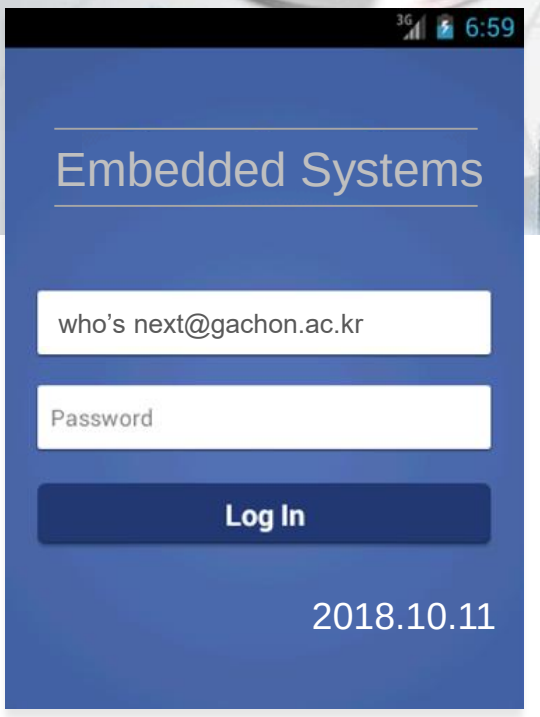


임베디드 시스템

컴퓨터공학과 이병문

강의일정

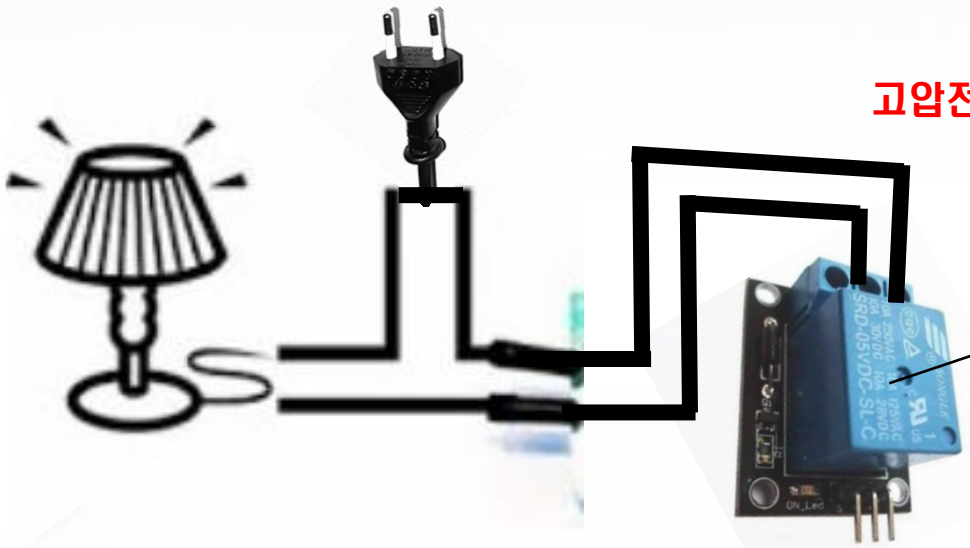
01	강의소개, 강의일정소개, 평가소개
02	사물인터넷, 라즈베리파이3 설치/구축
03	임베디드 액츄레이터/센서 제어 1
04	임베디드 액츄레이터/센서 제어 2
05	임베디드 액츄레이터/센서 제어 3
06	임베디드 액츄레이터/센서 제어 4
07	임베디드 액츄레이터/센서 제어 5
08	중간고사

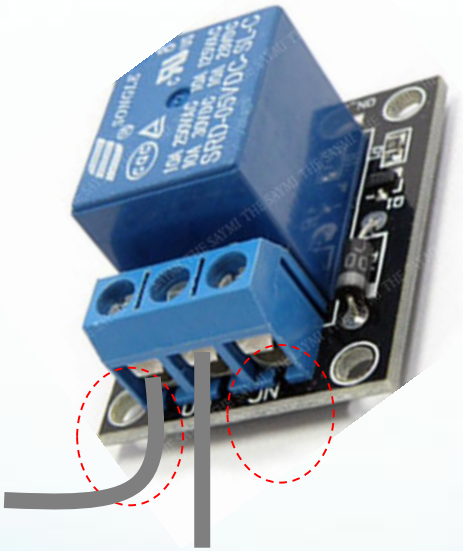
- Relay**제어**, **사운드센서제어**
- wiringPi GPIO API, **인터럽트제어**

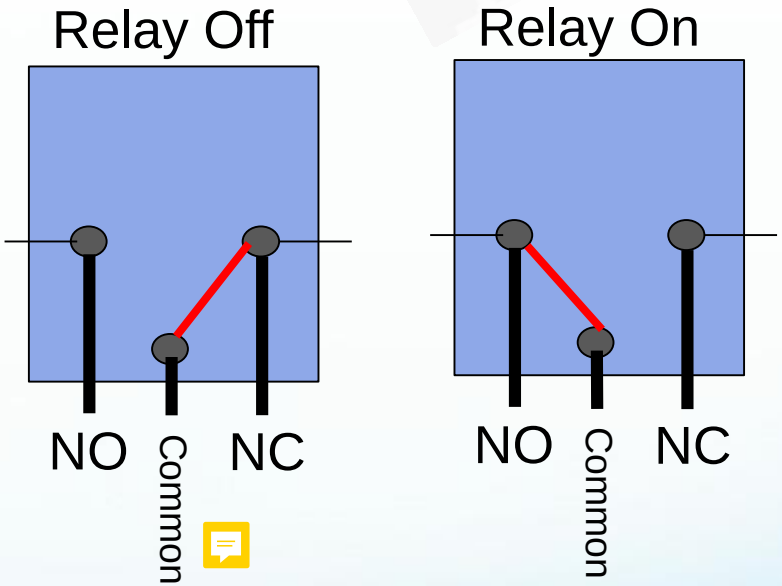
임베디드 액츄레이터/센서 제어

■ 하드웨어 구성

☑ Relay스위치 제어



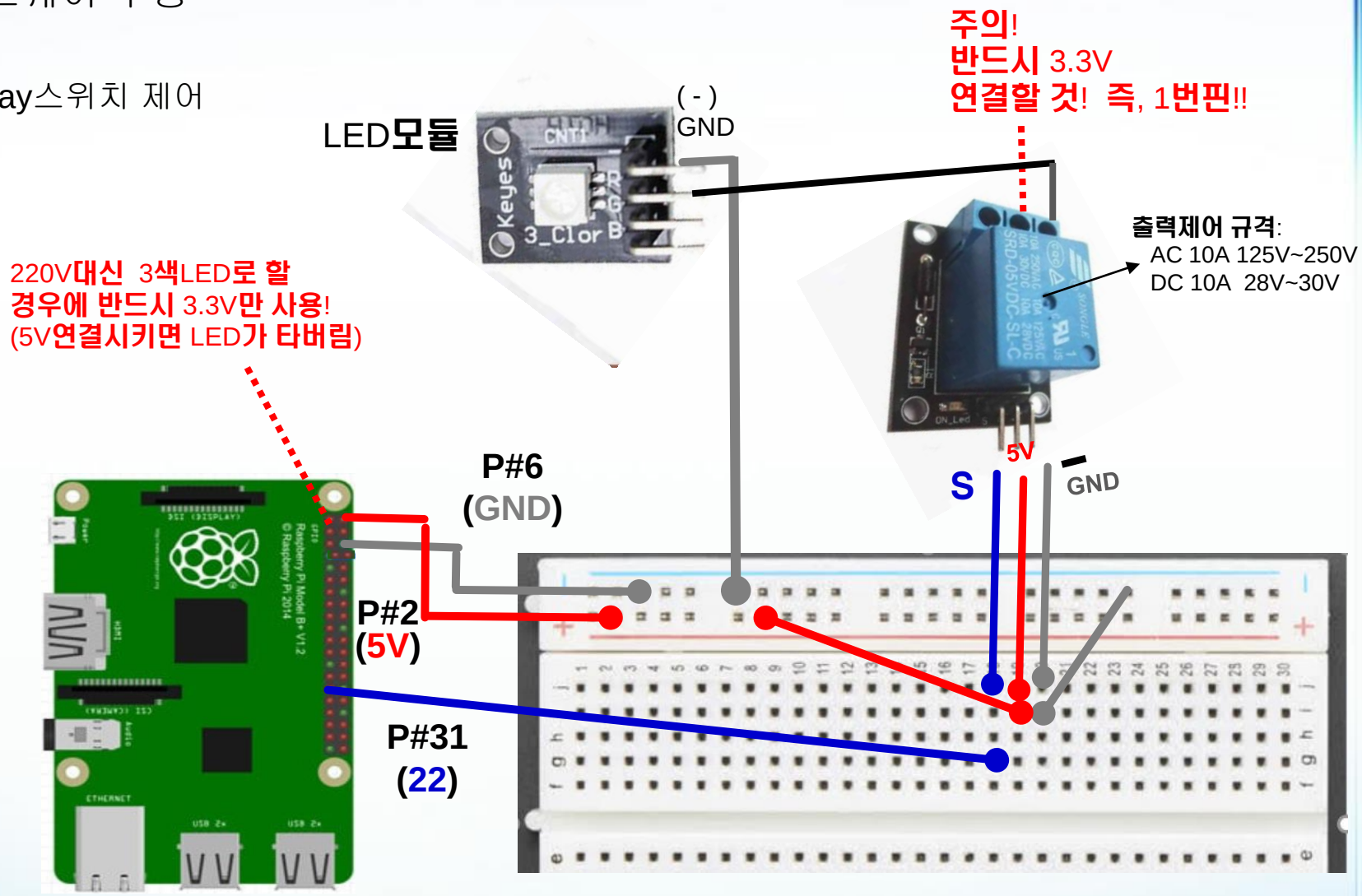




임베디드 액츄레이터/센서 제어

■ 하드웨어 구성

☑ Relay스위치 제어



임베디드 액츄레이터/센서 제어

■ Example code (자바스크립트 코드)

☑ Relay스위치 제어(On/Off)하는 코드

예제1 (relay.js)

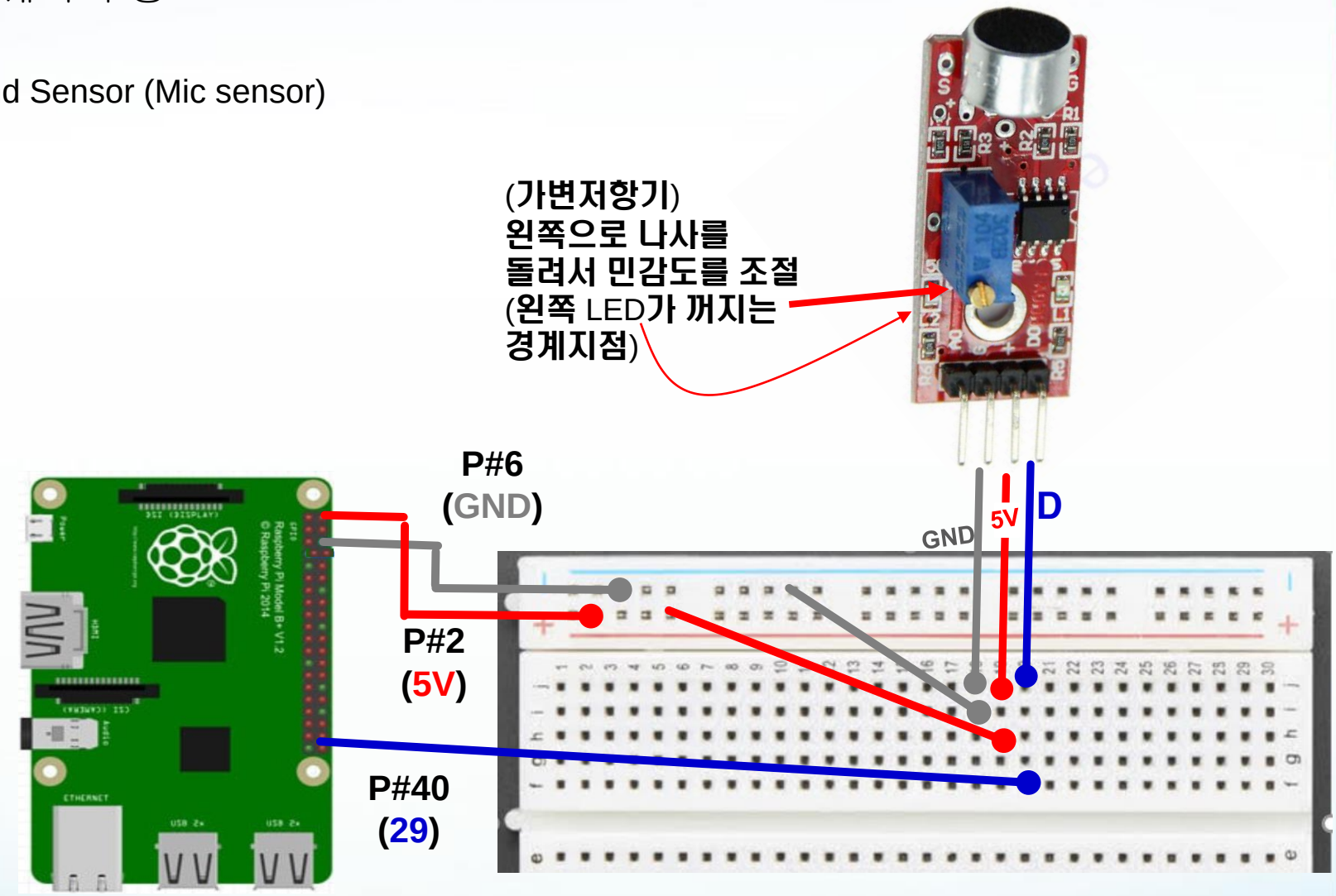
```
const  gpio = require('node-wiring-pi');
const  RELAY = 22;

const TurnOn = function() {
      gpio.digitalWrite(RELAY, gpio.HIGH);    // 3초 동안 전원공급
      console.log("Nodejs: RELAY on");
      setTimeout(TurnOff, 3000);
}

const TurnOff = function() {
      gpio.digitalWrite(RELAY, gpio.LOW);   // 3초동안 전원차단
      console.log("Nodejs: RELAY off");
      setTimeout(TurnOn, 3000);
}

gpio.wiringPiSetup();
gpio.pinMode(RELAY, gpio.OUTPUT);
setTimeout(TurnOn, 200);
```

```
$ vi   relay.js
$ sudo  node   relay.js
```

임베디드 액츄레이터/센서 제어

■ 하드웨어 구성

☑ Sound Sensor (Mic sensor)



임베디드 액츄레이터/센서 제어

■ Example code

☑ sound 센싱 코드

예제2 (sound.js)

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

const   DetectSound = function() {
    let  data = gpio.digitalRead(SOUND);
      if (data) {
           console.log("%d ! ", count++);
      }
      setTimeout(DetectSound, 10);
}

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

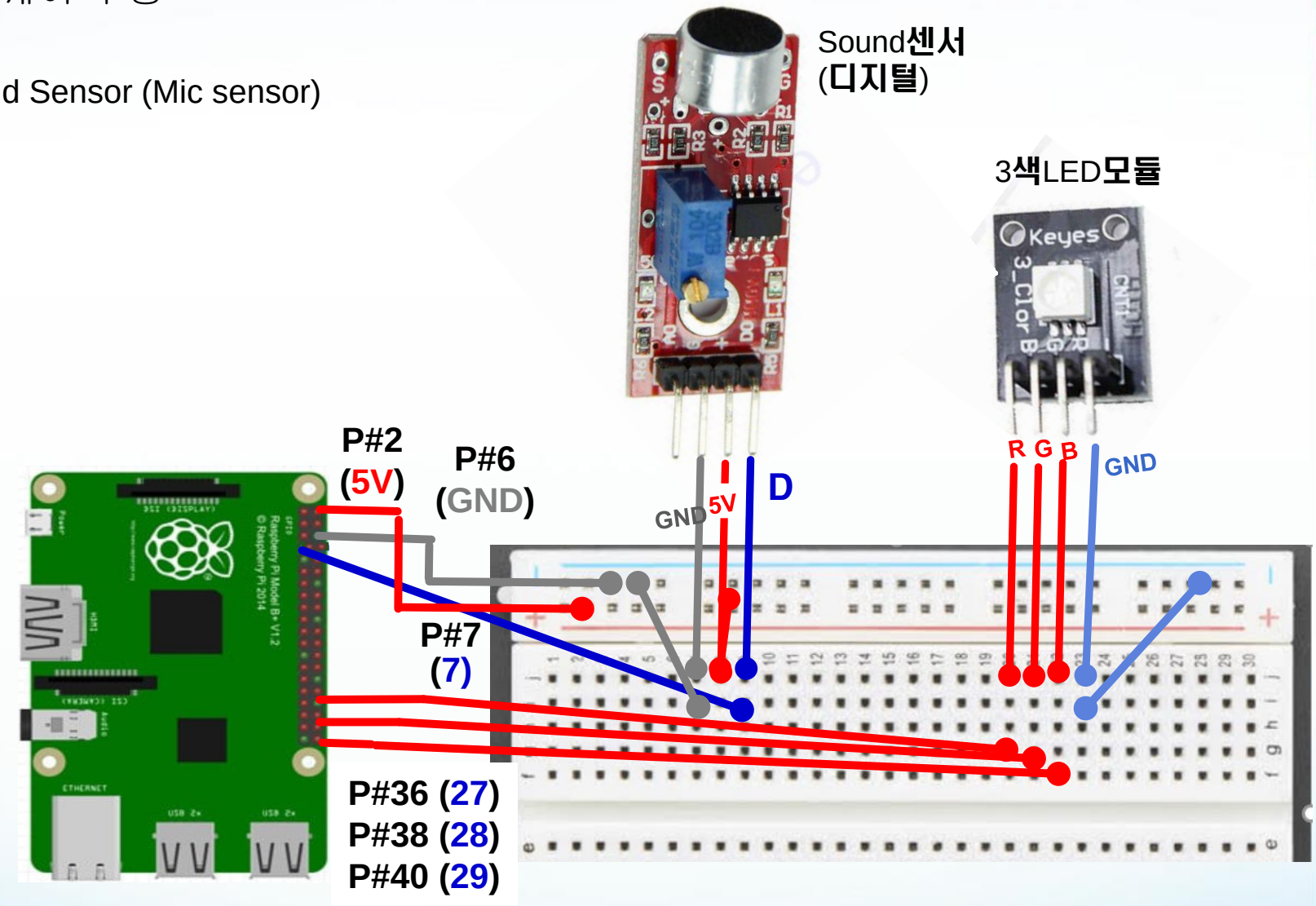

```
gpio. wiringPiSetup();
gpio. pinMode(SOUND, gpio. INPUT);
console.log("소리탐지중 … ");
setTimeout(DetectSound, 10);
```

```
$ vi   sound.js
$ sudo  node  sound.js
소리탐지중…
```

임베디드 액츄레이터/센서 제어

■ 하드웨어 구성

☑ Sound Sensor (Mic sensor)



임베디드 액츄레이터/센서 제어

■ Example code

☑ sound 를 탐지하면, 파랑색LED를 켜는 샘플코드

```
const    gpio = require('node-wiring-pi');
const    SOUND =  7;
const    BLUELED = 29;

const   DetectSound = function() {
      gpio.digitalWrite (BLUELED, 0);
      var data = gpio.digitalRead(SOUND);
      if (data) {
            gpio.digitalWrite(BLUELED, 1);
            console.log("Nodejs: it sounds loud !");
      }
      setTimeout(DetectSound, 10);
}

process.on('SIGINT', function() {
      gpio.digitalWrite(BLUELED, 0);
      console.log("Program Exit...");
      process.exit();
});
```

예제3 (led_sound.js)

```
gpio. wiringPiSetup();
gpio. pinMode(SOUND, gpio. INPUT);
gpio. pinMode(BLUELED, gpio.OUTPUT);
console.log("소리 탐지중...");
setTimeout(DetectSound, 1);
```

임베디드 액츄레이터/센서 제어

■ 실습일지

☑ 실습1

WiringPi GPIO API

■ WiringPi GPIO API (open source C libraries)

☑ WiringPi 란 ?

PIN based GPIO access library written in C for the Raspberry Pi.

☑ API 종류

- WiringPi Setup functions
- Core wiringPi functions

- Timing functions
- SPI library
- Software tone library
- Software PWM library

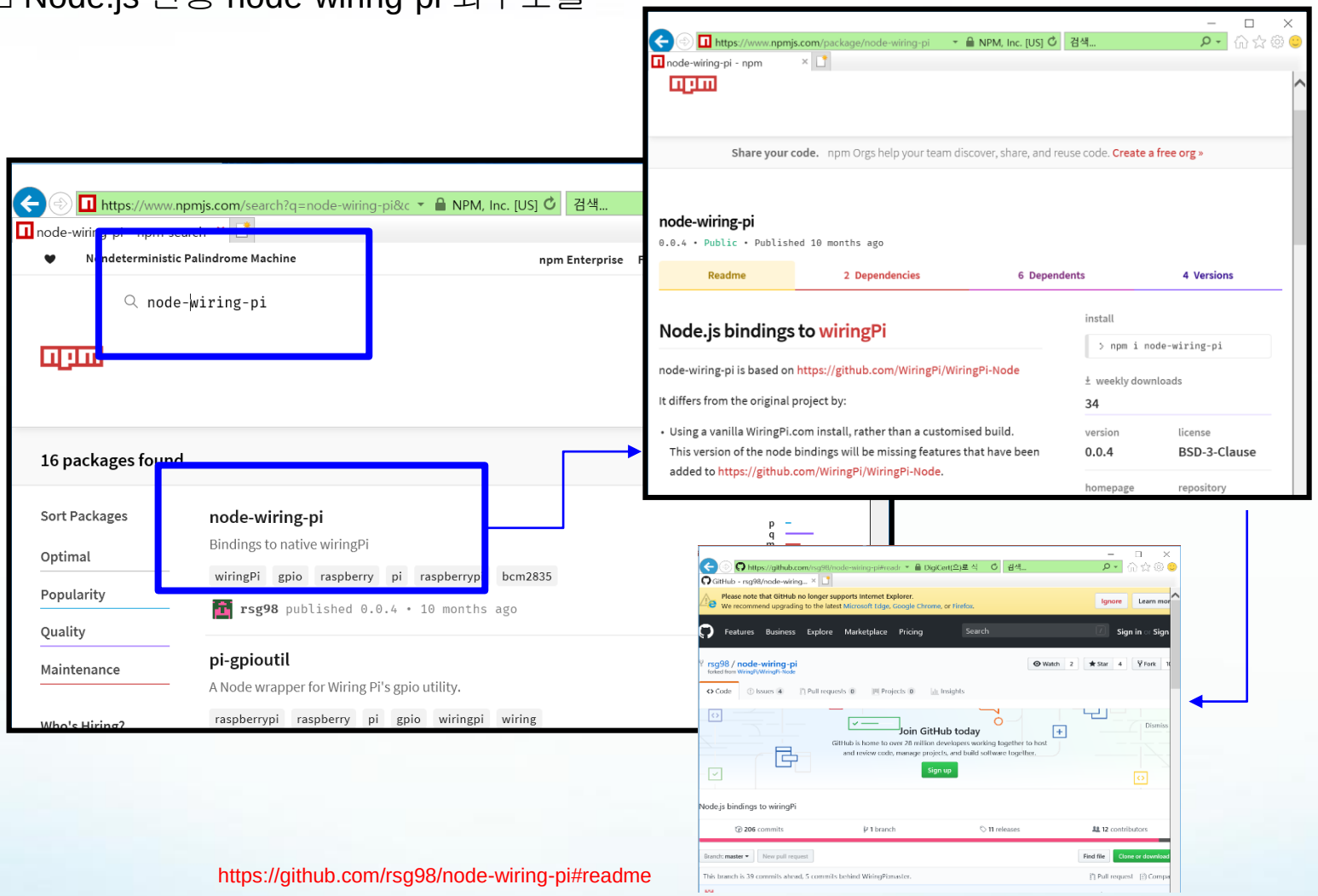
- Raspberry Pi specific functions
- Program priority, timing, threads
- Serial library
 -I2C library
 -Shift library

http://wiringpi.com

Wiring-pi / WiringPi-Node

- WiringPi - node.js

☑ Node.js 전용 node-wiring-pi 외부모듈



https://github.com/rsg98/node-wiring-pi#readme

WiringPi GPIO API

■ WiringPi GPIO API (open source API)

설치 · 추출	$ npm install node-wiring-pi
	const gpio = require('node-wiring-pi');

멤버함수(메소드)

Set up API

라즈베리파리 GPIO 핀을 초기화하는 함수 (반드시 초기에 호출되어야 함)

```
setup('wpi');                 // 'wpi'  'gpio'  'sys'  'phys'
wiringPiSetup ( ) ;           // GPIO핀을 wPi 핀번호로 설정
wiringPiSetupGpio ( ) ;       // GPIO핀을 BCM 핀번호로 설정
wiringPiSetupPhys ( ) ;       // GPIO핀을 Physical 핀번호로 설정
wiringPiSetupSys ( ) ;        // GPIO핀을 /sys/class/gpio로 설정
```

core API

GPIO 핀을 직접 제어하는 주요함수

```
pinMode (pin, mode) ;                 // mode =  INPUT, OUTPUT, PWM_OUTPUT
digitalWrite(pin, value) ;            // value = HIGH (1), LOW (0)
int     digitalRead(pin);
pullUpDnControl(pin,   pud);          // pud = PUD_OFF, PUD_DOWN, PUD_UP
pwmWrite(pin,  value);                // P#12, value = 0 ~ 1023
analogWrite(pin,  value) ;            // 추가적 아날로그모듈(또는 아날로고보드) 필요
int     analogRead(pin) ;
```

WiringPi GPIO API

■ WiringPi GPIO API

멤버함수(메소드)

timing API

시간지연함수, setup함수호출이후부터 경과한 시간을 측정하는 함수

```
void  delay(millisec);                    // 최대 49일까지 지연
void  delayMicroseconds(microsec);        // 최대 71분까지 지연
unsigned  int  millis(void) ;             // setup 이후 경과시간(millisecond)
unsigned  int  micros(void) ;             // setup 이후 경과시간(microsecond)
```

SPI API

(맨뒷장 부록참조)

ADC칩과 SPI 통신을 하기 위해, SPI 관련 핀을 제어하는 함수

```
int   wiringPiSPISetup(channel,  speed)    // 오류 -1리턴, 정상 > 0
Int   wiringPiSPIDataRW(channel,  data)    // 오류 -1리턴, 정상 > 0
```

channel … MCP3208은 8채널짜리 ADC칩이며, 센서가 연결된 채널을 지정
speed … Serial Clock speed (500,000 ~ 32,000,000 범위에서 지정)
data …… 라즈베리파이와 ADC칩간에 송수신하는 (버퍼에 저장된) 데이터

WiringPi GPIO API

■ WiringPi GPIO API

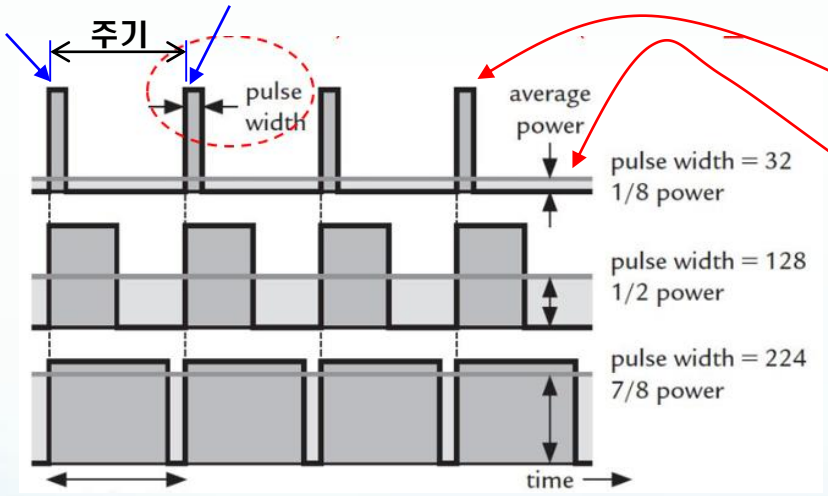
멤버함수(메소드)

Software PWM API

PWM(Pulse Width Modulation) – **펄스폭변조**
디지털 기기에서 아날로그 결과를 얻기 위한 기법
(DC**모터 속도제어**, LED**밝기제어에 활용되는 기술**)

```
int    softPwmCreate (pin, initialValue, pwmRange);      // 정상 0 리턴
void   softPwmWrite (pin,   value);
```

initialValue … **한 주기**(Frequency)**를 세분화했을때 시작 값**
pwmRange … **한 주기의 끝 값.** **예)** softPwmCreate(REDLED_PIN, 1, 100)
value …..... Duty cycle **크기값. 단, 범위에 있어야함.**
예) softPwmWrite(REDLED_PIN, 19); // 1~100 **중에서**
19만큼의 밝기로 셋팅



HIGH : **켜져있는 시간**
LOW : **꺼져있는 시간**

WiringPi GPIO API

■ WiringPi GPIO API (open source C libraries)

멤버함수(메소드)

Soft Tone API

소프트웨어로 사운드처리(소리 높낮이)를 가능하게 하는 함수

```
int     softToneCreate (pin) ;          // 성공 0 리턴
        softToneWrite (pin, frequency) ;  // frequency = 0 : 소리 OFF
        softToneStop(pin)
```

인터럽트 처리 API

무한반복(polling) 대신에, 인터럽트 처리방식으로 센서측정하는 기법/함수

```
int     wiringPiISR (pin, edgeType, callback)
```

특정pin에서, 지정된 인터럽트가 발생되면, 콜백함수가 자동적으로 호출된다.

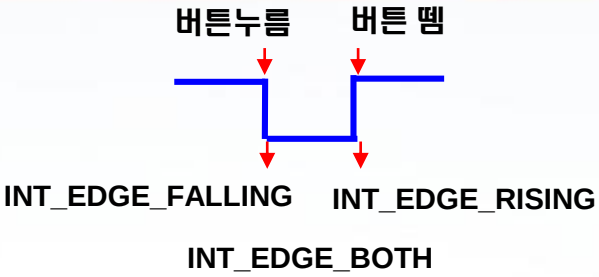
예)

```
void    MyHandler(void) {
    // 이곳에다가 처리할 프로그램 코드를 넣습니다.
    printf("인터럽트가 발생하면 함수가 자동으로 호출되어 실행됩니다. ₩n");
}

main() {
    ...
    wiringPiISR(BUTTON, INT_EDGE_RISING, &MyHander);
    ...
}
```

WiringPi GPIO API

■ Polling과 인터럽트 처리방식의 코드 비교

강추



Polling 방식

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

#define    BUTTON    29
void  main() {
   int  data = 0;

   wiringPiSetup();
   pinMode (BUTTON, INPUT);
   printf(" 버튼을 눌르세요 … \n");

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

인터럽트 방식

예제8 (button_isr.c)

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

#define    BUTTON    29

void  handler(void) {
     printf("Oh! Pressed!\n");    // 핸들러 코드
}

void  main() {

   wiringPiSetup();
   pinMode (BUTTON, INPUT);
    wiringPiISR (BUTTON, INT_EDGE_RISING, &handler);

    printf(" 버튼을 누르세요 …  \n");
    while  (1)
        sleep(10);
}
```

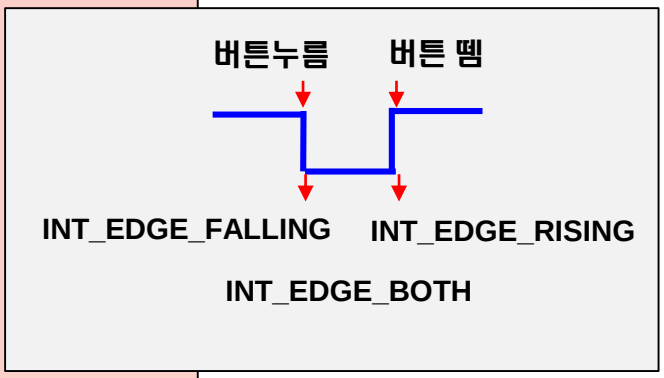
WiringPi GPIO API

■ 예제9를 Node.js 를 이용하여 이벤트처리 방식으로 변경하면

인터럽트(이벤트) 방식 (Node.js)

예제4 (led_sound_isr.js)

```
const  gpio = require('node-wiring-pi');
const  SOUND = 29;
const  BLUELED = 7;

const DetectSound = function() {
      gpio.digitalWrite(BLUELED, 1);
      gpio.delay(10);
      gpio.digitalWrite(BLUELED, 0);
}

process.on('SIGINT', function() {
      gpio.digitalWrite(BLUELED, 0);
      console.log("Program Exit...");
      process.exit();
});

gpio.wiringPiSetup();
gpio.pinMode(SOUND, gpio.INPUT);
gpio.pinMode(BLUELED, gpio.OUTPUT);
console.log("Node.js 이벤트방식: 소리 탐지중...");
gpio.wiringPiISR(SOUND, gpio.INT_EDGE_RISING, DetectSound);
```



Wiring-Pi 모듈 API (Node.js)

■ node-wiring-pi 모듈 API (open source Node.js module)

☑ API 종류

- APIs
  - Setup
  - Core functions
  - Interrupts
  - Raspberry Pi specific
  - I2C
  - SPI
  - Serial
  - Shift
  - Soft PWM
  - Soft Servo
  - Soft Tone
  - Extensions
- Extensions
  - dac7678
  - drcSerial
  - max31855
  - max5322
  - mcp23008
  - mcp23016
  - mcp23017
  - mcp23s08
  - mcp23s17
  - mcp3002
  - mcp3004/8
  - mcp3422/3/4
  - mcp4802/12/22

Wiring-Pi 모듈 API (Node.js)

■ 실습일지

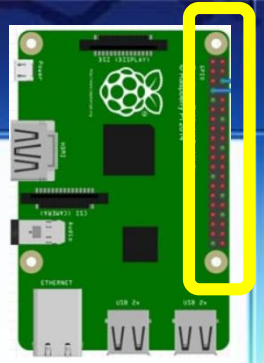
☑ 실습2

Wiring-Pi 모듈 API (Node.js)

■ 실습일지

☑ 실습2

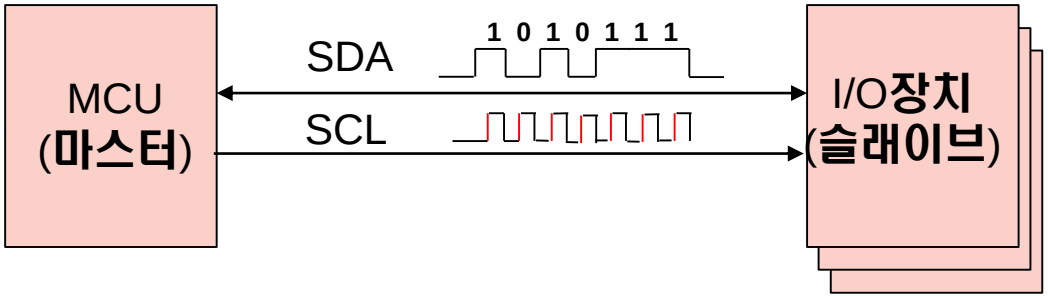
부록(I²C, SPI)



■ I²C, I2C 버스방식 (Inter-Integrated Circuit)

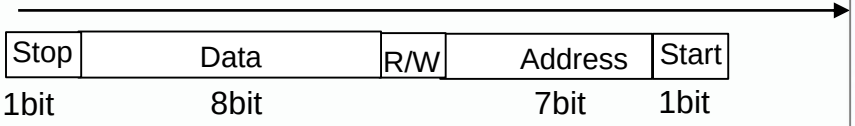
☑ MCU와 I/O 디바이스간의 양방향 시리얼(직렬)전송 버스

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



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

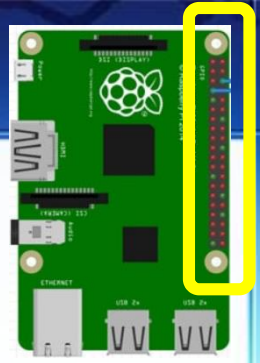
☑ SDA에서의 전송데이터 포맷



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 |
```

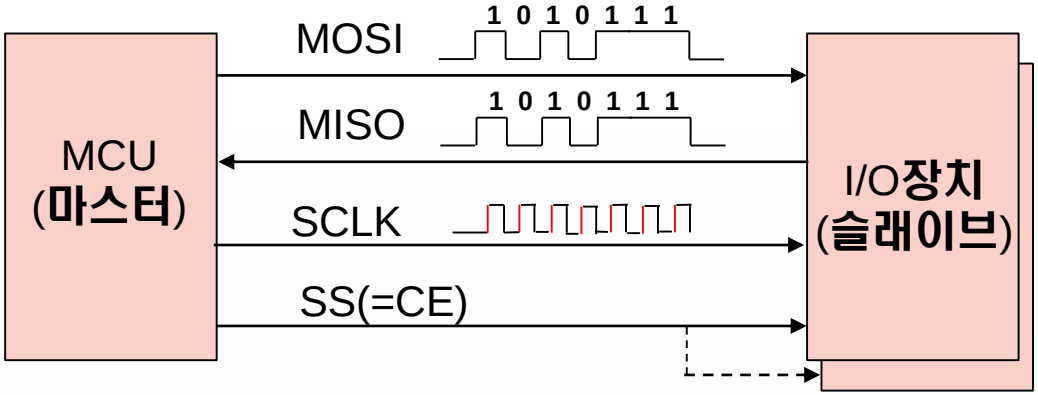
부록(I²C, SPI)



■ 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