

IoT 프로그래밍

라즈베리파이 GPIO 제어

Kyusik Chung
kchung@ssu.ac.kr

목차



1. 실험목표
2. 실험환경
3. 라즈베리파이 GPIO 기초
4. 라이브러리 함수를 이용한 GPIO 간접 제어
 - **실습1:** 라즈베리파이 GPIO 출력제어 (LED)
 - **실습2:** 라즈베리파이 GPIO 입력제어 (스위치)
 - **실습3:** 라즈베리파이 GPIO 출력제어 (PWM)

1. 실습목표

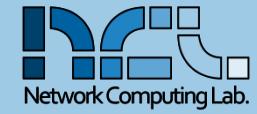


- 각종 센서, 모터, 입출력장치 등을 제어할 수 있는 GPIO pin 동작 및 제어 방법을 이해
- LED, 버튼 입력, PWM 제어하는 GPIO 프로그래밍 실습을 통해 라즈베리파이 GPIO 프로그래밍을 이해
- GPIO 제어 응용을 위한 기초학습

2. 실험 환경



- 라즈베리파이 보드 4B 사용
 - 라즈비안 설치
 - GCC 컴파일러 사용
 - C 코드 작성 및 테스트
- 라즈베리파이 보드를 서버모드로 접속
 - 우분투에서 ssh로 접속

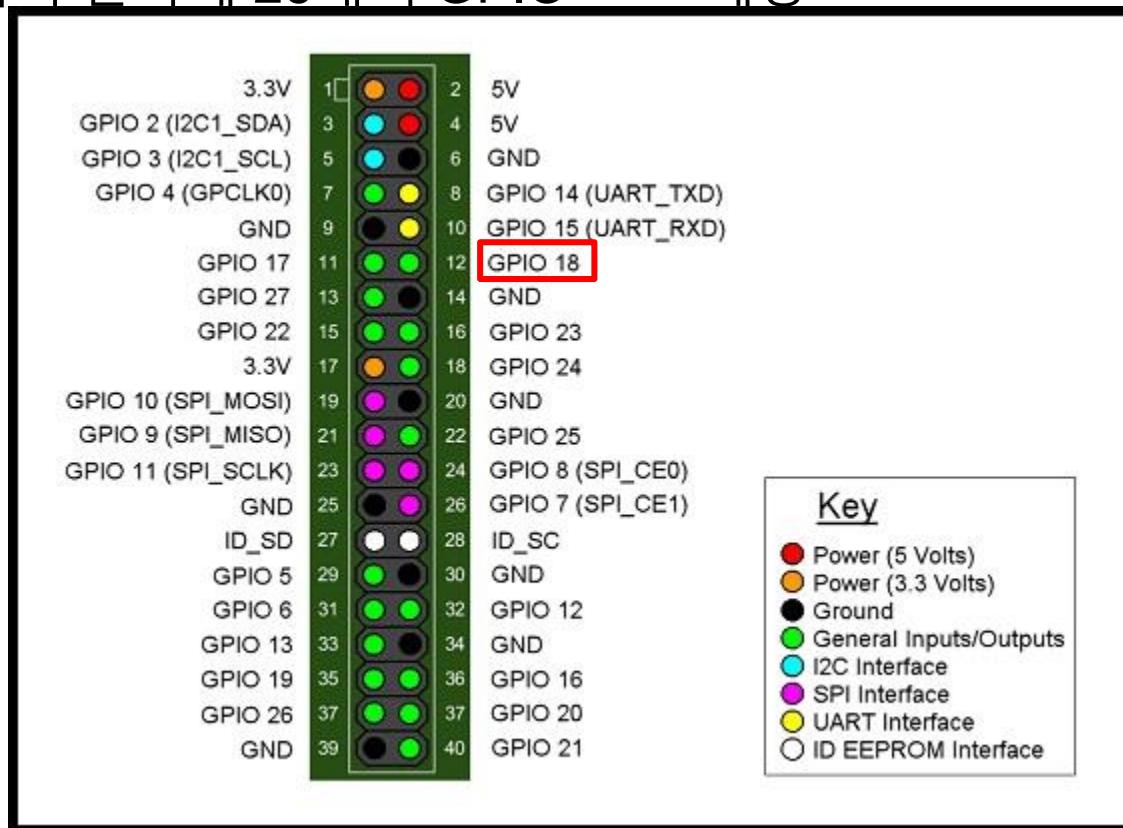


라즈베리파이 GPIO 기초

3. 라즈베리파이 GPIO 기초

GPIO (General Purpose Input / Output)

- 일반적인 입출력 포트 (3.3V 사용)
- 40개의 단자에 26개의 GPIO 포트 제공

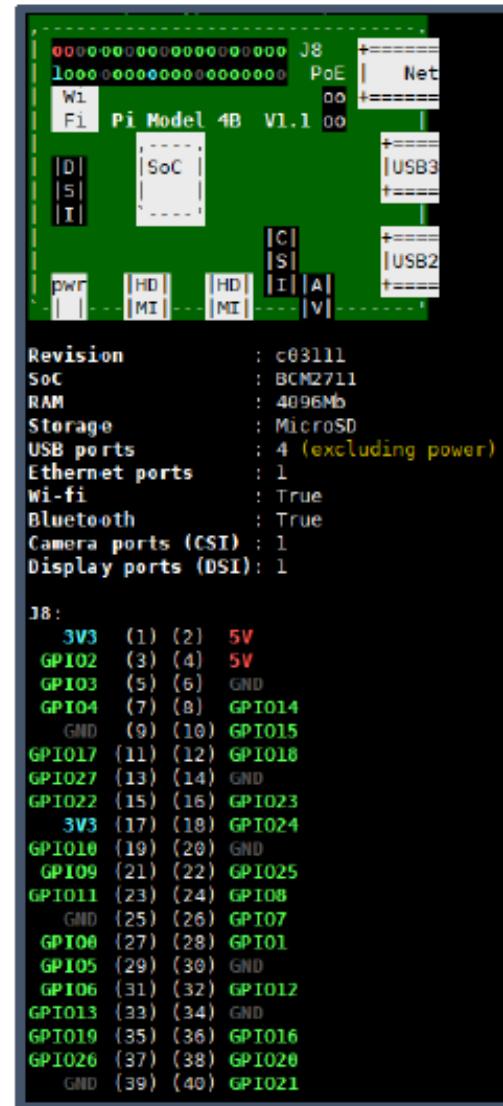


그림출처 : <http://www.hardcopyworld.com/ngine/arduino/wp-content/uploads/sites/3/2015/04/53bc258dc6c0425cb44870b50ab30621.jpg>

3. 라즈베리파이 GPIO 기초



- 핀 배열 확인
 - Pinout 명령
 - 라즈베리파이의 핀 배열 표시
 - <https://pinout.xyz/>
 - 핀에 관한 자세한 설명



3. 라즈베리파이 GPIO 기초

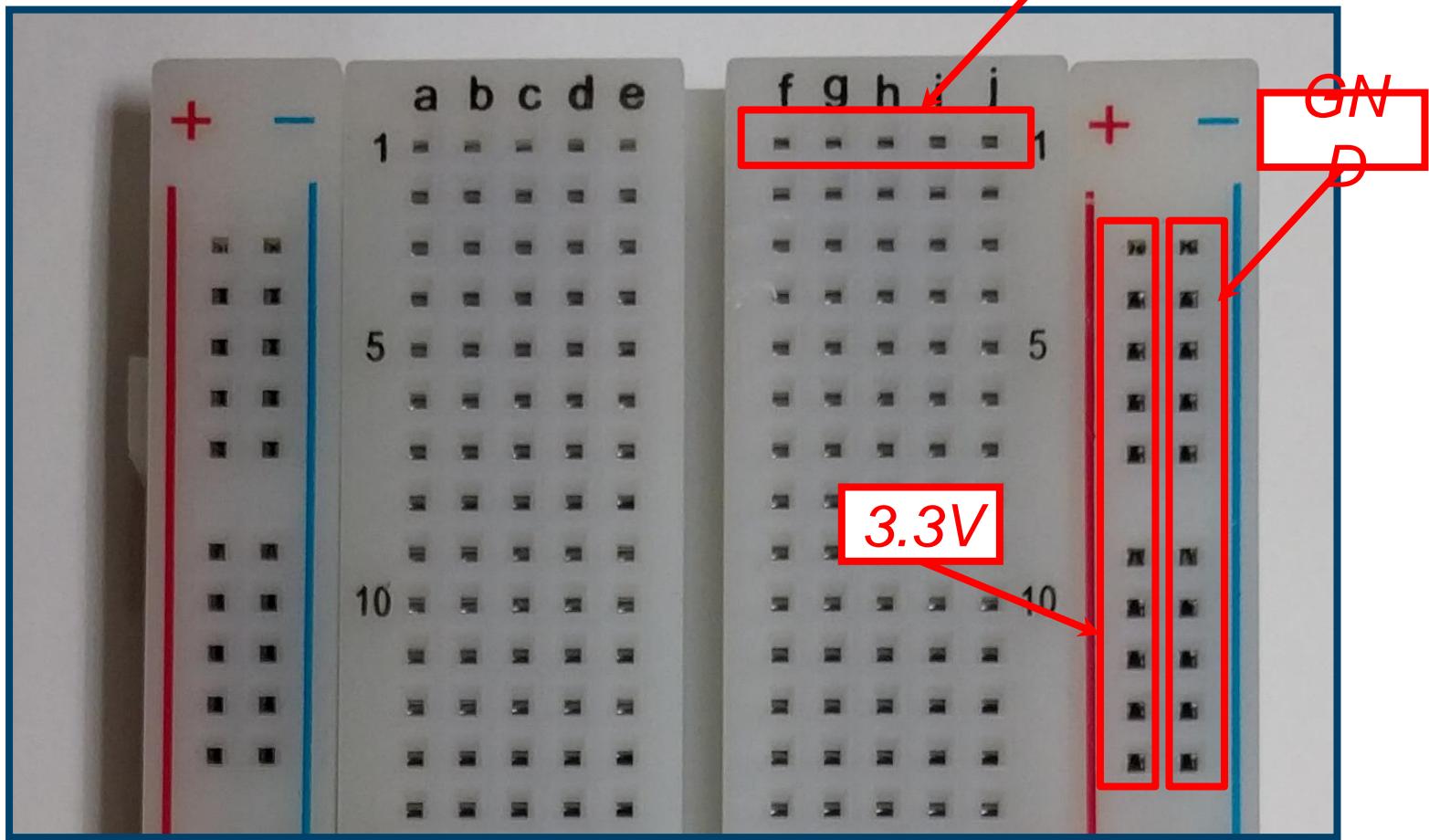
- 실습에 필요한 물품
 - 라즈베리파이 4 B
 - 브레드보드 (일명 빵판)
 - GPIO 커넥터
 - 전선
 - 저항
 - LED
 - 스위치
 - 스피커



3. 라즈베리파이 GPIO 기초



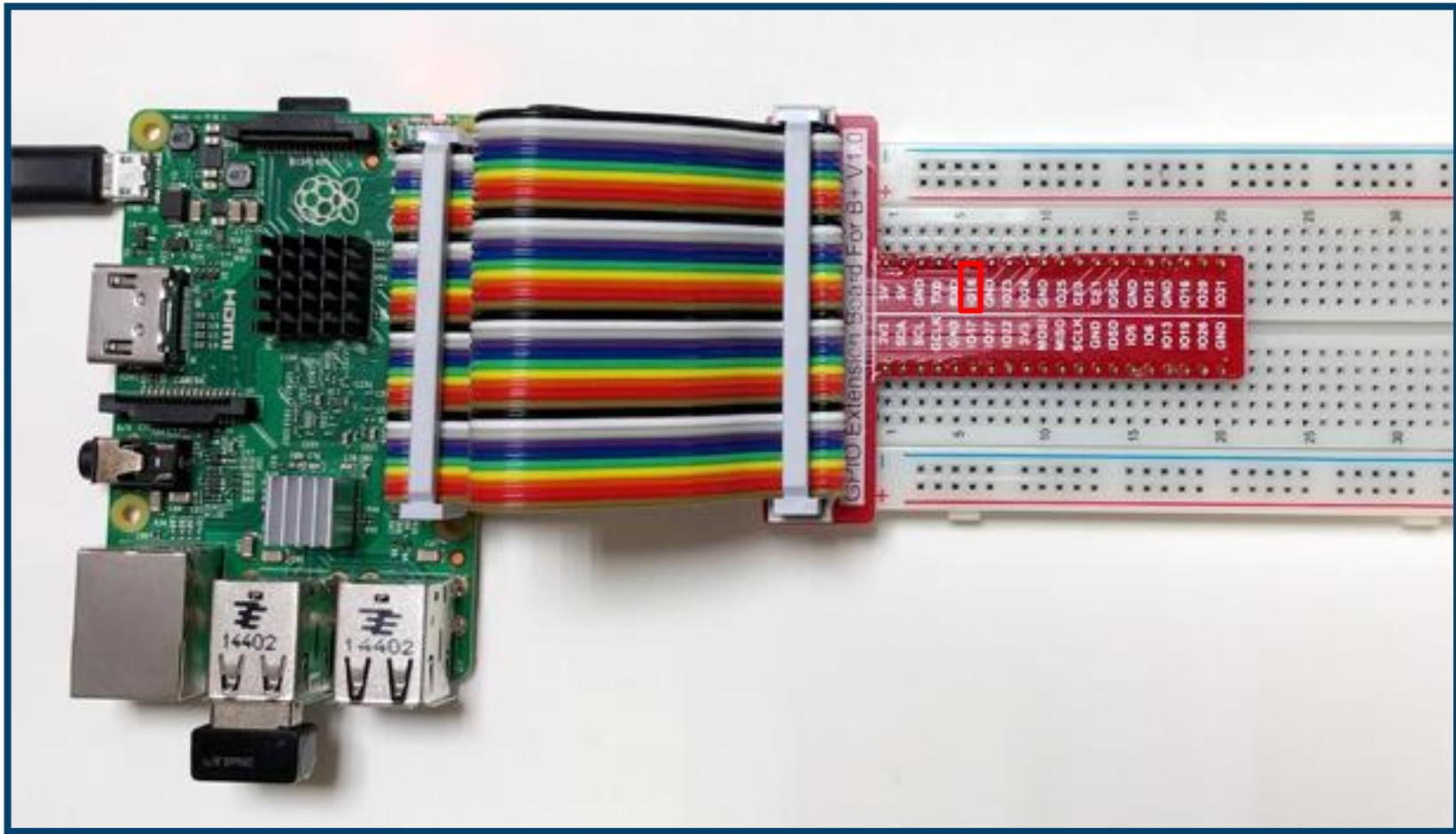
■ 브레드보드



3. 라즈베리파이 GPIO 기초

■ 라즈베리파이와 브레드보드 연결

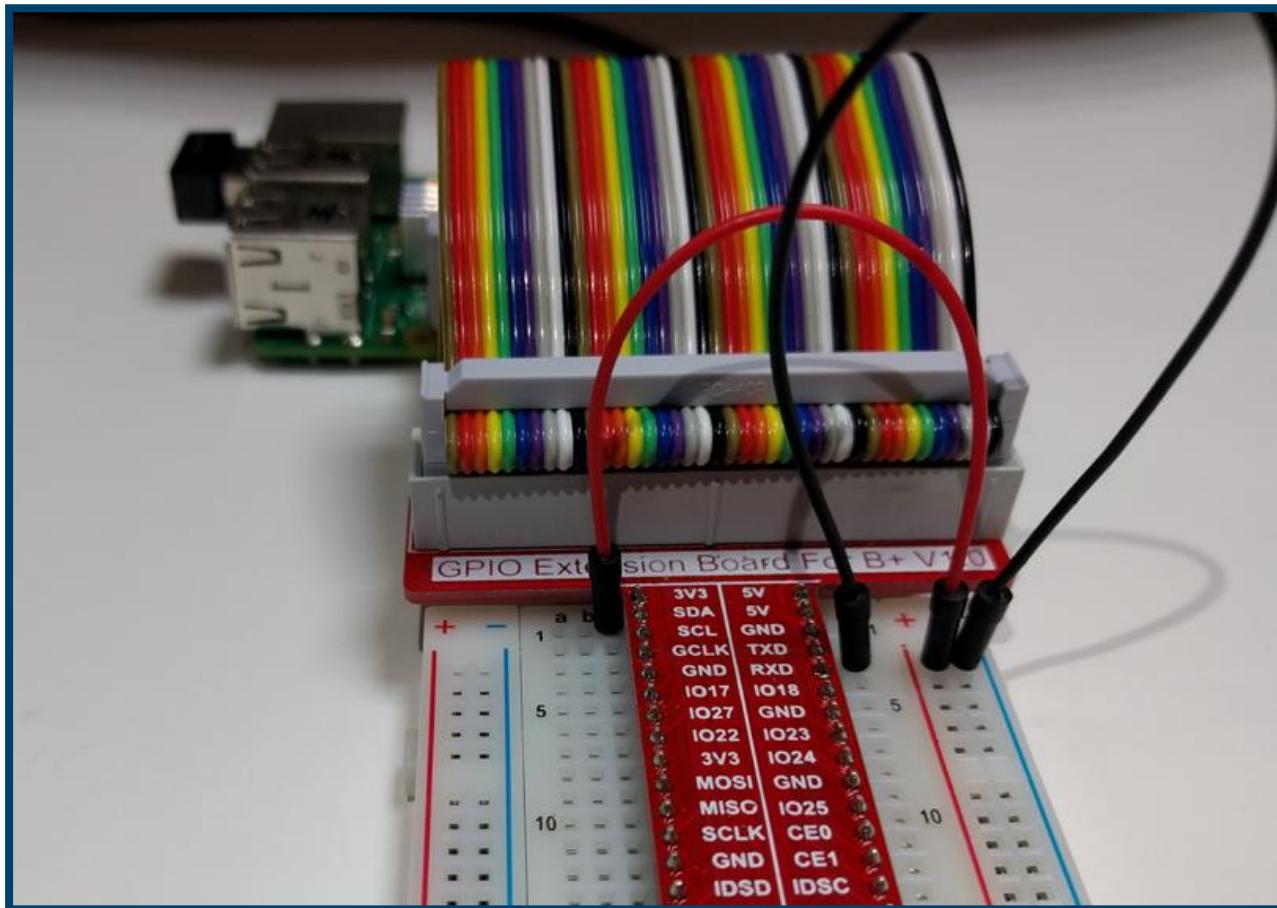
날짜



3. 라즈베리파이 GPIO 기초



3.3V와 GND 연결



3. 라즈베리파이 GPIO 기초



■ 옴의 법칙

$$V = I \times R$$

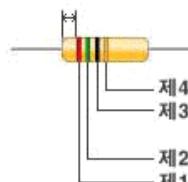
전압 전류 저항

3. 라즈베리파이 GPIO 기초

□ 저항

□ 저항값 읽기

끝에서부터 color까지의 풀이
좁은쪽 부터 읽어나갑니다.



제4색띠 : 저항값의 오차표시
제3색띠 : 셋째 수
(곱하는수,0의 갯수)
제2색띠 : 둘째 수
제1색띠 : 첫째 수

제 1 2 3 저
색 색 색 항
Ⅲ Ⅲ Ⅲ 오 차

- 1) 주황 주황 적색 금색
3 3 100 ±5% → $3300\Omega = 3.3k\Omega$, 오차±5%
- 2) 갈색 흑색 금색 금색
1 0 0.1 ±5% → 1Ω, 오차±5%
- 3) 노랑 보라 노랑 금색
4 7 10000 ±5% → $470000\Omega = 470k\Omega$, 오차±5%

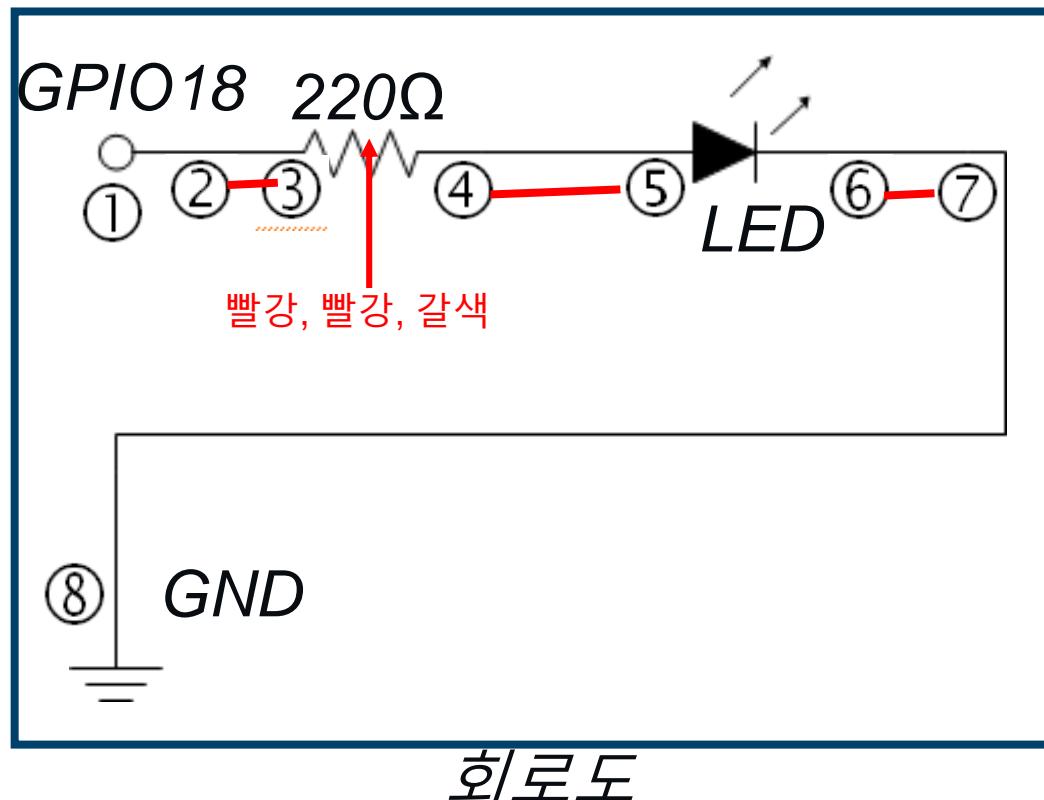
| 색상 COLOR | 저항환산표 | | | |
|-------------|-------|------|-------------|-------|
| | 첫째 수 | 둘째 수 | 셋째 수(곱하는 수) | 오차표시 |
| 검정(흑색) | 0 | 0 | 1 | |
| 밤색(갈색) | 1 | 1 | 10 | |
| 빨강(적색) | 2 | 2 | 100 | |
| 주황색(동색) | 3 | 3 | 1000 | |
| 노랑(황색) | 4 | 4 | 10000 | |
| 초록색(녹색) | 5 | 5 | 100000 | |
| 파랑색(청색) | 6 | 6 | 1000000 | |
| 보라색(자색) | 7 | 7 | 10000000 | |
| 회색(회색) | 8 | 8 | 100000000 | |
| 흰색(백색) | 9 | 9 | 1000000000 | |
| 금색 | | | 0.1 | ± 5% |
| 은색 | | | 0.01 | ± 10% |
| 무색 | | | | ± 20% |

그림출처 : http://kinimage.naver.net/storage/upload/2005/03/35/yjsrun_1111846591.gif

3. GPIO 출력 제어 (LED)

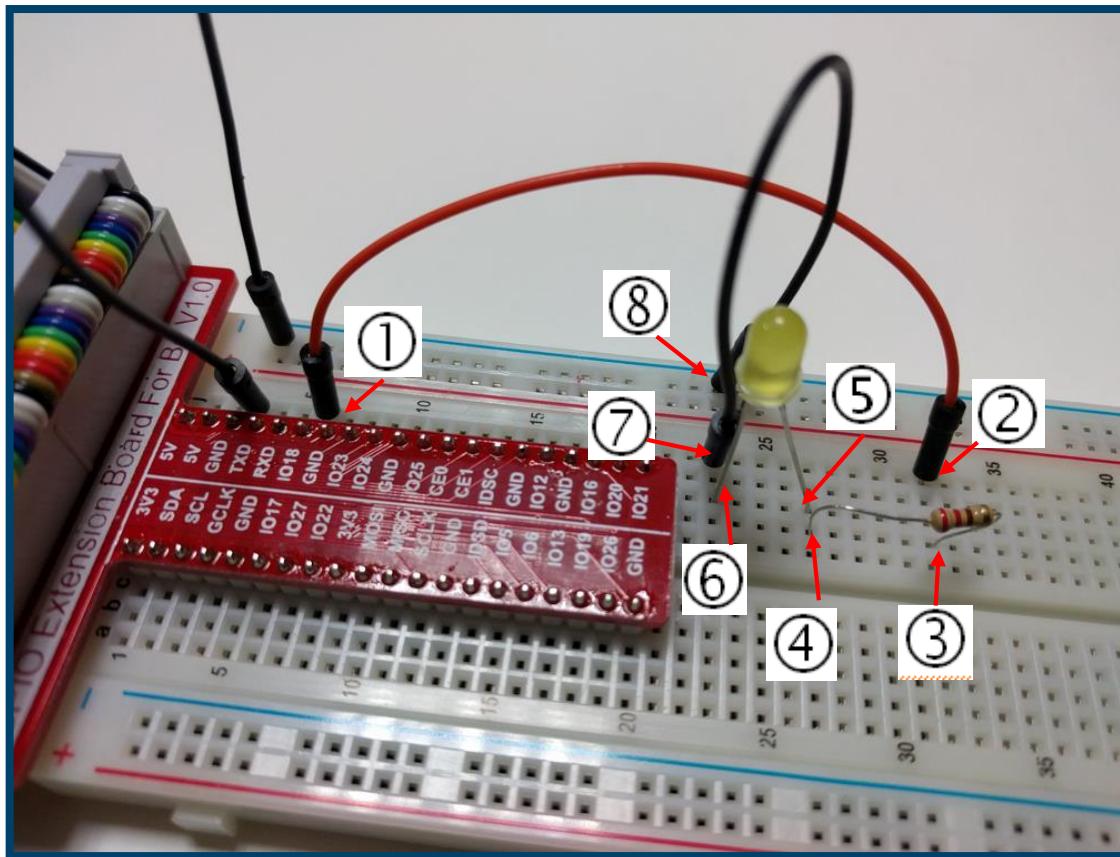


- GPIO 출력을 이용한 LED 제어
 - GPIO 출력핀에 LED를 연결하여 LED를 점등 / 소등



3. GPIO 출력 제어 (LED)

- 회로 구성



3. GPIO 제어 방법



□ GPIO 제어 방법

- device file에 리눅스 echo 명령어를 이용하여 입출력
- device file에 file read/write 함수를 이용하여 입출력
- raspi-gpio (GPIO debug용) 명령어를 이용
- C 프로그램으로 제어
 - 간접제어(라이브러리 함수 이용)
 - wiring pi 라이브러리(지원중단됨)
 - pigpio 라이브러리
 - 직접제어 - 메모리에 직접 접근하여 GPIO 제어
 - 주변기기 제어 레지스터들은 메모리의 특정 주소에 맵핑되어있음
 - 메모리 특정주소에 Read / Write 함으로써 하드웨어를 제어

3. GPIO 출력 제어 (LED) - 리눅스명령어



- echo "18" > /sys/class/gpio/export **GPIO18을 시스템에 추가**
- echo "out" > /sys/class/gpio/gpio18/direction **GPIO18을 output 모드로 사용**
- echo "1" > /sys/class/gpio/gpio18/value **GPIO18에 “1”을 출력**
- echo "0" > /sys/class/gpio/gpio18/value **GPIO18에 “0”을 출력**
- echo "18" > /sys/class/gpio/unexport **GPIO18을 시스템에서 삭제**

3. GPIO 출력 제어 (LED) - file read/write 함수



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

int ledControl(int gpio)
{
    int fd;
    char buf[BUFSIZ];
    fd = open("/sys/class/gpio/export", O_WRONLY);
    sprintf(buf, "%d", gpio);
    write(fd, buf, strlen(buf));
    close(fd);

    sprintf(buf, "/sys/class/gpio/gpio%d/direction", gpio);
    fd = open(buf, O_WRONLY);
    write(fd, "out", 3);
    close(fd);
```

3. GPIO 출력 제어 (LED) - file read/write 함수



```
sprintf(buf, "/sys/class/gpio/gpio%d/value", gpio);
fd = open(buf, O_WRONLY);
write(fd, "1", 1);
close(fd);

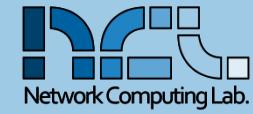
getchar();

fd = open("/sys/class/gpio/unexport", O_WRONLY);
sprintf(buf, "%d", gpio);
write(fd, buf, strlen(buf));
close(fd);
return 0;
}

int main(int argc, char **argv[])
{
    int gno;

    if(argc < 2)
    {
        printf("input error \n");
        return -1;
    }

    gno = atoi(argv[1]);
    ledControl(gno);
    return 0;
}
```



라이브러리 함수를 이용한 GPIO 간 접 제어

4.1 GPIO 출력 제어 (LED)-Wiring Pi



□ Wiring Pi 라이브러리를 이용한 GPIO (지원중단됨)

□ Wiring Pi?

- 라즈베리파이의 GPIO를 쉽게 하기 위한 인터페이스 라이브러리
- <http://wiringpi.com>
- gpio 유ти리티를 사용할 수 있게 함

□ gpio 유ти리티

- gpio 핀을 조작하기위한 스크립트
- 사용법
 - # gpio [옵션] [명령] [핀번호] [값]

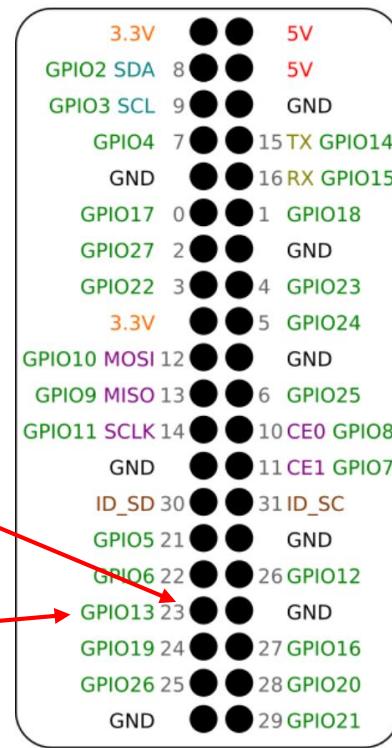
※ GPIO 핀번호를 BCM_GPIO로 사용하기 위해 -g 옵션 사용

4.1 GPIO 출력 제어 (LED)-Wiring PI



□ Wiring Pi C라이브러리

- C언어에서 사용할 수 있는 라이브러리
 - #include <wiringPi.h>를 소스코드에 추가하고, 컴파일(링크)시 -lwiringPi 추가
 - 라이브러리 함수
 - int wiringPiSetup(void);
 - Wiring Pi 설정 초기화, 가장 먼저 실행되어야 함
 - super user 권한 필요
 - Wiring Pi의 핀번호 사용
 - int wiringPiSetupGpio(void);
 - Wiring Pi 설정 초기화, 가장 먼저 실행되어야 함
 - super user 권한 필요
 - Broadcom GPIO 핀번호 사용



4.1 GPIO 출력 제어 (LED)-Wiring PI



- `void pinMode (int pin, int mode);`
 - 핀의 모드 설정
 - INPUT / OUTPUT / PWM_OUTPUT / GPIO_CLOCK
- `void digitalWrite (int pin, int value);`
 - HIGH or LOW 값을 핀에 입력
- `int digitalRead (int pin);`
 - 핀의 값을 읽음
- `void delay (unsigned int howLong);`
 - howLong 시간동안 프로그램 중지
 - 시간을 millisecond 단위로 입력

4.1 GPIO 출력 제어 (LED)-Wiring Pi



Wiring Pi C라이브러리 예제

1초동안 LED 점멸

pin을 out 모드로 전환

1초 정지

```
#include <wiringPi.h>

int main()
{
    int pin = 18;
    wiringPiSetupGpio();
    pinMode(pin, OUTPUT);
    digitalWrite(pin, HIGH);
    delay(1000);
    digitalWrite(pin, LOW);

    return 0;
}
```

Wiring Pi 초기화

pin에 HIGH (1) 입력

pin에 LOW (0) 입력

4.1 GPIO 출력 제어 (LED) - pigpio



pigpio

- General Purpose Input Outputs (GPIO) 라이브러리
- #include <pigpio.h> 를 추가하고, 컴파일 (링크) 시 -pthread -lpigpio -lrt 옵션 추가
- 주요 함수
 - int gpioInitialise()
 - GPIO 초기화
 - 성공 시 pigpio 버전 리턴, 실패 시 PI_INIT_FAILED 리턴

```
if (gpioInitialise() < 0)
{
    // pigpio initialisation failed.
}
else
{
    // pigpio initialised okay.
}
```

4.1 GPIO 출력 제어 (LED) - pigpio



- `void gpioTerminate()`
 - GPIO 사용 중지
- `int gpioSetMode(unsigned gpio, unsigned mode)`
 - 특정 핀의 모드를 선택
 - `gpio` : 핀 번호 (0-53)
 - `mode` : 설정할 모드
 - `PI_INPUT`
 - `PI_OUTPUT`
 - `PI_ALT0 ~ PI_ALT5`
 - 성공 시 0, 실패 시 `PI_BAD_GPIO` or `PI_BAD_MODE` 리턴

4.1 GPIO 출력 제어 (LED) - pigpio



- int gpioGetMode(unsigned gpio)
 - 특정 핀의 모드를 리턴
 - gpio : 핀 번호 (0-53)
 - 성공 시 모드 리턴, 실패 시 PI_BAD_GPIO 리턴

- int gpioRead(unsigned gpio)
 - 특정 핀의 상태를 읽음
 - gpio : 핀 번호 (0-53)
 - 성공 시 GPIO level (0, 1)을 리턴, 실패 시 PI_BAD_GPIO 리턴

```
printf("GPIO24 is level %d", gpioRead(24));
```

4.1 GPIO 출력 제어 (LED) - pigpio



- int gpioWrite(unsigned gpio, unsigned level)
 - 특정 핀에 값을 씁
 - gpio : 핀 번호 (0-53)
 - level : 쓸 값 (0, 1))
 - 성공 시 GPIO level (0, 1)을 리턴, 실패 시 PI_BAD_GPIO or PI_BAD_LEVEL 리턴

```
gpioWrite(24, 1); // Set GPIO24 high
```

- uint32_t gpioDelay(uint32_t micros)
 - 지정된 시간만큼 멈춤
 - micros : 멈출 시간 (microseconds)

4.1 GPIO 출력 제어 (LED) - pigpio

1초동안 LED 점멸

pin 을 out 모드
로 전환

1초 정지

```
#include <stdio.h>
#include <pigpio.h>
int main()
{
    int pin = 18;
```

gpiodInitialise();

```
gpiodSetMode(pin, PI_OUTPUT);
gpiodWrite(pin, 1);
gpiodDelay(1000000);
gpiodWrite(pin, 0);
```

```
gpiodTerminate();
return 0;
}
```

GPIO 초기화

pin off HIGH (1) 입력

pin on LOW (0) 입력

4.1 GPIO 제어 - pigpio와 wiring pi 비교

| | pigpio | wiring pi* |
|------------------|------------------------|-------------------------|
| 초기화함수 | gpioInitialise() | wiringPiSetupGpio() |
| 종료함수 | gpioTerminate() | X |
| 모드 set 함수(입력/출력) | gpioSetMode(pin, mode) | pinMode(pin, mode) |
| 모드 읽어오기함수 | gpioGetMode(pin) | X |
| read 함수 | gpioRead(pin) | digitalRead(pin) |
| write 함수 | gpioWrite(pin, level) | digitalWrite(pin, HIGH) |
| delay 함수 | gpioDelay(micros) | delay(micros) |

* <http://wiringpi.com/reference/core-functions/>

* <http://wiringpi.com/reference/raspberry-pi-specifics/>

4.1 GPIO 출력 제어 (LED) - pigpio

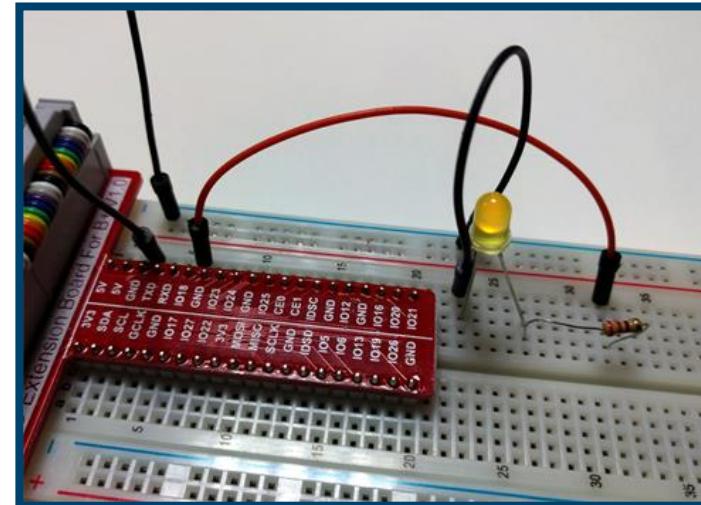
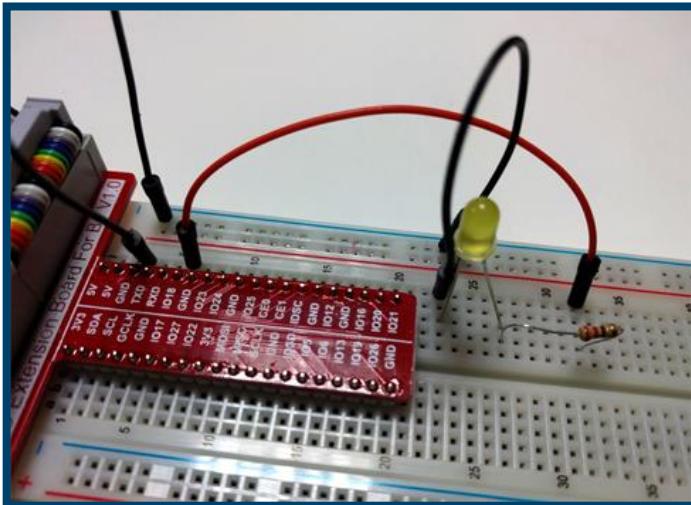


- 컴파일

```
# gcc -o led led.c -pthread -lpigpio -lrt
```

- 실행

```
# sudo ./led
```



1초간 점멸

4.1 GPIO 제어 - pigpio



OVERVIEW

ESSENTIAL

[gpioInitialise](#)

Initialise library
Stop library

BASIC

[gpioSetMode](#)

Set a GPIO mode

[gpioGetMode](#)

Get a GPIO mode

[gpioSetPullUpDown](#)

Set/clear GPIO pull up/down resistor

[gpioRead](#)

Read a GPIO

[gpioWrite](#)

Write a GPIO

PWM (overrides servo commands on same GPIO)

[gpioPWM](#)

Start/stop PWM pulses on a GPIO

[gpioSetPWMfrequency](#)

Configure PWM frequency for a GPIO

[gpioSetPWMrange](#)

Configure PWM range for a GPIO

[gpioGetPWMdutyCycle](#)

Get dutycycle setting on a GPIO

[gpioGetPWMfrequency](#)

Get configured PWM frequency for a GPIO

[gpioGetPWMrange](#)

Get configured PWM range for a GPIO

[gpioGetPWMrealRange](#)

Get underlying PWM range for a GPIO

출처: <http://abyz.me.uk/rpi/pigpio/cif.html>

4.1 GPIO 제어 - pigpio



□ pigpio 사용 예제

- GPIO 활용
- <https://github.com/joan2937/pigpio>

- 센서 활용
- <http://abyz.me.uk/rpi/pigpio/examples.html>

4.1 GPIO 출력이용한 LED제어 (실습 1)



□ LED 신호등 제작

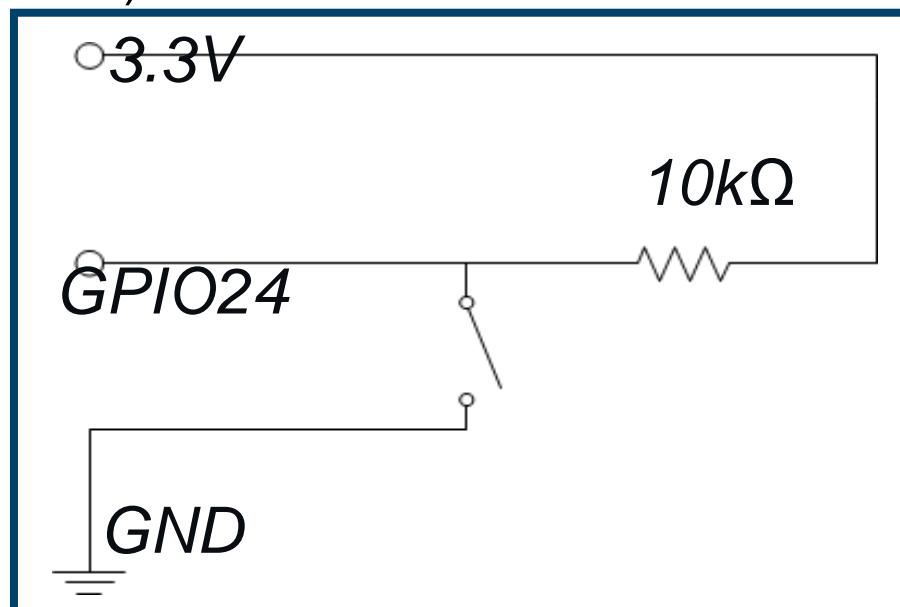
- 3개의 LED (빨강, 노랑, 녹색) 이용
- 일정한 순서와 시간동안 LED 점등
 - 녹색 ON – 노랑 ON - 빨강 ON – 녹색 ON - ... 반복
- 제작 방법
 - GPIO 18, 23, 25번에 각각 LED 연결
 - 18, 23, 25번핀을 순서에 맞게 ON 하고 delay() 함수로 일정 시간 (예:3초) 유지
 - 위의 과정을 무한 반복 (while(1))

4.2 GPIO 입력 (스위치)



GPIO 입력을 이용한 스위치 사용

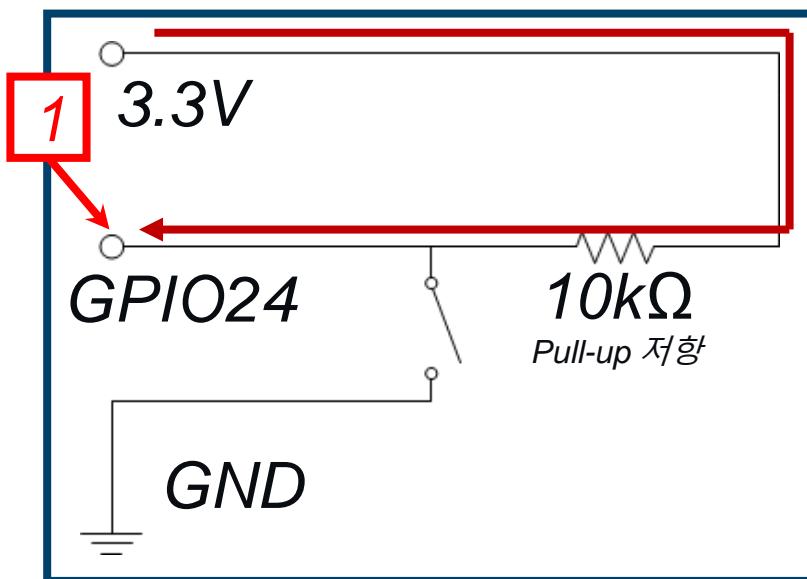
- GPIO 입력핀에 스위치를 연결하여 스위치 On/Off 여부를 입력 받음
- 입력핀(GPIO24)에 3.3V가 감지되면 1로 판단(스위치off), 아니면 0 (스위치 on)



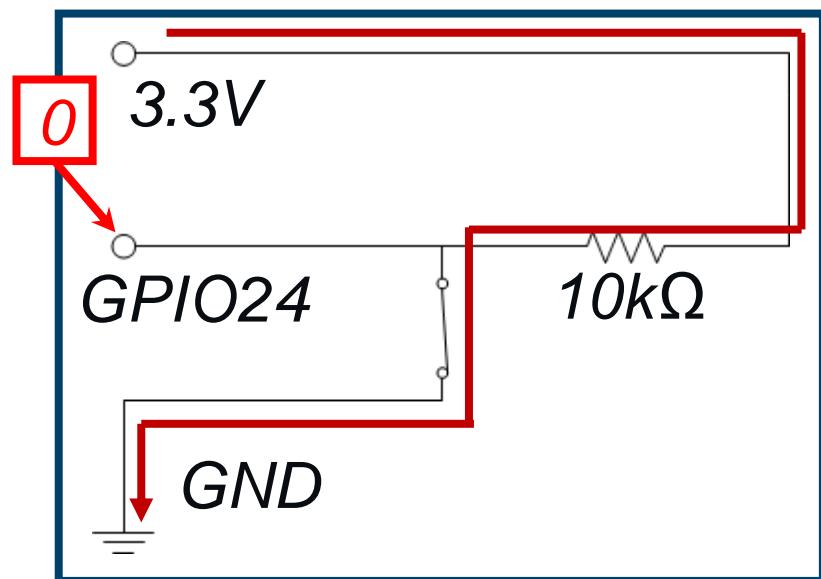
회로도

4.2 GPIO 입력 (스위치)

- 동작 원리



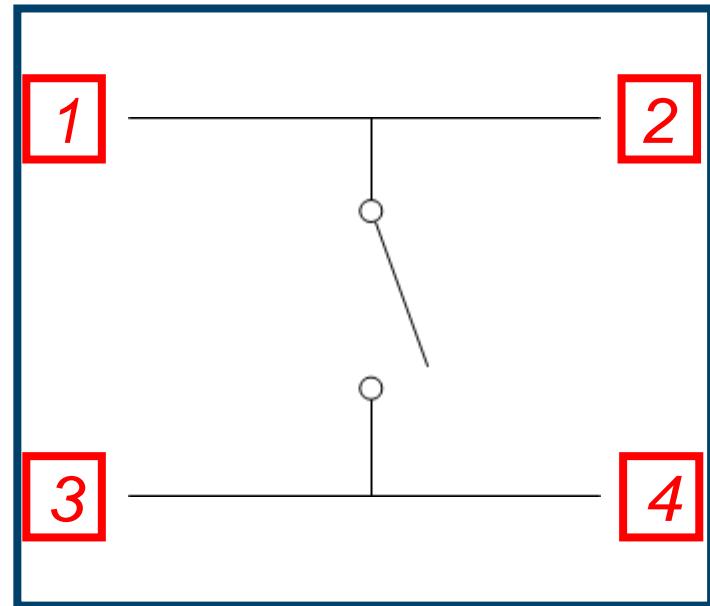
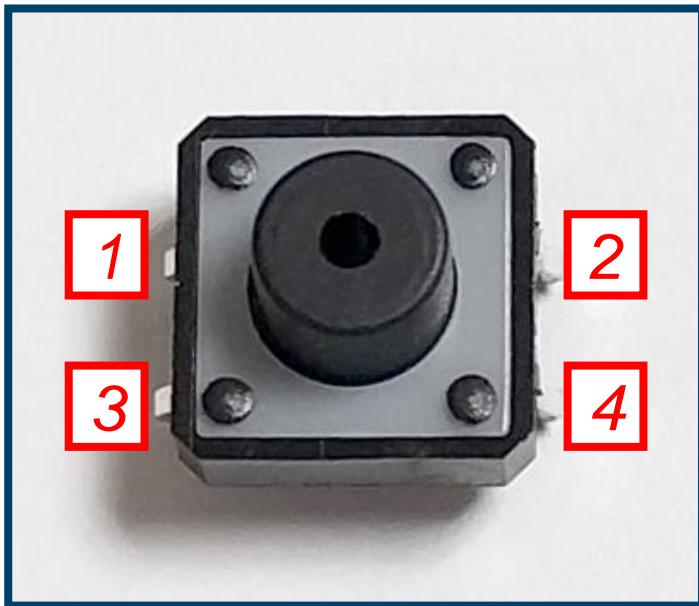
스위치 OFF



스위치 ON

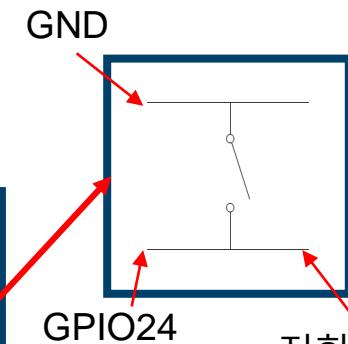
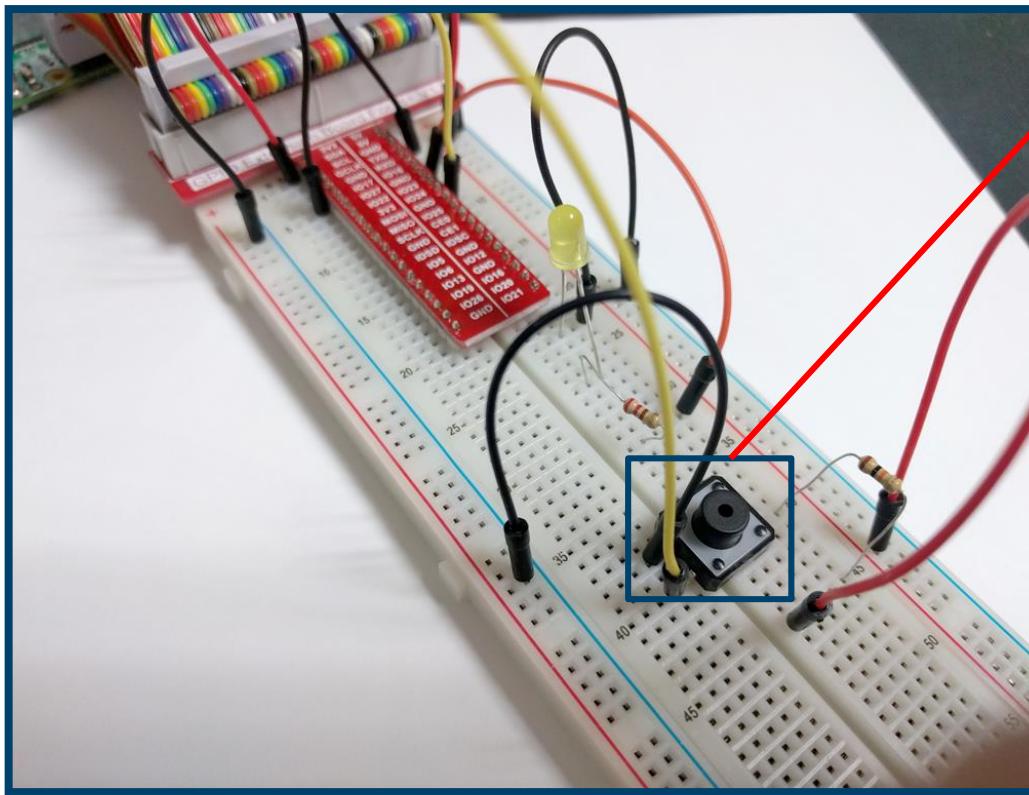
4.2 GPIO 입력 (스위치)

- 스위치 구조



4.2 GPIO 입력 (스위치)

- 회로 구성



GPIO24

GND

저항이
연결되면
반대편은
3.3V에
연결

4.2 GPIO 입력 (스위치)



Wiring Pi C라이브러리 예제

```
#include <wiringPi.h>

int main()
{
    int led_pin = 18;
    int switch_pin = 24;
    int i;

    wiringPiSetupGpio();
    pinMode(led_pin, OUTPUT);
    pinMode(switch_pin, INPUT);

    for(i=0: i<1000000000: i++) {
        if(digitalRead(switch_pin) == LOW) {
            digitalWrite(led_pin, HIGH);
            delay(100);
            digitalWrite(led_pin, LOW);
        }
    }

    return 0;
}
```

스위치가 눌렸는지 검사

스위치 연결 핀 모드 변경

4.2 GPIO 입력 (스위치)



■ pigpio C라이브러리 예제

24번 핀의 모드 변경

switch 입력 검사

0.1초 동안 정지

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

int main()
{
    int pin = 18;
    int switch_pin = 24;

    gpioInitialise();
    gpioSetMode(pin, PI_OUTPUT);
    gpioSetMode(switch_pin, PI_INPUT);

    while(1) {
        if(gpioRead(switch_pin) == 0) {
            gpioWrite(pin, 1);
            gpioDelay(100000);
            gpioWrite(pin, 0);
        }
    }

    gpioTerminate();
    return 0;
}
```

4.2 GPIO 입출력이용한 LED/스위치 실습(실습 2)



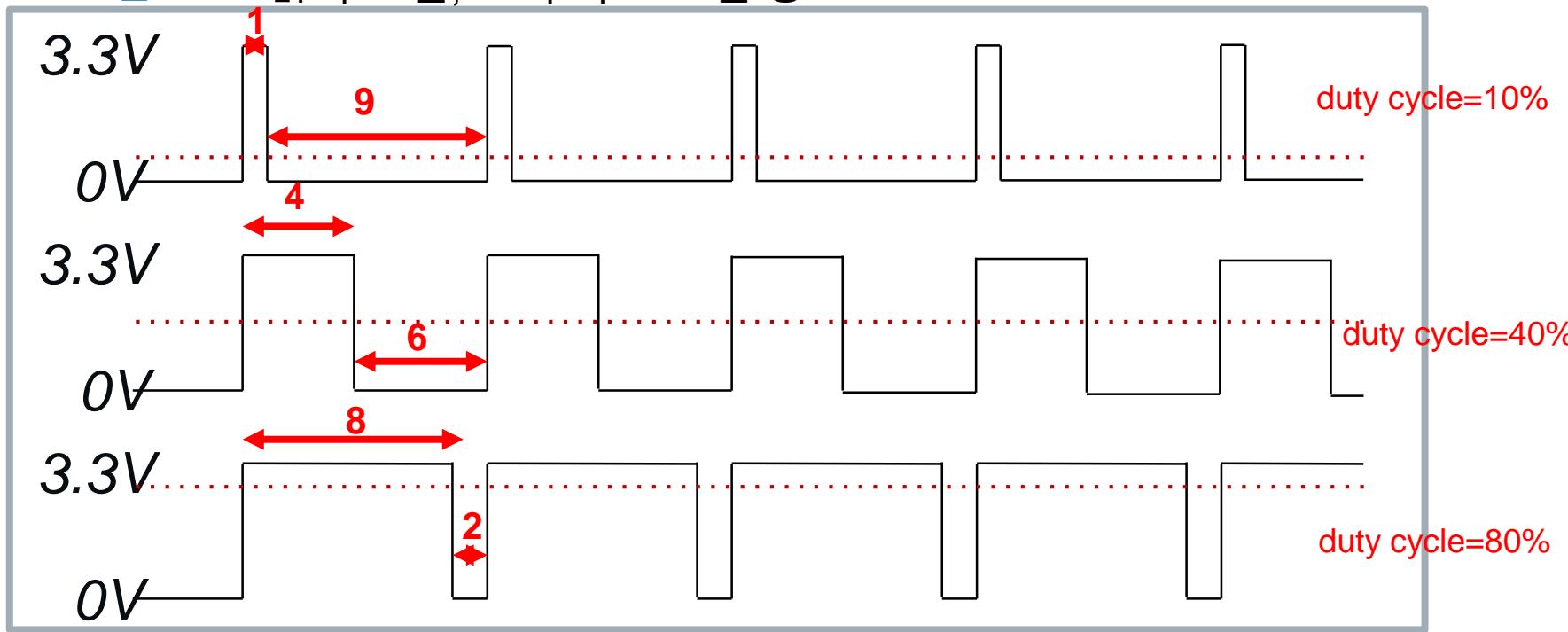
□ LED On/Off 스위치 제작

- 이전의 예제를 수정하여 스위치를 한 번 누르면 LED가 켜지고, 다시 한번 누르면 LED가 꺼지는 On/Off 스위치 제작
- 제작 방법
 - 스위치를 누를 때마다 gpioWrite() 함수에 HIGH, LOW가 교대로 들어가게 설정
 - 위의 과정을 무한 반복 (`while(1)`)
 - delay() 함수를 이용하여 스위치 입력이 짧은 시간 내에 여러번 들어가는 것을 막을 수 있음

4.3 GPIO 출력 (PWM)

PWM (Pulse Width Modulation)

- 펄스 폭 변조
- 디지털 회로에서 신호의 세기를 조절할 때 사용
 - LED 밝기 조절, 모터 속도 조절 등



4.3 GPIO 출력 (PWM)

■ 라즈베리파이의 PWM

- Hardware로 구현 (라이브러리에서 소프트웨어로도 사용 가능)
- 2개 채널 사용 가능 (사용할 수 있는 핀이 다른)
 - 12, 18 → 0번 채널
 - 13, 19 → 1번 채널

| | PWM0 | PWM1 |
|---------|-----------|-----------|
| GPIO 12 | Alt Fun 0 | - |
| GPIO 13 | - | Alt Fun 0 |
| GPIO 18 | Alt Fun 5 | - |
| GPIO 19 | - | Alt Fun 5 |
| GPIO 40 | Alt Fun 0 | - |
| GPIO 41 | - | Alt Fun 0 |
| GPIO 45 | - | Alt Fun 0 |
| GPIO 52 | Alt Fun 1 | - |
| GPIO 53 | - | Alt Fun 1 |

4.3 GPIO 출력 (PWM)



■ pigpio에서 PWM

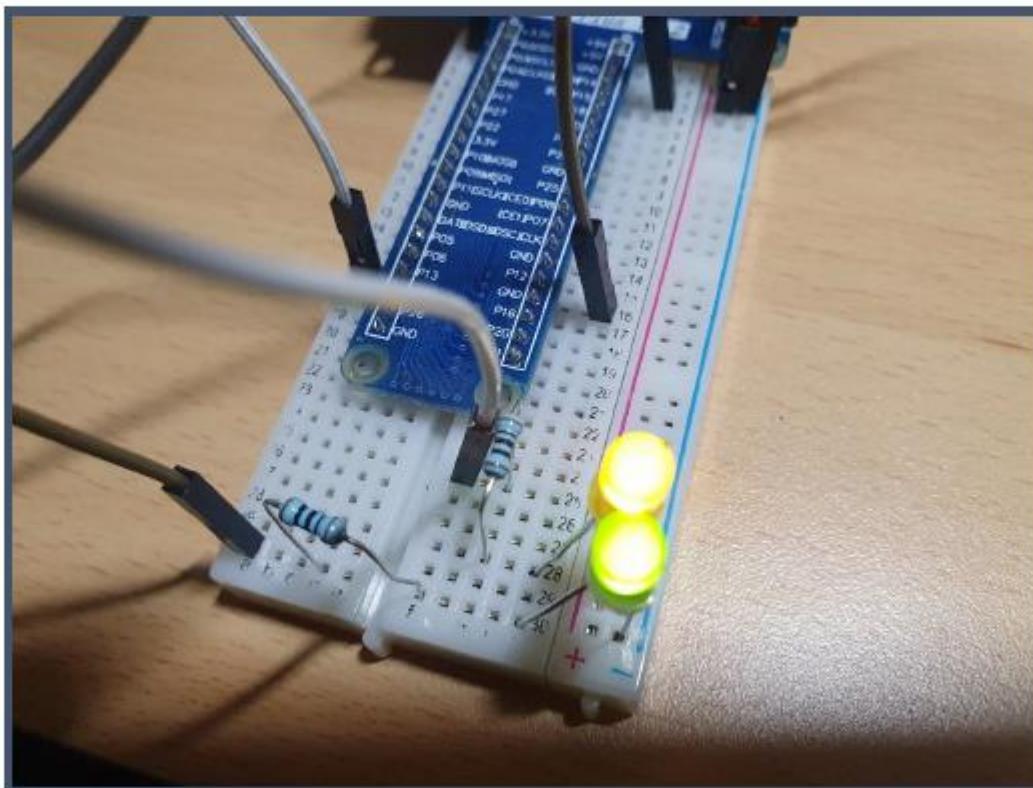
- Hardware PWM 사용
- `int gpioHardwarePWM(unsigned gpio, unsigned PWMfreq, unsigned PWMduty)`
 - `gpio` : 핀 번호
 - `PWMfreq` : PWM 주파수
 - 1-125M, 0은 꺼짐
 - 라즈베리파이 4는 1-137.5M
 - ex) 1 → 1초, 2 → 0.5초
 - `PWMduty` : duty cycle
 - 1-1000000, 0은 꺼짐
 - ex) 1000000 → 100%, 500000 → 50%, 300000 → 30%

4.3 GPIO 출력 (PWM)



■ PWM 예제

- LED를 1, 0.5초마다 깜빡이게 함
- 회로 구성 (12, 13번 핀에 LED 연결)



4.3 GPIO 출력 (PWM)



주파수 1, duty 50%

주파수 2, duty 50%

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

int main()
{
    int led1 = 12;
    int led2 = 13;

    gpioInitialise();

    gpioHardwarePWM(led1, 1, 500000);
    gpioHardwarePWM(led2, 2, 500000);

    gpioTerminate();

    return 0;
}
```

4.3 GPIO 출력 (PWM)



■ PWM 예제 (LED 밝기 조절)

- LED 밝기를 점차 밝아지고 점차 어두워지게 함
- duty cycle을 조절

duty cycle 증가 혹은 감소
(step +, -에 따라 달라짐)

duty cycle $\geq 100\%$ 거나
 ≤ 0 이면 역방향으로 전환

delay만큼 정지 (0.01초)

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

int main()
{
    int led1 = 12;
    int step = 10000;
    int delay = 10000;
    int freq = 10000;
    int duty = 0;

    gpioInitialise();

    while(1) {
        duty += step;
        if(duty >= 1000000 || duty <= 0) {
            step = -step;
        }
        gpioHardwarePWM(led1, freq, duty);

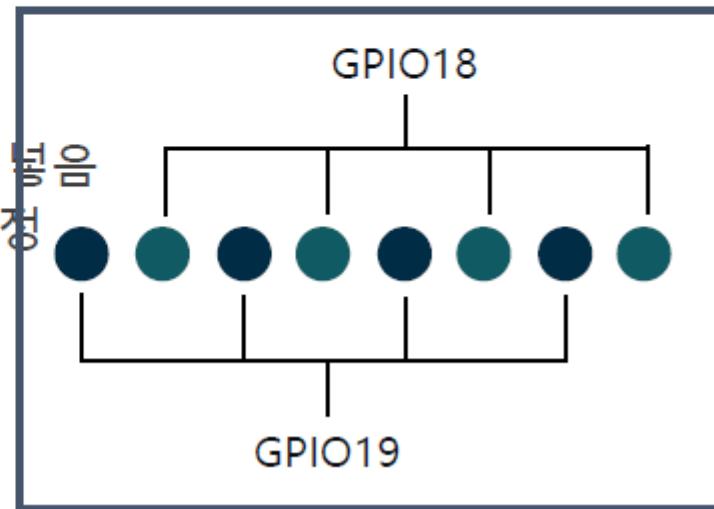
        gpioDelay(delay);
    }

    gpioTerminate();
    return 0;
}
```

4.3 GPIO 출력 (PWM) (실습 3)

■ LED 점멸 조명 제작

- 여러개의 LED의 밝기를 주기적으로 조절
- 홀수번째 LED와 짝수번째 LED는 밝기를 반대로 조절
- 제작방법
 - 두 개의 GPIO 핀에 LED를 병렬 연결
 - 두 개의 핀 PWM 값을 반대로 넣음
 - duty를 시간에 따라 변하게 설정



4.3 GPIO 출력 (PWM) (실습 3)



■ LED 밝기조절 스위치제작

- 두개의 스위치를 이용하여 LED 밝기 UP / DOWN기능추가
- 10단계밝기조절(duty cycle 0%, 10%, 20%,....90%)

■ UP 버튼

- duty cycle 증가

■ DOWN 버튼

- duty cycle 감소