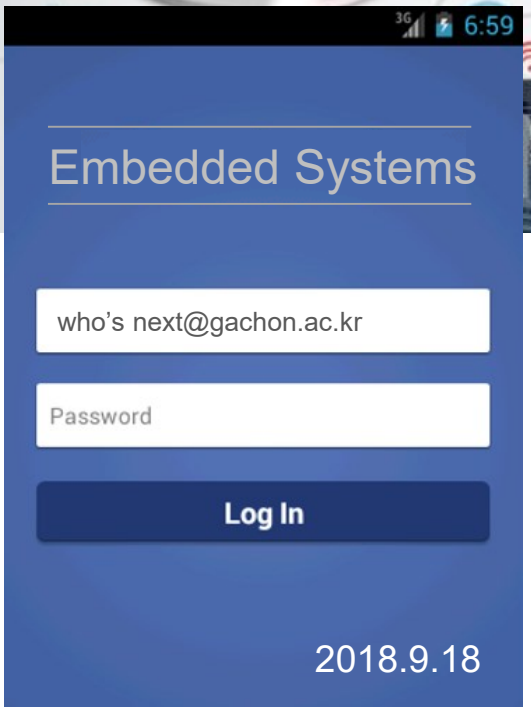


임베디드 시스템

컴퓨터공학과 이병문

강의일정

01	강의소개, 강의일정소개, 평가소개
02	사물인터넷, 라즈베리파이3 설치/구축
03	임베디드 액츄레이터/센서 제어 1
04	임베디드 액츄레이터/센서 제어 2
05	임베디드 액츄레이터/센서 제어 3
06	임베디드 액츄레이터/센서 제어 4
07	임베디드 액츄레이터/센서 제어 5
08	중간고사

- **임베디드 실습환경**
- **임베디드 액츄레이터/센서 제어**
- Node.js**기반 임베디드 제어**

실습준비 – 화요일 강좌듣는 학생들만 해당됨!!!

계정생성 하세요! (지난 시간에 생성했던 (계정이 삭제되었으므로), 다시 생성!)

1) **라즈베리파이3 부팅 & pi 계정으로 로그인!**

2) **일반사용자계정(본인 학번) 생성**

```
$ sudo   useradd   -m   -s   /bin/bash   학번1
$ sudo   passwd    학번1

$ sudo   useradd   -m   -s   /bin/bash   학번2
$ sudo   passwd    학번2
```

학번, 예) 20160000
암호, 예) gachon654321

3) **권한설정**

```
$ sudo  vi   /etc/sudoers

#
# This file MUST be ….
#
….
20160000   ALL=(ALL)   NOPASSWD:   ALL

$
```

편집과정에서
절대로 백스페이스
누르면 안됨!!!

예) 20160000

<ESC> :wq!

임베디드 실습환경

■ 하드웨어 구성

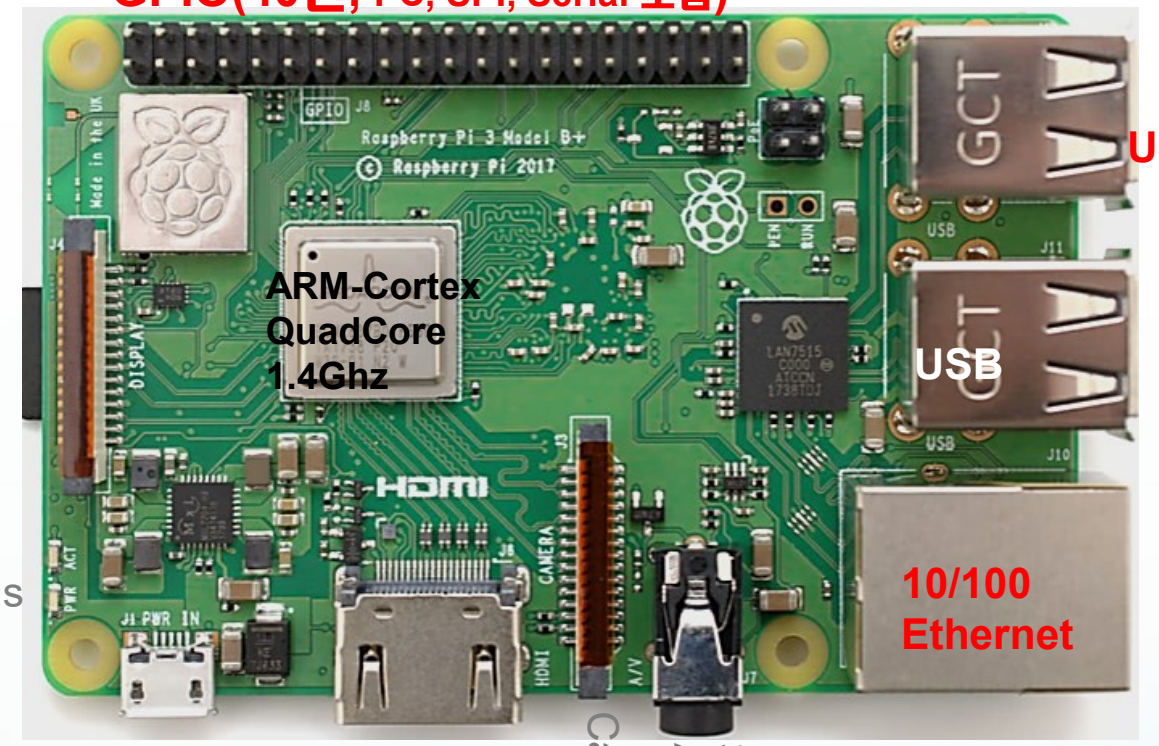
☑ **Raspberry Pi 3 Model B+**

(2017년 버전)

Broadcom **BCM2837B0**
(Cortex-A53 (ARMv8) 64-bit)

WiFi(802.11n/ac)
BLE 4.2

GPIO(40핀, I²C, SPI, Serial 포함)



Chip Antenna

전용디스플레이 (7인치)

DSI Connector

(뒤면) MicroSD Memory Slot

Status LED

USB

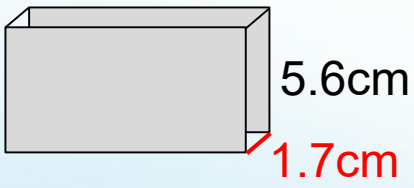
MicroUSB 5핀전원
(5V/2.5A)

HDMI

Camera
전용카메라

3.5mm Audio

RAM 1GB (**뒷면**)



임베디드 실습환경

■ 하드웨어 구성

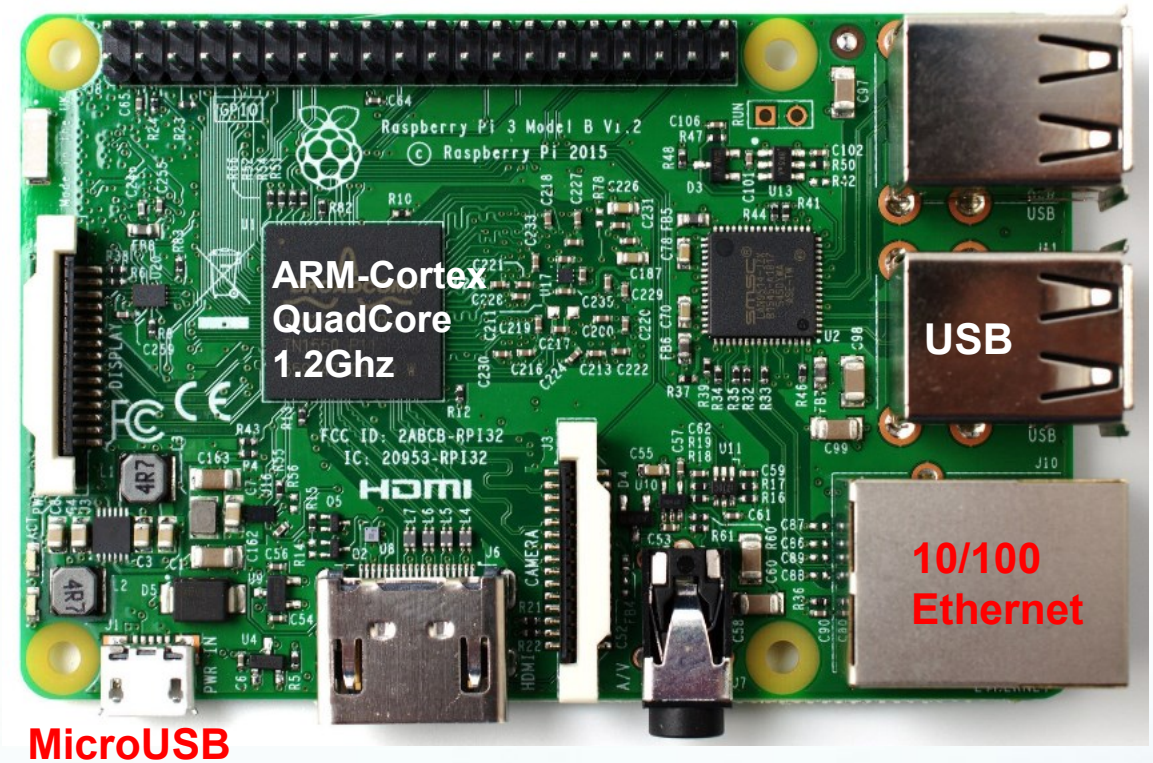
☑ **Raspberry Pi 3 Model B**

(2015년 버전)

Broadcom **BCM2837**
(Cortex-A53 (ARMv7) 64-bit)

WiFi(802.11n)
BLE 4.1

GPIO(40핀, I²C, SPI, Serial 포함)



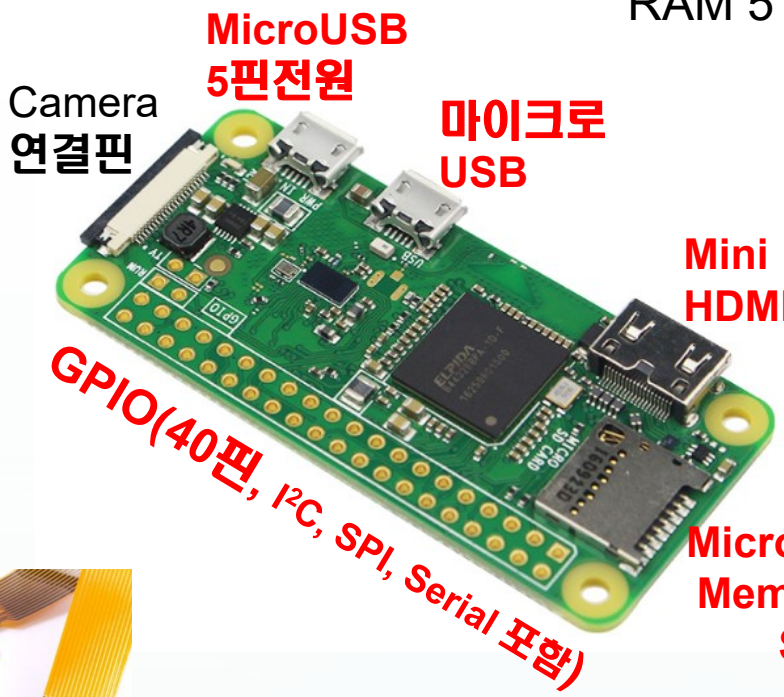
MicroUSB
5핀전원
(5V/2.5A)

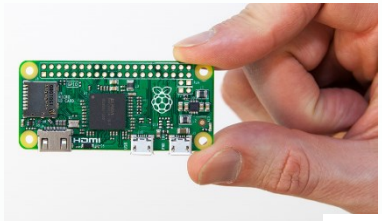
임베디드 실습환경

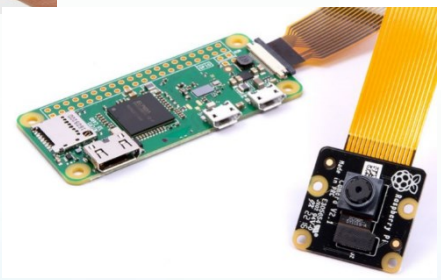
■ 하드웨어 구성

☑ **Raspberry Pi 0 W**

BCM2835
1Ghz CPU
RAM 512MB







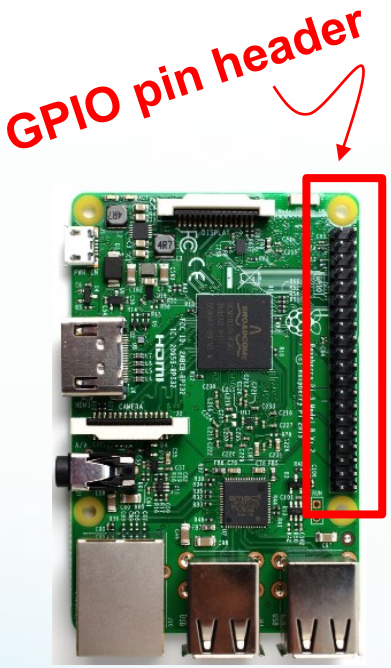
임베디드 실습환경

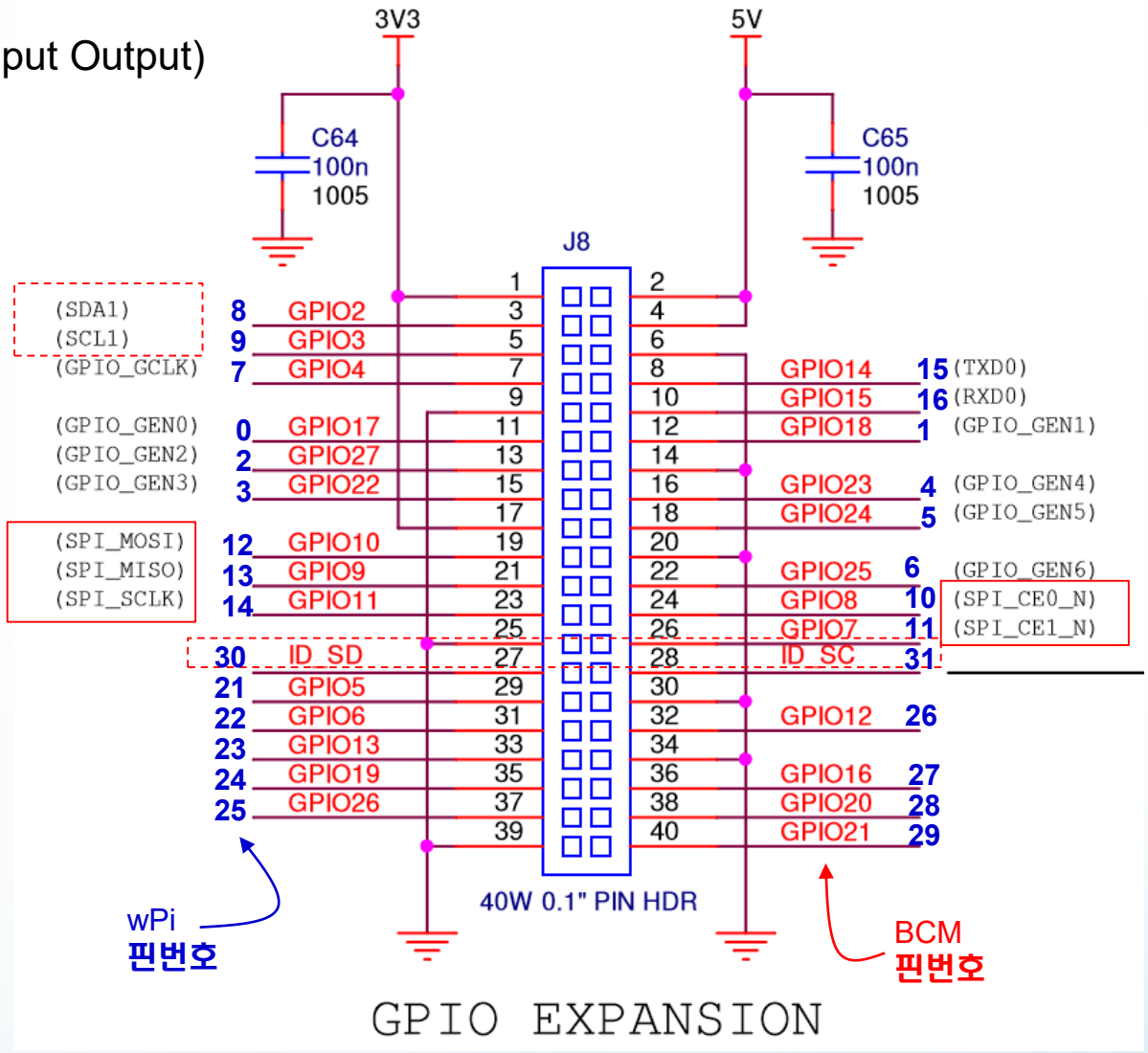
■ 라즈베리파이3 하드웨어 구성

☑ GPIO(General Purpose Input Output)

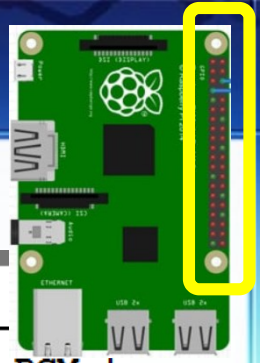
I²C

SPI





임베디드 실습환경

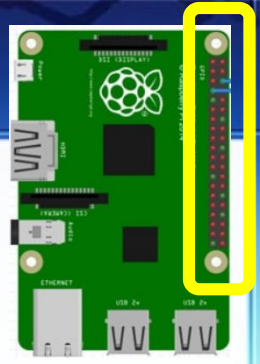


■ **GPIO Pin 구성**

$ gpio readall

```
pi@raspberrypi:~/mysrc $ gpio readall
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5v      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 0 | IN   | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | IN   | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 1 | 11 || 12 | 0 | OUT  | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 1 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | IN   | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | IN   | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |  OUT | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |  OUT | 1 | 35 || 36 | 0 | OUT  | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |  OUT | 0 | 37 || 38 | 0 | OUT  | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
```

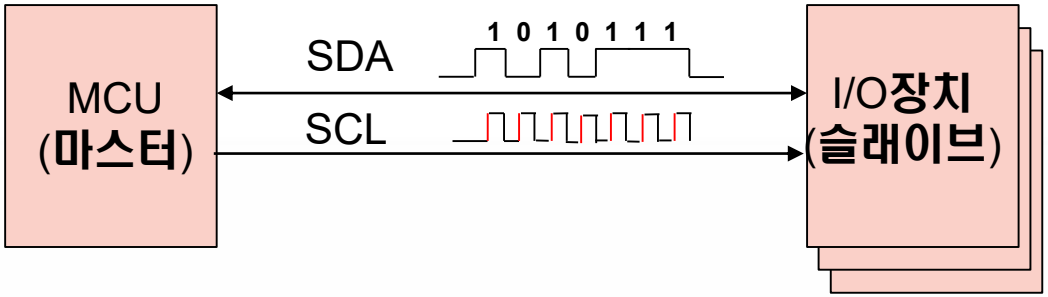
임베디드 실습환경



■ I²C, I2C 버스방식 (Inter-Integrated Circuit)

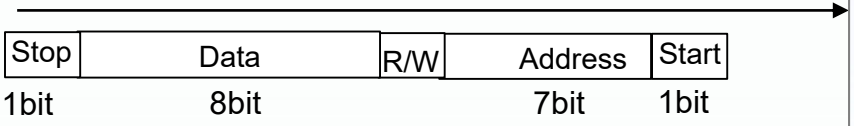
☑ MCU와 I/O 디바이스간의 양방향 시리얼(직렬)전송 버스

☑ 2개의 버스(SDA, SCL)를 이용하여 데이터 전송함 (half duplex 방식)



Serial DAta line
데이터 비트의 신호선
Serial Clock Line
동기용 클럭 신호선

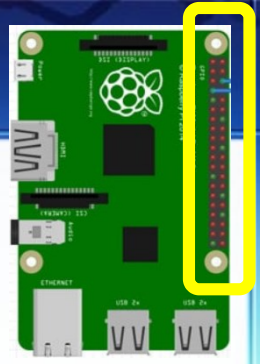
☑ SDA에서의 전송데이터 포맷



GPIO Pin layout (**실제시스템**, **$ gpi**

```
pi@raspberrypi:~/mysrc $ gpio readall
 +-----+-----+---------+------+---+---Pi
 | BCM | wPi |   Name  | Mode | V | Physi
 +-----+-----+---------+------+---+----+
 |     |     |    3.3v |      |   |  1 |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 |
 |     |     |      0v |      |   |  9 |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 | 28 | 1 | IN | SCL.0 | 31 | 1 |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 | 30 |   |    |   0v  |    |   |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 | 32 | 0 | IN | GPIO.26 | 26 | 12 |
```

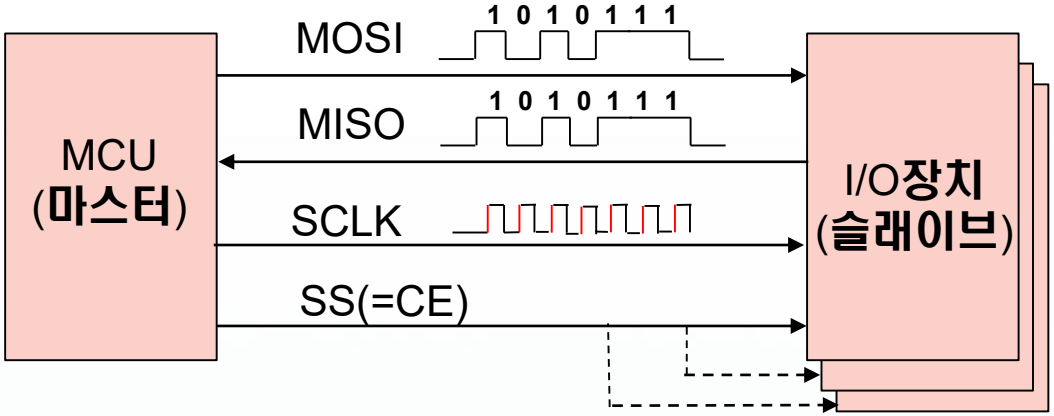
임베디드 실습환경



■ SPI (Serial Peripheral Interface) 버스방식

☑ MCU와 I/O 디바이스간의 양방향 시리얼(직렬)전송 버스

☑ 4개의 버스(SCLK, MOSI, MISO, SS)를 이용하여 데이터 전송함 (full duplex 방식)



Master Output Slave Input
Master Input Slave Output
Serial CLocK
Slave Select (=Chip Enable)

22	3	GPIO. 3	IN	1	15	16	0	IN	GPIO. 4	4	23
		3.3v			17	18	0	IN	GPIO. 5	5	24
10	12	MOSI	IN	0	19	20			0v		
9	13	MISO	IN	0	21	22	0	IN	GPIO. 6	6	25
11	14	SCLK	IN	0	23	24	1	IN	CE0	10	8
		0v			25	26	1	IN	CE1	11	7
0	30	SDA.0	IN	1	27	28	1	IN	SCL.0	31	1
5	21	GPIO.21	IN	1	29	30			0v		
6	22	GPIO.22	IN	1	31	32	0	IN	GPIO.26	26	12

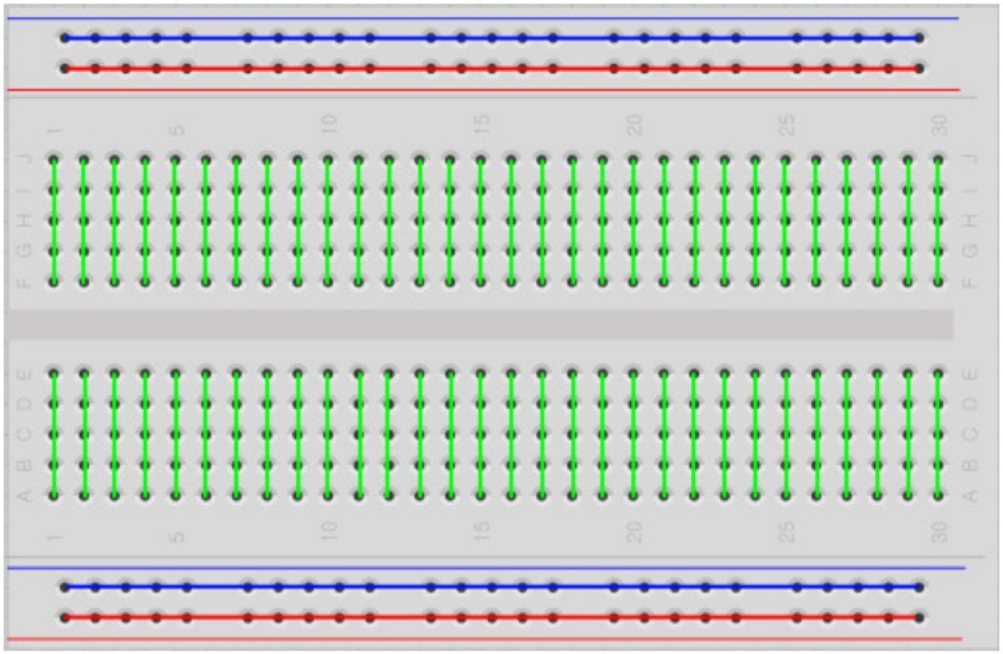
임베디드 실습환경

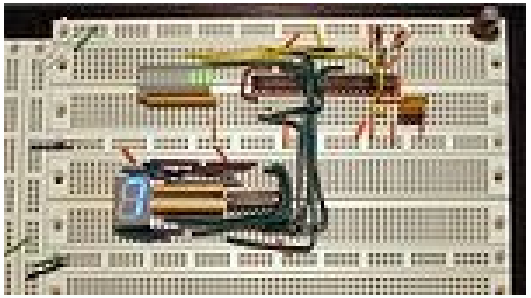
■ Breadboard

☑ 전자회로의 시제품(시작품)을 만드는 데 사용하는 무땜납 장치

☑ 천공 아래에 많은 납이 도금된 인청동 스프링 클립이 있는 플라스틱 천공 블록

버스
- 전원(+)을 연결하여 사용(3.3V, 5V 전원)
- 전원(- 또는 GND)을 연결하여 사용 (0v 전원)
- 부품의 핀을 꼽거나, 점퍼케이블을 끼워서 회로구성

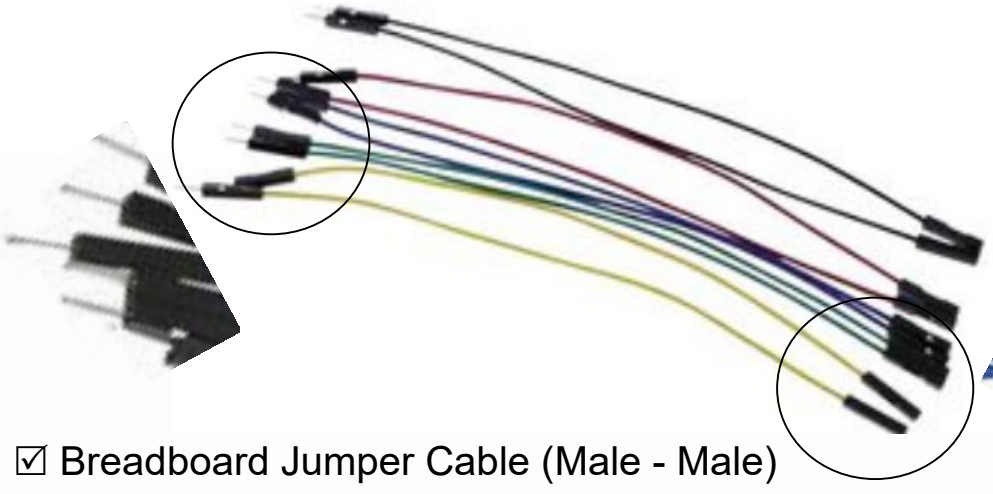




임베디드 실습환경

■ 점퍼케이블

☑ Breadboard Jumper Cable (Male - Female)



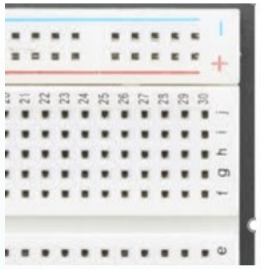
수-수
암-암
암-수

☑ Breadboard Jumper Cable (Male - Male)

☑ Breadboard Jumper Cable (Female – Female)

- www.sparkfun.com
- www.adafruit.com
- **네이버쇼핑**, **옥션**,11**번가 쇼핑몰**

RaspberryPi



Bread board

임베디드 액츄레이터/센서 제어

■ LED 하드웨어 구성

☑ LED (2 Color)

☑ LED (3 Color)

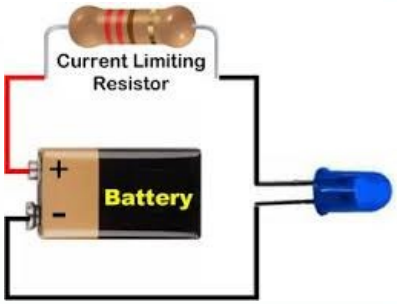
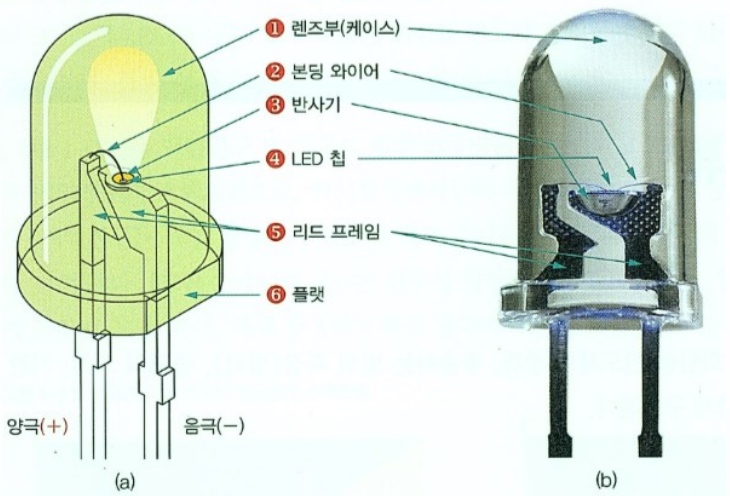
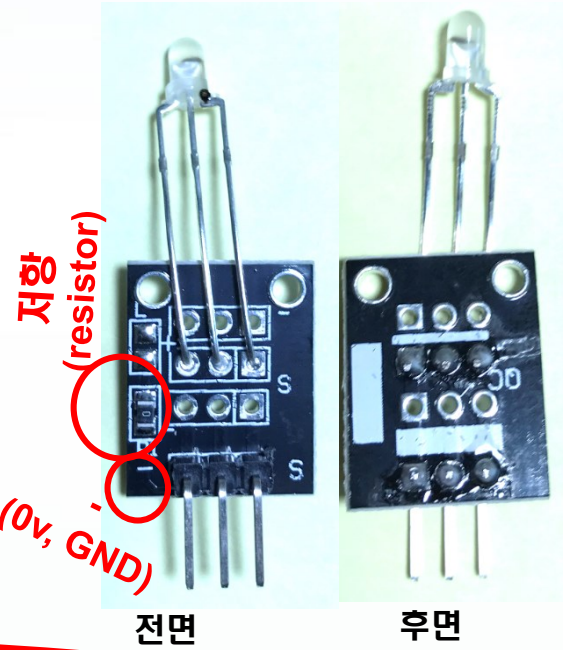


그림 1-13 램프형 LED





램프형 LED 모듈

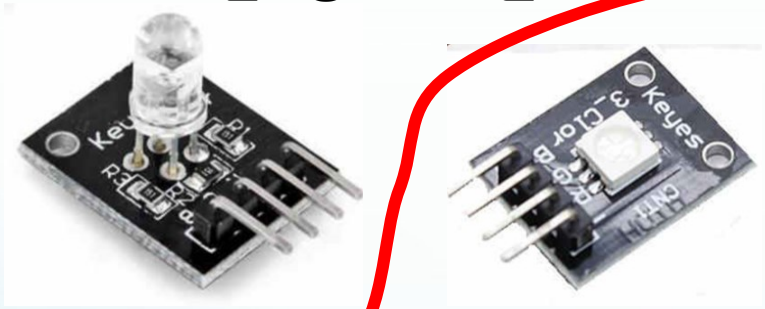
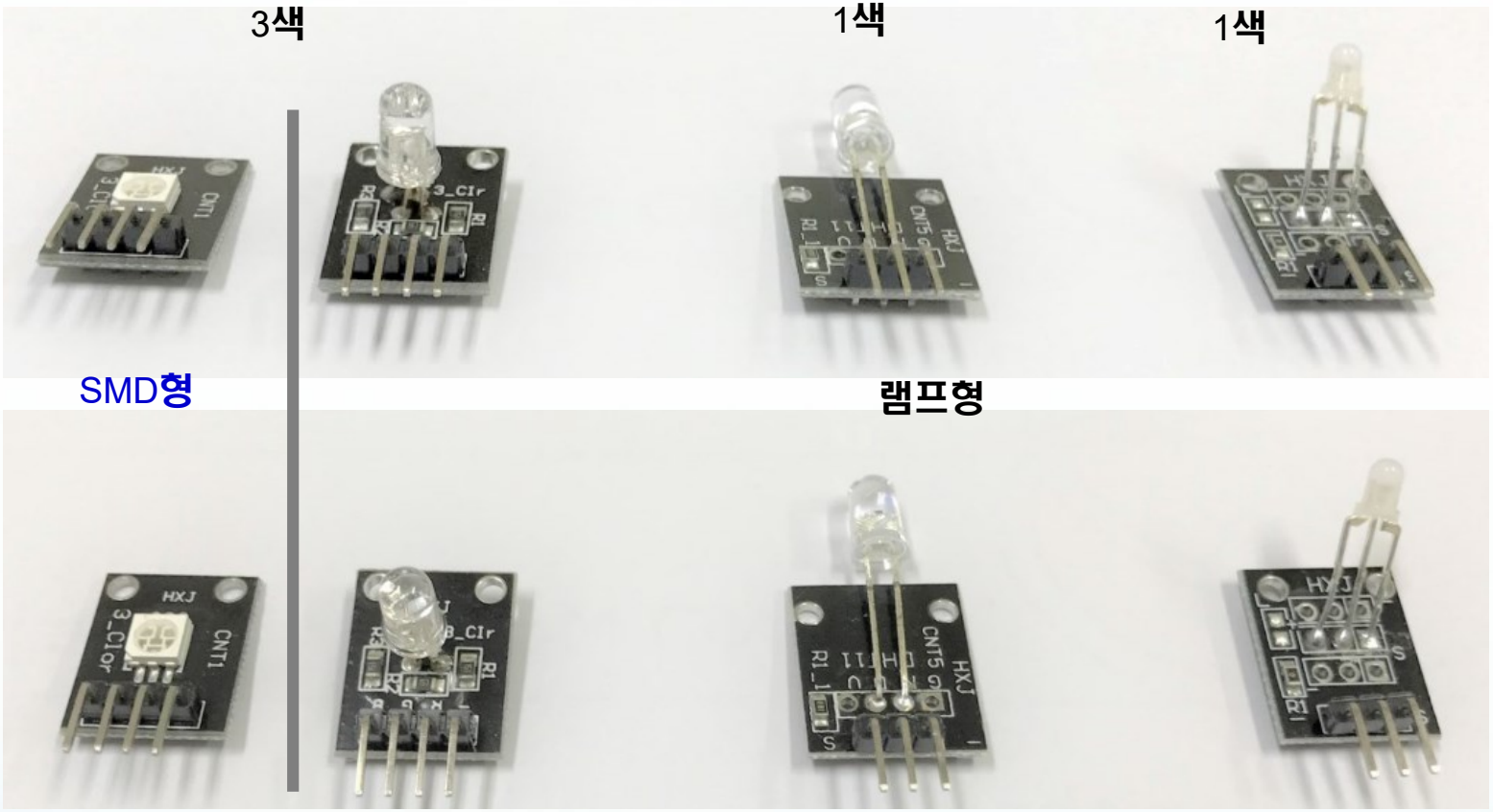


그림 1-19 'SMD 5050'을 이용해 만든 조명 제품

SMD형 LED 모듈

임베디드 액츄레이터/센서 제어

■ LED 하드웨어 구성

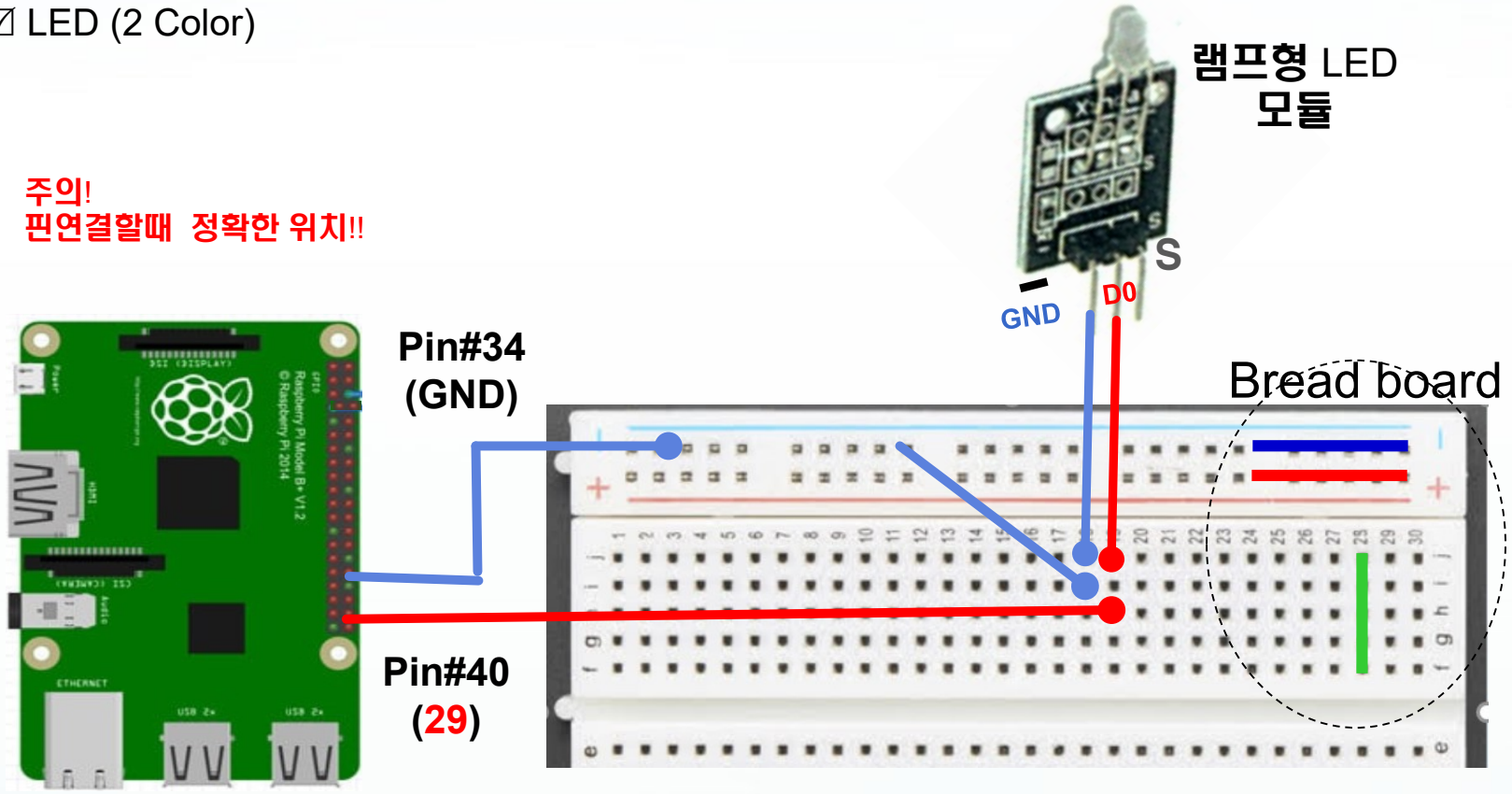


임베디드 액츄레이터/센서 제어

■ 하드웨어 구성

☑ LED (2 Color)

주의!
핀연결할때 정확한 위치!!



임베디드 액츄레이터/센서 제어

■ Example code

☑ LED 를 제어(On/Off)하는 C코드

wPi **핀번호가** 29

예제1 (led1.c)

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

#define    LEDPIN     29
void main() {

     wiringPiSetup();
     pinMode(LEDPIN,  OUTPUT);
     printf("LED1 will be blink...\n^C to stop\n");

     while (1) {
        digitalWrite(LEDPIN, 1); delay(200);      // 1 = on
        digitalWrite(LEDPIN, 0); delay(200);      // 0 = off
     }
}
```

```
pi@raspberrypi: ~/mysrc/work
pi@raspberrypi:~/mysrc/work $ sudo ./led1
LED1 will be blink...
^C to stop
^Cpi@raspberrypi:~/mysrc/work $
```

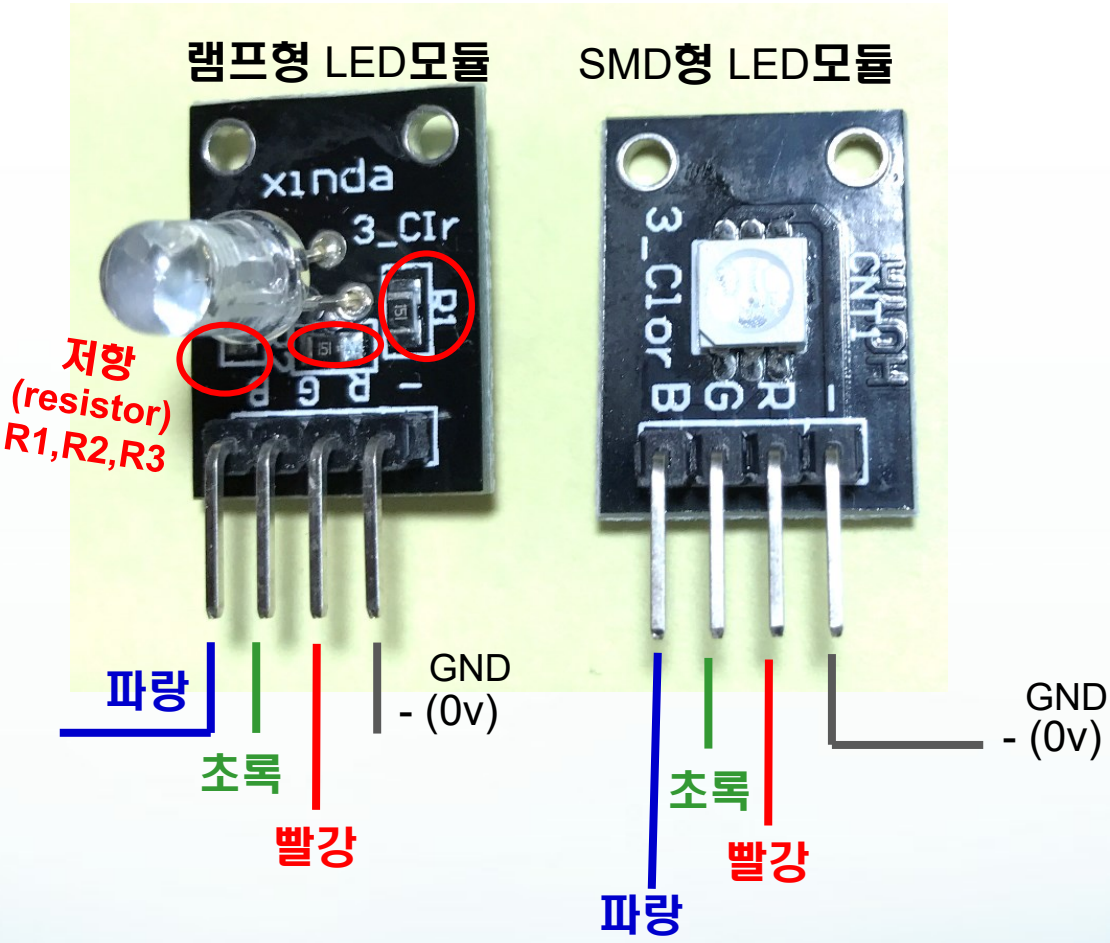
```
$ vi  led1.c
$ cc  -o  led1   led1.c   -lwiringPi
$ sudo  ./led1
```

wiringPi **라이브러리** API

임베디드 액츄레이터/센서 제어

■ 하드웨어 구성

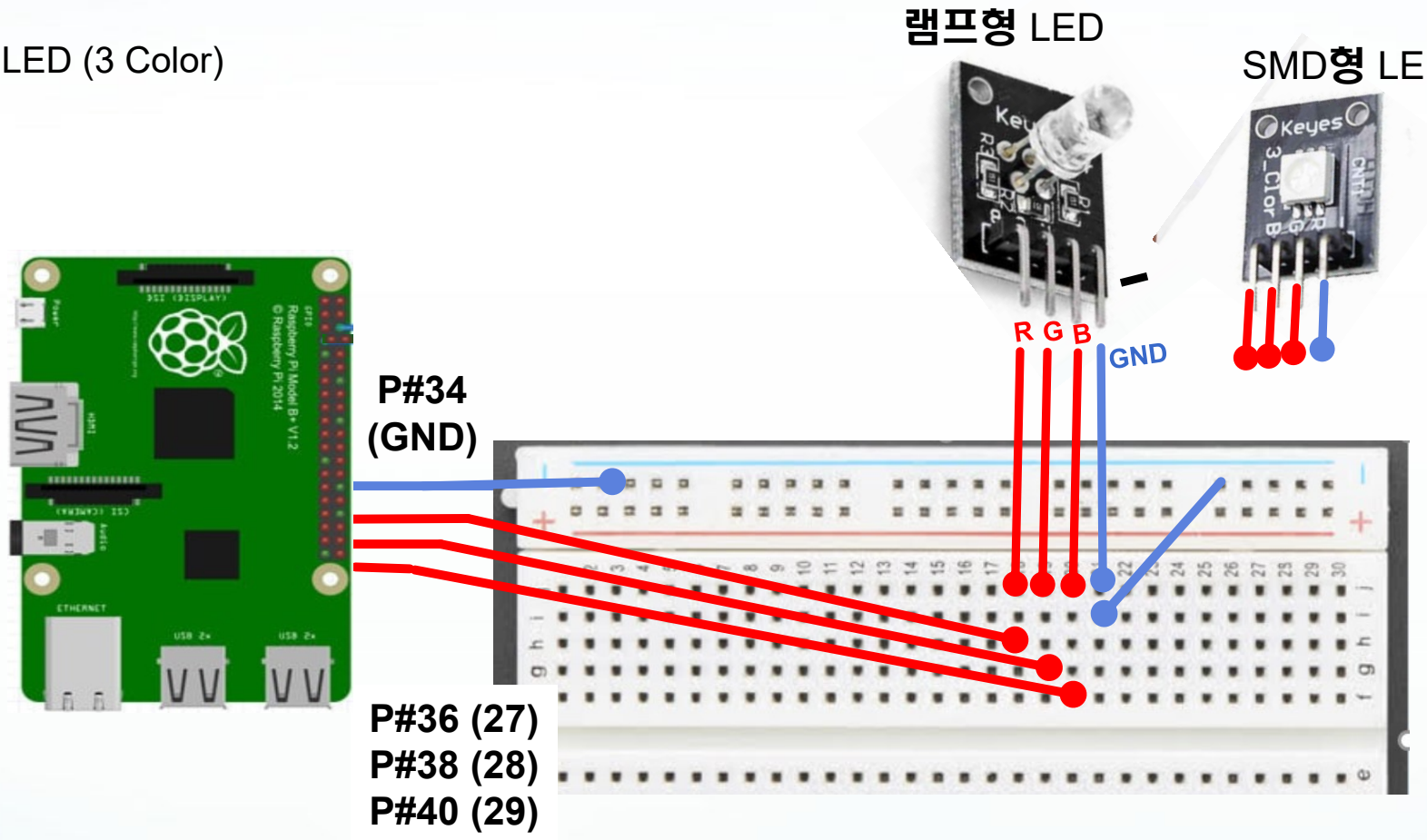
☑ LED (3 Color)



임베디드 액츄레이터/센서 제어

■ 하드웨어 구성

☑ LED (3 Color)



임베디드 액츄레이터/센서 제어

■ Example code

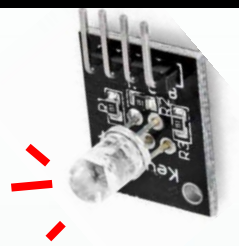
☑ LED(3 Color) 를 제어(On/Off)하는 C코드

예제2 (led3.c)

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

#define   LED31    27    /* blue */
#define   LED32    28    /* green */
#define   LED33    29     /* red */

main() {

  wiringPiSetup();
  pinMode (LED31, OUTPUT);
  pinMode(LED32, OUTPUT);
  pinMode LED33, OUTPUT);
  printf("LED2 will be blink (red, gree, blue)\n^C to stop\n");
  while (1) {
    digitalWrite(LED31, 1); delay(400);
    digitalWrite(LED31, 0);
    digitalWrite (LED32, 1); delay(400);
    digitalWrite (LED32, 0);
    digitalWrite (LED33, 1); delay(400);
    digitalWrite (LED33, 0);
  }
}
```

```
$ vi  led3.c
$ cc  -o  led3   led3.c   -lwiringPi
$ sudo  ./led3
```

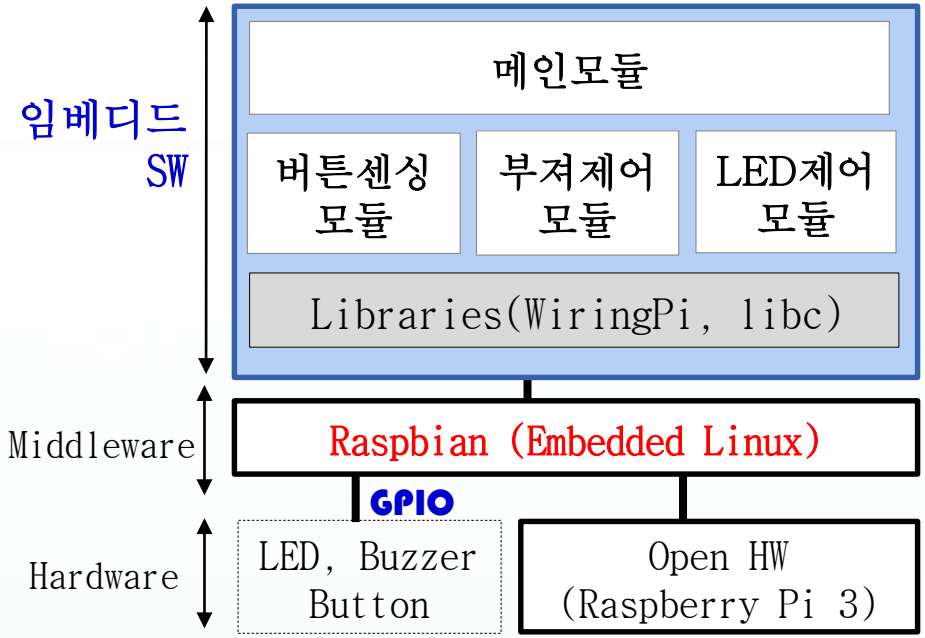
```
pi@raspberrypi: ~/mysrc/work
pi@raspberrypi:~/mysrc/work $ sudo ./led3
LED3 will be blink (red, gree, blue)
^C to stop
```



임베디드 액츄레이터/센서 제어

■ C 프로그램을 이용한 임베디드 SW구조

☑ LED(3 Color) 를 제어(On/Off)하는 C코드



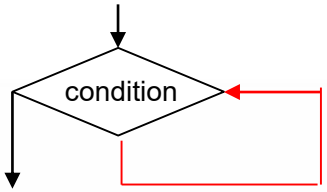
Node.js 기반 임베디드 제어

■ (Node.js 기반) 프로그래밍은 왜 ?

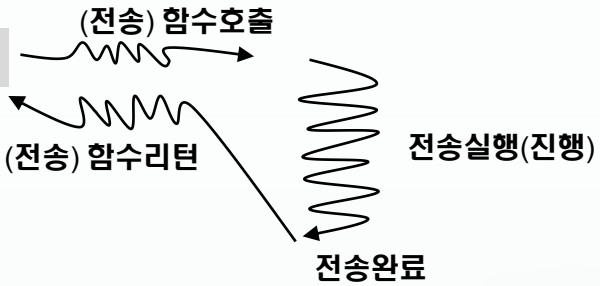
☑ Event driven service (≠ Polling service)

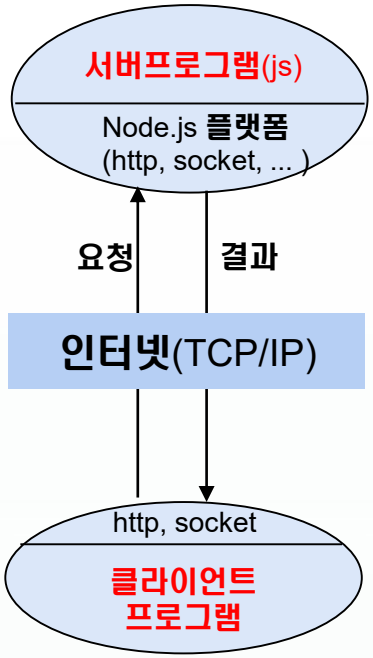
☑ Asynchronous I/O service (= Non-blocking I/O ≠ Blocking I/O)

```
While (1) {
  if (condition == 1)
      break;
}
```




```
send_all_data(data);
```





☑ 장점: 성능, 프로그램용이, High speed network application 에 적합

☑ 안정화된 플랫폼, 네트워크 프로토콜 지원 -> V8 크롬엔진

Node.js 기반 임베디드 제어

■ (Node.js 기반) 프로그래밍은 왜 ?

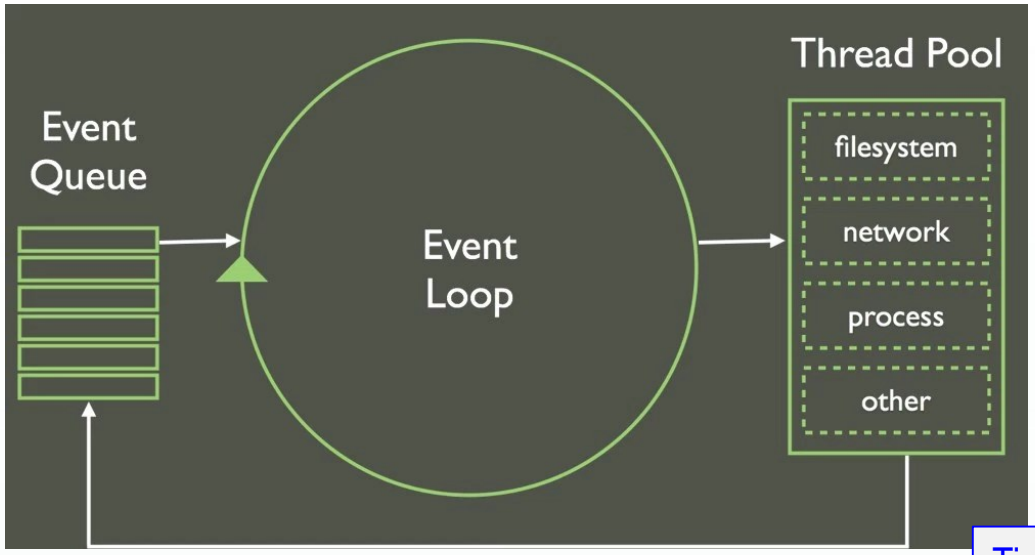
☑ 다양한 플랫폼에서 프로그램을 개발할 수 있음
(서버플랫폼, 모바일플랫폼, 임베디드플랫폼)

Node.js 기반 임베디드 제어

■ Node.js 의 Event loop

☑ Event loop vs Polling

☑ Callback function, Register callback function, Handling callback function




```
Timer_handler( ) {
   console.log("1초가 경과했음!");
}

SetTimeout(Timer_handler, 1000);
```

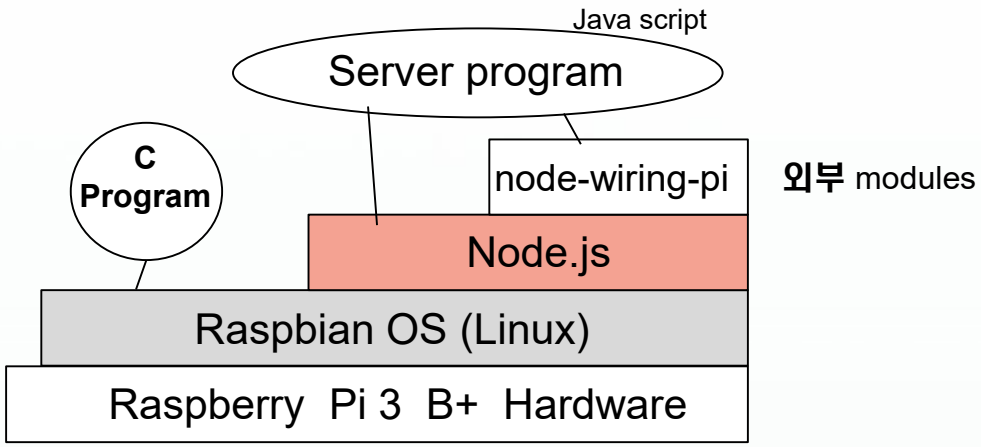
Node.js 기반 임베디드 제어

■ 실습환경(개발환경) 구축

☑ 플랫폼SW 구축 (Raspberry Pi 3 B+, Raspbian OS)

☑ Node.js 설치 (Node.js 패키지설치) + npm(노드용 SW관리자, module 설치용tool)

☑ node-wiring-pi 설치



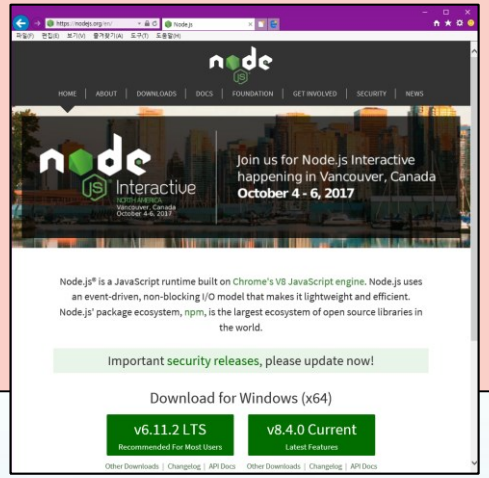
Node.js 기반 임베디드 제어

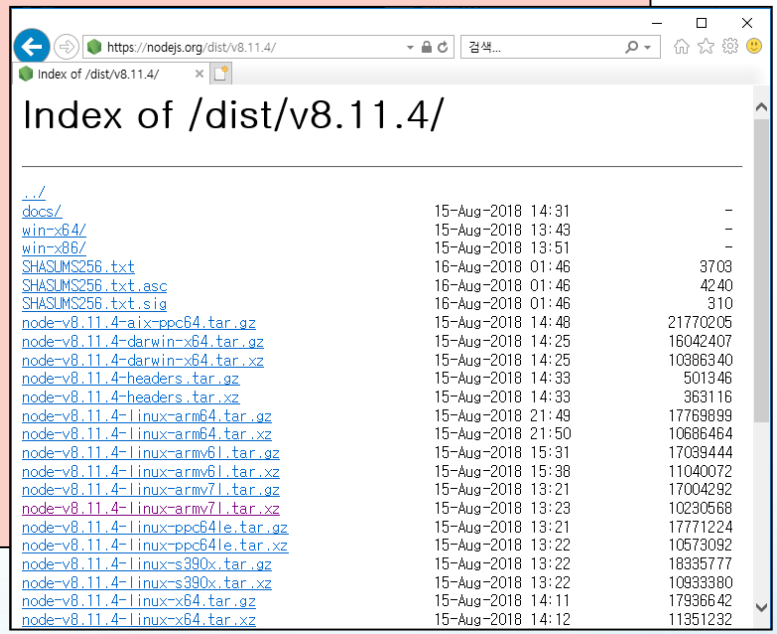
■ Node.js 플랫폼 설치

☑ Node.js 플랫폼(node.js) 설치

팀에서 1명만 할 것!

```
$ mkdir  download
$ cd  download
$ wget  https://nodejs.org/dist/v8.11.4/node-v8.11.4-linux-armv7l.tar.xz
$ tar  -xvf  node-v8.11.4-linux-armv7l.tar.xz
$ cd  node-v8.11.4-linux-armv7l
$ sudo  cp  -R  *  /usr
$ cd ../..

$ node  -v
V8.11.4
$ npm  -v
5.6.0
$ gpio  -v
gpio version: 2.46
```



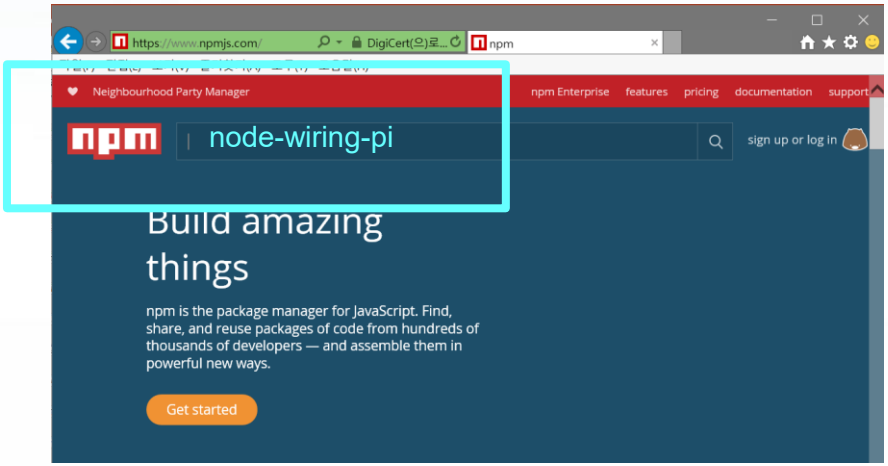


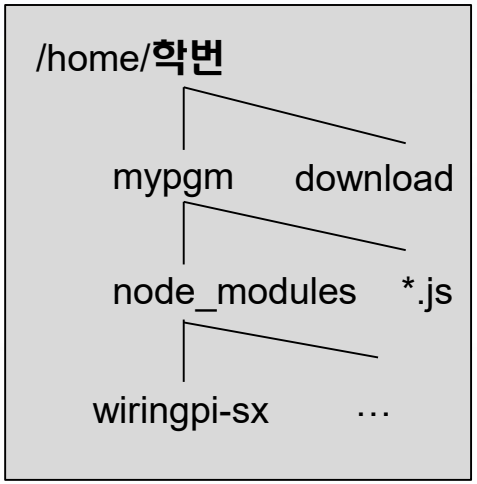
www.nodejs.org

Node.js 기반 임베디드 제어

- Node모듈 설치
- ☑ 외부모듈검색 및 설치

www.npmjs.com






```
$ mkdir  mypgm
$ cd  mypgm
$ su
암 호: *
# npm   init
# npm   install   node-wiring-pi
# chown  -R  학번계정명.학번계정그룹명   *
# exit
$
```

root **암호**

엔터키를 계속 눌러서, 기본값 설정

wiringpi-sx **외부모듈 설치**

예) chmod -R 20160000.20160000 *

Node.js 기반 임베디드 제어

■ Example code (node.js 소스코드)

☑ LED 를 제어(On/Off)하는 코드

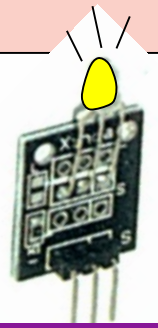
단, 미리 하드웨어를 구성하여야 함!

```
const   gpio = require('node-wiring-pi');
const   LEDPIN  = 29;
var      count = 0;

const   TimeOutHandler = function ( ) {
        if  (count > 0)  {
              gpio.digitalWrite(LEDPIN,  1);
              console.log("Node: LED on");
              count = 0;
         }
         else {
              gpio.digitalWrite(LEDPIN, 0);
              console.log("Node: LED off");
             count = 1;
        }
        setTimeout (TimeOutHandler, 1000);
}
gpio.setup('wpi');
gpio.pinMode(LEDPIN, gpio.OUTPUT);
setTimeout(TimeOutHandler, 1000);
```

예제3 (led1.js)

15페이지 슬라이드 참조

```
$ cd  mypgm
$ vi   led1.js
$ sudo  node  led1.js
```



```
20160000@raspberrypi: ~/mypgm
20160000@raspberrypi:~/mypgm $ sudo node led1.js
Node: LED off
Node: LED on
Node: LED off
Node: LED on
Node: LED off
```

실습미션

■ 실습일지

☑ 실습1

☑ 실습2

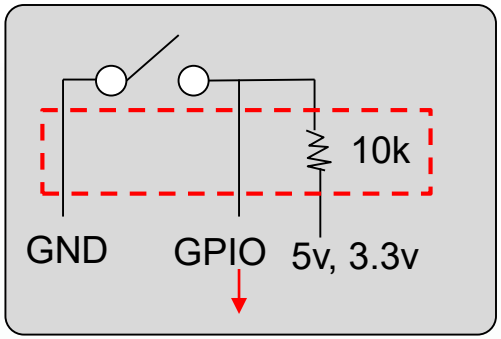
☑ 실습3

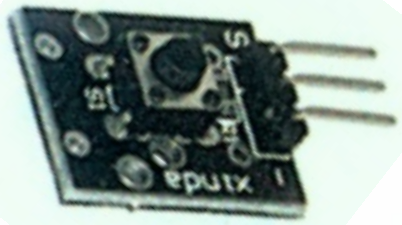
임베디드 센서(Button) 제어

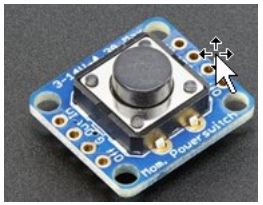
■ 하드웨어 구성

☑ Push Button sensor 모듈(= switch)

- 순간적으로 버튼의 누르는 순간의 상태값(높음 또는 낮음)을 측정하는 센서모듈



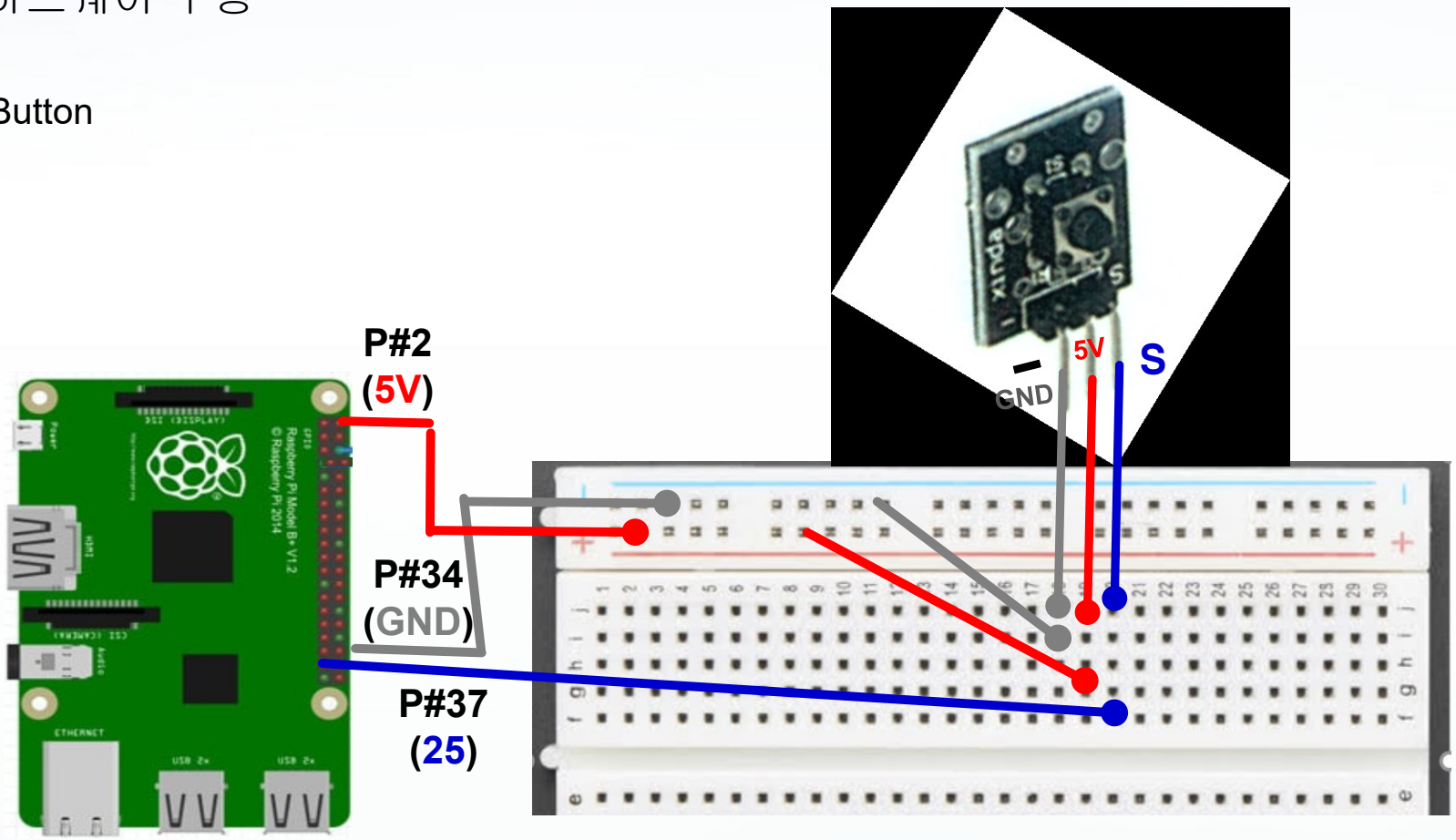




임베디드 센서(Button) 제어

■ 하드웨어 구성

☑ Button

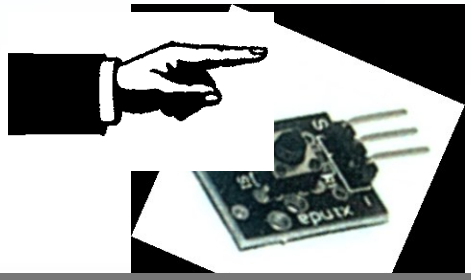


임베디드 센서(Button) 제어

■ Example code

☑ Button 값을 측정하는 C코드

예제4 (button.c)



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

#define    BUTTON_PIN    25    // wPi번호
void  main() {
   int  button_data = 0, ret = 0;

   wiringPiSetup();
   pinMode (BUTTON_PIN, INPUT);
   printf("Wait for press button …\n ^C to stop\n");

  while (1) {
      button_data = digitalRead(BUTTON_PIN);
      if  ( ! button_data ) {
            printf("Oh! Pressed!\n");  delay(500);
      }
  }
}
```

```
pi@raspberrypi: ~/mysrc/work
pi@raspberrypi:~/mysrc/work $ sudo ./button
 Wait for press button ...
 ^C to stop
Oh! Pressed!
Oh! Pressed!
Oh! Pressed!
Oh! Pressed!
Oh! Pressed!
```

```
$ vi   button.c
$ cc   -o   button   button.c   -lwiringPi
$ sudo  ./button
```

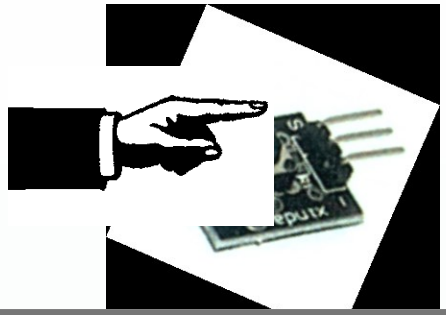
임베디드 센서(Button) 제어

■ Example code

☑ Button 값을 측정하는 JS 코드

예제5 (button.js)

```
const  gpio = require('node-wiring-pi');
const  BUTTON = 25;

const CheckButton = function() {
      let   data = gpio.digitalRead(BUTTON);
      if (!data)
            console.log("Nodejs: Button was pressed!");
      setTimeout(CheckButton, 300);
}

process.on('SIGINT', function() {
      console.log("Program Exit...");
      process.exit();
});

gpio.setup('wpi');
gpio.pinMode(BUTTON, gpio.INPUT);
setImmediate(CheckButton);
```



```
20160000@raspberrypi: ~/emapp
20160000@raspberrypi:~/emapp $ sudo node button.js
Nodejs: Button was pressed!
Nodejs: Button was pressed!
Nodejs: Button was pressed!
```

```
$ cd  mypgm
$ vi   button.js
$ sudo  node   button.js
```

실습미션

■ 실습일지

☑ 실습4

☑ **(심화문제1)** 실습일지를 끝낸 사람만, 아래 문제를 풀 것!!!

버튼을 클릭해도 잘 인식되지 못하는 경우를 발견할 수 있었을 것입니다.

이것을 해결하려면, (버튼클릭에 대한) 민감도를 조절하여야 합니다.

민감도 조절은 **timeout** 값으로 할 수 있으며, 가장 최적의 **timeout** 시간 값이 몇 **ms** 인지

수많은 시도를 통해서 찾아내기 바랍니다.

예) setTimeout(CheckButton, 300);

실습미션

■ 라즈비안(리눅스) 종료

☑ 실습을 종료하기 전에, 리눅스를 먼저 "시스템종료" 시켜야 합니다.

☑ 리눅스 "시스템종료" 방법 (아래 방법들 중에서 하나만 선택해서 하면 됩니다.)

```
$ sudo systemctl isolate runlevel0.target
```
또는

```
$ sudo systemctl isolate poweroff.target
```
또는

```
$ sudo shutdown -h now
```
또는

```
$ sudo init 0
```
또는

```
$ sudo telinit 0
```

강의 **Q&A**

실습장비 반납시 주의사항

1. **라즈베리파이3에 마이크로SD메모리가 있는지 확인!**
2. **37종 센서키트에 37개를 모두 넣었는지 확인!**
3. **LAN케이블 다시 빼서, 깔끔하게 정리하고**
4. **키트상자내에 모든 잡자재.부품/재료를 원래대로 넣고**
5. **(반납 담당자는) 아래 실습장비를 반드시 확인후에 제출하세요!**
   1) **라즈베리파이3 세트**
   2) **37종 센서키트**
   3) Breadboard, **점퍼케이블**
   4) LAN**케이블**