

IoT Platform 4th Week

- Interfacing to RPi Inputs and Outputs -

Jaeseok Yun

Soonchunhyang University

RPi GPIO

General purpose input output

RPi GPIO

■ 디지털 출력

- 디지털 값 (0 또는 1)을 출력하여 전자 회로 (예, 릴레이)를 구동
- 각 핀에 대해 **3.3V, 2-3 mA**까지만 출력 가능 (아두이노 40 mA와 비교)
- 예, LED 점등 제어

■ 디지털 입력

- 디지털 값 (0 또는 1)을 읽어 RPi 애플리케이션에서 활용
- 예, 온도 센서에서 온도 값 활용

■ 아날로그 출력

- PWM (pulse width modulation)을 이용해 아날로그 전압 레벨을 모사
- 예, 서보 모터 제어, LED 밝기 제어

■ 아날로그 입력

- RPi는 **아날로그-디지털 변환기** (ADC: analog-to-digital converter)가 **없음**
- 외부 ADC를 추가하여 디지털 입력 (또는 버스)를 통해 기능 확장

RPi GPIO 핀 헤더^{1,2,3}



Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I2C)	I ² C	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)	SPI0	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

¹ <http://www.raspberry-pi-geek.com/howto/GPIO-Pinout-Rasp-Pi-1-Model-B-Rasp-Pi-2-Model-B>

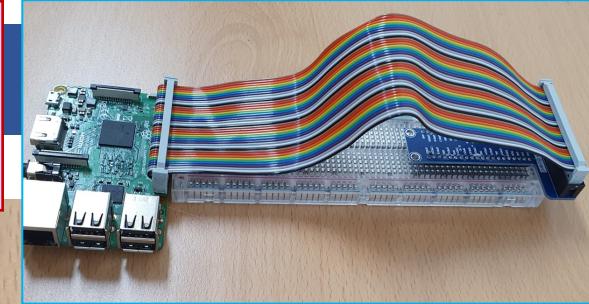
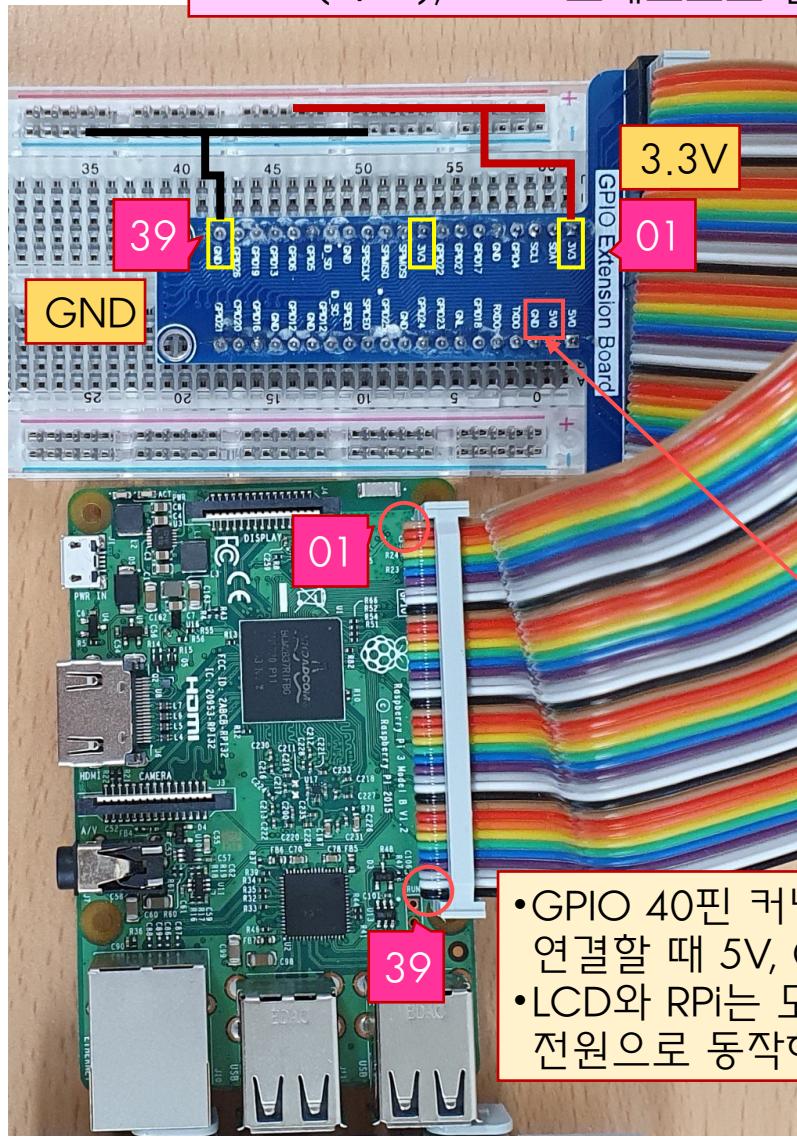
² 익스플로링 라즈베리 파이, p233

³ <https://pinout.xyz/pinout/>

RPi GPIO T바 연결



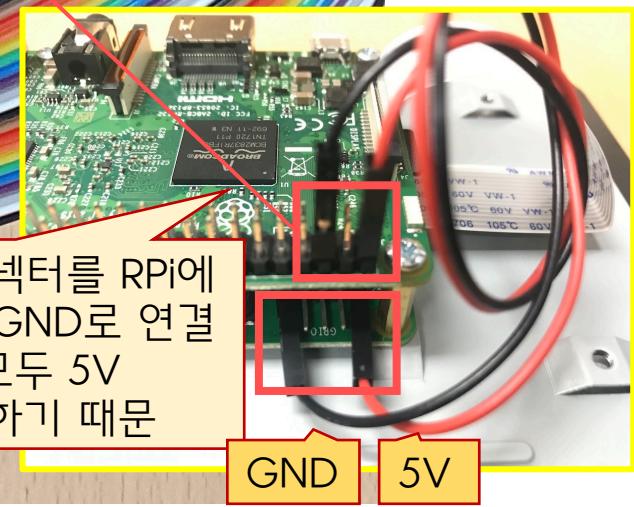
- 뒤로 갈수록 하드웨어 구성이 복잡해지기 때문에 **GPIO 확장 보드** (T-바) 활용 필요!
- Vcc (3.3V), GND 브레드보드 연결



- LCD 전원 어댑터 연결



- GPIO 40핀 커넥터를 RPi에 연결할 때 5V, GND로 연결
- LCD와 RPi는 모두 5V 전원으로 동작하기 때문



5V
GND

GPIO

sysfs

RPi GPIO 핀 헤더^{1,2,3}



Pin#	NAME
01	3.3v DC Power
02	GPIO02 (SDA1_I2C)
03	• 디지털 입력 • 풀다운 저항 활용 • 누를 때 1을 입력하는 스위치 꾸미기
04	(GPIO_GEN0)
13	GPIO27 (GPIO_GEN2)
15	GPIO22 (GPIO_GEN3)
17	3.3v DC Power
19	GPIO10 (SPI_MOSI)
21	GPIO09 (SPI_MISO)
23	GPIO11 (SPI_CLK)
25	Ground
27	ID_SD (I2C ID EEPROM)
29	GPIO05
31	GPIO06
33	GPIO13
35	GPIO19
37	GPIO26
39	Ground

NAME	Pin#
DC Power 5v	02
DC Power 5v	04
Ground	06
(TXD0) GPIO14	08
(RXD0) GPIO15	10
(GPIO_GEN1) GPIO18	12
Ground	14
(GPIO_GEN4) GPIO23	16
(GPIO_GEN5) GPIO24	18
Ground	20
(GPIO_GEN6) GPIO25	22
(SPI_CE0_N) GPIO08	24
(SPI_CE1_N) GPIO07	26
(I2C ID EEPROM) ID_SC	28
Ground	30
GPIO12	32
Ground	34
GPIO16	36
GPIO20	38
GPIO21	40

¹ <http://www.raspberry-pi-geek.com/howto/GPIO-Pinout-Rasp-Pi-1-Model-B-Rasp-Pi-2-Model-B>

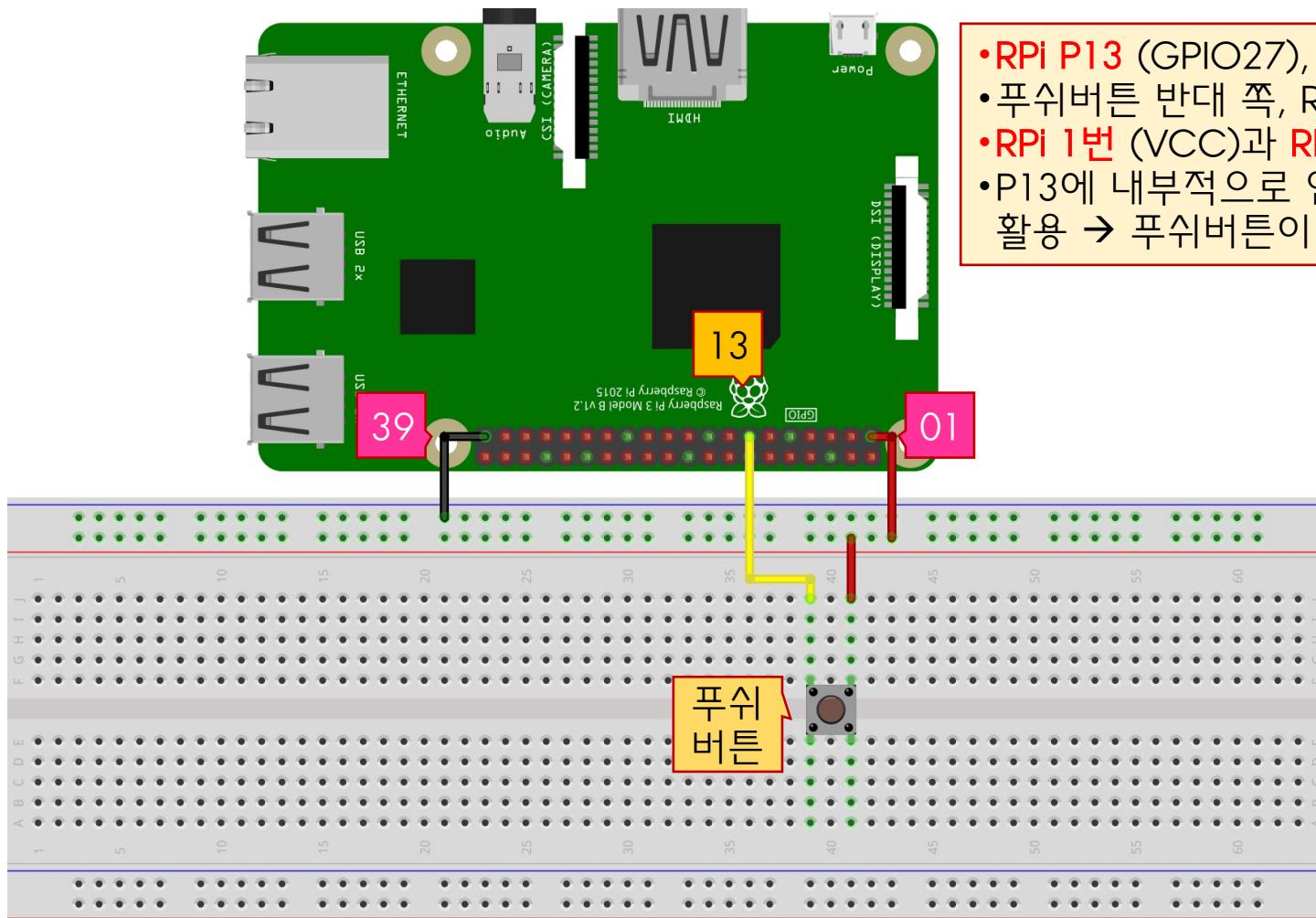
² 익스플로링 라즈베리 파이, p233

³ <https://pinout.xyz/pinout/>

디지털 입력 회로 구성

Practice

■ 리눅스 sysfs (파일 시스템) 활용 디지털 입력



- RPi P13 (GPIO27), 푸쉬버튼 한쪽 연결
- 푸쉬버튼 반대 쪽, RPi VCC 연결
- RPi 1번 (VCC)과 RPi 39번 (GND) 활용
- P13에 내부적으로 연결된 풀다운 저항 활용 → 푸쉬버튼이 눌리지 않으면 0V

디지털 입력 sysfs

Practice

■ 리눅스 sysfs (파일 시스템) 활용 GPIO 직접 제어

```
pi@raspberrypi: /sys/class/gpio/gpio27
pi@raspberrypi:/sys/class/gpio $ ls
export gpiochip0 gpiochip100 gpiochip128 unexport
pi@raspberrypi:/sys/class/gpio $ echo 27 > export
pi@raspberrypi:/sys/class/gpio $ ls
export gpio27 gpiochip0 gpiochip100 gpiochip128 unexport
pi@raspberrypi:/sys/class/gpio $ cd gpio27
pi@raspberrypi:/sys/class/gpio/gpio27 $ ls
active_low device direction edge power subsystem uevent value
pi@raspberrypi:/sys/class/gpio/gpio27 $ echo in > direction
pi@raspberrypi:/sys/class/gpio/gpio27 $ cat direction
in
pi@raspberrypi:/sys/class/gpio/gpio27 $ cat value
0
pi@raspberrypi:/sys/class/gpio/gpio27 $ cat value
1
pi@raspberrypi:/sys/class/gpio/gpio27 $ cat value
1
pi@raspberrypi:/sys/class/gpio/gpio27 $ cat value
0
pi@raspberrypi:/sys/class/gpio/gpio27 $ cat value
0
pi@raspberrypi:/sys/class/gpio/gpio27 $
```



- RPi의 GPIO27를 활성화
- 활성화된 GPIO27 디렉토리 이동

- **direction**: 입력 (in) 설정
- **cat**으로 **direction** 확인 가능

- **cat**으로 **value** 확인 가능
- 푸쉬 버튼을 누르면 '1' 출력
- 푸쉬 버튼을 떼면 '0' 출력

WiringPi

GPIO 제어 라이브러리

WiringPi¹

- RPi를 위해 C로 쓰인 GPIO 인터페이스 라이브러리
- 고든 헨더슨 (Gorden Henderson)이 개발 유지
- 아두이노의 Wiring 라이브러리와 유사
- 파이썬, 루비, 펄에서 연계하여 사용하기 위한 **3rd 파티
바인딩** 제공
 - 3rd 파티 바인딩이란 C로 쓰인 WiringPi 라이브러리를 다른 프로그래밍 언어 (예, 파이썬)에서 활용할 수 있게 하는 래퍼 (wrapper) 제공을 의미
- RPi에서 빠른 GPIO 스위칭 제공
- RPi 아닌 **일반적인 임베디드 리눅스 시스템에 적용 불가**
 - BCM2835, BCM2836, BCM2837 SoC를 위해 제작됨

WiringPi 설치¹

■ 설치 확인

```
pi@raspberrypi: ~ $ gpio -v
gpio version: 2.46
Copyright (c) 2012-2018 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Raspberry Pi Details:
  Type: Pi 3, Revision: 02, Memory: 1024MB, Maker: Unknown05
  * Device tree is enabled.
  *--> Raspberry Pi 3 Model B Rev 1.2
  * This Raspberry Pi supports user-level GPIO access.
pi@raspberrypi: ~ $
```

■ 설치 (git이 설치되었다는 가정하에)

- cd ~/
- git clone https://github.com/WiringPi/WiringPi
- cd ~/WiringPi
- ./build
- gpio -v

• RPi OS Desktop에 포함되어
있지 않으므로 설치 필요!
(2022/3 기준)

WiringPi 설치

■ 설치 후 테스트

```
pi@raspberrypi: ~
```

• 물리 번호 • wPi 번호 • 브로드컴 (BCM) 번호
• GPIOXX

```
pi@raspberrypi: ~ $ gpio readall
```

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	0	11 12	0	IN	GPIO. 1	1	18
27	2	GPIO. 2	IN	0	13 14			0v		
22	3	GPIO. 3	IN	0	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	IN	0	33 34			0v		
19	24	GPIO.24	IN	0	35 36	0	IN	GPIO.27	27	16
26	25	GPIO.25	IN	0	37 38	0	IN	GPIO.28	28	20
		0v			39 40	0	IN	GPIO.29	29	21
		Name	Mode	V	Physical	V	Mode	Name	wPi	BCM
					Pi 3					

```
pi@raspberrypi: ~ $
```

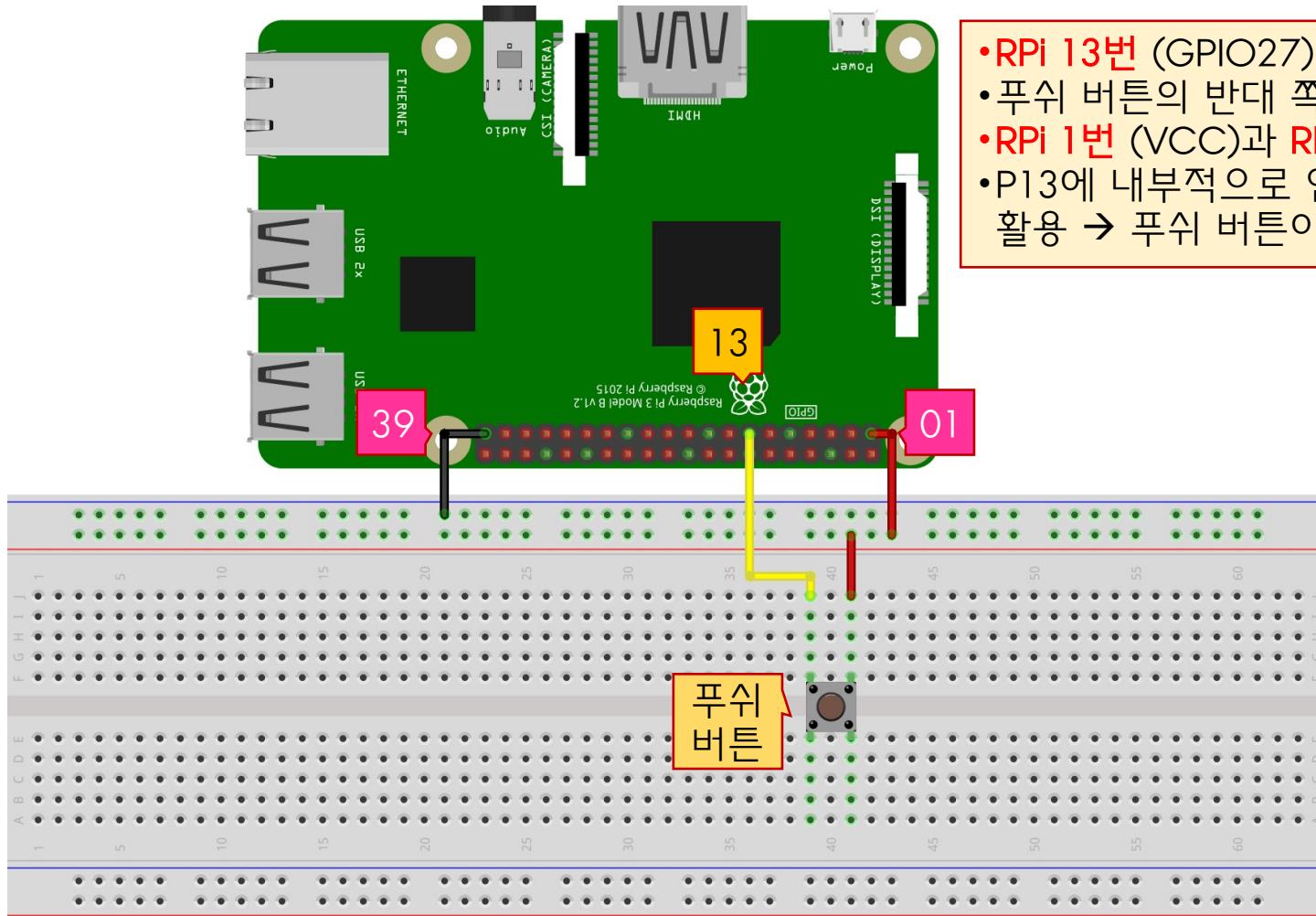
디지털 입력

WiringPi

디지털 입력 회로 구성

Practice

■ wPi를 이용한 디지털 입력



- RPi 13번 (GPIO27), 푸쉬 버튼 한쪽 연결
- 푸쉬 버튼의 반대 쪽, RPi VCC 연결
- RPi 1번 (VCC)과 RPi 39번 (GND) 활용
- P13에 내부적으로 연결된 풀다운 저항 활용 → 푸쉬 버튼이 눌리지 않으면 0V

디지털 입력 wPi 핀 확인

- 물리적 physical 핀 (13), GPIO (27), wPi (2)

```
pi@raspberrypi: ~
pi@raspberrypi: ~ $ gpio readall
+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | BCM |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   |   | 3.3v |   |   | 1 2 |   |   | 5v | 09 | Ground
|   |   |     |   |   | 3 4 |   |   | 5v | 11 | GPIO17 (GPIO_GEN0)
|   |   | SDA.1 | IN | 1 | 5 6 |   |   | 0v | 13 | GPIO27 (GPIO_GEN2)
|   |   | SCL.1 | IN | 1 | 7 8 | 0 | IN | TxD | 15 | GEN3)
|   |   | GPIO. 7 | IN | 1 | 9 10 | 1 | IN | RxD | 17 | 
|   |   | 0v |   |   | 11 12 | 0 | IN | GPIO | 18 | 
|   |   | GPIO. 0 | IN | 0 | 13 14 | 0 | IN | GPIO | 19 | 
|   |   | GPIO. 2 | IN | 0 | 15 16 | 0 | IN | GPIO | 20 | 
|   |   | GPIO. 3 | IN | 0 | 17 18 | 0 | IN | GPIO | 21 | 
|   |   | 3v |   |   | 22 23 | 0 | IN | GPIO. 6 | 6 | 25
|   |   | SI |   |   | 24 25 | 1 | IN | CE0 | 10 | 8
|   |   | SO |   |   | 26 27 | 1 | IN | CE1 | 11 | 7
|   |   | 0v |   |   | 28 29 | 1 | IN | SCL.0 | 31 | 1
|   |   | 0A.0 |   |   | 30 31 | 0 | IN | 0v |   |   |
|   |   | 0.21 |   |   | 32 33 | 0 | IN | GPIO.26 | 26 | 12
|   |   | 6 | 22 | GPIO.22 | IN | 1 | 31 34 | 0 | IN | GPIO.27 | 27 | 16
|   |   | 13 | 23 | GPIO.23 | IN | 0 | 33 35 | 0 | IN | GPIO.28 | 28 | 20
|   |   | 19 | 24 | GPIO.24 | IN | 0 | 36 37 | 0 | IN | GPIO.29 | 29 | 21
|   |   | 26 | 25 | GPIO.25 | IN | 0 | 38 39 | 0 | IN | 0v |   |   |
|   |   | 0v |   |   | 40 41 | 0 | IN |   |   |   |   |
+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | BCM |
+-----+-----+-----+-----+-----+-----+-----+-----+
pi@raspberrypi: ~ $
```

• GPIO 번호는 27

• wPi 핀 번호는 2

• 물리적 핀 번호는 13

wPi SW 개발 과정에서 필요 회로 구성에서 필요

• GPIO 번호는 27

디지털 입력 wPi 명령 사용

Practice

```
pi@raspberrypi: ~
```

```
pi@raspberrypi:~ $ gpio mode 2 in
pi@raspberrypi:~ $ gpio read 2
0
pi@raspberrypi:~ $ gpio read 2
1
pi@raspberrypi:~ $ gpio read 2
1
pi@raspberrypi:~ $ gpio read 2
0
pi@raspberrypi:~ $ gpio -g read 27
0
pi@raspberrypi:~ $ gpio -g read 27
1
pi@raspberrypi:~ $ gpio -g read 27
1
pi@raspberrypi:~ $ gpio -g read 27
0
pi@raspberrypi:~ $
```

- `gpio mode <pin> <mode>`

• 특정 GPIO 핀 동작 모드를 설정

• <`pin`>: wPi 핀 번호, '-g' 옵션과 함께 GPIO 핀 번호 사용 가능
 • <`mode`>: 특정 pin의 모드 (in, out, pwm, up, down, tri) 를 지정

- `gpio read <pin>`

• 특정 GPIO 핀 디지털 값을 읽음

• <`pin`>: wPi 핀 번호, '-g' 옵션과 함께 GPIO 핀 번호 사용 가능

• wPi 핀 번호 (2) 대신, '-g' 옵션과 함께 GPIO 핀 번호 (27)을 사용하여 디지털 값을 읽음

• 모든 wPi 모든 명령이 '-g' 옵션을 지원하는 것은 아님

• 주로 wPi 번호 체계를 사용하는 것이 좋음

- GPIO

번호는 27

27
2

wPi 핀
번호는 2

wPi SW 개발
과정에서 필요



• 물리적 핀
번호는 13

회로 구성
에서 필요

디지털 입력 wait for interrupts (wfi)

Practice

- `gpio mode <pin> <mode>`
 - 특정 GPIO 핀 동작 모드를 설정
 - <**pin**>: wPi 핀 번호, '-g' 옵션과 함께 GPIO 핀 번호 사용 가능
 - <**mode**>: 특정 pin의 모드 (in, out, pwm, up, down, tri) 를 지정

- `gpio wfi <pin> <mode>`
 - 특정 GPIO 핀 상태 변경을 기다림
 - <`pin`>: wPi 핀 번호
 - <`mode`>: 상승 에지 (`rising`), 하강 에지 (`falling`), 상승/하강 에지 (`both`) 인터럽트가 발생 할 때까지 대기

- 이 경우 버튼이 눌리자 마자 (상승 에지)에서 제어 반화

- 이 경우 버튼에서 손을 떼자 마자 (하강 에지)에서 제어 반환

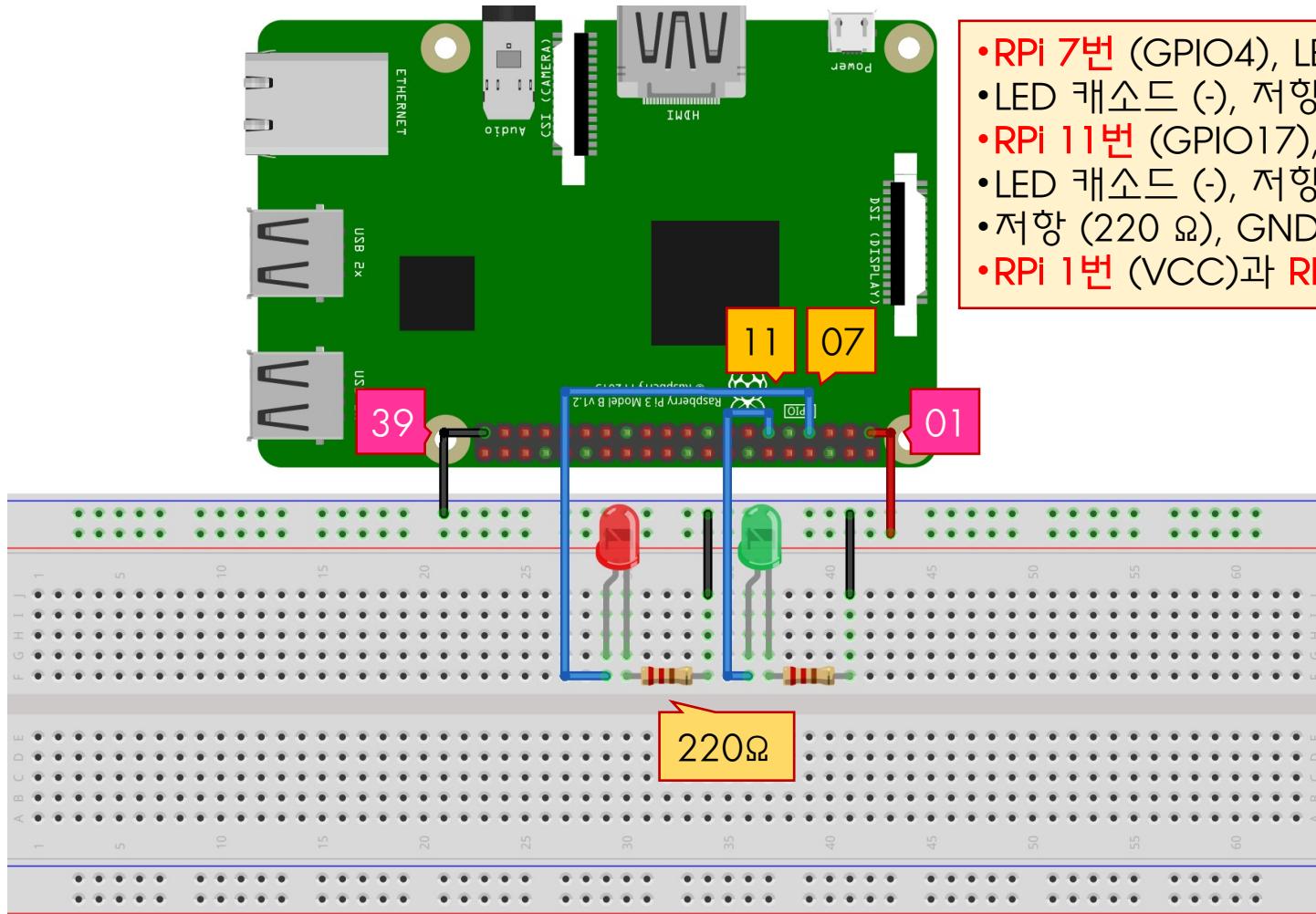
디지털 출력

WiringPi

디지털 출력 회로 구성

Practice

■ wPi를 이용한 디지털 출력

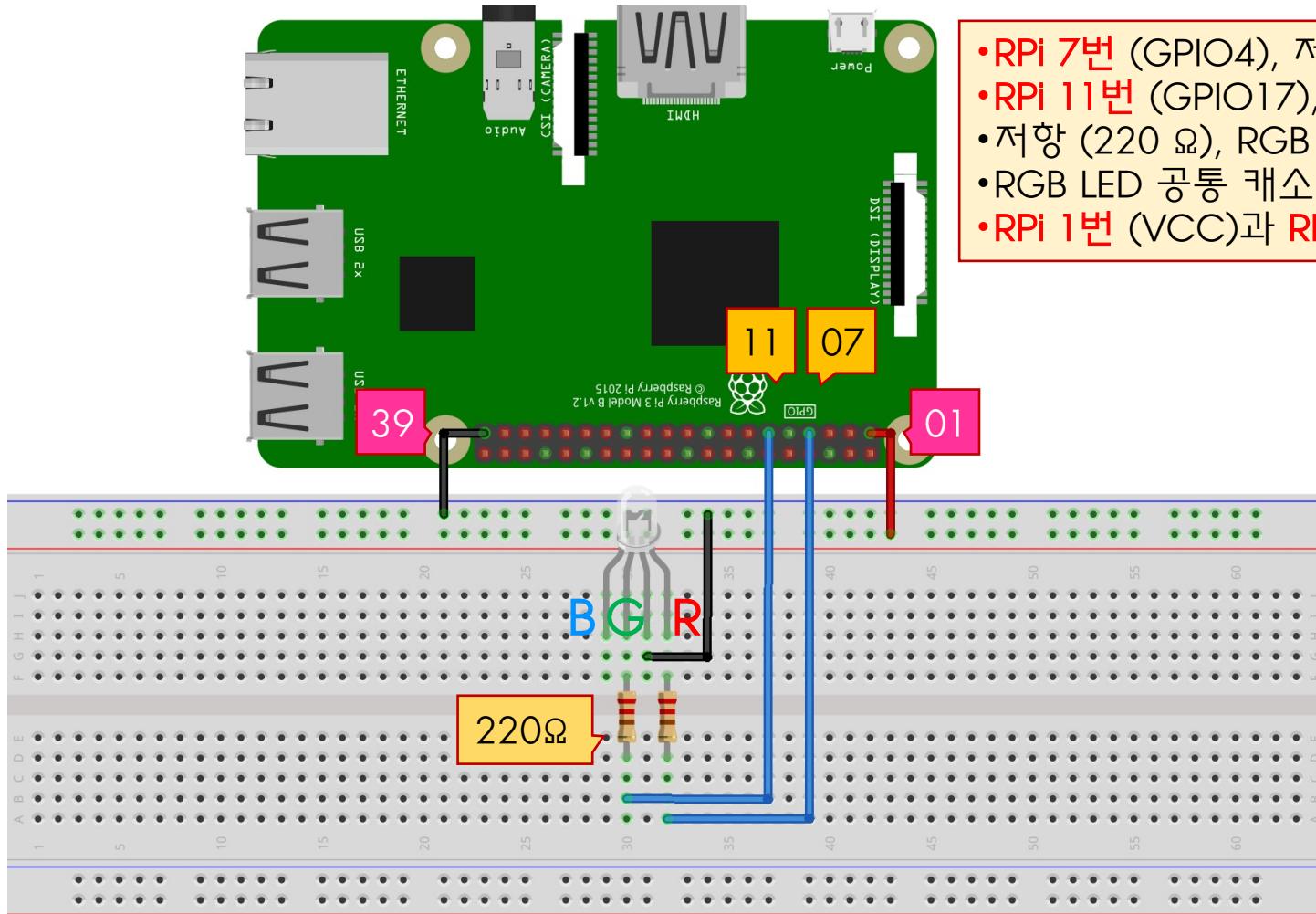


- RPi 7번 (GPIO4), LED 애노드 (+) 연결
- LED 캐소드 (-), 저항 (220 Ύ) 연결
- RPi 11번 (GPIO17), LED 애노드 (+) 연결
- LED 캐소드 (-), 저항 (220 Ύ) 연결
- 저항 (220 Ύ), GND 연결
- RPi 1번 (VCC)과 RPi 39번 (GND) 연결

디지털 출력 회로 구성

Practice

■ wPi를 이용한 디지털 출력



- RPi 7번 (GPIO4), 저항 (220 Ω) 연결
- RPi 11번 (GPIO17), 저항 (220 Ω) 연결
- 저항 (220 Ω), RGB LED R/G/B 핀 연결
- RGB LED 공통 캐소드, RPi GND 연결
- RPi 1번 (VCC)과 RPi 39번 (GND) 활용

디지털 입력 wPi 핀 확인

■ 물리적 physical 핀 (11), GPIO (17), wPi (0)

```
pi@raspberrypi: ~
pi@raspberrypi: ~ $ gpio readall
+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name |
+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     | 3.3v |     |   |     |   |     |     |
| 2   | 8   | SDA.1 | IN  | 1  | 3   | 4   | IN  | 5v  |
| 3   | 9   | SCL.1 | IN  | 1  | 5   | 6   | 0v  |
| 4   | 7   | GPIO. 7 | IN  | 1  | 7   | 8   | 0   |
|       |     | 0v    |     |   | 9   | 10  | 1   |
| 17  | 0   | GPIO. 0 | IN  | 0  | 11  | 12  | 0   |
|       |     | 2v    |     |   | 13  | 14  | 1   |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 9   | 13  | MISO | IN  | 0  | 29  | 30  | IN  | 6   |
| 11  | 14  | MOSI | IN  | 0  | 31  | 32  | 0   | 10  |
| 0   | 30  | SCLK | IN  | 1  | 33  | 34  |     | 11  |
| 5   | 21  | GPIO.21| IN  | 1  | 35  | 36  | 0   | 26  |
| 6   | 22  | GPIO.22| IN  | 1  | 37  | 38  | 0   | 27  |
| 13  | 23  | GPIO.23| IN  | 0  | 39  | 40  | IN  | 28  |
| 19  | 24  | GPIO.24| IN  | 0  |       |     | IN  | 29  |
| 26  | 25  | GPIO.25| IN  | 0  |       |     | IN  | 21  |
|       |     | 0v    |     |   |       |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+
| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name |
+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     | 3.3v |     |   |     |   |     |     |
| 5   | 05  | GPIO03 | SCL1 | 5v | 05  | 07  | 5v  |
|       |     | I2C   |     |   |     | 09  | 0v  |
| 7   | 07  | GPIO04 | GCLK | 0v | 09  | Ground |     |
|       |     |        |     |   |     |     |     |
| 11  | 11  | GPIO17 | GEN0 | RxD | 11  | 13  | 0v  |
|       |     |        |     |   |     | 13  | 0v  |
|       |     |        |     |   |     |     |     |
|       |     | 0v    |     |   |     |     |     |
|       |     | 2v    |     |   |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 27  | 2  | GPIO. 2 | IN  | 0  | 29  | 30  | IN  | 6   |
|       |     | 2v    |     |   | 31  | 32  | 0   | 10  |
|       |     | 0v    |     |   | 33  | 34  |     | 11  |
|       |     | 0v    |     |   | 35  | 36  | 0   | 26  |
|       |     | 0v    |     |   | 37  | 38  | 0   | 27  |
|       |     | 0v    |     |   | 39  | 40  | IN  | 28  |
|       |     | 2v    |     |   |       |     | IN  | 29  |
|       |     | 0v    |     |   |       |     |     | 21  |
+-----+-----+-----+-----+-----+-----+-----+-----+
pi@raspberrypi: ~ $
```

디지털 입력 wPi 명령 사용

Practice

pi@raspberrypi: ~

```
pi@raspberrypi:~ $ gpio mode 7 out  
pi@raspberrypi:~ $  
pi@raspberrypi:~ $  
pi@raspberrypi:~ $ gpio mode 0 out  
pi@raspberrypi:~ $  
pi@raspberrypi:~ $  
pi@raspberrypi:~ $ gpio write 7 1  
pi@raspberrypi:~ $  
pi@raspberrypi:~ $  
pi@raspberrypi:~ $ gpio write 7 0  
pi@raspberrypi:~ $  
pi@raspberrypi:~ $  
pi@raspberrypi:~ $ gpio write 0 1  
pi@raspberrypi:~ $  
pi@raspberrypi:~ $  
pi@raspberrypi:~ $ gpio write 0 0  
pi@raspberrypi:~ $
```

- **gpio mode <pin> <mode>**
- 특정 GPIO 핀 동작 모드를 설정
- <pin>: wPi 핀 번호, '-g' 옵션과 함께 GPIO 핀 번호 사용 가능
- <mode>: 특정 pin의 모드 (in, out, pwm, up, down, tri) 를 지정

- **gpio write <pin> <value>**
- 특정 GPIO 핀 디지털 값을 쓰기
- <pin>: wPi 핀 번호
- <value>: 출력할 디지털 값으로 0 또는 1

wPi 프로그래밍

LED on/off

wRi 프로그래밍

LED 점등

아두이노 스케치 구조와 비교

Practice

```
#include <wiringPi.h>
#include <iostream>
using namespace std;
#define LED_GPIO 17
```

```
int main() {
    wiringPiSetupGpio();
```

```
cout << "Starting fast GPIO toggle on GPIO" << LED_GPIO << endl;
cout << "Press CTRL+C to quit..." << endl;
```

```
pinMode(LED_GPIO, OUTPUT);
```

```
while(1) {
```

```
    digitalWrite(LED_GPIO, HIGH);
//    for(int i=0; i<50; i++) { }
    delay(1000);
```

```
    digitalWrite(LED_GPIO, LOW);
//    for(int i=0; i<49; i++) { }
    delay(1000);
}
```

```
return 0;
}
```

• wPi 라이브러리 사용을 위한 헤더 파일 포함 선언

• GPIO (17), 물리 번호 (11) 핀 사용 선언

• wiringPiSetup() 은 wPi를 초기화, wPi 번호 사용

• wiringPiSetupGpio() 은 wPi 초기화, wPi 번호가 아닌 GPIO 번호 사용

• wiringPiSetupPhys() 은 wPi 초기화, wPi 번호가 아닌 물리 번호 사용

BCM	wPi	Name	Mode	V	Physical
		3.3v			1 2
2	8	SDA.1	IN	1	3 4
3	9	SCL.1	IN	1	5 6
4	7	GPIO. 7	IN	1	7 8
		0v			9 10
17	0	GPIO. 0	IN	0	11 12
27	2	GPIO. 2	IN	0	13 14

• pinMode()는 디지털 입출력 핀의 모드 (출력 또는 입력) 설정

• 첫 번째 인수는 설정하고자 하는 핀 번호

• 두 번째 인수는 'OUTPUT' (출력) 또는 'INPUT' (입력)

• digitalWrite()는 특정 디지털 핀의 출력 값 (1 또는 0) 설정

• 첫 번째 인수는 설정하고자 하는 핀 번호

• 두 번째 인수는 'HIGH' (논리값 1) 또는 'LOW' (논리값 0)

• delay()는 인수로 주어진 milliseconds 동안 대기

• RPi에서는 delay()를 사용하면 블록킹을 해버리기 때문에 사용하지 않는 것을 권장

wRi 프로그래밍 LED 점등

교재 '익스플로링 RPi'의 pp. 272-273
wPi 함수 사용 실습 필요

Practice

```
pi@raspberrypi: ~/exploringrpib/chp06/wiringPi
pi@raspberrypi:~/exploringrpib/chp06/wiringPi $ ls
build          buttonLEDdb           clock.cpp      fasttoggle     info.cpp    servo
buttonLED      buttonLEDdebounce.cpp fadeLED       fasttoggle.cpp pwm        servo.cpp
buttonLED.cpp   clock              fadeLED.cpp   info          pwm.cpp
pi@raspberrypi:~/exploringrpib/chp06/wiringPi $ cp fasttoggle.cpp delaytoggle.cpp
pi@raspberrypi:~/exploringrpib/chp06/wiringPi $ ls
build          buttonLEDdb           clock.cpp      fadeLED.cpp   info        pwm.cpp
buttonLED      buttonLEDdebounce.cpp delaytoggle.cpp fasttoggle   info.cpp   servo
buttonLED.cpp   clock              fadeLED      fasttoggle.cpp pwm        servo.cpp
pi@raspberrypi:~/exploringrpib/chp06/wiringPi $ nano delaytoggle.cpp
pi@raspberrypi:~/exploringrpib/chp06/wiringPi $
pi@raspberrypi:~/exploringrpib/chp06/wiringPi $ •앞 소스 코드 추가
pi@raspberrypi:~/exploringrpib/chp06/wiringPi $
pi@raspberrypi:~/exploringrpib/chp06/wiringPi $ g++ delaytoggle.cpp -o delaytoggle [-lwiringPi]
pi@raspberrypi:~/exploringrpib/chp06/wiringPi $ ls
build          buttonLEDdb           clock.cpp      fadeLED      fasttoggle.cpp  pwm        servo.cpp
buttonLED      buttonLEDdebounce.cpp delaytoggle    fadeLED.cpp   info        pwm.cpp
buttonLED.cpp   clock              delaytoggle.cpp fasttoggle   info.cpp   servo
pi@raspberrypi:~/exploringrpib/chp06/wiringPi $ ./delaytoggle
Starting fast GPIO toggle on GPIO17
Press CTRL+C to quit...
^C
pi@raspberrypi:~/exploringrpib/chp06/wiringPi $
```

• wPi 라이브러리
위치 지정 컴파일

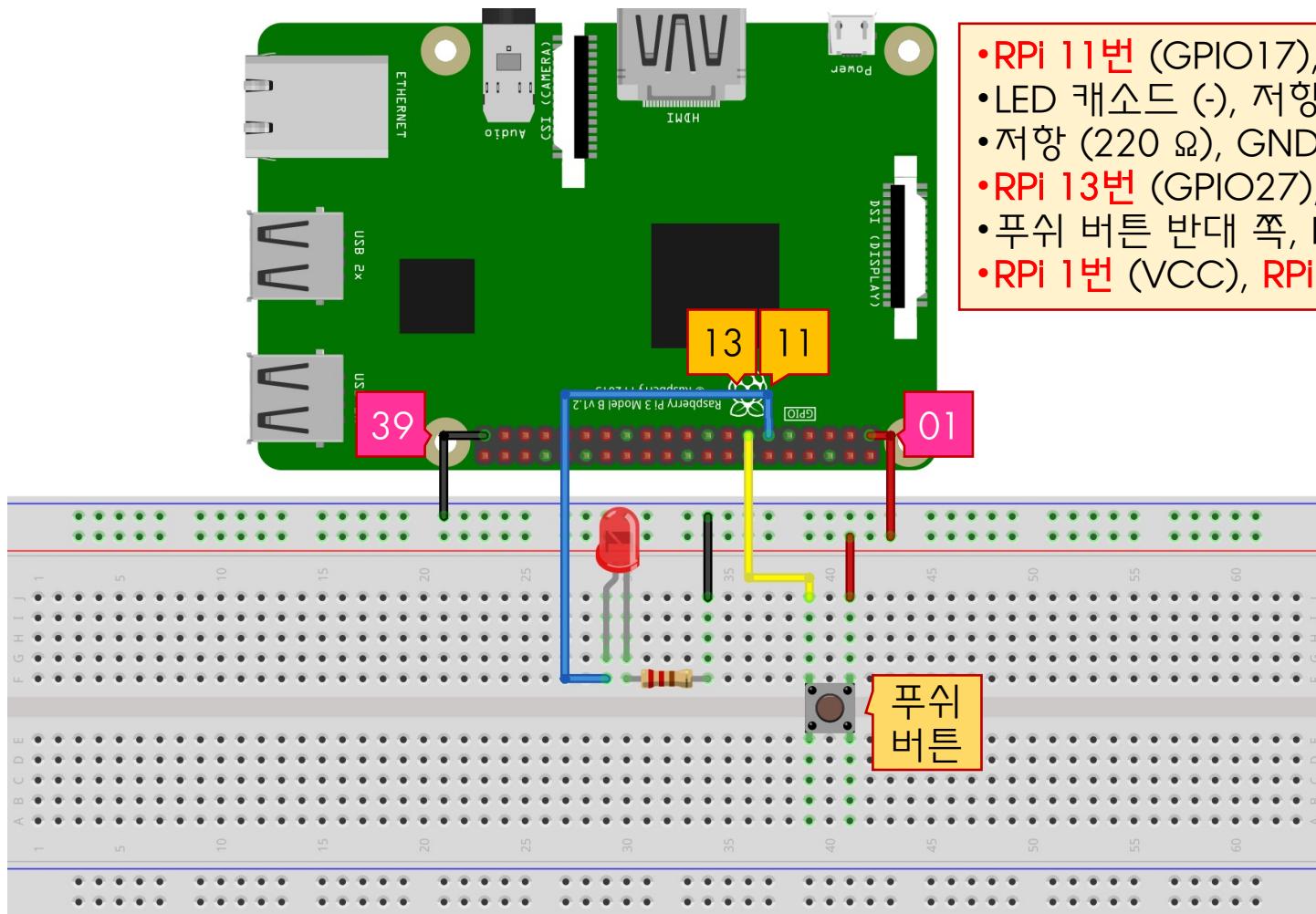
```
digitalWrite(LED_GPIO, HIGH);
// for(int i=0; i<50; i++) { }
delay(1000);

digitalWrite(LED_GPIO, LOW);
// for(int i=0; i<49; i++) { }
delay(1000);
```

wPi 프로그래밍

Interrupt service routine

■ wPi를 이용한 디지털 입출력



- RPi 11번 (GPIO17), LED 애노드 (+) 연결
- LED 캐소드 (-), 저항 (220 Ω) 연결
- 저항 (220 Ω), GND 연결
- RPi 13번 (GPIO27), 푸쉬 버튼 한쪽 연결
- 푸쉬 버튼 반대 쪽, RPi VCC 연결
- RPi 1번 (VCC), RPi 39번 (GND) 구성

wRi 프로그래밍 ISR

digitalRead(int pin) 과 비교!
<http://wiringpi.com/reference/core-functions/>

Practice

```
#include <iostream>
#include <wiringPi.h>
#include <unistd.h>
using namespace std;
#define LED_GPIO      17
#define BUTTON_GPIO   27
```

```
void lightLED(void){
    static int x = 1;
    digitalWrite(LED_GPIO, HIGH);
    cout << "Button pressed " << x++ << " times! LED on" << endl;
}
```

```
int main() {
    wiringPiSetupGpio();
    pinMode(LED_GPIO, OUTPUT);
    pinMode(BUTTON_GPIO, INPUT);
    digitalWrite (LED_GPIO, LOW);
    cout << "Press the button on GPIO "
        << BUTTON_GPIO << endl;
```

- GPIO (17), 물리 번호 (11) 핀을 LED 사용
- GPIO (27), 물리 번호 (13) 핀을 푸쉬 버튼 사용

- 인터럽트가 발생할 때 호출되는 함수 ISR (인터럽트 서비스 루틴)
- ISR이 다중 호출 될 때 상태를 저장하기 위해 **static** 변수 선언

- wiringPiISR()**는 특정 핀 인터럽트 발생을 처리할 ISR 등록
- 첫 번째 인수는 핀 번호
- 두 번째 인수는 인터럽트 이벤트 설정
INT_EDGE_RISING, INT_EDGE_FALLING, INT_EDGE_BOTH
- 세 번째 인수는 콜백 함수 즉 ISR의 주소

```
wiringPiISR(BUTTON_GPIO, INT_EDGE_RISING, &lightLED);
```

```
for(int i=10; i>0; i--){
    cout << "You have " << i << " seconds remaining..." << endl;
    sleep(1);
} return 0;
```

- sleep()**는 해당 스레드를 초단위로 대기
- 10초 동안 '버튼 입력'을 대기한 후 종료

wRi 프로그래밍 ISR

Practice

```
pi@raspberrypi: ~exploringrp1/chp06/wiringPi
pi@raspberrypi:~/exploringrp1/chp06/wiringPi $ ls
build          buttonLEDdb           clock.cpp        fadeLED      fasttoggle.cpp   pwm          servo.cpp
buttonLED      buttonLEDdebounce.cpp delaytoggle     fadeLED.cpp  info          pwm.cpp       servo
buttonLED.cpp   clock               delaytoggle.cpp fasttoggle    info.cpp      servo
pi@raspberrypi:~/exploringrp1/chp06/wiringPi $ ./buttonLED
Press the button on GPIO 27
You have 10 seconds remaining...
You have 9 seconds remaining...
You have 8 seconds remaining...
Button pressed 1 times! LED on
You have 7 seconds remaining...
Button pressed 2 times! LED on
You have 6 seconds remaining...
You have 5 seconds remaining...
Button pressed 3 times! LED on
Button pressed 4 times! LED on
Button pressed 5 times! LED on
Button pressed 6 times! LED on
Button pressed 7 times! LED on
You have 4 seconds remaining...
You have 3 seconds remaining...
You have 2 seconds remaining...
Button pressed 8 times! LED on
Button pressed 9 times! LED on
You have 1 seconds remaining...
pi@raspberrypi:~/exploringrp1/chp06/wiringPi $
```

- 버튼 (GPIO 27)이 눌릴 때마다 ISR이 호출되어 메시지 출력
- 버튼 (GPIO 27)이 눌릴 때마다 ISR이 호출되어 메시지 출력
- 기계적 채터링으로 인한 '스위치 바운싱' 문제
- 교재 p276 소프트웨어적 디바운스 방법 참조

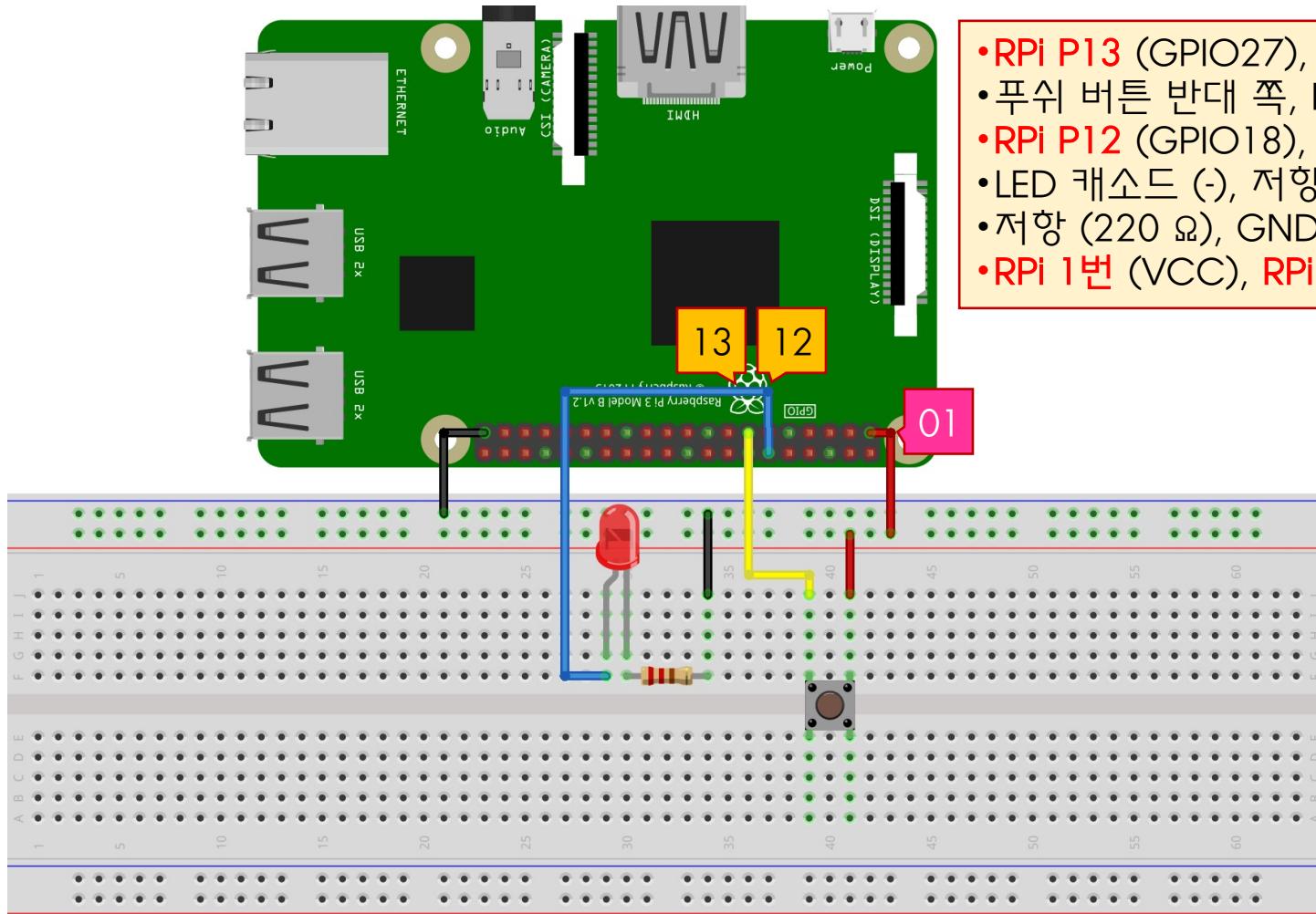
wPi 프로그래밍

Pulse width modulation

PWM LED 페이딩 회로 구성

Practice

■ wPi를 이용한 PWM 신호 출력



- RPi P13 (GPIO27), 푸쉬 버튼 한쪽 연결
- 푸쉬 버튼 반대 쪽, RPi VCC (P1) 연결
- RPi P12 (GPIO18), LED 애노드 (+) 연결
- LED 캐소드 (-), 저항 (220 Ω) 연결
- 저항 (220 Ω), GND 연결
- RPi 1번 (VCC), RPi 39번 (GND) 활용

PWM LED 페이딩 회로 구성

Practice

```
#include <iostream>
#include <wiringPi.h>
#include <unistd.h>
using namespace std;
#define PWM_LED
#define BUTTON_GPIO
bool running = true;
```

```
int main() {
    wiringPiSetupGpio();
    pinMode(PWM_LED, PWM_OUTPUT);
    pinMode(BUTTON_GPIO, INPUT);
    wiringPiISR(BUTTON_GPIO,
        INT_EDGE_RISING, &buttonPress);
    pwmSetRange(1000);
```

```
void buttonPress(void) {
    cout << "Button was pressed - "
        "start graceful end." << endl;
    running = false;
}
```

- 버튼이 눌렸을 때 호출되는 ISR

18

27

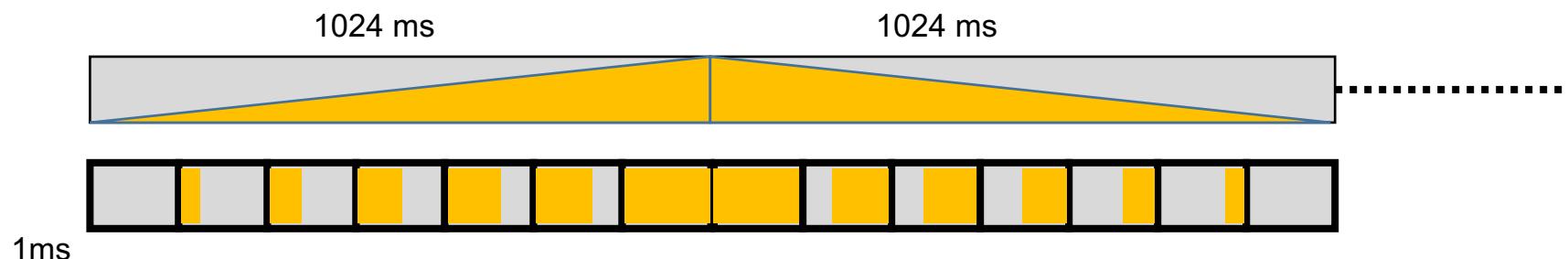
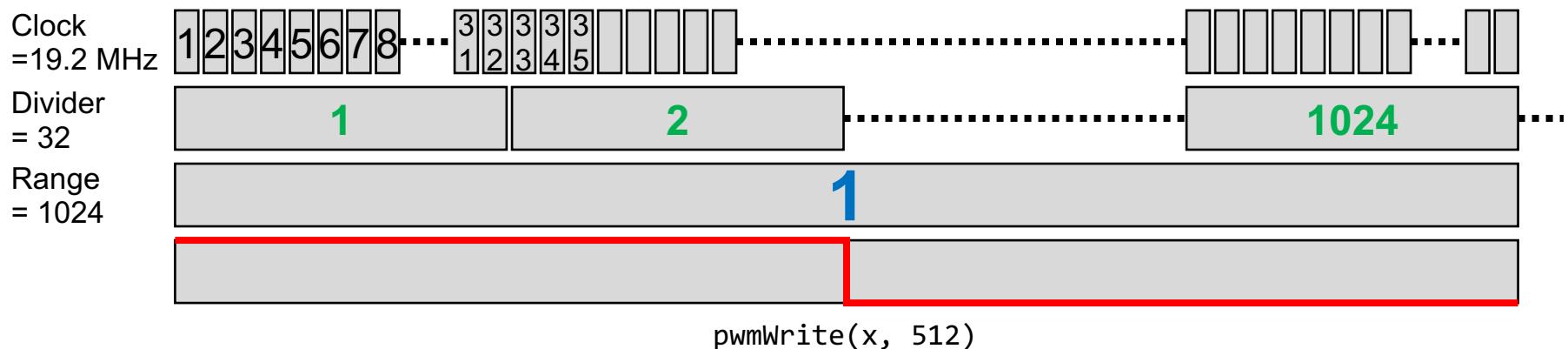
- GPIO (27), 물리 번호 (13) 핀을 푸쉬 버튼 사용
- GPIO (18), 물리 번호 (12) 핀을 LED 사용
- 푸쉬 버튼이 눌렸을 때 running 값 설정

- **pwmSetRange()** 는 PWM 범위 레지스터 설정
2 - 4,095 사이의 값, 기본 값은 1024
- **pwmSetClock()** 은 PWM 클럭 제수 (divider)
기본 값은 32
- PWM 클럭 주파수는 = $19.2 \text{ Mhz} / (\text{제수} * \text{범위})$
- 만약 범위가 1000,이고 제수가 32라면 600 Hz PWM

```
cout << "Fading the LED in/out until the button is pressed" << endl;
while(running) {
    for(int i=1; i<1000; i++) {
        pwmWrite(PWM_LED, i); usleep(1000);
    }
    for(int i=999; i>=0; i--) {
        pwmWrite(PWM_LED, i); usleep(1000);
    }
}
cout << "LED Off: Program has finished gracefully!" << endl;
return 0;
```

- **pwmWrite()** 는 **pwmSetRange()**로 설정한 범위
내의 값을 듀티비로 설정
- 예로서 범위가 1000일 때 **pwmWrite()** 설정 값이
500이면 1000/500 으로 50% 듀티비를 가짐

PWM LED 페이딩 해석



```
for (i=0; i<1024; i++) {
    pwmWrite(x, i);
    usleep(1000);
}
```

```
for (i=1023; i>=0; i--) {
    pwmWrite(x, i);
    usleep(1000);
}
```

interval = i
range = r
step = s

$$i * r/s = 1020$$

PWM LED 페이딩 회로 구성

```
pi@raspberrypi: ~/exploringrp1/chp06/wiringPi
pi@raspberrypi:~/exploringrp1/chp06/wiringPi $ ls
build          buttonLEDdb           clock.cpp        fadeLED      fasttoggle.cpp   pwm       servo.cpp
buttonLED      buttonLEDbounce.cpp  delaytoggle    fadeLED.cpp  info          pwm.cpp   servo
buttonLED.cpp   clock              delaytoggle.cpp  fasttoggle  info.cpp     servo
pi@raspberrypi:~/exploringrp1/chp06/wiringPi $ sudo ./fadeLED
Fading the LED in/out until the button is pressed
Button was pressed -- start graceful end.
Button was pressed -- start graceful end.
Button was pressed -- start graceful end.
LED Off: Program has finished gracefully!
pi@raspberrypi:~/exploringrp1/chp06/wiringPi $
```

•root 권한으로 실행 필요

•스위치 바운스 문제 해결 필요

•교재 276 페이지 참조¹

Summary

- RPi GPIO
- WiringPi library
- wPi: digital input with
- wPi: digital output with wPi
- wPi: interrupt service routines
- wPi: pulse width modulation

Thank you

Questions?

Contact: eclass.sch.ac.kr
(순천향대학교 학습플랫폼 LMS)