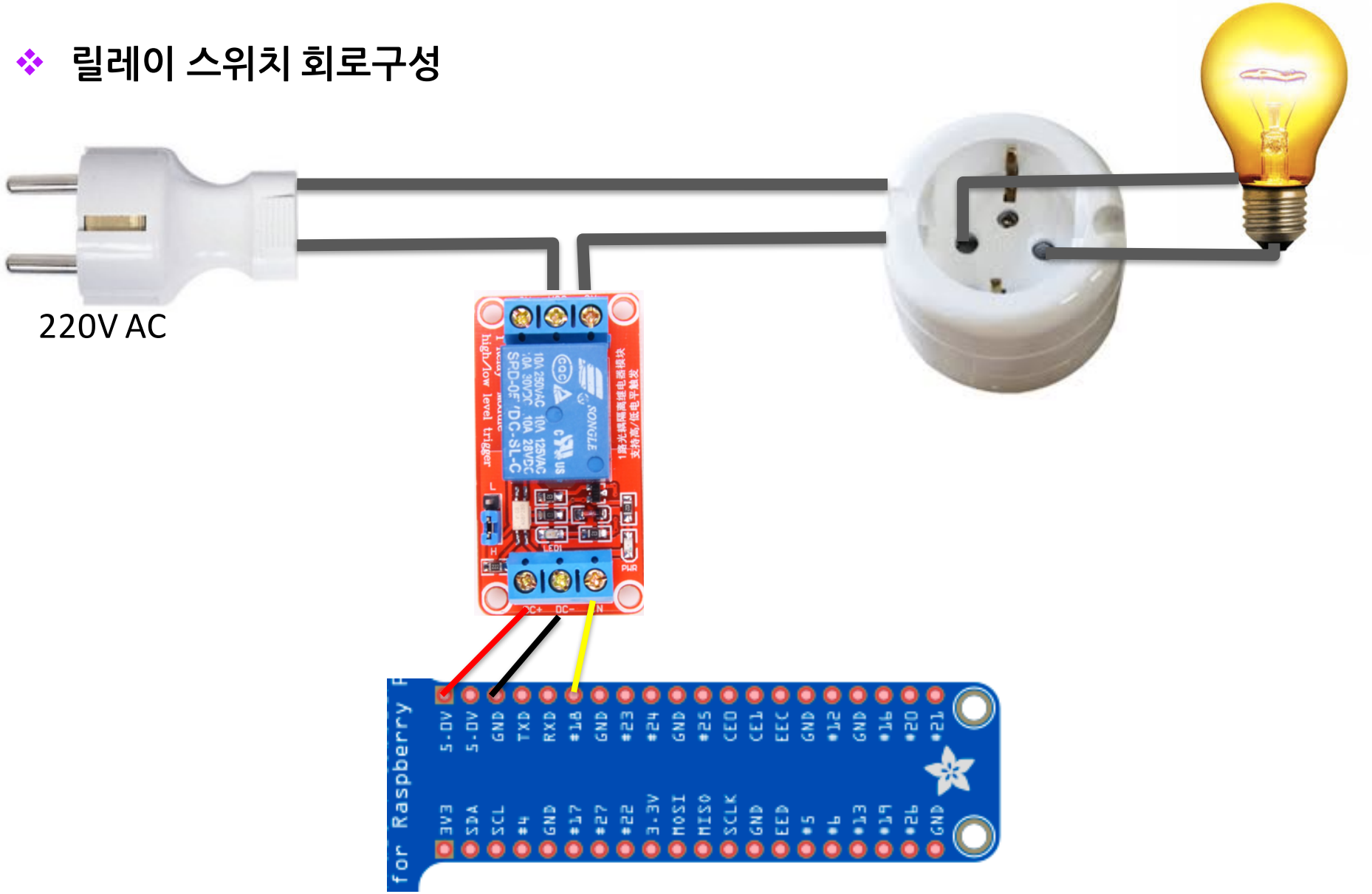
Digital Output

- **릴레이 스위치**
  - 220V 가전제품 켜고 끄기
  - 보드의 5V 전원 제어 이외 가전제품
  - 필요 부품
    - 릴레이 스위치
      - IN, 5V, GND
    - 무접점 반도체 릴레이
      - IN, GND
    - 220V 플러그 암/수



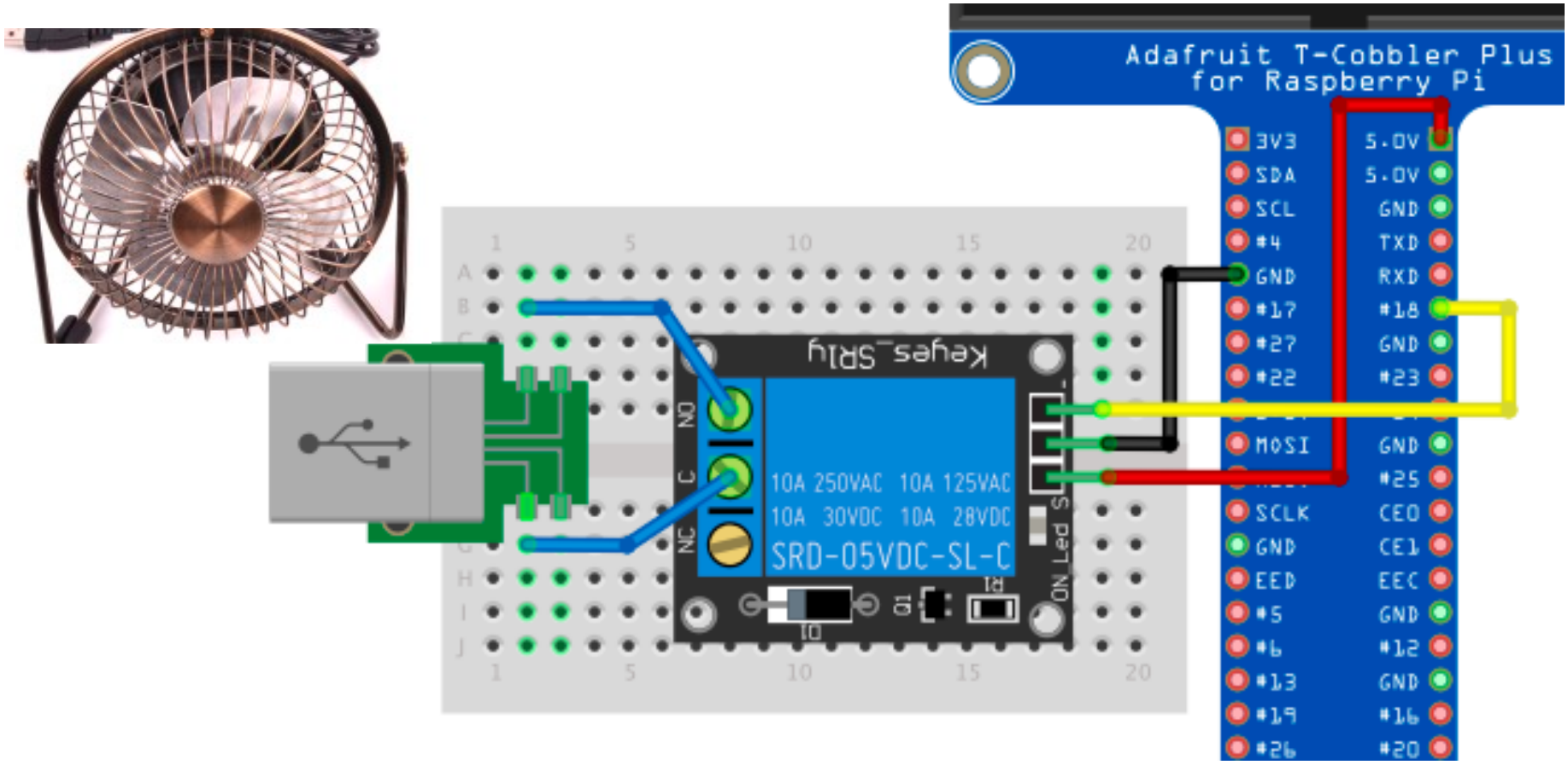
Digital Output

❖ **릴레이 스위치 회로구성**



Digital Output

❖ 릴레이 스위치 회로구성



Digital Output

❖ 릴레이 스위치 Code

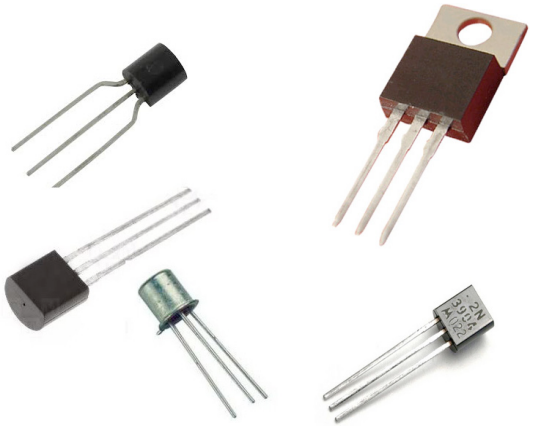
- LED Blink와 동일

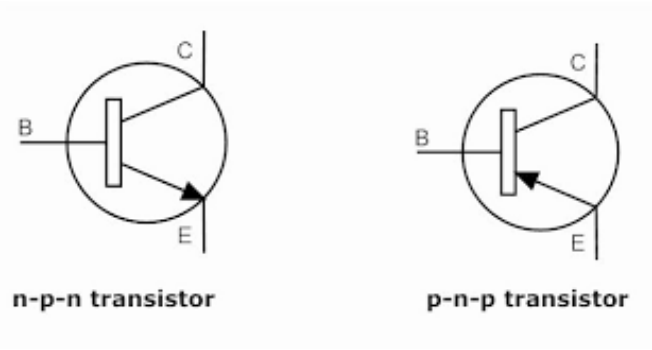
```
import RPi.GPIO as GPIO
import time

fan_pin = 18

try:
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(fan_pin, GPIO.OUT)

    while True:
        val = input("1:on, 0:off")
        GPIO.output(fan_pin, val)
finally:
    print 'clean up'
    GPIO.cleanup()
```

Digital Output

❖ 트랜지스터

- Transistor = Trans + Resistor
- 대표적인 반도체 소자
- 3개 단자:C(Collector), B(Base), E(Emitter)
- Base 단자 전류(전압)에 따라 내부 저항 변화
- 증폭 회로, 스위칭 회로에 활용
  - 자동차 가속 패달
- NPN
  - Base의 전압이 이미터 보다 높으면 동작
- PNP
  - Base의 전압이 이미터 보다 낮으면(0.6v) 동작





Digital Output

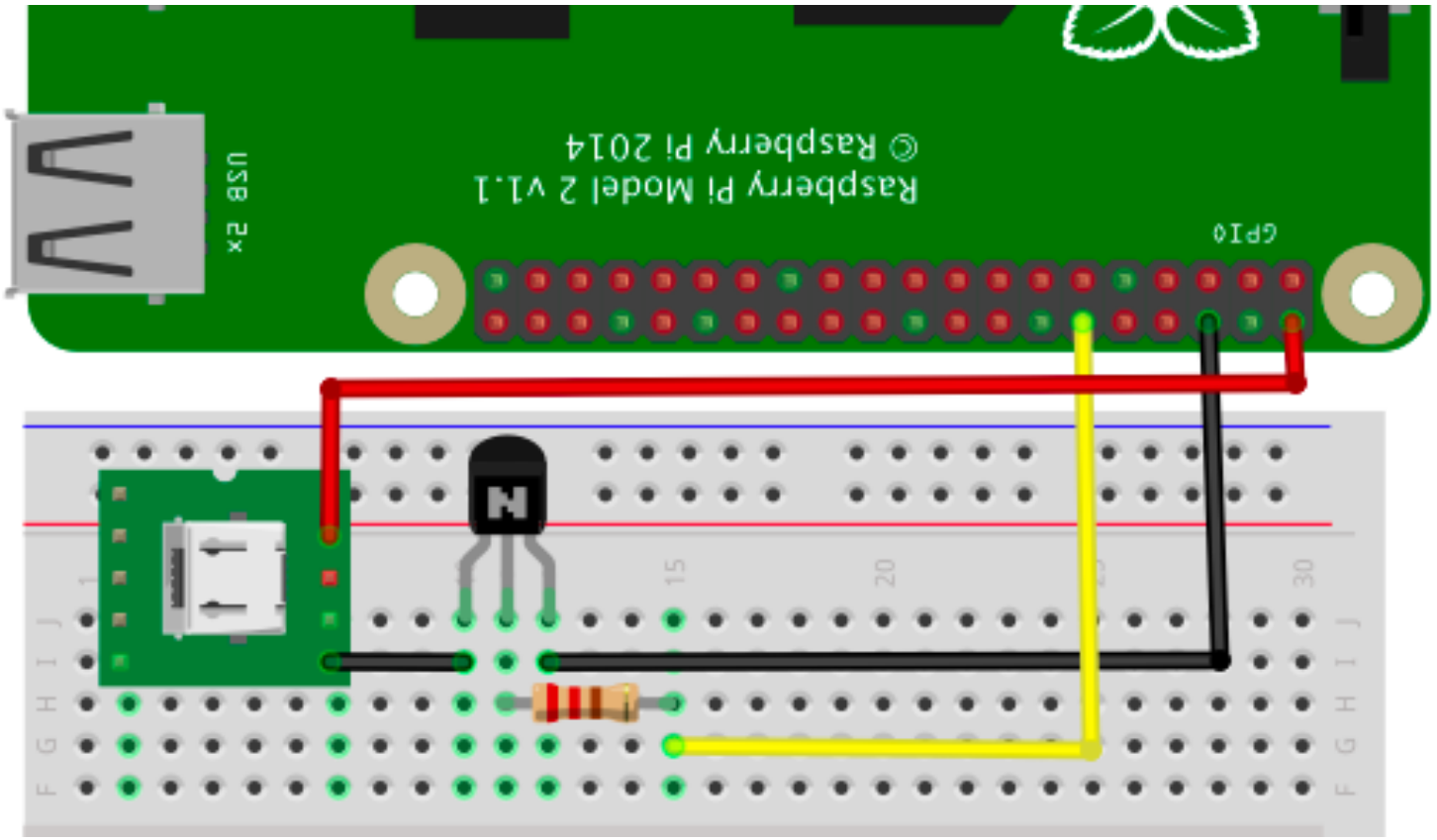
❖ 트랜지스터 스위치

- 2n2222
  - 많이 사용하는 PNP 접합형 트랜지스터
  - 전압이 아닌 전류를 제어
  - 저항을 이용해서 트랜지스터 보호
- 회로 구성
  - USB Connector Vcc - R-PI 5V
  - USB Connector GND - Collector
  - R-PI GPIO18 - Base(220옴)
  - R-PI GND - Emitter

Pin Configuration :
1. Emitter
2. Base
3. Collector

Digital Output

❖ **트랜지스터 스위치 회로구성**



❖ 트랜지스터 스위치 Code

- LED Blink와 동일

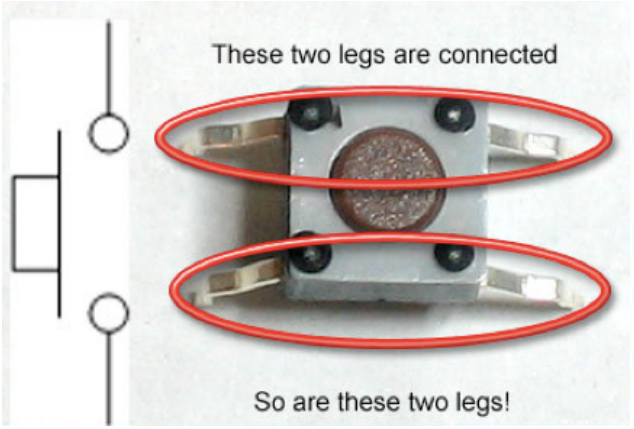
```
pin = 18

try:
   GPIO.setmode(GPIO.BCM)
   GPIO.setup(pin, GPIO.OUT)

   while True:
      val = input("swtich [on:1, off:0]")
      GPIO.output(pin, val)
finally:
   print 'clean up'
   GPIO.cleanup()
```

세부목차

1. Introduction
2. OS(Rasbian) Installation
3. Development Environment
4. GPIO
5. Digital Output
6. **Digital Input**
7. Analog Output - PWM
8. Analog Input - RC Circuit
9. 고수준 Sensor Modules

Digital Input

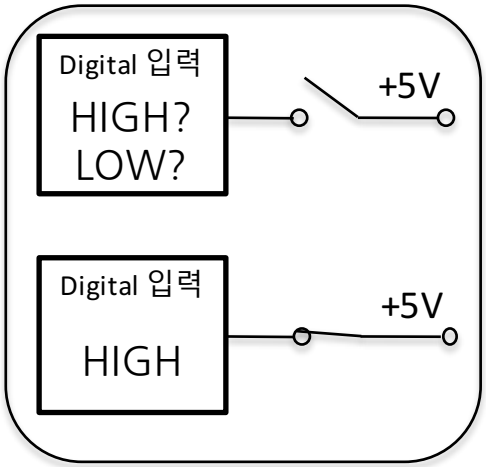
- **푸쉬 버튼 스위치 입력**
  - 버튼 스위치로 LED를 켜고 끔
  - 버튼 스위치
    - 두개의 다리는 붙어 있다
    - 4개 중 2개만 연결해도 된다.
  - 회로 연결
    - 13번 - LED - GND
    - 풀다운 저항
      - 5V - 버튼 - 10k 저항 - GND
      - 버튼 - Digital 7번
    - 풀업 저항
      - 5V - 10k저항 - 버튼 - Digital 7번
      - 버튼 - GND

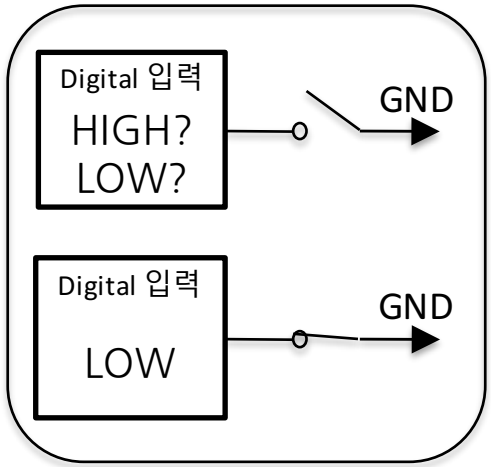


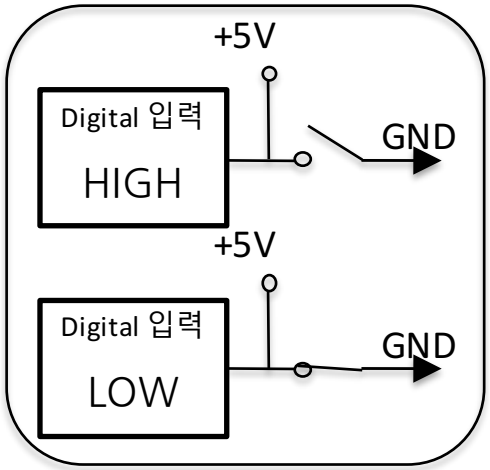
1. Digital Input

❖ 풀업(Pull Up), 풀다운(Pull Down) 저항

- 플로팅(Floating) 상태
  - 스위치가 열려있는 동안 어떤 상태인지 알 수 없는 상태
  - 주변 핀의 전압, 정전기 등 잡음에 취약
  - 스위치가 열려있는 동안 Vcc(5V) 또는 0V(GND)를 연결해서 해결
  - Vcc와 GND를 그대로 연결하면 단락되어 과전류 문제
    - 일반적으로 10KΩ 저항을 사용하여 해결
  - 저항을 전원(Vcc)에 연결하면 풀업(Pull Up), GND 연결하면 풀다운(Pull Down)

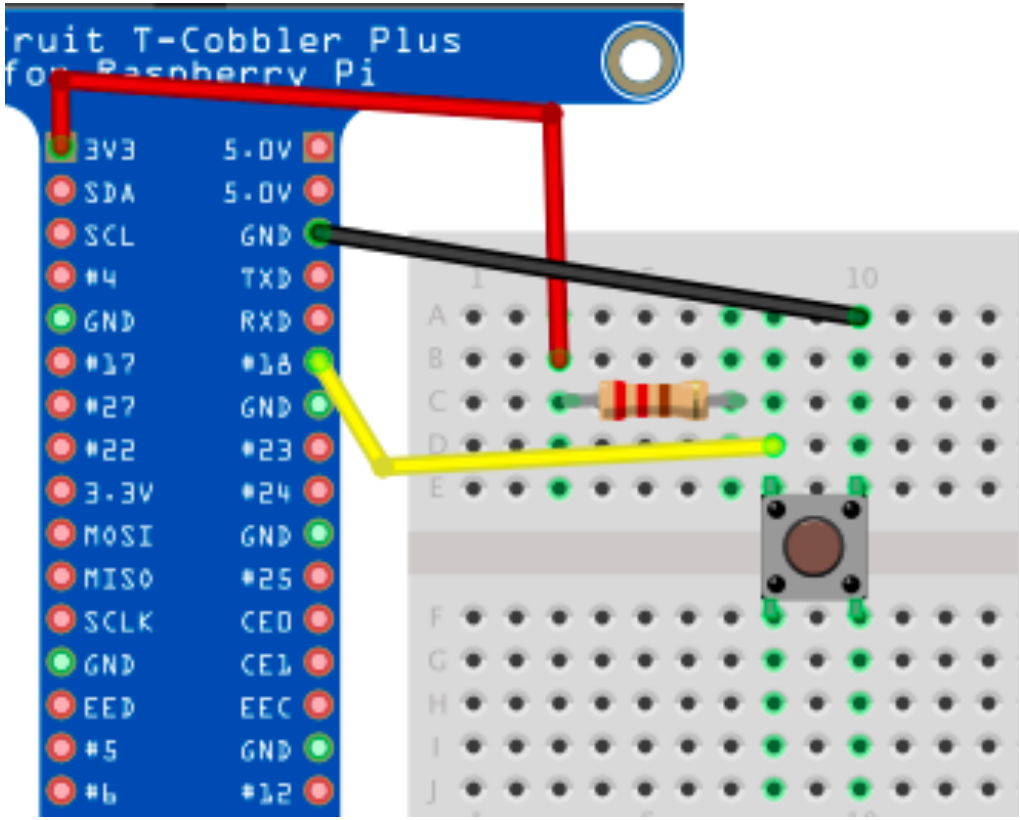






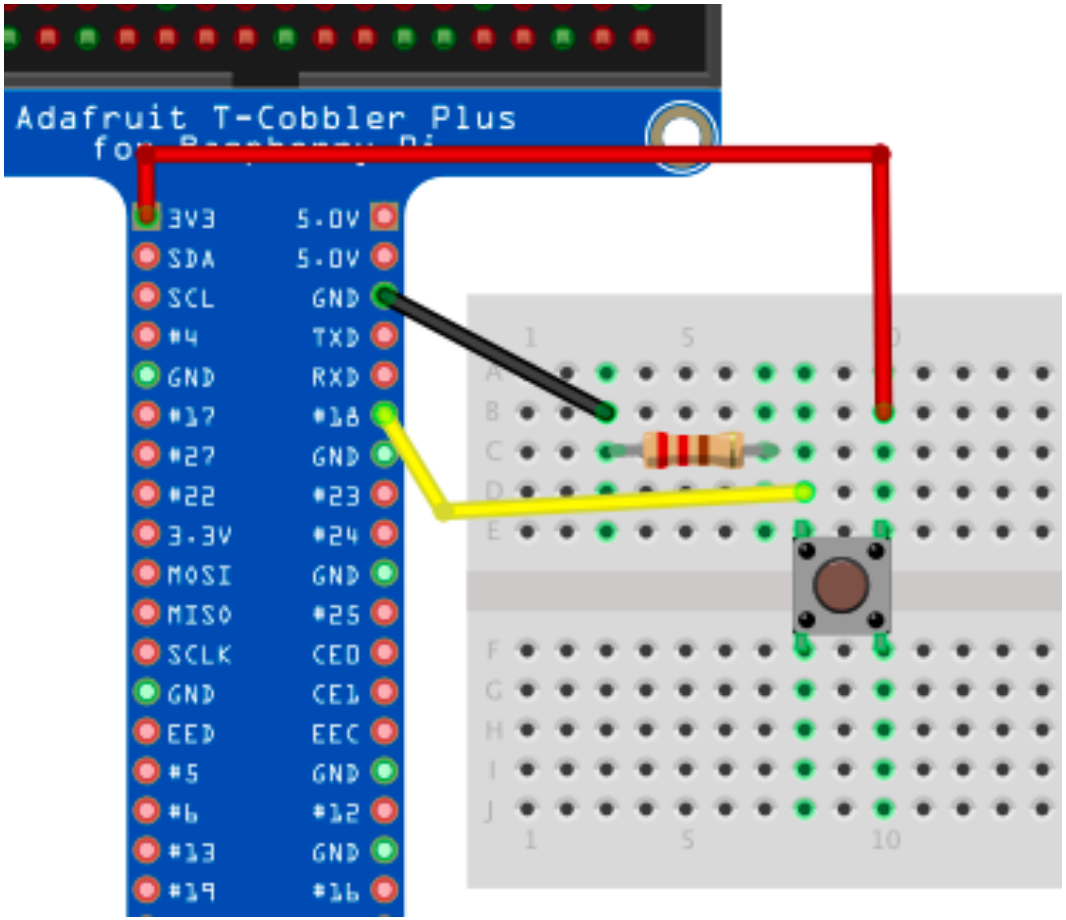
Digital Input

❖ **푸쉬버튼 스위치 회로(풀업 저항)**



Digital Input

❖ **푸쉬버튼 스위치 회로(풀다운 저항)**



Digital Input

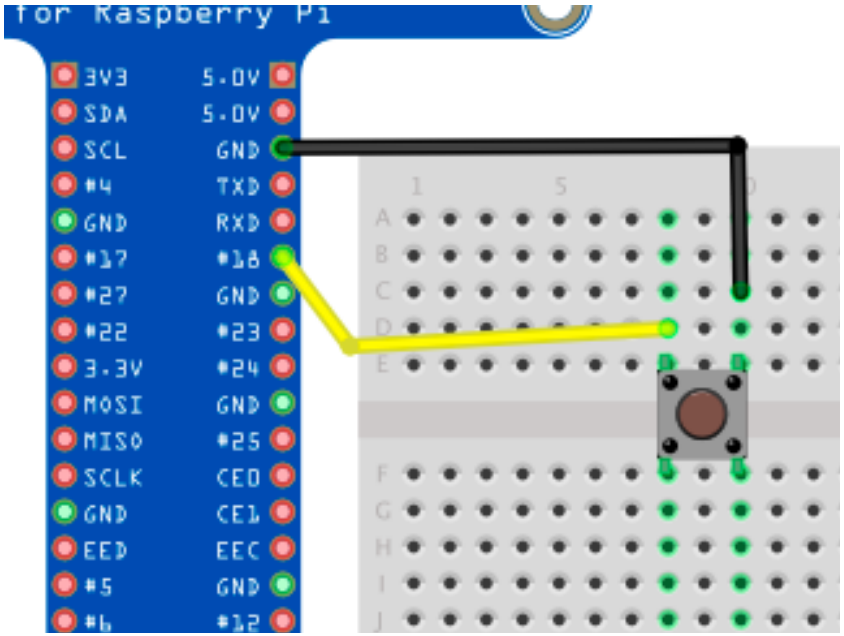
❖ 푸쉬버튼 스위치 Code

```
import RPi.GPIO as GPIO

pin = 18
try:
   GPIO.setmode(GPIO.BCM)
   GPIO.setup(pin, GPIO.IN)

   while True:
      print GPIO.input(pin)
finally:
   GPIO.cleanup()
```

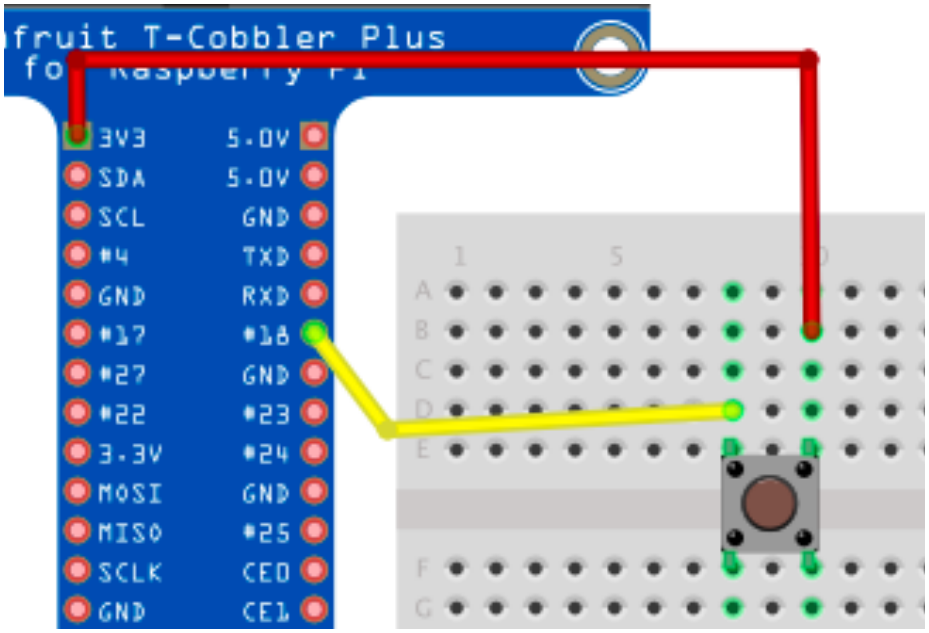
Digital Input

❖ 푸쉬버튼 회로(내부 풀업/풀다운)

- 풀업저항을 내부적으로 제공
- GPIO.setup(pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
- GPIO.setup(pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)



GPIO.PUD_UP



GPIO.PUD_DOWN

Digital Input

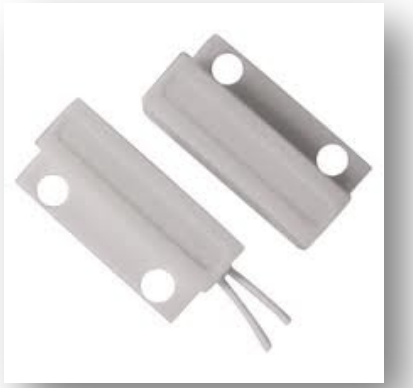
❖ 푸쉬버튼 스위치 내부 풀업/다운 Code

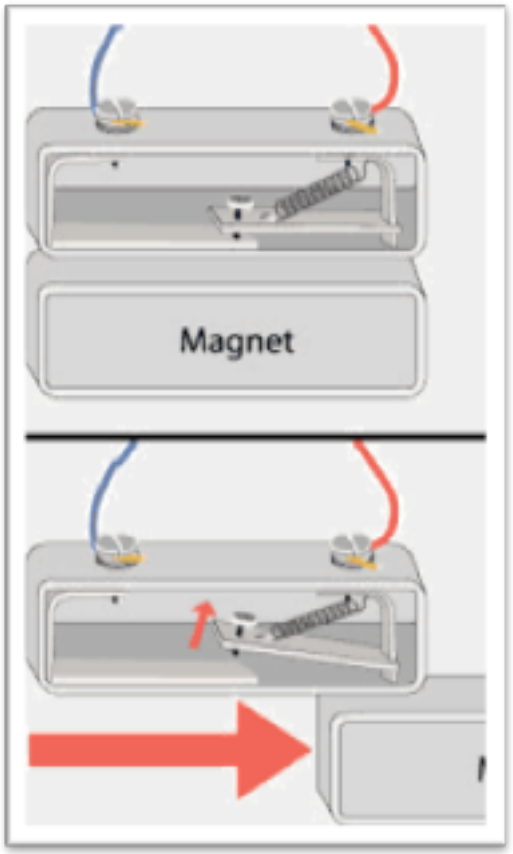
```
import RPi.GPIO as GPIO

pin = 18
try:
   GPIO.setmode(GPIO.BCM)
   #GPIO.setup(pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
   GPIO.setup(pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
   while True:
      print GPIO.input(pin)
finally:
   GPIO.cleanup()
```

Digital Input

❖ Magnetic Door Switch

- 문열림 탐지
- 두개의 자석이 붙고 떨어짐에 따라 동작하는 스위치
- 필요 부품
  - 자석 도어 스위치
  - 10KΩ 저항
- Push Button 과 동일한 회로와 스케치 사용





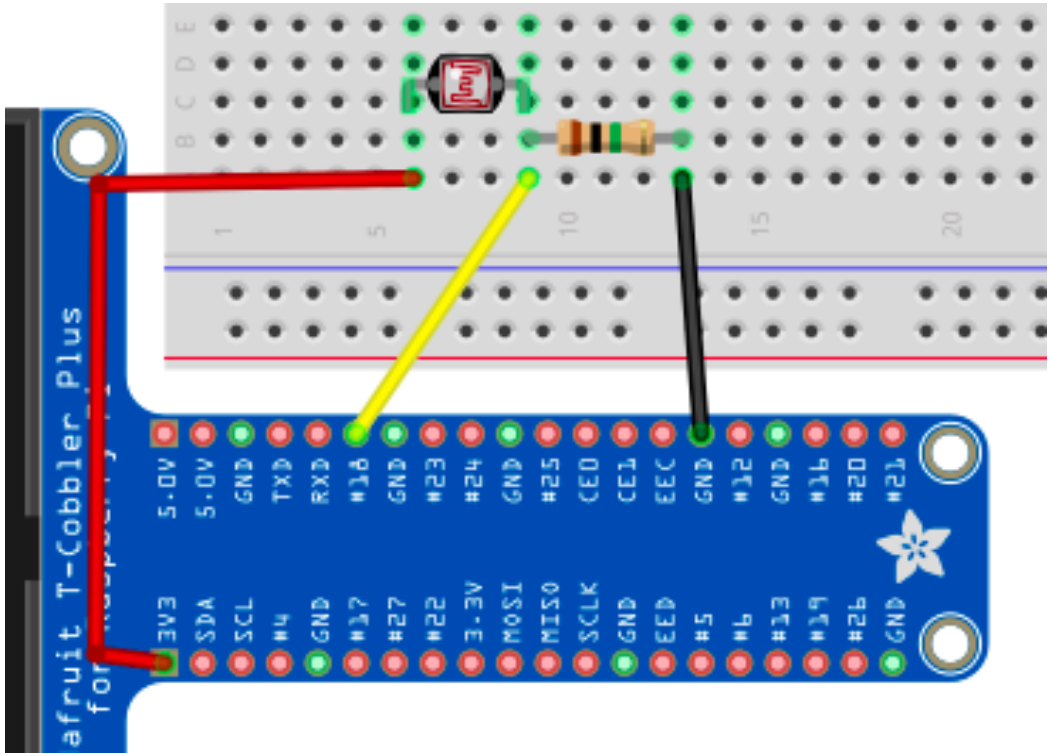
Digital Input

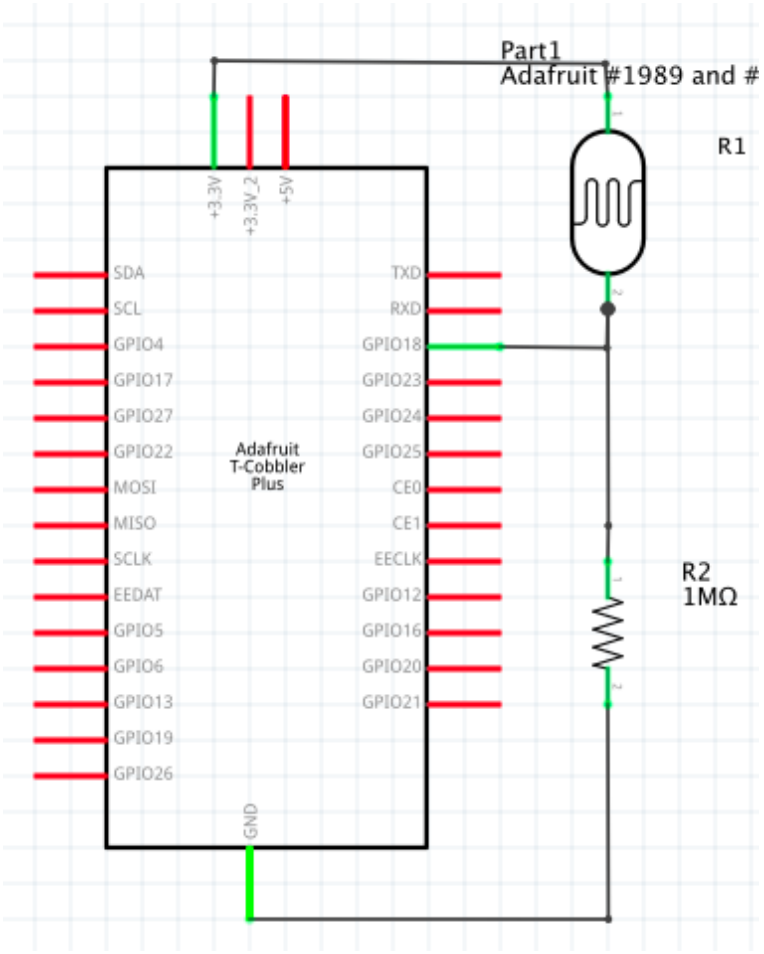
❖ LDR(Light Dependent Resistor)

- 조광센서
- 양음극 없음
- 가변저항, 빛에 따라 저항값 변화
  - 밝으면 저항 감소, 어두우면 저항 증가
  - 10Lux : 20 ~ 50kΩ
  - 0Lux : 2MΩ
- 회로
  - 1MΩ 풀다운 저항
  - 가변적인 전압의 특정 값 이상/이하 입력
  - Analog 신호를 Digital로 인식

Digital Input

❖ **LDR(조도센서) 회로구성**





Digital Input

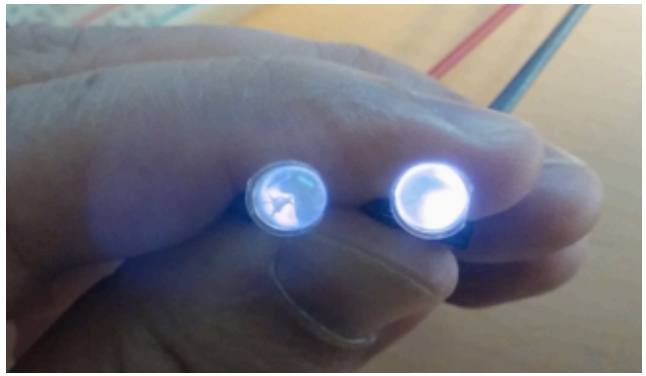
❖ LDR(조도센서) Code

```
import RPi.GPIO as GPIO
import time


try:
    pin = 18
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(pin, GPIO.IN)
    val = -1
    while True:
        read = GPIO.input(pin)
        if read != val:
            val = read
            print time.strftime("%Y%m%d-%H%M%S"), val
        #time.sleep(0.1)
finally:
    print "clean up."
    GPIO.cleanup()
```

Digital Input

❖ 적외선 센서(Infrared Sensor)

- 적외선 발생기(IRED)
  - Infrared Emitting Diode
  - LED 모양
  - 일반적으로 리모콘 끝 부분에 장착
  - 육안으로 빛을 볼 수 없슴
    - 카메라 뷰파인더로 확인 가능
  - 동작 전류 : 100mA
  - 동작 전압 : 1.3v ~ 1.7v
  - 필요 저항 : 5v 일때 35Ω
    - (5 - 1.5)v / 0.1A = 35



Absolute maximum ratings

Characteristic	Symbol	Ratings	Unit
Power Dissipation	P_D	150	mW
Forward Current	I_F	100	mA
*1Peak Forward Current	I_{FP}	1	A
Reverse Voltage	V_R	4	V
Operating Temperature	T_{opr}	-25~85	℃
Storage Temperature	T_{stg}	-30~100	℃
*2Soldering Temperature	T_{sol}	260℃ for 5 seconds	

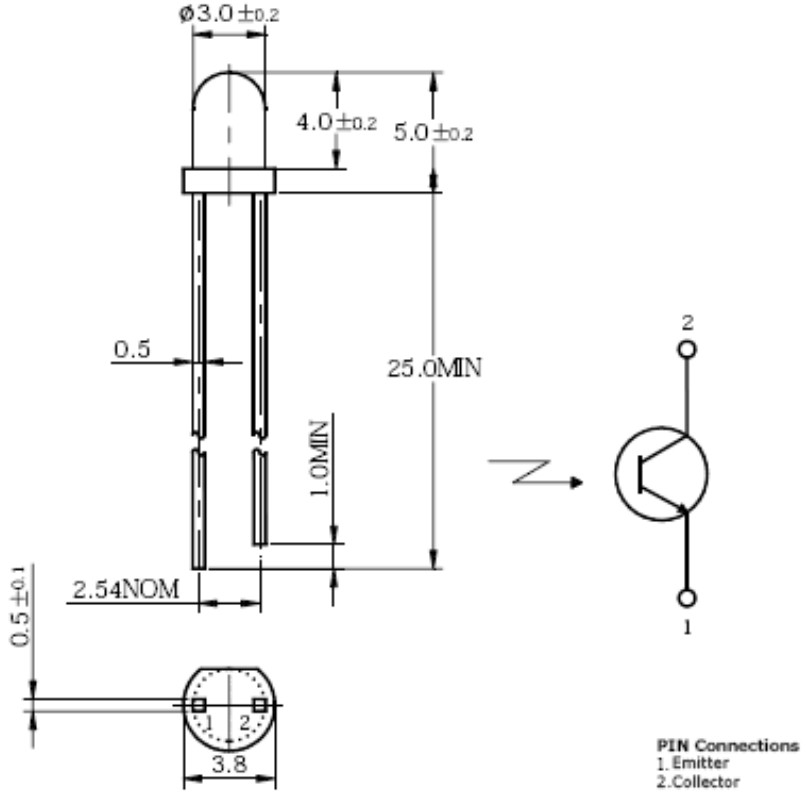
*1.Duty ratio = 1/16, Pulse width = 0.1ms
*2.Keep the distance more than 2.0mm from PCB to the bottom of IRED package

Electrical Characteristics

Characteristic	Symbol	Test Condition	Min.	Typ.	Max.	Unit
Forward Voltage	V_F	I_F= 50mA	-	1.3	1.7	V
Radiant Intensity	I_E	I_F= 50mA	30	70	-	mW/Sr

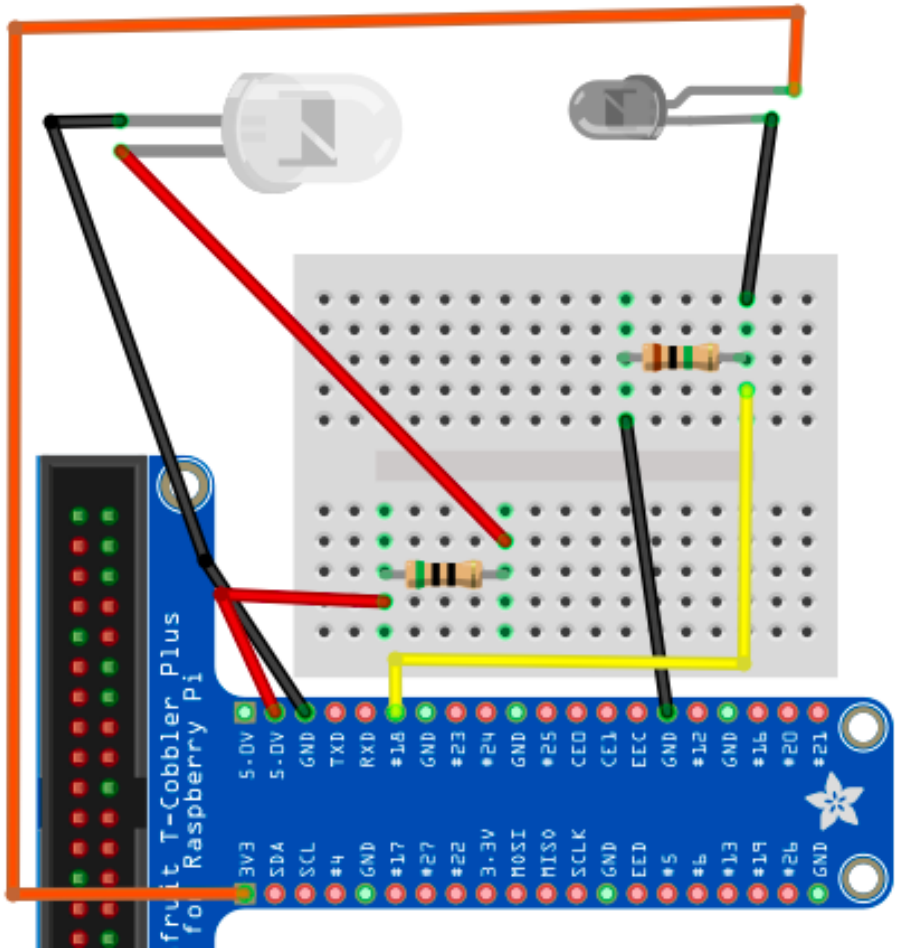
❖ 적외선 센서(Infrared Sensor)

- 적외선 수신기(Photo Transistor)
  - 적외선 수신에 따른 트랜지스터
  - 적외선 값에 따라 Collector-Emitter 저항 변화
  - Pull-down 저항 1MΩ



Digital Input

❖ **적외선 센서 회로구성**



Digital Input

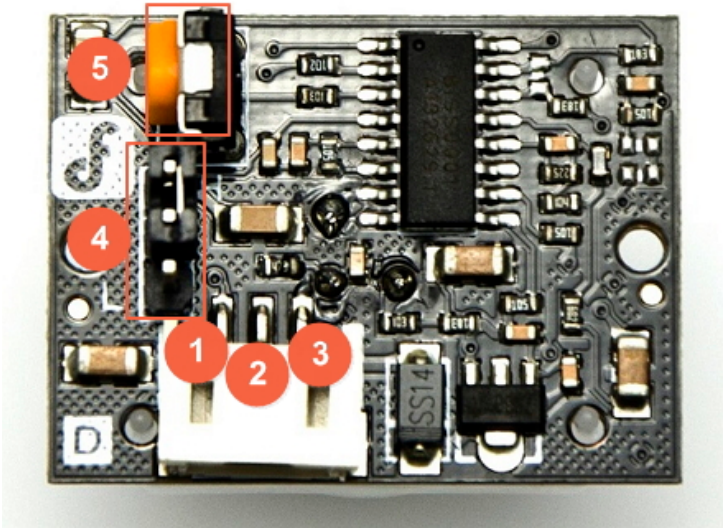
❖ 적외선 센서 Code

```
import RPi.GPIO as GPIO
import time


try:
    pin = 18
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(pin, GPIO.IN)
    val = -1
    while True:
        read = GPIO.input(pin)
        if read != val:
            val = read
            print time.strftime("%Y%m%d-%H%M%S"), val
        #time.sleep(0.1)
finally:
    print "clean up."
    GPIO.cleanup()
```

Digital Input

❖ 동작 감지하기

- 주변에 움직이는 것이 있는지 감지
- 사람이 움직이면 LED 켜짐
- 필요 부품
  - Passive Infrared, PIR 센서
    - 1 : Out
    - 2: Vcc
    - 3: GND
    - 4:
      - H: Repeatable
      - L : Unrepeatable
    - 5 : latency , 0.5s ~ 50s



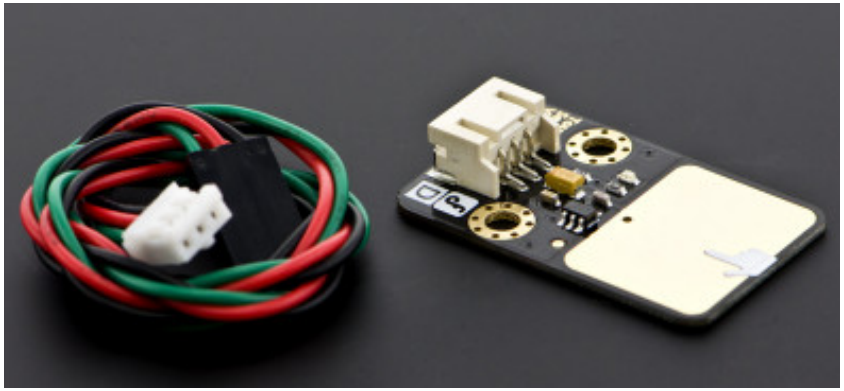
Digital Input

❖ PIR 센서 Code

```
import RPi.GPIO as GPIO
import time
from datetime import datetime

pri_pin = 18
try:
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(pri_pin, GPIO.IN)
    val = -1
    while True:
        read = GPIO.input(pri_pin)
        if val != read:
            val = read
            if val== 0:
                print str(datetime.now()), "No intruder"
            elif val == 1:
                print str(datetime.now()), "Intruder dectected"
        time.sleep(0.5)
finally:
    print 'clean up'
    GPIO.cleanup()
```

Digital Input

- **터치 센서**
  - 온도/습도 센서를 모듈로 구성
  - 3핀
    - Red : Vcc (3~5V)
    - Green : Data Out
    - Black : GND
  - 단순한 Digital Input



Digital Input

❖ 터치 센서 Code

```
import time
import datetime
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)

pad_pin = 5
GPIO.setup(pad_pin, GPIO.IN)

while True:
    pad_pressed = GPIO.input(pad_pin)

    if pad_pressed:
        current_datetime = datetime.datetime.now().strftime("%I:%M%p:%S on %B %d, %Y")
        print("pressed! - " + current_datetime)

    time.sleep(0.1)
```

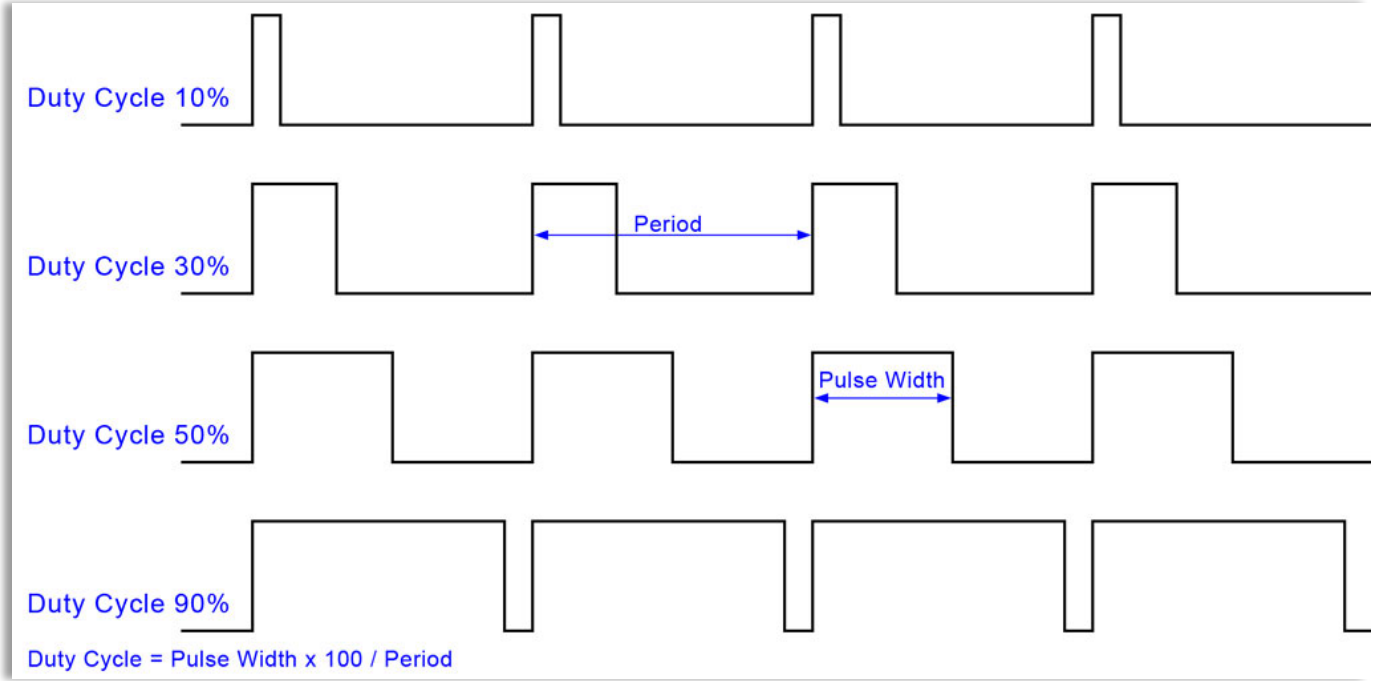
세부목차

1. Introduction
2. OS(Rasbian) Installation
3. Development Environment
4. GPIO
5. Digital Output
6. Digital Input
7. **Analog Output - PWM**
8. Analog Input - RC Circuit
9. 고수준 Sensor Modules

Analog Output - PWM

❖ Analog Out

- PWM(Pulse Width Modulation) 펄스 폭 변조
  - 지정된 주파수의 펄스의 폭을 조절하여 아날로그 신호로 사용
  - 주파수(Frequency) : Hz, 1초에 일어날 펄스의 갯수
  - 주기(Period) : 한 펄스의 지속 시간
  - Pulse Width : 하나의 Period에서 활성화된 기간
  - Duty Cycle : 한 주기 내에서 HIGH 상태 시간 비율

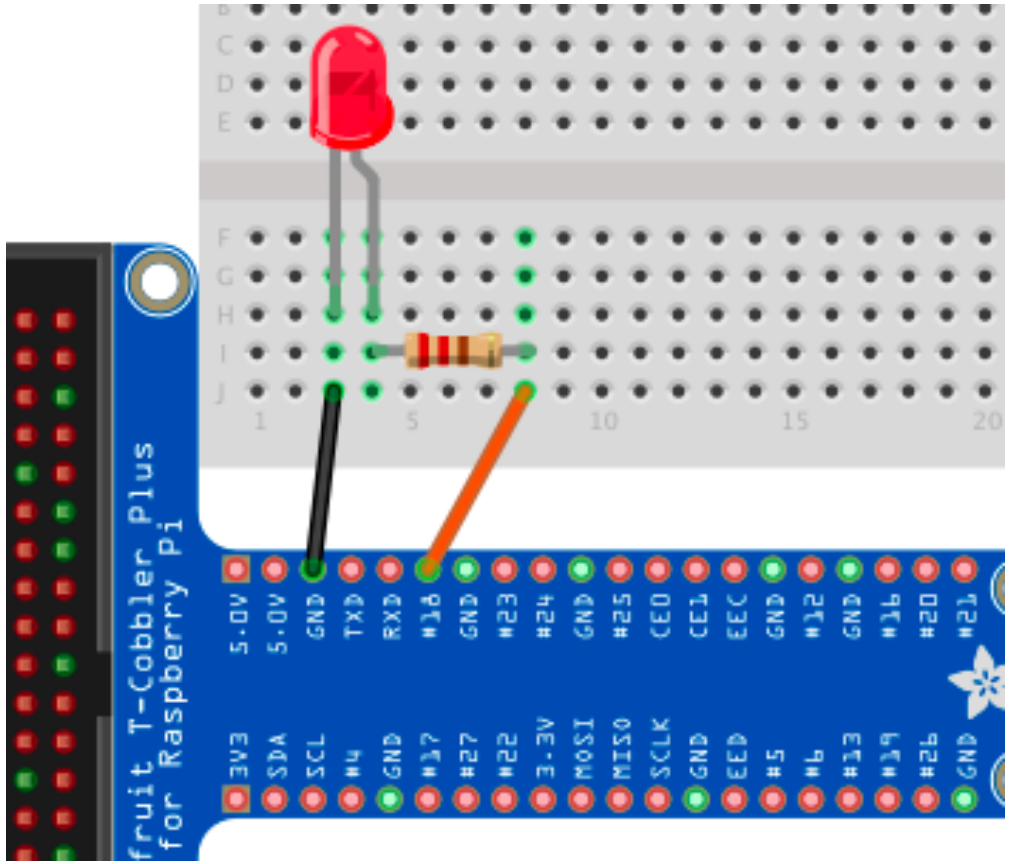


Analog Output - PWM

- **Analog Out**
  - Raspberry-Pi PWM
    - Hardware PWM : BCM2835 SoC
      - 레지스터 AL0
        - GPIO12,13 - PWM0, PWM1
      - 레지스터 AL1
        - GPIO18, 19 - PWM0, PWM1
    - Software PWM : Wiring-Pi Libarary
      - GPIO 제한 없슴
      - 정확도 떨어짐
  - LED Fading
    - GPIO18

Analog Output - PWM

❖ **LED Fade 회로 구성**



Analog Output - PWM

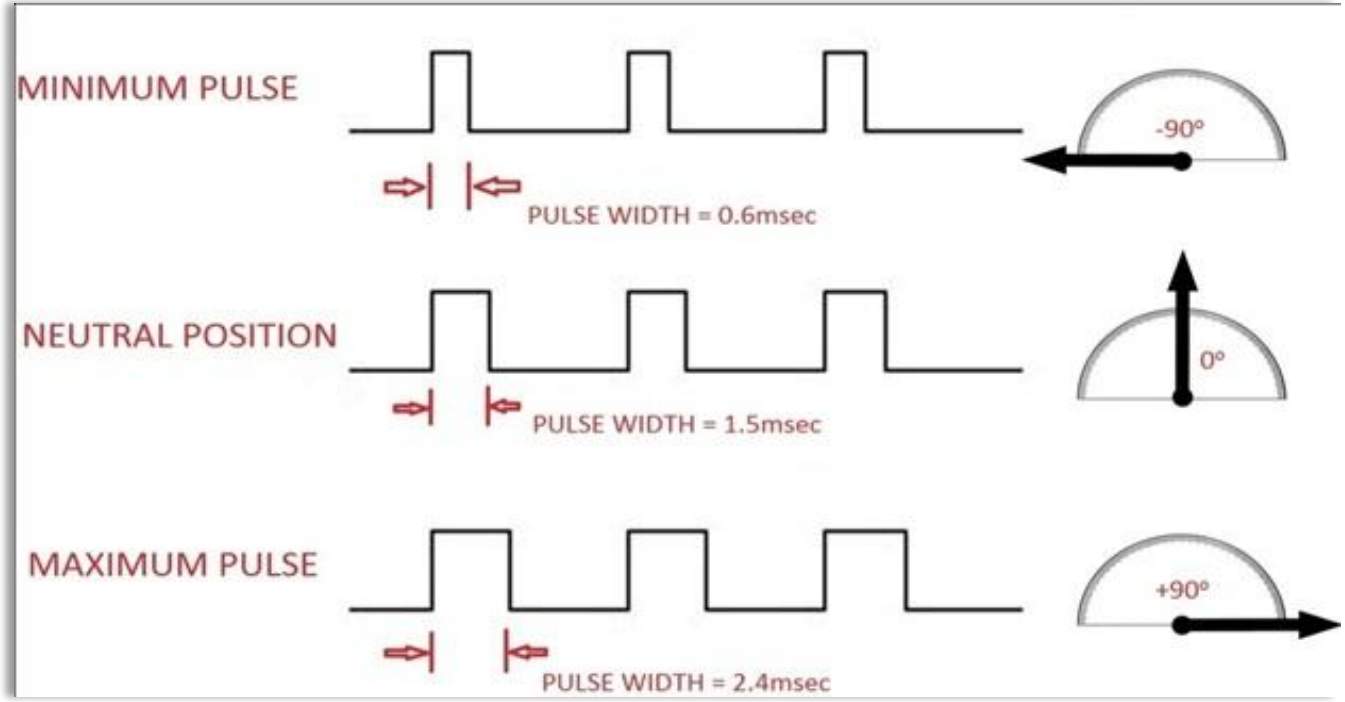
❖ LED Fade Code

```
import RPi.GPIO as GPIO
import time

try:
   pin = 26
   GPIO.setmode(GPIO.BCM)
   GPIO.setup(pin, GPIO.OUT)

   pwm = GPIO.PWM(pin, 100)
   pwm.start(0)

   while True:
      for i in range(0, 101):
         pwm.ChangeDutyCycle(i)
         time.sleep(0.05)
      for i in range(100, -1, -1):
         pwm.ChangeDutyCycle(i)
         time.sleep(0.05)
finally:
   pwm.stop()
   GPIO.cleanup()
```

Analog Output - PWM

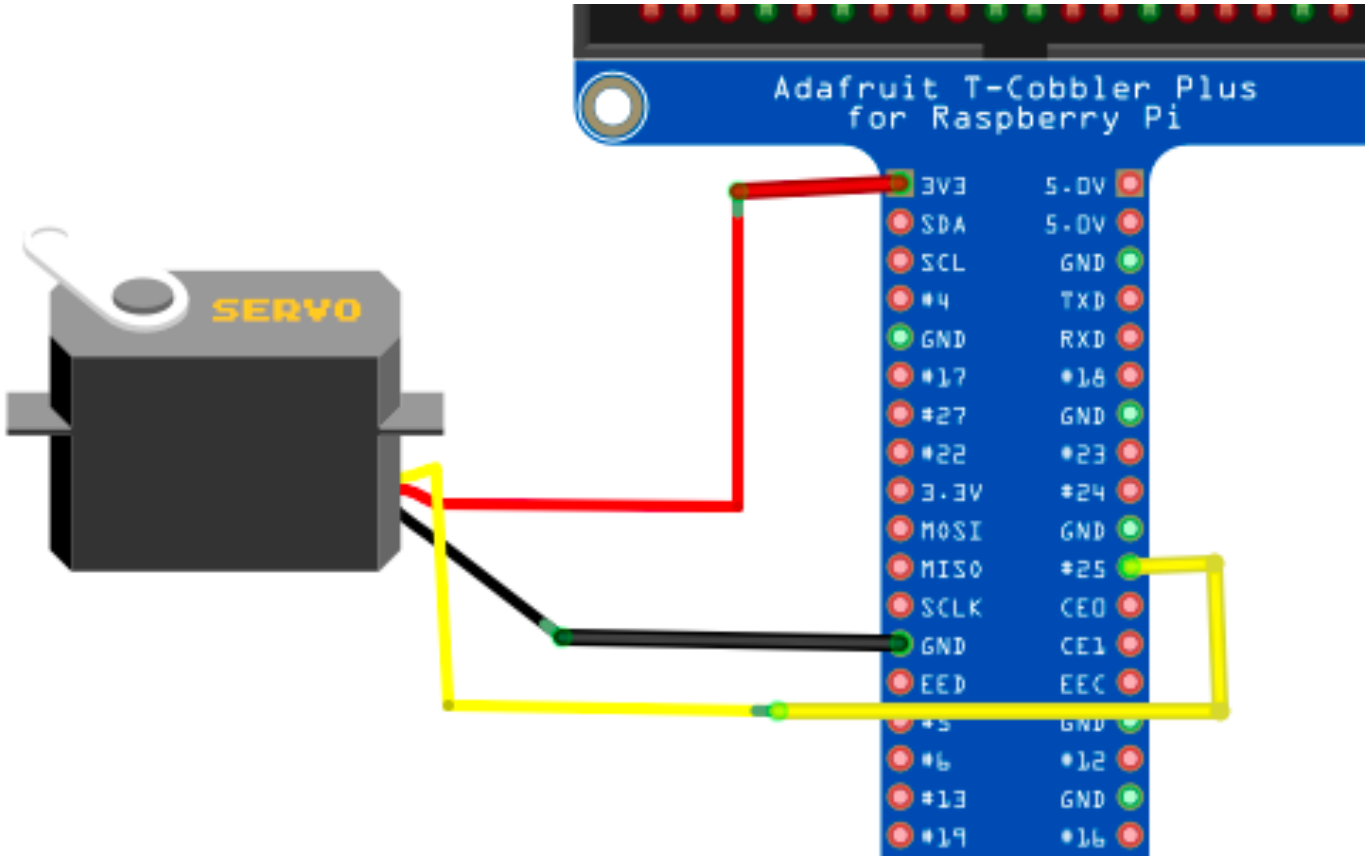
❖ Servo Motor

- 180º 회전 가능
- 각도 별 제어 가능
- PWM 펄스의 지속시간으로 제어
  - 0.5ms : -90º
  - 1.5ms : 0º
  - 2.5ms : +90º



Analog Output - PWM

❖ **Servo Motor 회로 구성**



Analog Output - PWM

❖ Servo Motor Code

- Frequency : 100Hz
  - 1초(1000ms)에 100Hz
  - 1Period = 10ms(1000/100)
- -90º : 0.5ms
  - 1Period의 5%
    - 0.5 / 10
- 0º : 1.5ms
  - 1Period의 15%
    - 1.5 / 10
- +90º : 2.5ms
  - 1Period의 25%
    - 2.5 / 10

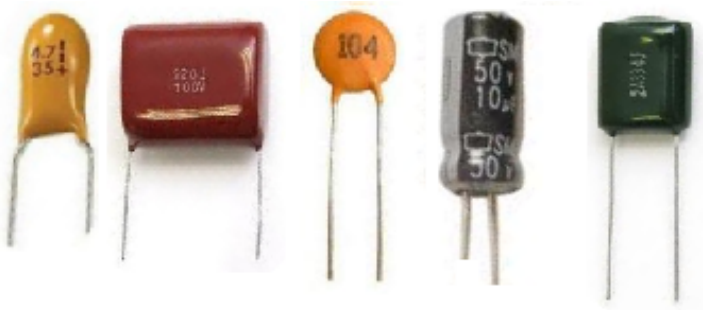
```
import RPi.GPIO as GPIO
import time

pin = 25
try:
   GPIO.setmode(GPIO.BCM)
   GPIO.setup(pin, GPIO.OUT)
   p = GPIO.PWM(pin,100)
   p.start(5)
   while True:
      p.ChangeDutyCycle(5) #-90degree
      time.sleep(1)
      p.ChangeDutyCycle(15) #0 dgree
      time.sleep(2)
      p.ChangeDutyCycle(25) #+90 degree
      time.sleep(1)
      p.ChangeDutyCycle(15) #0 dgree
      time.sleep(2)
except KeyboardInterrput:
   p.stop()
   GPIO.cleanup()
```

세부목차

1. Introduction
2. OS(Rasbian) Installation
3. Development Environment
4. GPIO
5. Digital Output
6. Digital Input
7. Analog Output - PWM
8. **Analog Input - RC Circuit**
9. 고수준 Sensor Modules

Analog Input - RC Circuit

- **Capacitor를 이용한 Analog Input**
  - 디지털 입력 핀으로 아날로그 흉내
  - RC회로(Resistor * Capacitor)
  - Capacitor (콘덴서)
    - 교류는 통과, 직류는 차단
    - 축전지
    - 교류 전압 충전/방전
    - 시 상수 (Time Constant)
      - 전압의 63.2%를 충전하는데 걸리는 시간(초)
      - 반복 : 나머지 전압의 63.2% 충전
      - TC = R x C

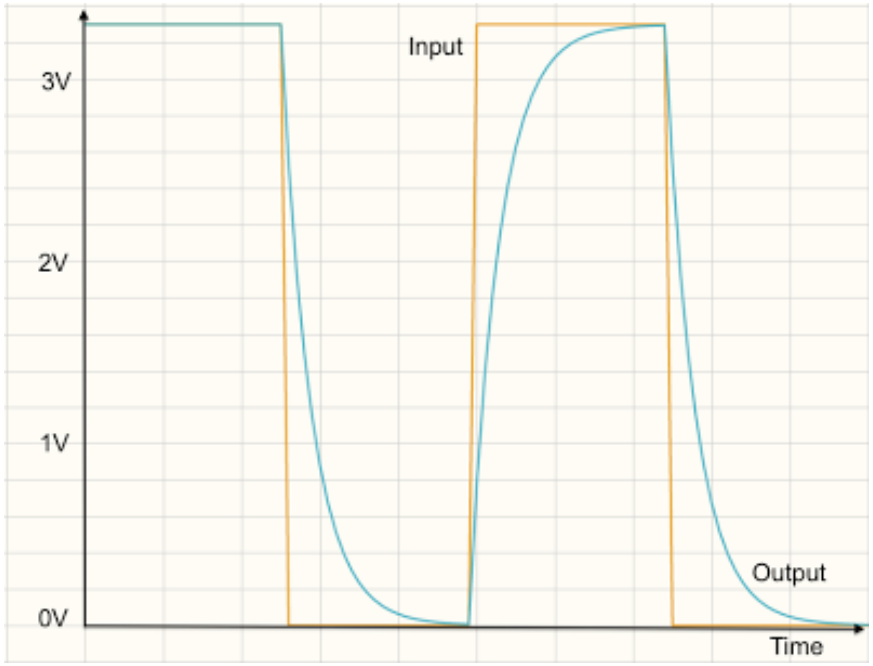


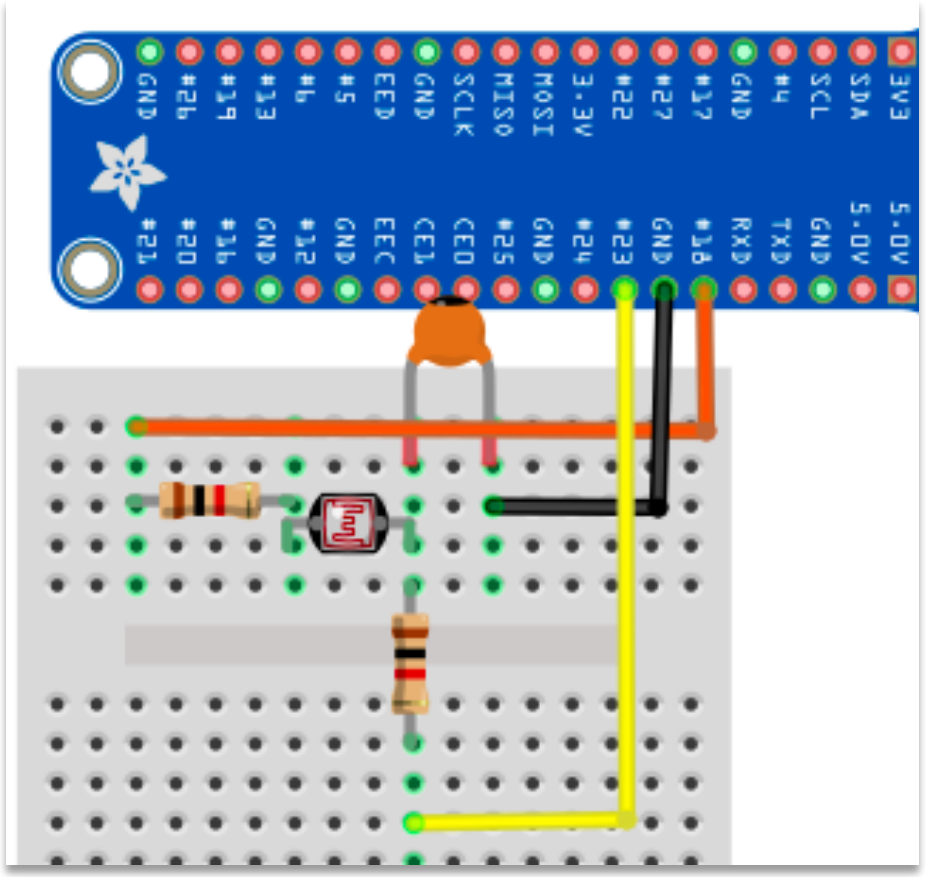
Symbol

Analog Input - RC Circuit

❖ Capacitor를 이용한 Analog Input

- 저항의 크기에 따라 충전 시간 변화
- 캐패시터의 충전 시간 측정
  - 가변저항 값 유추
- 고정저항 + 가변저항
  - 1KΩ + LDR





Analog Input - RC Circuit

❖ LDR Analog Input

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
a_pin = 18
b_pin = 23

GPIO.setmode(GPIO.BCM)

def charge_time():
    GPIO.setup(b_pin, GPIO.IN)
    GPIO.setup(a_pin, GPIO.OUT)
    count = 0
    GPIO.output(a_pin, True)
    while not GPIO.input(b_pin):
        count = count + 1
    return count
```

```
def analog_read():
    GPIO.setup(a_pin, GPIO.IN)
    GPIO.setup(b_pin, GPIO.OUT)
    GPIO.output(b_pin, False)
    time.sleep(0.1)
    return charge_time()


while True:
    print  analog_read()
    #time.sleep(1)
```

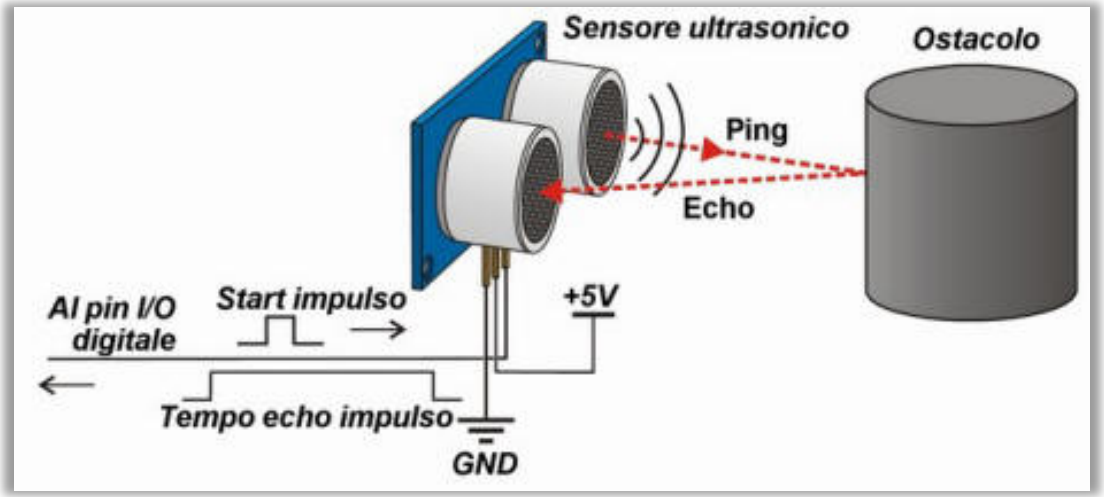
세부목차

1. Introduction
2. OS(Rasbian) Installation
3. Development Environment
4. GPIO
5. Digital Output
6. Digital Input
7. Analog Output - PWM
8. Analog Input - RC Circuit
9. **고수준 Sensor Modules**

고수준 Sensor Modules

❖ 초음파 거리 센서

- HC SR-04
  - Vcc : 5v
  - Trigger : GPIO 24 , 초음파 발생
  - Echo : GPIO 23, 반사되는 음파 인식
  - GND : Ground
- 초음파를 발생시켜서 반사되는 시간으로 거리를 계산
  - Trigger pin에 10us 동안 HIGH
    - 8번의 40hz 초음파 펄스 발생
  - Echo pin Low상태
    - 펄스 발생 중
  - Echo pin HIGH 상태
    - 펄스 발생 종료
  - Echo pin LOW 상태
    - 반사 음을 수신
  - 소요 시간
    - HIGH 상태인 동안의 시간



고수준 Sensor Modules

❖ 초음파 거리 센서 Source

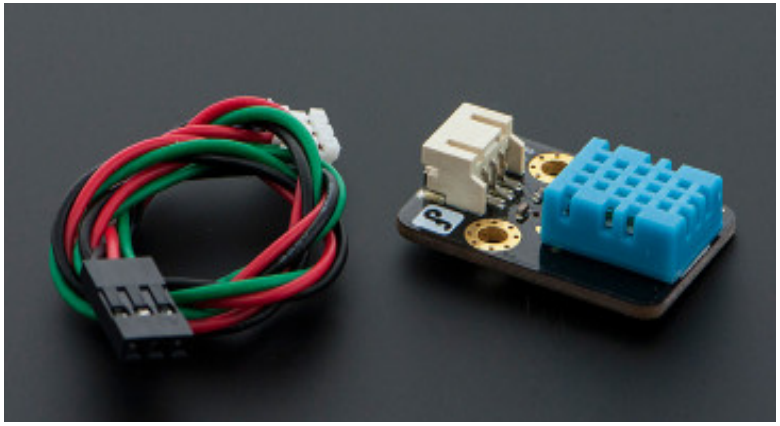
- 소요시간으로 거리 계산
- 음속 : 340m/s, 34000cm/s
- 34000 = distance / time
  - 34000 = distance / (time/2)
    - 왕복 시간
  - 17000 = distance / time
  - 17000 * time = distance

$$Speed = \frac{Distance}{Time}$$

```
trig_pin = 24
echo_pin = 23
try:
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(trig_pin, GPIO.OUT)
    GPIO.setup(echo_pin, GPIO.IN)
    while True:
      GPIO.output(trig_pin, False)
      print "ready for mesurement."
      time.sleep(0.2)
      GPIO.output(trig_pin, True)
      time.sleep(0.00001) #set HIGH for 10us
      GPIO.output(trig_pin, False)
      while GPIO.input(echo_pin) == 0:
        start_time = time.time()
      while GPIO.input(echo_pin) == 1:
        end_time = time.time()
      travel_time = end_time - start_time;
      distance = travel_time * 17150 #32300/2
      distance = round(distance, 2)
      print 'Distance:%dcm'%distance
  finally:
    GPIO.cleanup()
```

고수준 Sensor Modules

❖ DHT-11 온도/습도 모듈

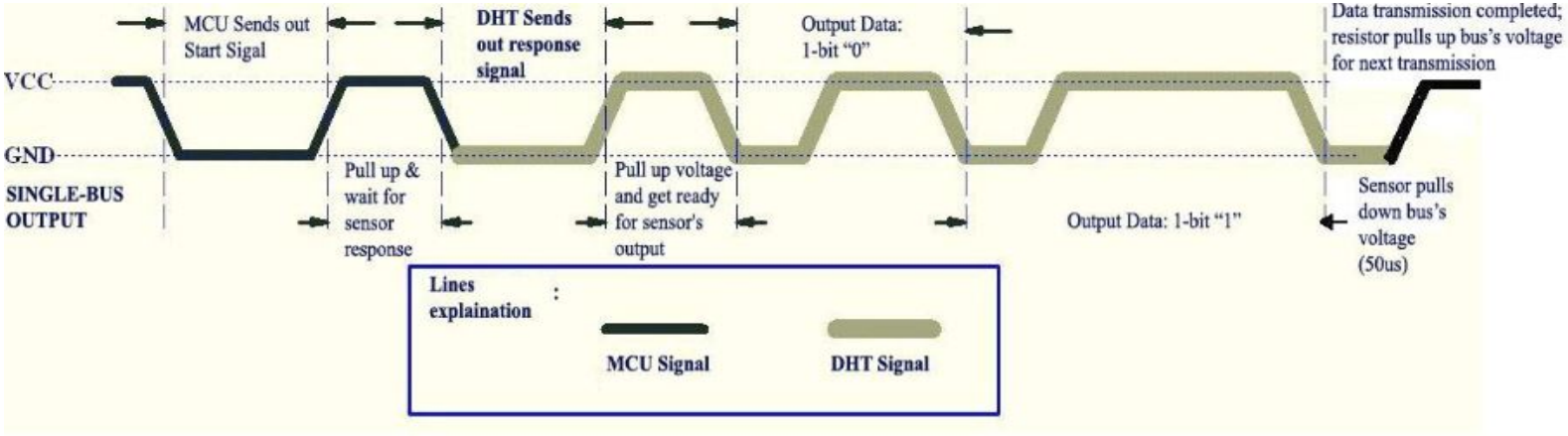


- 온도/습도 센서를 모듈로 구성
- 3핀
  - Red : Vcc (3~5V)
  - Green : Data Out
  - Black : GND
- 1Wire 통신
- 주어진 통신 체계에 마추어 신호 전달
  - MCU 요청신호가 전달되면 동작
  - HIGH 레벨 지속 시간에 따라 0, 1 구분
  - 40bit 데이타(습도: 16bit, 온도: 16bit, CheckSum: 8bit
  - Real-time 이 아니면 통신 실패 확율 높음
  - Rpi.GPIO로 구현한 것은 잦은 실패
  - http://www.uugear.com/portfolio/dht11-humidity-temperature-sensor-module/

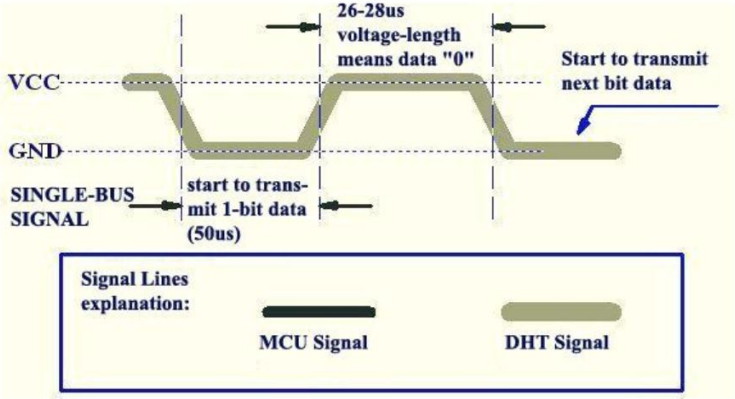
고수준 Sensor Modules

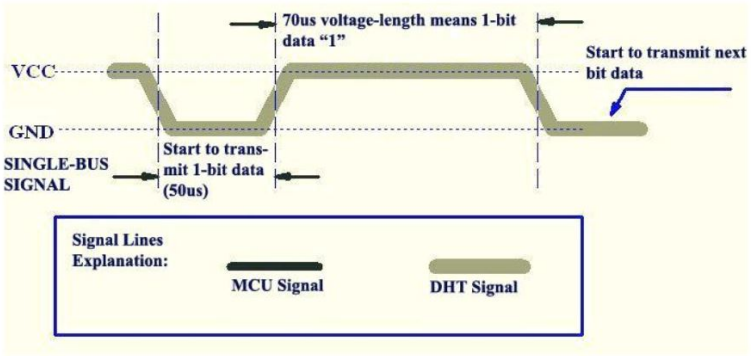
❖ DHT-11 온도/습도 모듈

- 요청 신호
  - HIGH→LOW → PULL_UP



- 0 :26~28us HIGH, 1: 70us HIGH





고수준 Sensor Modules

- **DHT-11 온도/습도 모듈**
  - 40bit 전송 데이타
    - 16bit 습도 (예: 652 → 65.2%)
      - 8bit : 습도 상위 비트 (0000 0010)
      - 8bit : 습도 하위 비트 (1000 1100)
    - 16bit 온도 (예: 351 → 35.1º)
      - 8bit : 온도 상위 비트 (0000 0001)
      - 8bit : 온도 하위 비트 (0101 1111)
    - 8bit : Check Sum (예: 1110 1110)

      0000 0010
      + 1000 1100
      + 0000 0001
      + 0101 1111

고수준 Sensor Modules

❖ DHT-11 Source

- Rpi.GPIO로 구현
  - https://github.com/netikras/r-pi_DHT11/blob/master/dht11.py

```
import RPi.GPIO as GPIO
import time
import sys

def bin2dec(string_num):
    return str(int(string_num, 2))

data = []
effectiveData = []
bits_min=999;
bits_max=0;
HumidityBit = ""
TemperatureBit = ""
crc = ""
crc_OK = False;
Humidity = 0
Temperature = 0
pin=4

GPIO.setmode(GPIO.BCM)
```

```
def pullData():
  global data
   global effectiveData
   global pin

   data = []
   effectiveData = []

   GPIO.setup(pin,GPIO.OUT)
   GPIO.output(pin,GPIO.HIGH)
   time.sleep(0.025)
   GPIO.output(pin,GPIO.LOW)
   time.sleep(0.14)

   GPIO.setup(pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

   for i in range(0,1000):
      data.append(GPIO.input(pin))
```

고수준 Sensor Modules

❖ DHT-11 Source

- Rpi.GPIO로 구현 〈계속〉

```
def analyzeData():
    seek=0;
    bits_min=9999;
    bits_max=0;

    global HumidityBit
    global TemperatureBit
    global crc
    global Humidity
    global Temperature

    HumidityBit = ""
    TemperatureBit = ""
    crc = ""

    while(seek < len(data) and data[seek] == 0):
        seek+=1;

    while(seek < len(data) and data[seek] == 1):
        seek+=1;
```

```
for i in range(0, 40):

    buffer = "";

    while(seek < len(data) and data[seek] == 0):
        seek+=1;

    while(seek < len(data) and data[seek] == 1):
        seek+=1;
        buffer += "1";

    if (len(buffer) < bits_min):
        bits_min = len(buffer)

    if (len(buffer) > bits_max):
        bits_max = len(buffer)

    effectiveData.append(buffer);
```

고수준 Sensor Modules

❖ DHT-11 Source

- Rpi.GPIO로 구현 〈계속〉

```
for i in range(0, len(effectiveData)):
    if (len(effectiveData[i]) < ((bits_max + bits_min)/2)):
        effectiveData[i] = "0";
    else:
        effectiveData[i] = "1";
 for i in range(0, 8):
    HumidityBit += str(effectiveData[i]);

 for i in range(16, 24):
    TemperatureBit += str(effectiveData[i]);


 for i in range(32, 40):
    crc += str(effectiveData[i]);

 Humidity = bin2dec(HumidityBit)
 Temperature = bin2dec(TemperatureBit)
```

```
def isDataValid():

   global Humidity
   global Temperature
   global crc

   print "isDataValid(): H=%d, T=%d, crc=%d"% (int(Humidity),
int(Temperature), int(bin2dec(crc)))
   if int(Humidity) + int(Temperature) == int(bin2dec(crc)):
      return True;
   else:
      return False;
def printData():
  global Humidity
  global Temperature

  print "H: "+Humidity
  print "T: "+Temperature
```

고수준 Sensor Modules

❖ DHT-11 Source

- Rpi.GPIO로 구현 〈계속〉

```
while (not crc_OK):
   pullData();
   analyzeData();
   if (isDataValid()):
      crc_OK=True;
      print "\r",
      printData();
   else:
      sys.stderr.write(".")
      time.sleep(2);
```

고수준 Sensor Modules

❖ DHT-11 Source

- Adafruit DHT 모듈 활용
  - https://learn.adafruit.com/dht/overview

```
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
```

```
import Adafruit_DHT

sensor = Adafruit_DHT.DHT11
pin = 7

while True:
    humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)

    if humidity is not None and temperature is not None:
        print "Temp={0:0.1f}*C Humidity={1:0.1f}%".format(temperature, humidity)
    else:
        print "Failed to get reading."
```