

1

라즈베리파이 GPIO 입 · 출력



1) 라즈베리파이 GPIO 연결 개요

라즈베리파이 시스템의 구조 학습

- BCM 2835 processor, 3.3V (3V3) power on pins , SD card is like the hard drive...

리눅스(Linux)에 관한 학습

- The root directory : /, the super user designation : sudo, change permissions : chmod 777 filename...

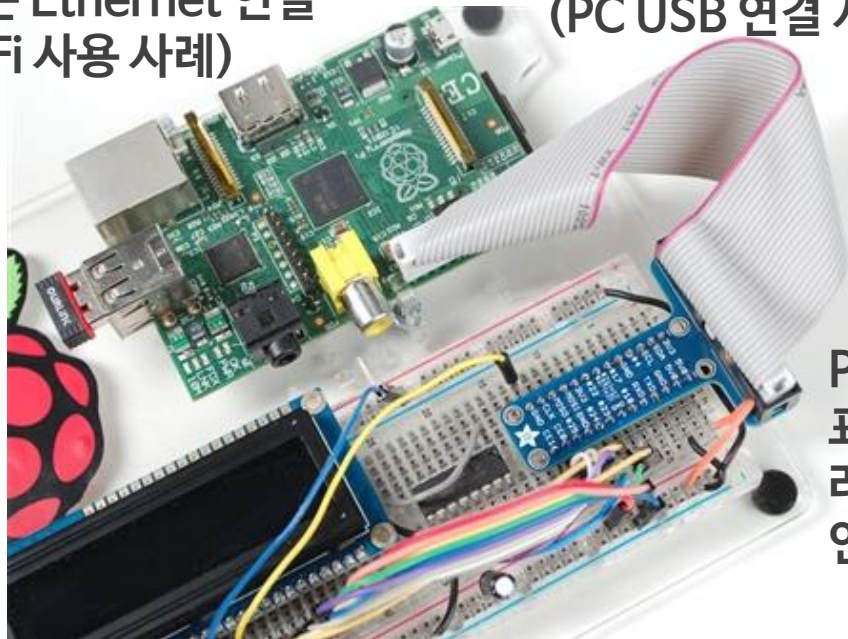
라즈베리파이 네트워킹에 관한 설정 학습

- ssh id@“네트워크 IP address”
- ifconfig
- /etc/network/interfaces

1) 라즈베리파이 GPIO 연결 개요

Wi-Fi 또는 Ethernet 연결
(Wi-Fi 사용 사례)

전원어댑터 연결
(PC USB 연결 시, 전력이 부족할 수 있음)



Pin1은 빨강색으로
표시되어 있으므로
라즈베리파이의 1번 Pin에
연결하여야 함

※ 라즈베리파이 GPIO 포트에 케이블을 연결하는 데 있어, Pin 번호가
올바르게 접속되어야 하므로 주의하여야 함

2

라즈베리파이의 GPIO 활용



1) 라즈베리파이 GPIO Pins

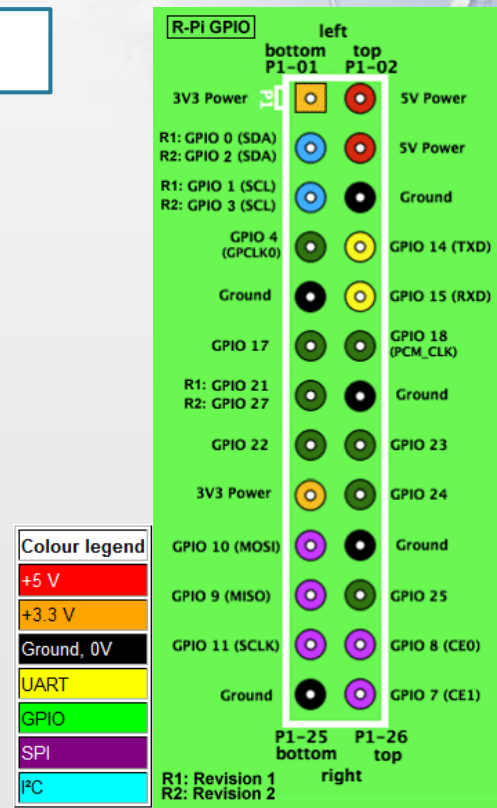
17개의 GPIO Pin을 활용함

- 각 Pin은 대부분 Alternated Function이 가능함

UART를
위한
2개 Pin

I2C를
위한
2개 Pin

SPI를
위한
6개 Pin



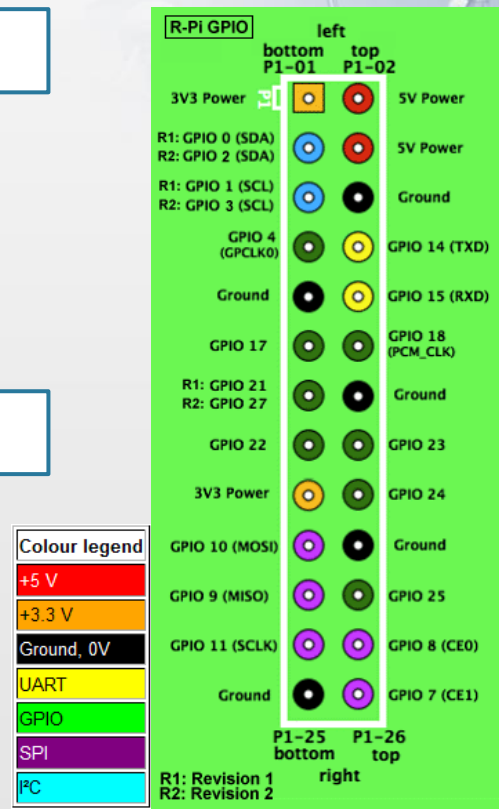
1) 라즈베리파이 GPIO Pins

17개의 Pin은 모두 GPIO로 사용할 수 있음

- 입 · 출력이 가능함
- 모든 Pin은 인터럽트를 지원함
- 각 Pin에 대해서 Pull-up 및 Pull-down 저항을 지원함

라즈베리에서 각 Pin은 3.3V로 동작함

- 아두이노에서 Pin은 5V로 동작함
- 최대 허용되는 전류량은 50mA임

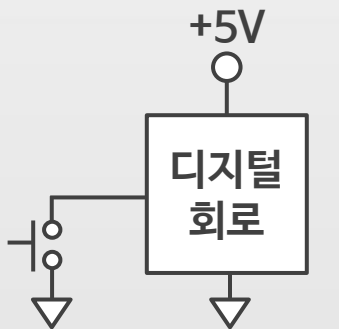


1) 라즈베리파이 GPIO Pins

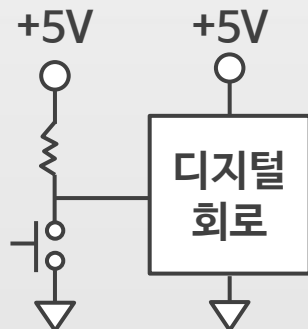
Pull-up 저항

Pull-up 저항

논리적으로 High Level 상태를 유지하기 위해 신호의
입 · 출력 단자와 전원(Vcc) 단자 사이에 접속하는 저항



스위치 입력



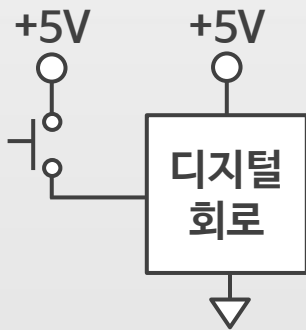
Pull-up 저항 사용

1) 라즈베리파이 GPIO Pins

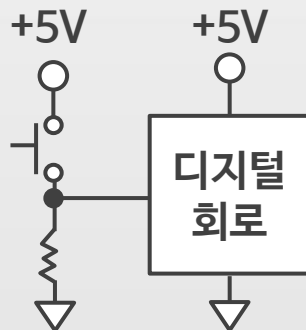
Pull-down 저항

Pull-down 저항

논리적으로 Low Level 상태를 유지하기 위해서 신호의
입 · 출력 단자와 접지 단자 사이에 접속하는 저항



스위치 입력



Pull-down 저항 사용

1) 라즈베리파이 GPIO Pins

 리눅스에서 각 GPIO Pin에 데이터를 읽고 (Read) 쓰는 (Write) 방법

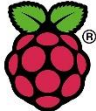
1 파일 시스템을 통한 File-type Access 방식

2 SoC 메모리 레이아웃에 기반한 GPIO 주변기기의 메모리 주소에 데이터를 Read/Write하는 방식

			P1		
<50mA	3V3		1 2	5V	
BCM GPIO00/02	SDA0/1	8	3 4	5V	
BCM GPIO01/03	SCL0/1	9	5 6	GND	
BCM GPIO04		7	7 8	15 TX	BCM GPIO14
	GND		9 10	16 RX	BCM GPIO15
BCM GPIO17		0	11 12	1 PWM0	BCM GPIO18
BCM GPIO21/27		2	13 14	GND	
BCM GPIO22		3	15 16	4	BCM GPIO23
<50mA	3v3		17 18	5	BCM GPIO24
BCM GPIO10	SPIMOSI	12	19 20	GND	
BCM GPIO9	SPIMOSO	13	21 22	6	BCM GPIO25
BCM GPIO11	SPI SCLK	14	23 24	10 SPI CE0 N	BCM GPIO08
	GND		25 26	11 SPI CE1 N	BCM GPIO07
			P5		
<50mA	3V3		2 1	5V	
BCM GPIO29	SCL0	18	4 3	17 SDA0	BCM GPIO28
BCM GPIO31		20	6 5	19	BCM GPIO30
	GND		8 7	GND	


리눅스에서 각 GPIO Pin에 데이터를 읽고 (Read) 쓰는 (Write) 방법






Raspberry Pi

GPIO Cheat Sheet



CYNTech
COMPONENTS
F. FETCO MECHANICAL SPECIALISTS



PC

A low-speed interface used to communicate with multiple simple devices and sensors via a two-wire interface.

CLK

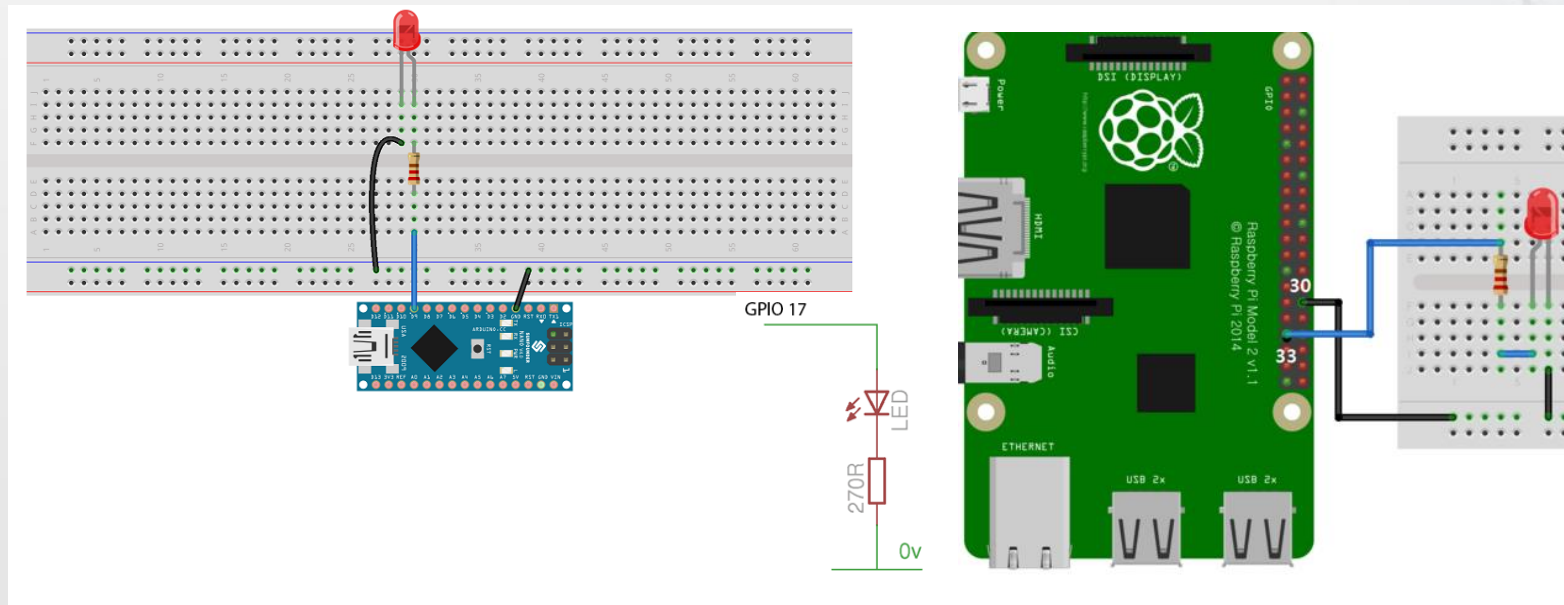
Inter-Integrated Circuit (I2C) is a serial bus interface which supports multiple devices and only requires two wires for communication (no separate clock or device select needed). It is, however, limited to relatively low speeds (usually 10-1000kb/s).

GPIO

2) 라즈베리파이 GPIO Pins에 LED 연결

📖 라즈베리파이 GPIO Pin에 LED를 연결하고 구동시키는 예제

Power On 시, Input으로 설정되기 때문에 초기에는 **LED Off** 상태임



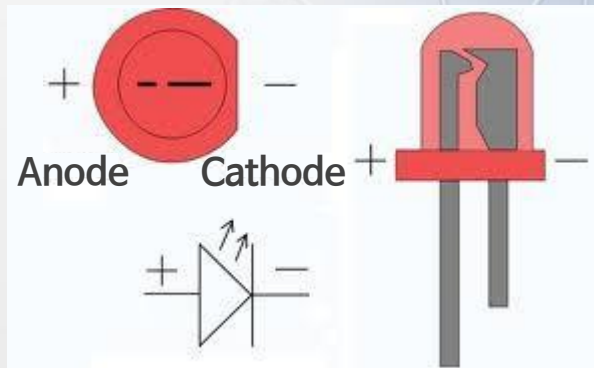
2) 라즈베리파이 GPIO Pins에 LED 연결

LED 연결방법

저항 220~330옴을 라즈베리파이의 P1 헤더핀의 16번, 18번Pin에 연결함

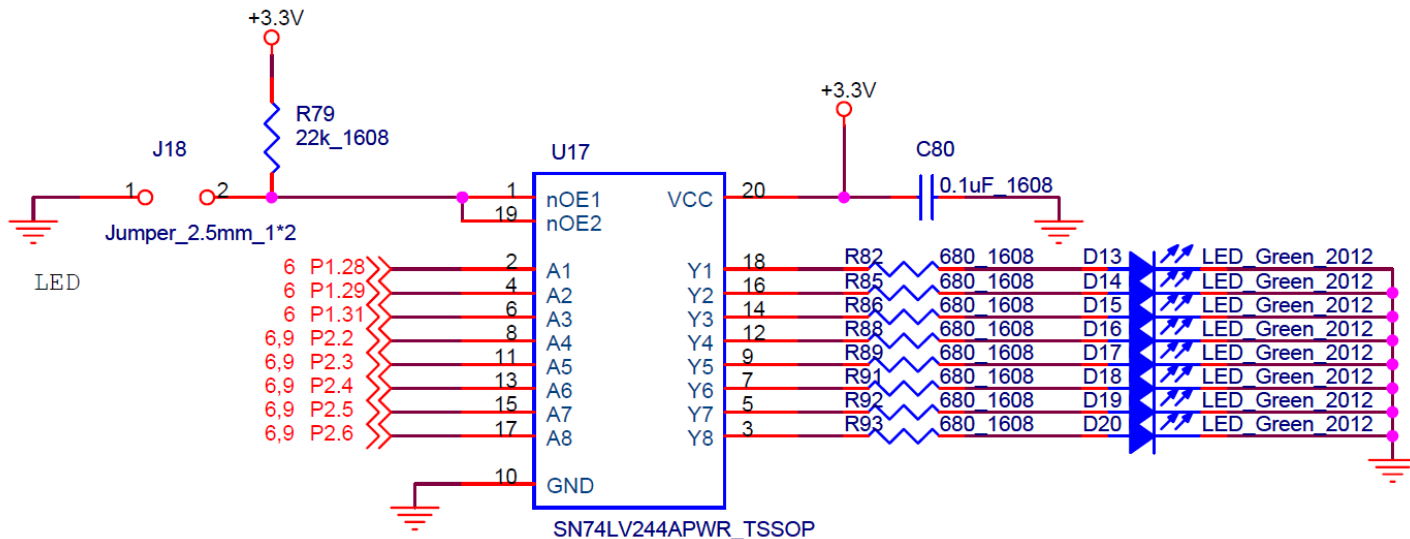
GPIO의 Output 값을
High로 설정 시 LED가 **On**, Low로 설정 시 **Off**임

➡ LED는 다리가 짧은 쪽이 **Cathode -**,
긴 쪽이 **Anode +**임



2) 라즈베리파이 GPIO Pins에 LED 연결

LED 연결방법

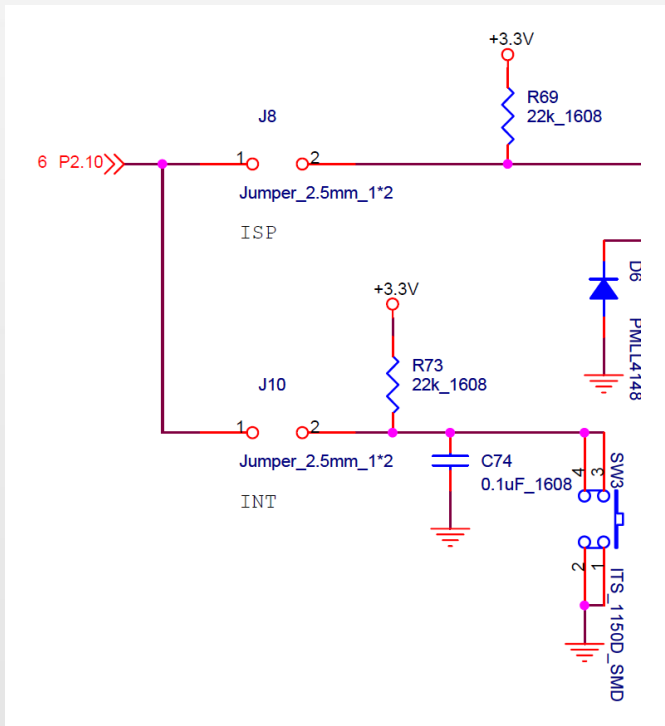


상태 표시를 위한 LED는 포트1의 28, 29, 31번 핀과 포트2의 2~6번 핀이 연결되어 있다고 가정할 때, 해당 핀에 **1**을 출력하면 LED가 켜짐

2) 라즈베리파이 GPIO Pins에 LED 연결



LED 연결방법



포트2의 10번 핀에
스위치3이 연결되어
있다고 가정할 때
스위치를 눌렀을 경우,
입력 핀에 **0**이 입력됨

2) 라즈베리파이 GPIO Pins에 LED 연결

파일 시스템을 사용한 File Type Access 방식의 예제

Sudo 명령어 사용

Shell script 사용

▪ sudo ./blink.sh

```
#!/bin/sh
echo 17 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio17/direction
while true
do
    echo 1 > /sys/class/gpio/gpio17/value
    sleep 1
    echo 0 > /sys/class/gpio/gpio17/value
    sleep 1
done
```



명령어 (echo 17 > /sys/class/gpio/unexport)를 통해 17번 Pin을 사용할 수 있도록 함

2) 라즈베리파이 GPIO Pins에 LED 연결

파일 시스템을 사용한 File Type Access 방식의 예제

Sudo 명령어 사용

Shell Script 사용

- **nano blink.sh** 명령어를 사용하여 LED가 깜박거리도록 하는 스크립트를 수행함
- 저장은 **Ctrl-w**, 종료는 **Ctrl-x**임
- blink.sh 파일의 권한(Permission)을 **chmod 755 blink.sh**로 변경함
- Blink.sh를 실행함

```
sudo ./blink.sh
```

➡ blink.sh 파일이 저장되어 있는 디렉토리에서 실행하여야 함

- 스크립트가 무한히 수행됨으로써 LED가 무한히 깜박거리는 상황을 제거하기 위해서 **Ctrl-c** 버튼을 눌러 종료할 수 있음

2) 라즈베리파이 GPIO Pins에 LED 연결

파일 시스템을 사용한 File Type Access 방식의 예제

모든 명령어는 한 번에 하나씩 수행하며, root 권한으로 수행할 필요가 있을 때에는 앞부분에 sudo 명령어를 사용할 수 있음

/sys/class/gpio 디렉토리에 있는 File 및 콘텐츠를 통해 GPIO를 다룰 수 있음

2) 라즈베리파이 GPIO Pins에 LED 연결

파일 시스템을 사용한 File Type Access 방식의 예제

리눅스 시스템에서 모든 것은 파일로 가능함

- `/dev/ttyUSB0`, `/sys/class/net/eth0/address`,
`/dev/mmcblk0p2`...

커널 모듈에서 Sysfs는 `/sys/class`에 위치한 Device에 대해서 접근하고자 할 때, 가상 파일 시스템을 제공함

- 시스템 레벨(Kernel Level)에서 다룰 수 있는 장치들에 대한 Interface 방법으로, User-space에서 수행할 수 있는 코드를 사용자들에게 제공함

2) 라즈베리파이 GPIO Pins에 LED 연결

장 · 단점

장점

- 사용자영역 (User-space)에서 기존에 파일을 다루는 Interface를 활용하여 디지털 GPIO 포트를 제어할 수 있음

단점

- 커널에 대하여 모드 전환 (Mode Switch)이 필요하며, 실제적으로 커널에서 장치의 입 · 출력을 다루어야 하고, 다시 User Mode로 모드 전환이 되어야 함
- 아두이노에서 제공하는 `digitalWrite()` / `digitalRead()` 보다는 속도가 느림

3) 라즈베리파이 Wiring Pi 라이브러리 활용법

BCM2835 라즈베리파이를 위해 C언어로 작성된
GPIO 접근 라이브러리

- GPIO에 할당된 메모리에 대하여 GPIO 주소에
데이터를 읽고/쓰기

라즈베리파이에 아두이노와 유사한 명령어로
IO 연산을 사용할 수 있도록 하는 Library

3) 라즈베리파이 Wiring Pi 라이브러리 활용법

특징

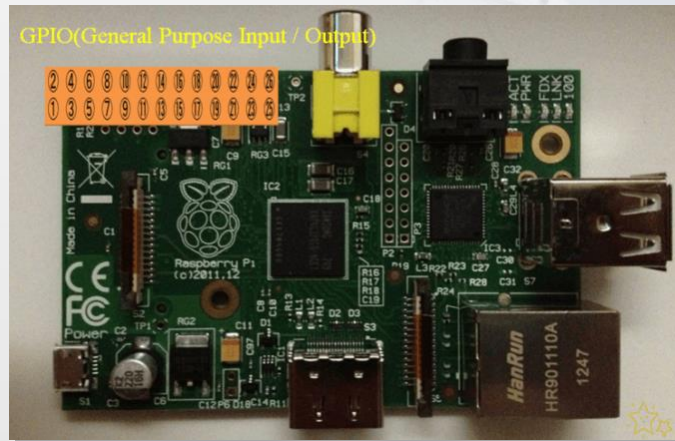
- 1 Command-line 기반 Utility 방식의 GPIO 사용
- 2 GPIO Pin에 대한 아날로그 Read/Write 지원

3) 라즈베리파이 Wiring Pi 라이브러리 활용법

설치방법

<http://wiringpi.com/download-and-install/>

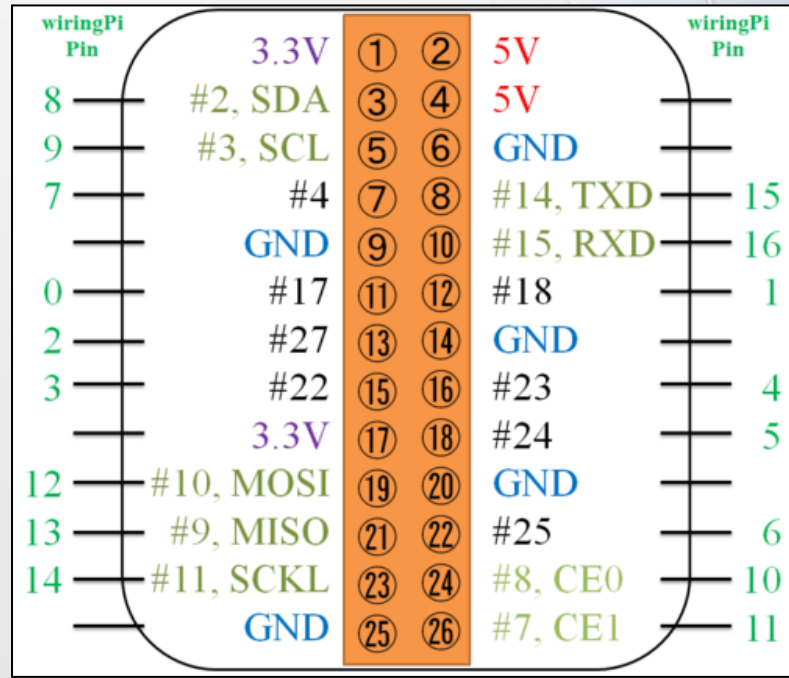
- `sudo apt-get install git-core`
- `sudo apt-get update`
- `sudo apt-get upgrade`
- `git clone git://git.drogon.net/wiringPi`
- `cd wiringPi`
- `git pull origin`
- `cd wiringPi`
- `./build`



3) 라즈베리파이 Wiring Pi 라이브러리 활용법

 강의에서 사용한 GPIO : 23번, 24번

wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin
-	-	3.3v	1 2	5v	-	-
8	R1:0/R2:2	SDA0	3 4	5v	-	-
9	R1:1/R2:3	SCL0	5 6	0v	-	-
7	4	GPIO7	7 8	TxD	14	15
-	-	0v	9 10	RxD	15	16
0	17	GPIO0	11 12	GPIO1	18	1
2	R1:21/R2:27	GPIO2	13 14	0v	-	-
3	22	GPIO3	15 16	GPIO4	23	4
-	-	3.3v	17 18	GPIO5	24	5
12	10	MOSI	19 20	0v	-	-
13	9	MISO	21 22	GPIO6	25	6
14	11	SCLK	23 24	CE0	8	10
-	-	0v	25 26	CE1	7	11
wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin



3) 라즈베리파이 Wiring Pi 라이브러리 활용법



Wiring Pi 라이브러리를 활용한 LED 점멸 예

<1/2>

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

// LED Pin - wiringPi pin 0 is BCM_GPIO 17.
#define LED 0

int main (void) {
    printf ("Raspberry Pi blink\n");
    wiringPiSetup ();           // ← note the setup method chosen
    pinMode (LED, OUTPUT);
```

3) 라즈베리파이 Wiring Pi 라이브러리 활용법



Wiring Pi 라이브러리를 활용한 LED 점멸 예

〈2/2〉

```
for (;;) {  
    digitalWrite (LED, HIGH) ; // On  
    delay (500) ;                // mS  
    digitalWrite (LED, LOW) ; // Off  
    delay (500) ;  
}  
return 0 ;  
}
```

3) 라즈베리파이 Wiring Pi 라이브러리 활용법

📖 Wiring Pi 라이브러리를 활용한 LED 점멸 예

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

#define LED1 4 // BCM_GPIO 23
#define LED2 5 // BCM_GPIO 24

int main (void)
{
    if (wiringPiSetup () == -1)
        return 1 ;

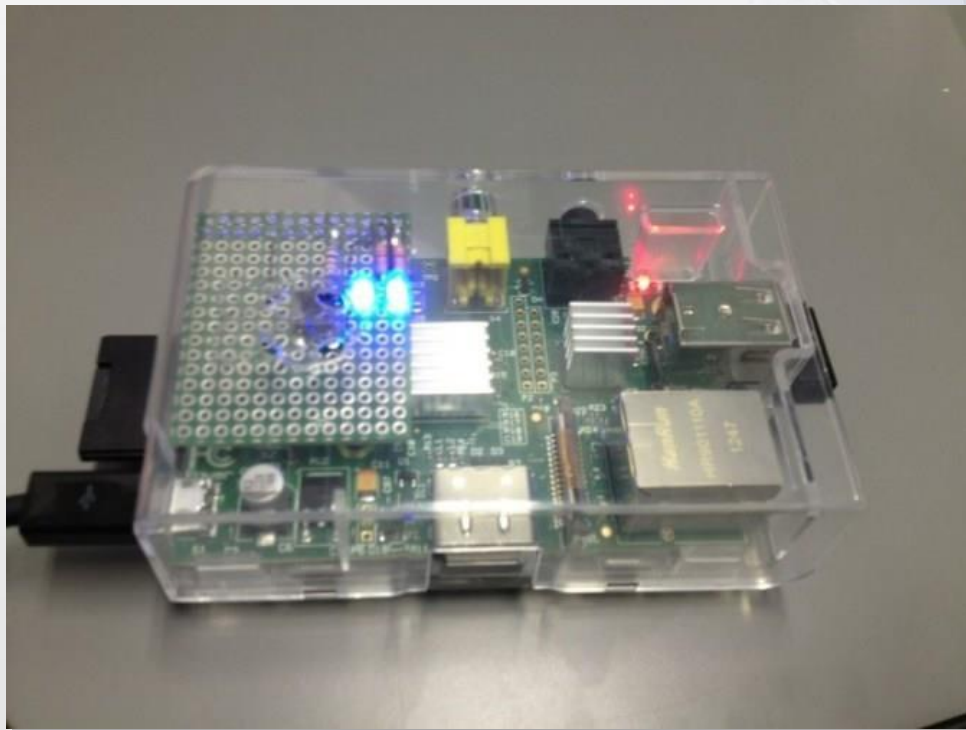
    pinMode (LED1, OUTPUT) ;
    pinMode (LED2, OUTPUT) ;

    for (;;)
    {
        digitalWrite (LED1, 1) ; // On
        digitalWrite (LED2, 1) ; // On

        delay (1000) ; // ms

        digitalWrite (LED1, 0) ; // Off
        digitalWrite (LED2, 0) ; // Off

        delay (1000) ;
    }
    return 0 ;
}
```



3) 라즈베리파이 Wiring Pi 라이브러리 활용법

Wiring Pi 라이브러리를 활용한 LED 점멸 예

컴파일
방법

```
gcc -o gpio-test1 gpio-test1.c -lwiringPi
```

프로그램
실행

```
sudo ./gpio-test1
```

3) 라즈베리파이 Wiring Pi 라이브러리 활용법



Wiring Pi 라이브러리를 활용한 LED 점멸 예

LED 점멸 프로그램을 컴파일하고 실행하는 방법

```
gcc -Wall -o blink blink.c -lwiringPi  ← Compile  
sudo ./blink                          ← Run
```

중단 없이 계속 수행될 경우, **Ctrl-c**를 사용함

Note : One of the four wiring *setup* functions **must** be called at the start of your program or your program will not work correctly

3) 라즈베리파이 Wiring Pi 라이브러리 활용법



라즈베리파이 GPIO Pins에 LED 연결 예제

〈1/3〉

```
#include <stdio.h> // Used for printf() statements
#include <wiringPi.h> // Include WiringPi library!
const int pwmPin = 18; // PWM LED - Broadcom pin 18, P1 pin 12
const int ledPin = 23; // Regular LED - Broadcom pin 23, P1 pin 16
const int butPin = 17; // Active-low button - Broadcom pin 17, P1 pin 11
const int pwmValue = 75; // Use this to set an LED brightness
int main(void)
{
    // Setup stuff:
    wiringPiSetupGpio(); // Initialize wiringPi -- using Broadcom pin number
```


3) 라즈베리파이 Wiring Pi 라이브러리 활용법



라즈베리파이 GPIO Pins에 LED 연결 예제

〈2/3〉

```
pinMode(pwmPin, PWM_OUTPUT); // Set PWM LED as PWM output
pinMode(ledPin, OUTPUT); // Set regular LED as output
pinMode(butPin, INPUT); // Set button as INPUT
pullUpDnControl(butPin, PUD_UP); // Enable pull-up resistor on button
while(1)
{
    if (digitalRead(butPin)) // Button is released if this returns 1
    {
        pwmWrite(pwmPin, pwmValue); // PWM LED at bright setting
        digitalWrite(ledPin, LOW); // Regular LED off
    }
}
```

3) 라즈베리파이 Wiring Pi 라이브러리 활용법



라즈베리파이 GPIO Pins에 LED 연결 예제

〈3/3〉

```
else // If digitalRead returns 0, button is pressed
{
    pwmWrite(pwmPin, 1024 - pwmValue); // PWM LED at dim setting
    // Do some blinking on the ledPin:
    digitalWrite(ledPin, HIGH); // Turn LED ON
    delay(75); // Wait 75ms
    digitalWrite(ledPin, LOW); // Turn LED OFF
    delay(75); // Wait 75ms again
}
}
return 0
}
```

3) 라즈베리파이 Wiring Pi 라이브러리 활용법

라즈베리파이의 초기화 방법 4가지

1 int wiringPiSetup (void);

2 int wiringPiSetupGpio (void);

3 int wiringPiSetupPhys (void);

4 int wiringPiSetupSys (void);

3

라즈베리파이의 직렬통신방법



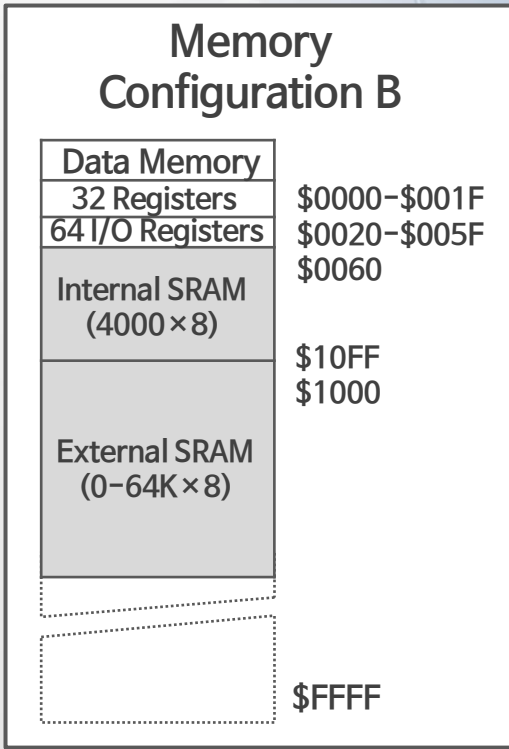
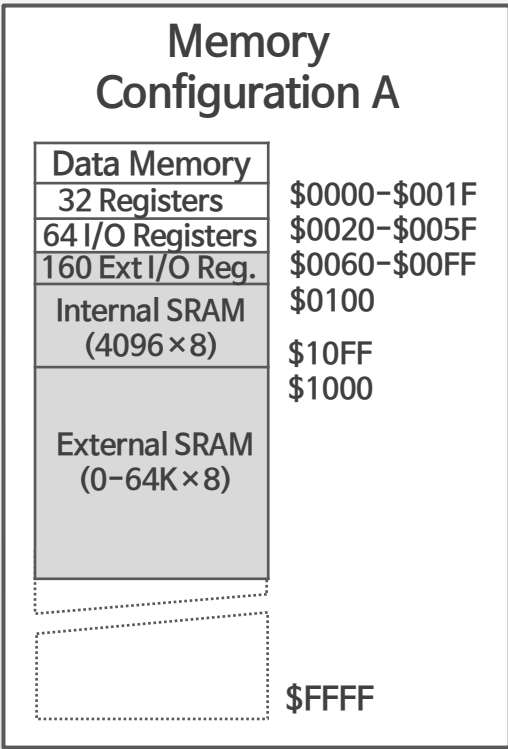
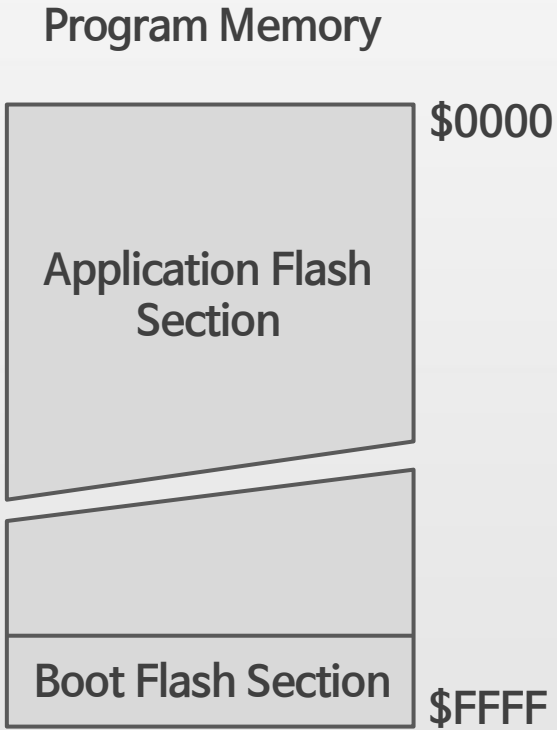
1) 입·출력 제어방식

프로그램드 I/O 방식

메모리 맵 접근방식

- 메모리뿐만 아니라 하드웨어 주변장치까지도 메모리 주소를 부여함으로써 일반적인 메모리의 읽고(Read)/쓰기(Write) 명령어를 통해 주변장치를 접근하는 방식

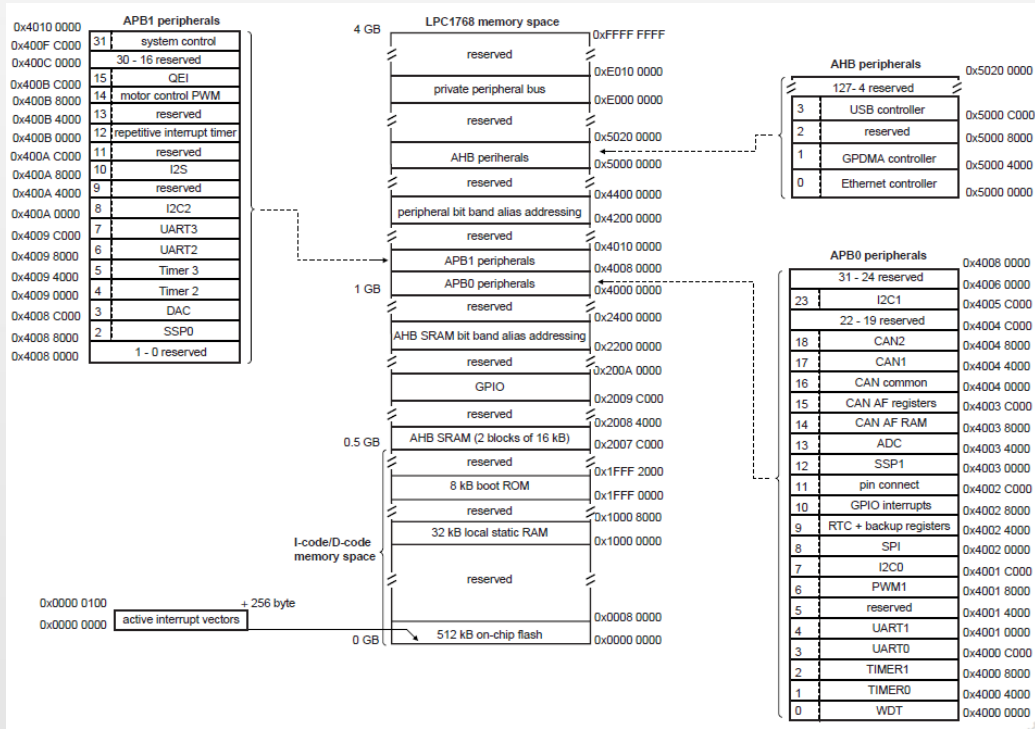
1) 입·출력 제어방식



1) 입 · 출력 제어방식



LPC1768(ARM Cortex-M3 프로세서) 메모리 맵



1) 입·출력 제어방식



LPC1768(ARM Cortex-M3 프로세서) 메모리 맵

Address range	General Use	Address range details and description	
0x0000 0000 to 0x1FFF FFFF	On-chip non-volatile memory	0x0000 0000 - 0x0007 FFFF	For devices with 512 kB of flash memory.
		0x0000 0000 - 0x0003 FFFF	For devices with 256 kB of flash memory.
		0x0000 0000 - 0x0001 FFFF	For devices with 128 kB of flash memory.
		0x0000 0000 - 0x0000 FFFF	For devices with 64 kB of flash memory.
		0x0000 0000 - 0x0000 7FFF	For devices with 32 kB of flash memory.
	On-chip SRAM	0x1000 0000 - 0x1000 7FFF	For devices with 32 kB of local SRAM.
		0x1000 0000 - 0x1000 3FFF	For devices with 16 kB of local SRAM.
		0x1000 0000 - 0x1000 1FFF	For devices with 8 kB of local SRAM.
	Boot ROM	0x1F00 0000 - 0x1FFF 1FFF	8 kB Boot ROM with flash services.
0x2000 0000 to 0x3FFF FFFF	On-chip SRAM (typically used for peripheral data)	0x2007 C000 - 0x2007 FFFF	AHB SRAM - bank 0 (16 kB), present on devices with 32 kB or 64 kB of total SRAM.
		0x2008 0000 to 0x2008 3FFF	AHB SRAM - bank 1 (16 kB), present on devices with 64 kB of total SRAM.
	GPIO	0x2009 C000 to 0x2009 FFFF	GPIO.
0x4000 0000 to 0x5FFF FFFF	APB Peripherals	0x4000 0000 to 0x4007 FFFF	APB0 Peripherals, up to 32 peripheral blocks, 16 kB each.
		0x4008 0000 to 0x400F FFFF	APB1 Peripherals, up to 32 peripheral blocks, 16 kB each.
	AHB peripherals	0x5000 0000 to 0x501F FFFF	DMA Controller, Ethernet interface, and USB interface.
0xE000 0000 to 0xE00F FFFF	Cortex-M3 Private Peripheral Bus	0xE000 0000 to 0xE00F FFFF	Cortex-M3 related functions, includes the NVIC and System Tick Timer.