
Arduino

Rev. A511

세부목차

Arduino

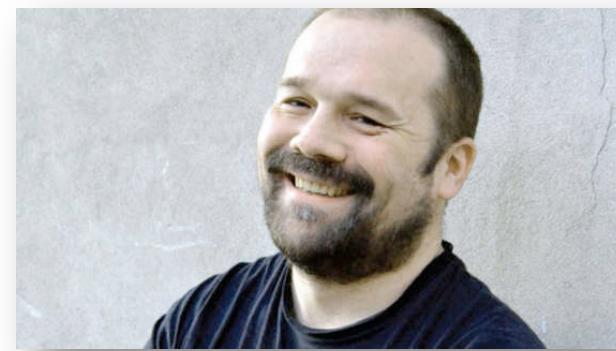
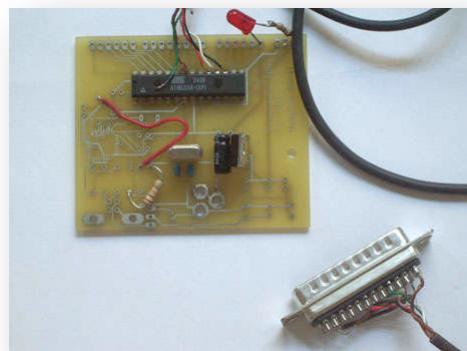
1. Introduction
2. 개발환경
3. Digital I/O
4. Analog I/O
5. 외부 라이브러리
6. 인터럽트와 타이머
7. Advanced I/O
8. 시리얼 통신
9. 아두이노 메모리

Introduction

Arduino

❖ Arduino?(아르duino)

- 이름의 유래
 - Arduin : 이탈리아 이브레아 지역의 왕(1002년), 독일 헨리 2세에 의해 쫓겨남
 - Bar di Re Arudino : 아두이노 개발자의 단골 술집
- Massimo Banzi(마시모 반지)
 - IDII(Interaction Design Institute Ivera) 교수
 - 이탈리아 이브레아 소재
 - 예술 IT 융합 전문 대학원
 - 콜롬비아 출신 IDII 대학원생 석사 논문인 Wiring을 파생
 - 동료 다비드 쿠아르티아예스와 아두이노 공동개발
 - MIT에서 만든 디자이너들을 위한 프로그램 언어인 Processing에서 영감을 얻음



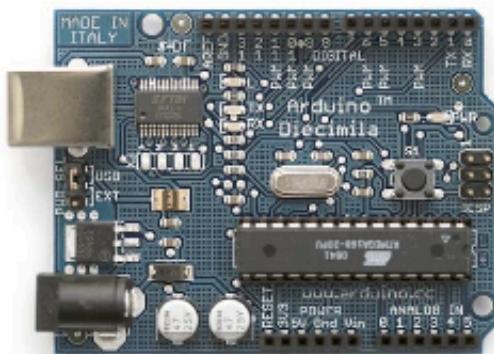
Introduction

Arduino

❖ Arduino

- Open Source H/W Prototyping Platform
- Arduino Board + IDE + Community

A physical piece of hardware



A programming environment

```
Arduino - 0010 Alpha
Blink.ino

/*
 * The basic Arduino example. Turns on an LED on for one second,
 * then off for one second, and so on... We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only one LED.
 */

// http://www.arduino.cc/en/Tutorial/Blink

int ledPin = 13; // LED connected to digital pin 13

void setup() // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT); // sets the digital pin as output
}

void loop() // can over and over again
{
  digitalWrite(ledPin, HIGH); // sets the LED on
  delay(1000); // waits for a second
  digitalWrite(ledPin, LOW); // sets the LED off
  delay(1000); // waits for a second
}
```

A community & philosophy



Arduino playground

:: About the Arduino Playground ::

Welcome to the Arduino Playground, a wiki where all the users of Arduino can contribute and benefit from their collective research.

This is the place to post and share your own code, circuit diagrams, tutorials, DIY instructions, tips and tricks, and after all the hard work, to show off your project!

Arduino Playground is a work in progress. We can use all the help you can give, so please read the [Participate](#) section and get your fingers typer!

:: RoadMap: What Needs to be Done? ::

There is a lot to do...most of the people are just starting, simple playgrounders waiting for you to fill them up...however here is a small roadmap of things I personally think should be developed first (again, this is a wish as you are more than welcome to add in

Arduino playground

Search: []

Projects Built with Arduino

Manuals

- Arduino Tutorials
- Official Tutorial page
- Information on the Arduino Mini
- Bluetooth tutorial
- Xbee (Zigbee) tutorial
- IR Arduino Shield
- Servo LCD shield
- Learning Electronics
- Learning Programming
- Learn About and Arduino
- Let the Arduino Shine
- Sharing Knowledge
- CourseWare
- Arduino and WiFi
- Programming the Bluetooth WiFi
- Reconfiguring Decimila AutoReset
- How to use "Debian" Upgrade

Run Arduino on...

- Unix
- Ubuntu Linux
- Debian Linux
- CentOS
- SUSE Linux
- Fedora Linux
- The command line

Insert Arduino with...

- Instant Reality (OSS)
- Flash
- Processing
- PC (Pure Data)

Introduction

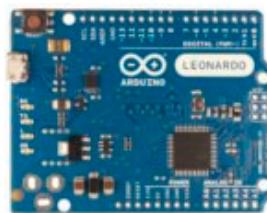
Arduino

❖ Arduino Boards

- <http://arduino.cc/en/Main/Products>



Arduino Uno



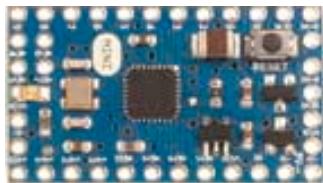
Arduino Leonardo



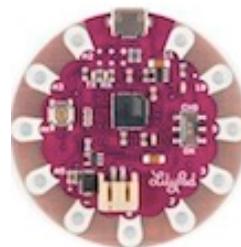
Arduino Yun



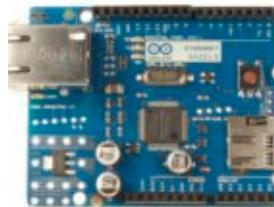
Arduino Micro



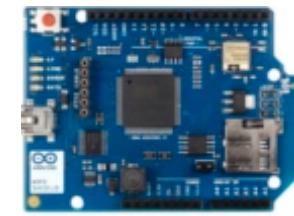
Arduino Mini



LilyPad Arduino USB



Ethernet Shield

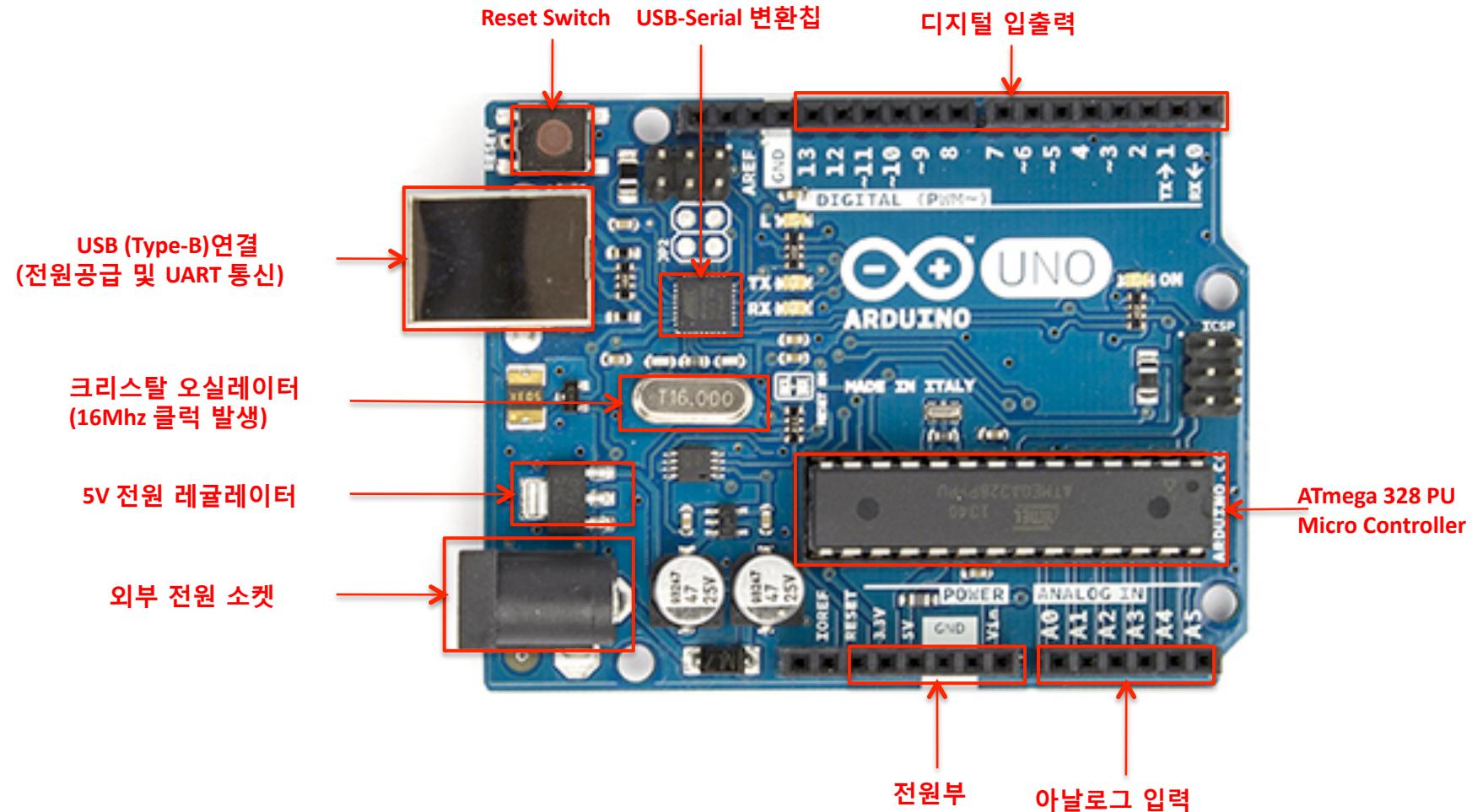


Wifi Shield

Introduction

Arduino

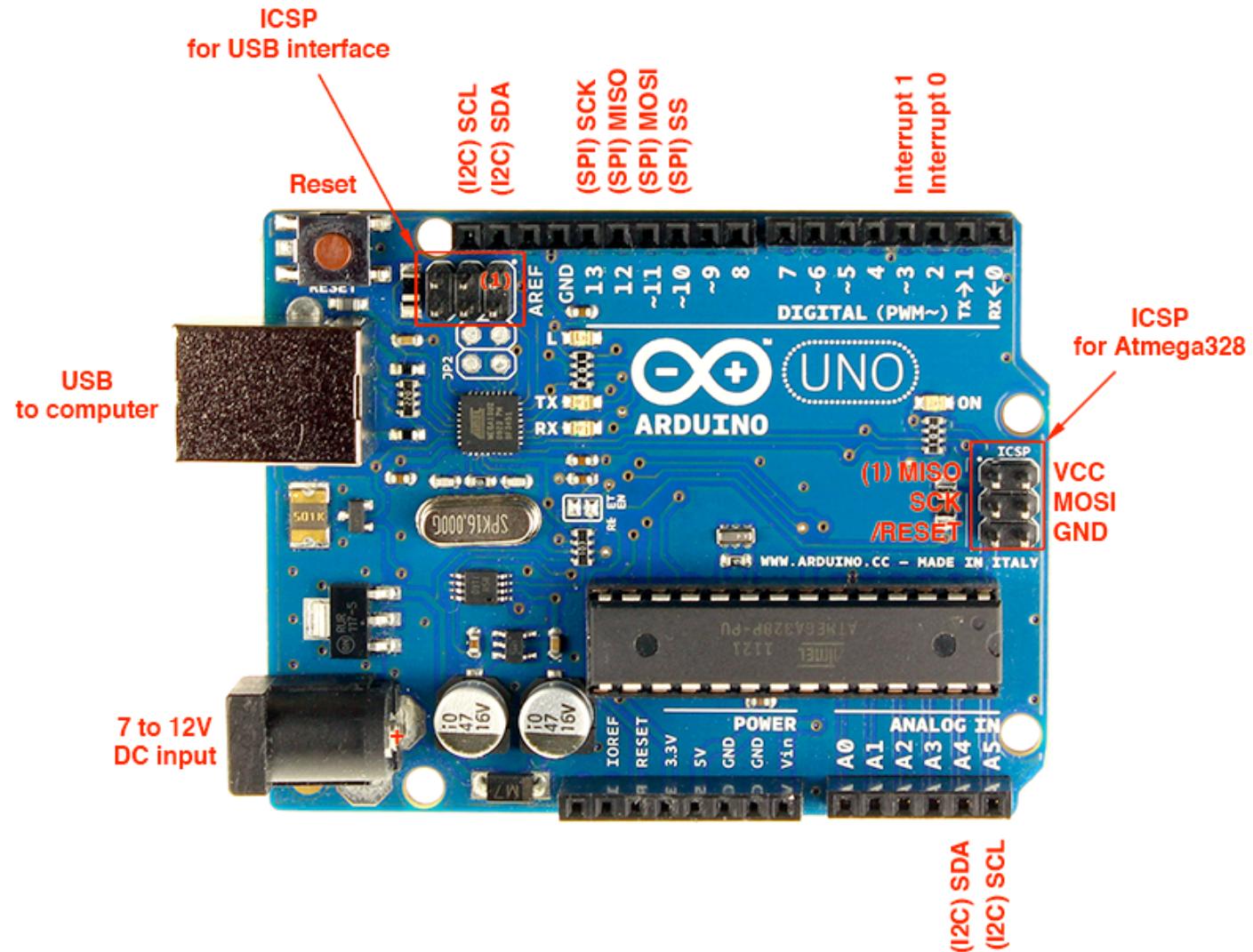
❖ Arduino



Introduction

Arduino

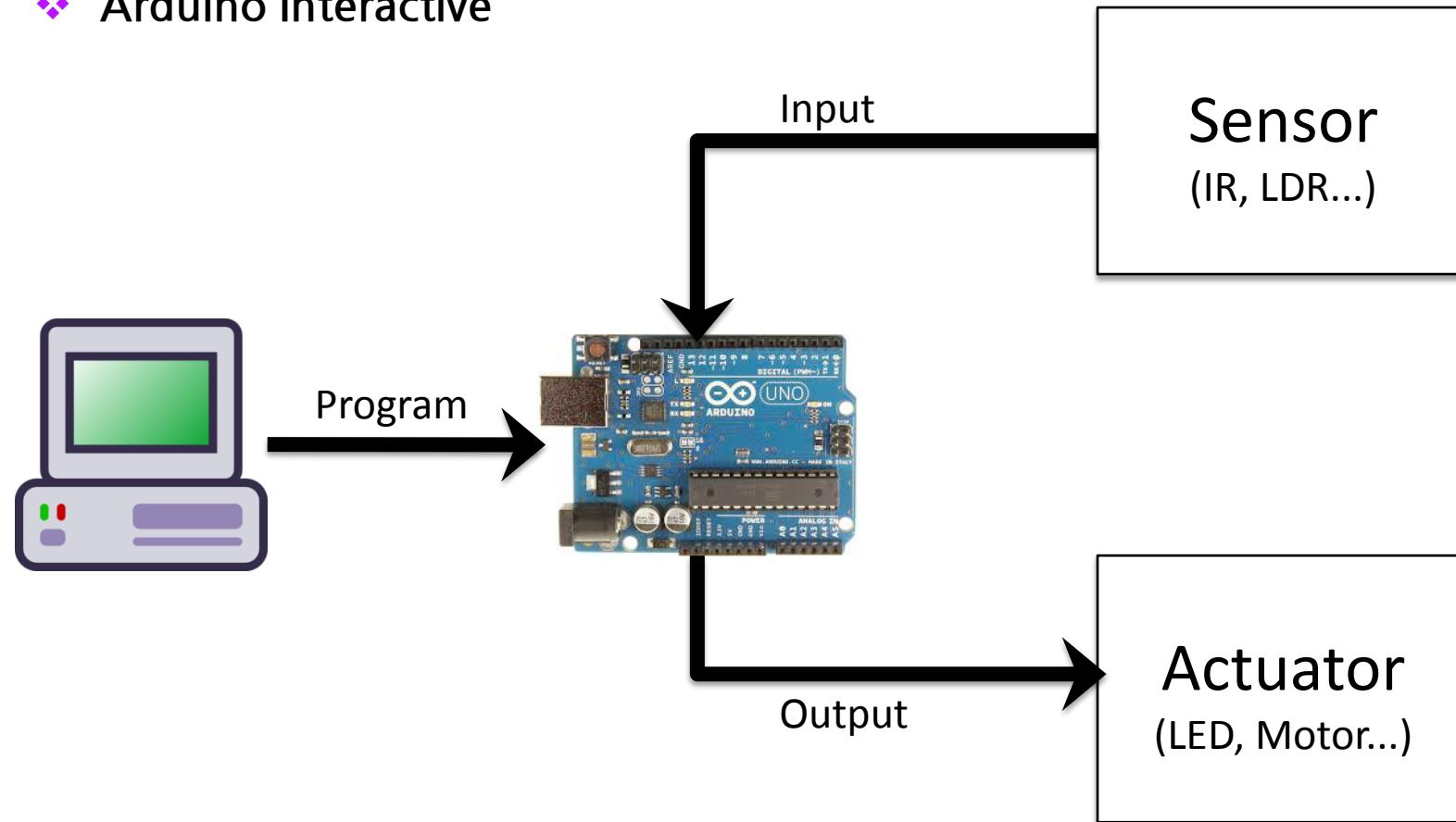
❖ Arduino



Introduction

Arduino

❖ Arduino Interactive



세부목차

Arduino

1. Introduction
2. 개발환경
3. Digital I/O
4. Analog I/O
5. 외부 라이브러리
6. 인터럽트와 타이머
7. Advanced I/O
8. 시리얼 통신
9. 아두이노 메모리

개발환경

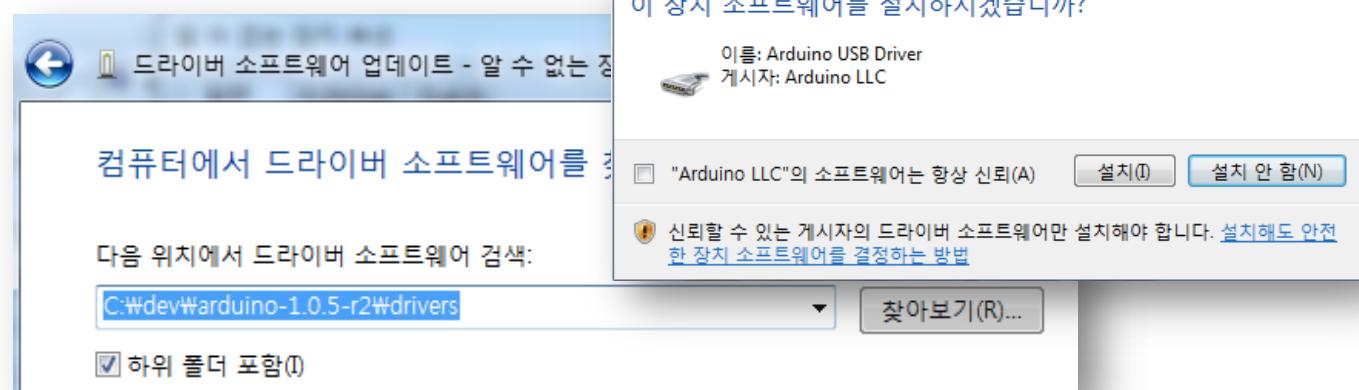
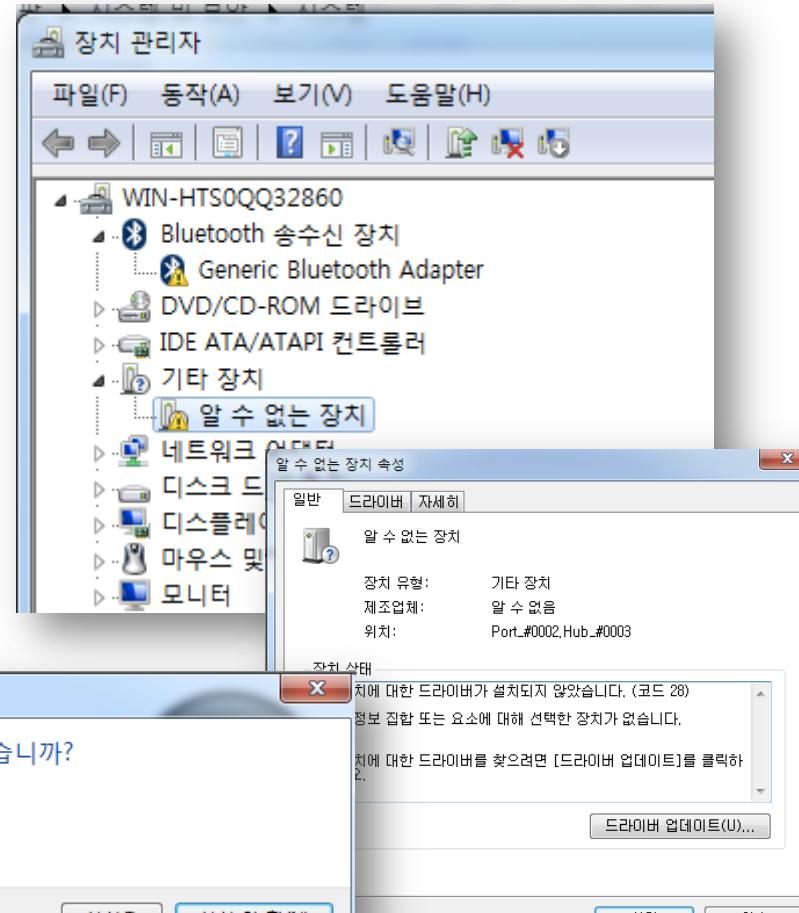
Arduino

❖ Arduino IDE

- <http://arduino.cc/en/Main/Software>
- Windows, Mac, Linux

❖ Driver 설치

- Windwos의 경우
- PC와 아두이노 연결
- 시작 -> 제어판 -> 시스템 -> 장치관리자
- COMXX 목록에서 드라이버 업데이트
- 아두이노 설치디렉토리/Drivers 선택

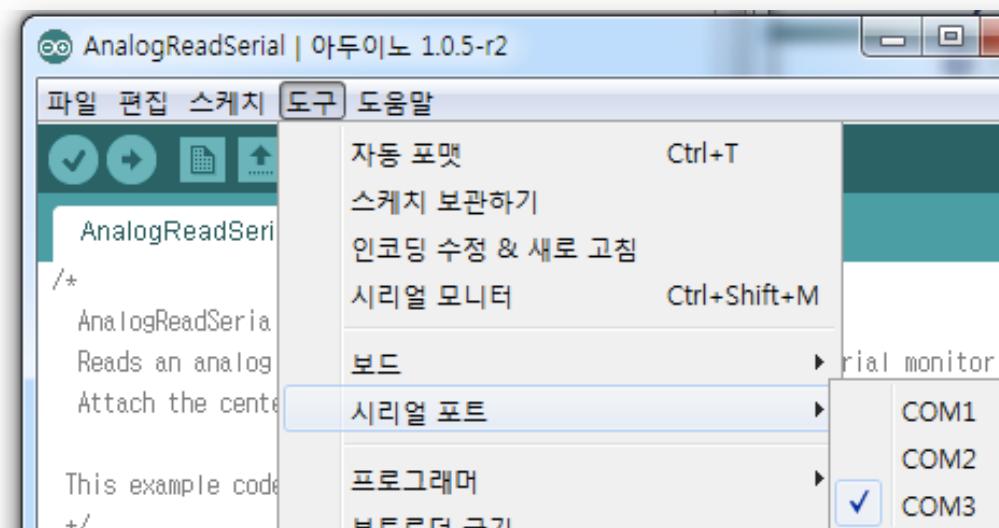
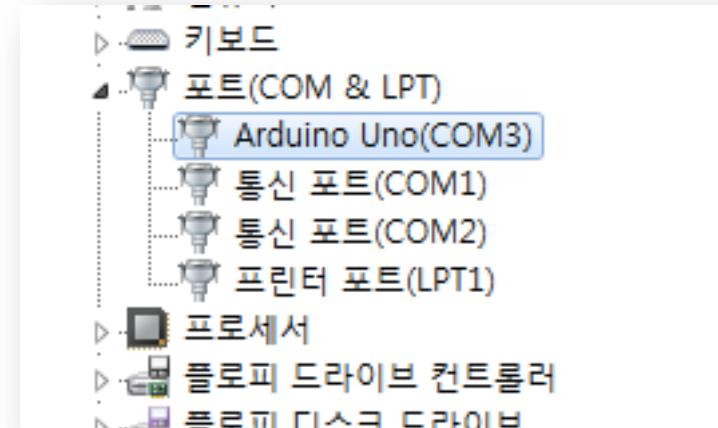


개발환경

Arduino

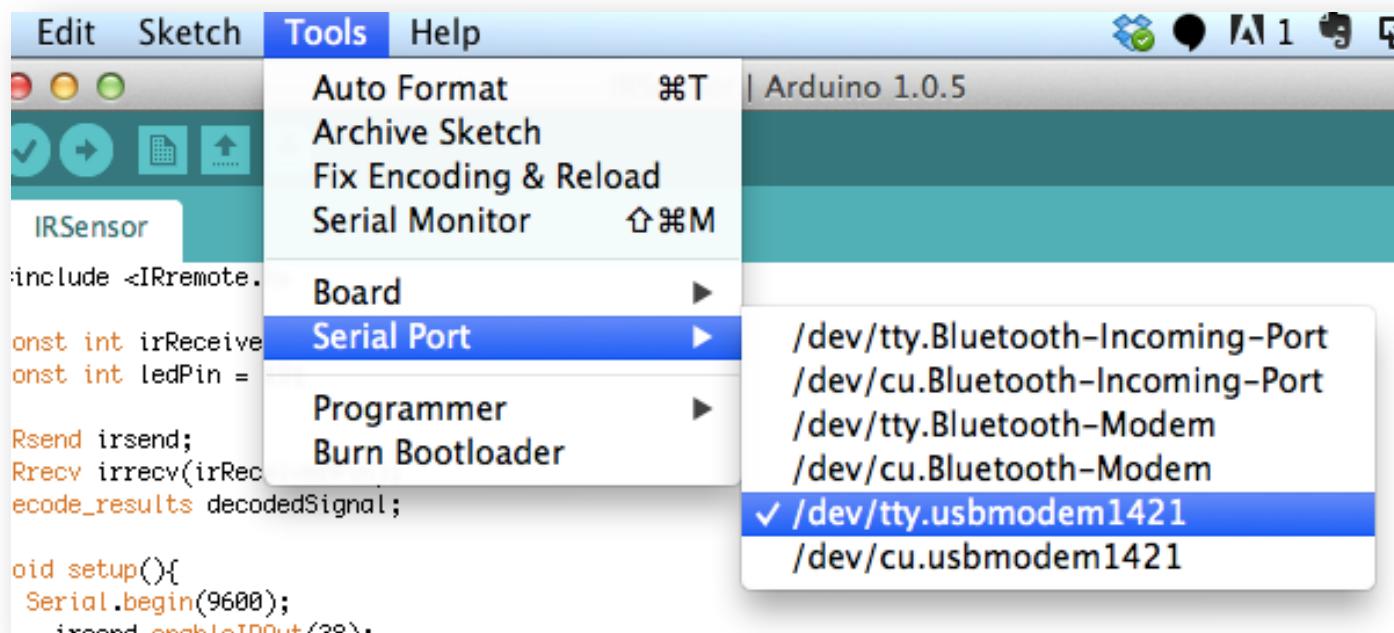
❖ Arduino IDE - Windows

- Driver 설치 후 장치관리자
- COM & LPT 항목 확인
 - COM 포트 번호는 기기마다 달라짐
- ArdulDE 실행
 - Arduino.exe
 - 도구 > 시리얼 포트
 - 장치에서 확인 한 COM포트 선택



❖ Arduino IDE - Mac

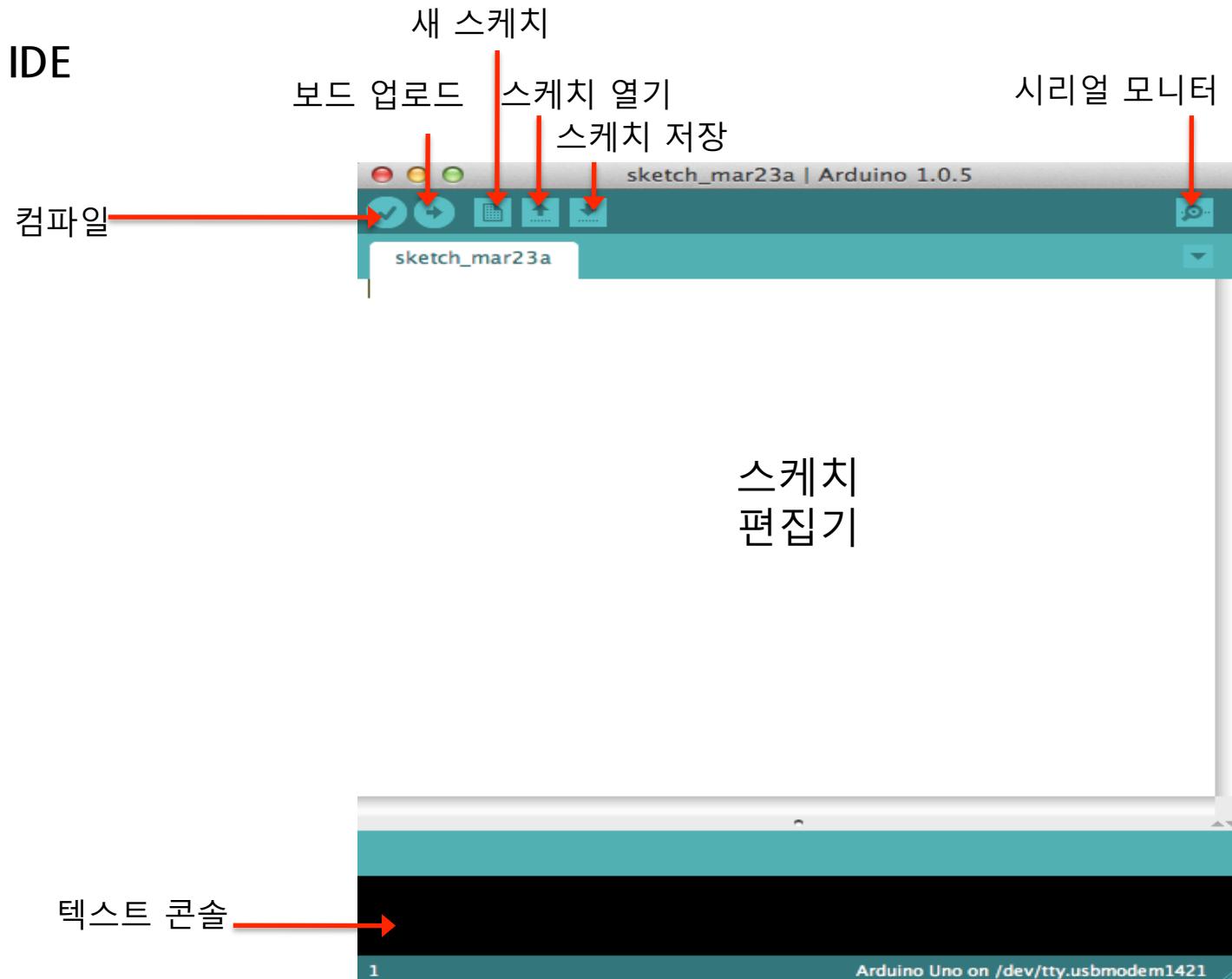
- Driver 설치 필요 없음
- ArdulDE 실행
 - Arduino.app
 - 도구 > 시리얼 포트
 - /dev/tty.usbmodemXXX 선택



개발환경

Arduino

❖ Arduino IDE

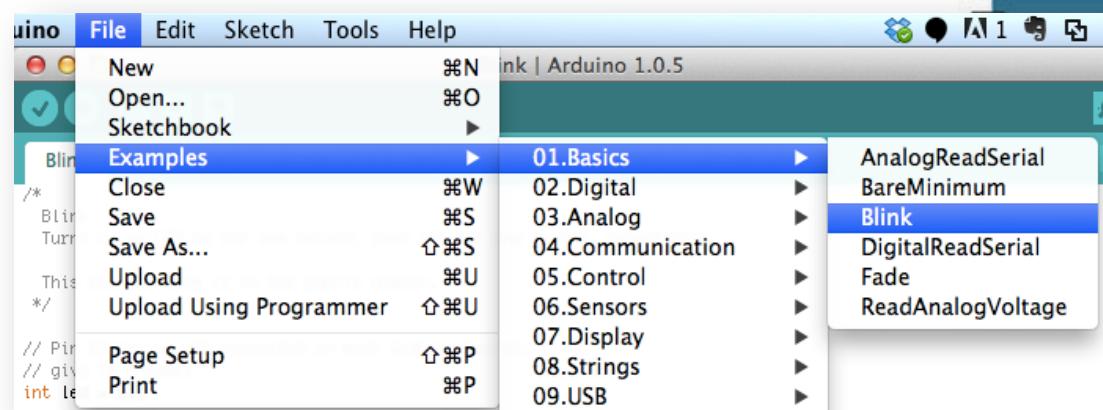
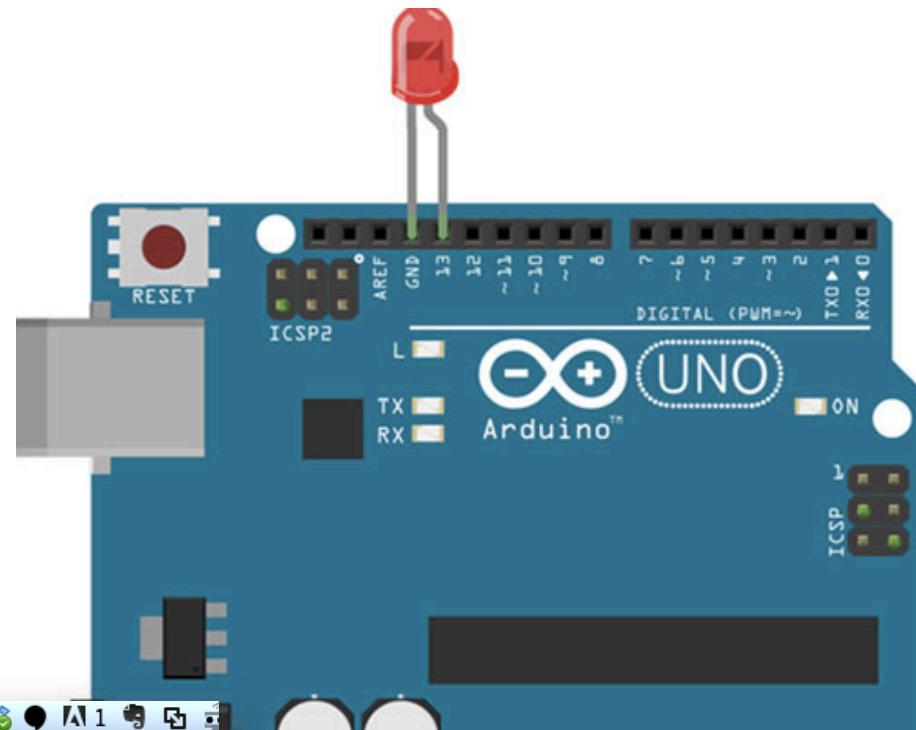


개발환경

Arduino

❖ Arduino HelloWorld

- 회로 구성
 - LED
 - 짧은 다리 GND
 - 긴 다리 13번
- 스케치
 - 파일 > 예제 > 01.Basics > Blink
 - 컴파일, 업로드
- 실험 결과
 - LED 및 13번 포트 옆 LED 깜박임



세부목차

Arduino

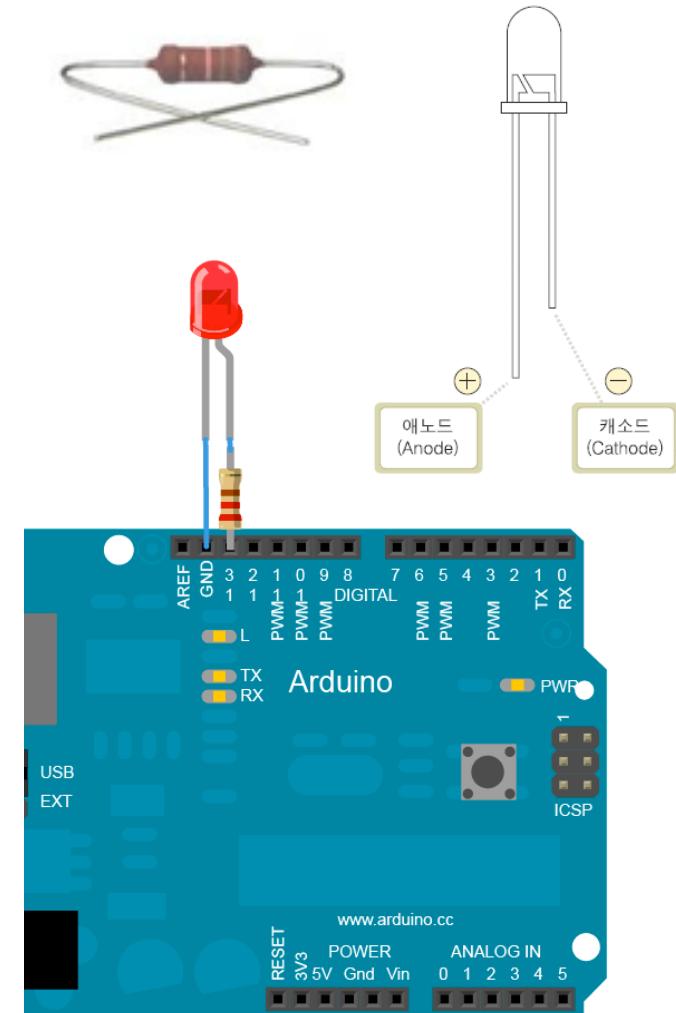
1. Introduction
2. 개발환경
3. Digital I/O
4. Analog I/O
5. 외부 라이브러리
6. 인터럽트와 타이머
7. Advanced I/O
8. 시리얼 통신
9. 아두이노 메모리

Digital I/O

Arduino

❖ LED Blink

- 아두이노의 HelloWorld
- LED 깜박이기
- 부품
 - LED(Light Emitting Diode)
 - 짧은 다리 : Cathod(캐소드), 음극(-)
 - 긴 다리 : Anode(애노드), 양극(+)
 - 캐소드 쪽 머리 테두리가 깎여 있다.
 - 저항(Register)
 - 전류의 흐름을 방해
 - 전압, 전류 저하
 - 전극이 없음
 - 단위 : Ω(ohm, 옴)
 - 회로 연결
 - LED 긴 다리 : 13번 포트 + 저항
 - LED 짧은 다리 : GND



Digital I/O

Arduino

❖ 저항 값 구하기

- LED 전압, 전류 확인
 - Data sheet
 - 5mm 적색 기준
 - 전류 20mA
 - 전압 약 1.8V~2.2V
- 옴의 법칙
 - $V = I * R$
 - $R = V / I$
 - (공급전압 - LED전압) / 전류
 - $(5 - 2) / 0.02 = 150$
 - 150에 근사한 값의 저항 사용 : 220Ω

Absolute Maximum Ratings: (Ta=25°C).			
ITEMS	Symbol	Absolute Maximum Rating	Unit
Forward Current	I _F	20	mA
Peak Forward Current	I _{FPK}	20	mA
Suggestion Using Current	I _{SU}	16-18	mA
Reverse Voltage (V _R =5V)	I _R	10	mA
Power Dissipation	P _D	105	mW
Operation Temperature	T _{OPR}	-40 ~ 85	°C
Storage Temperature	T _{STG}	-40 ~ 100	°C
Lead Soldering Temperature	T _{SOL}	Max. 260°C for 3 Sec. Max. (3mm from the base of the epoxy bulb)	

Absolute Maximum Ratings: (Ta=25°C)					
ITEMS	Symbol	Test condition	Min.	Typ.	Max.
Forward Voltage	V _F	I _F =20mA	1.8	---	2.2
Wavelength (nm) or TC(k)	Δ λ	I _F =20mA	620	---	625
*Luminous intensity	I _v	I _F =20mA	150	---	200
50% Viewing Angle	2 θ 1/2	I _F =20mA	40	---	60

Forward Current	I _F	20	mA
Forward Voltage	V _F	I _F =20mA	1.8 --- 2.2 V

Digital I/O

Arduino

❖ 저항값 읽기

- 과거 숫자 인쇄 기술 부족
- 4개 또는 5개의 색갈띠
- 금색 또는 은색을 오른쪽에
- 4색인 경우
 - 3번째는 승수(0의 갯수)
 - 4번째는 오차범위
- 5색인 경우
 - 4번째는 승수(0의 갯수)
 - 5번째는 오차범위
- 예시
 - 갈색(1), 검정(0), 오렌지(10^3), 금색
 - $10,000\Omega = 10K\Omega, \pm 5\%$
 - 빨강(2), 빨강(2), 갈색(10^1), 금색
 - $220\Omega, \pm 5\%$
 - 오렌지(3), 오렌지(3), 갈색(10^1), 금색
 - $330\Omega, \pm 5\%$
 - 갈색(1), 검정(0), 검정(0), 노랑(10^4), 갈색
 - $1,000,000\Omega = 1M\Omega, \pm 1\%$

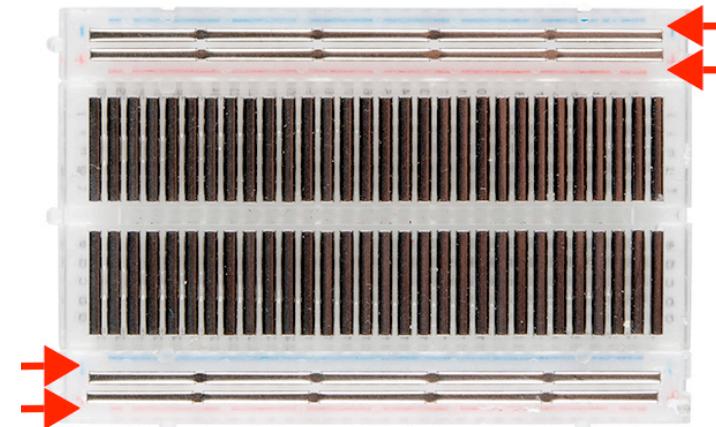
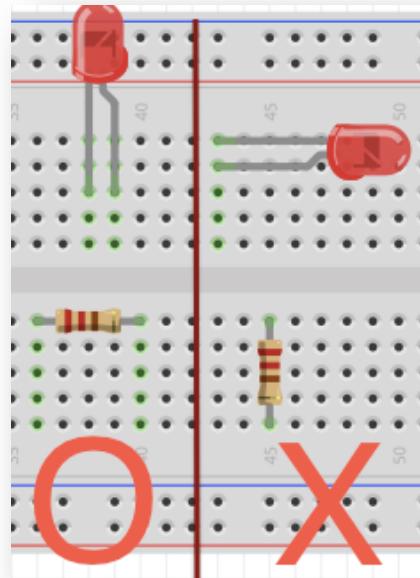
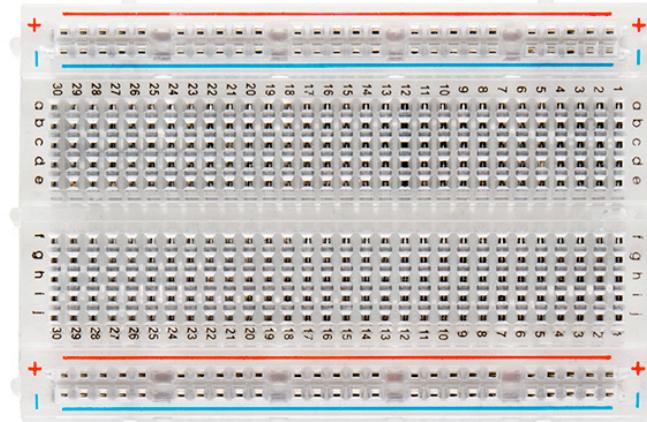
색명 (COLOR)		색띠			승수 (MULTIPLIER)	허용차 (TOLERANCE)
		1ST BAND	2ND BAND	3RD BAND		
흑색		0	0	0	0	
BLACK						
갈색		1	1	1	1	$\pm 1\%(F)$
BROWN						
빨강		2	2	2	2	$\pm 2\%(G)$
RED						
등색		3	3	3	3	
ORANGE						
황색		4	4	4	4	
YELLOW						
녹색		5	5	5	5	$\pm 0.5\%(D)$
GREEN						
청색		6	6	6	6	$\pm 0.25\%(C)$
BLUE						
자색		7	7	7	7	$\pm 0.1\%(B)$
GVIOLET						
회색		8	8	8	8	$\pm 0.05\%(A)$
GRAY						
백색		9	9	9	9	
WHITE						
금색					-1	$\pm 5\%(J)$
GOLD						
은색					-2	$\pm 10\%(K)$
SILVER						

Digital I/O

Arduino

❖ Bread Board (빵판)

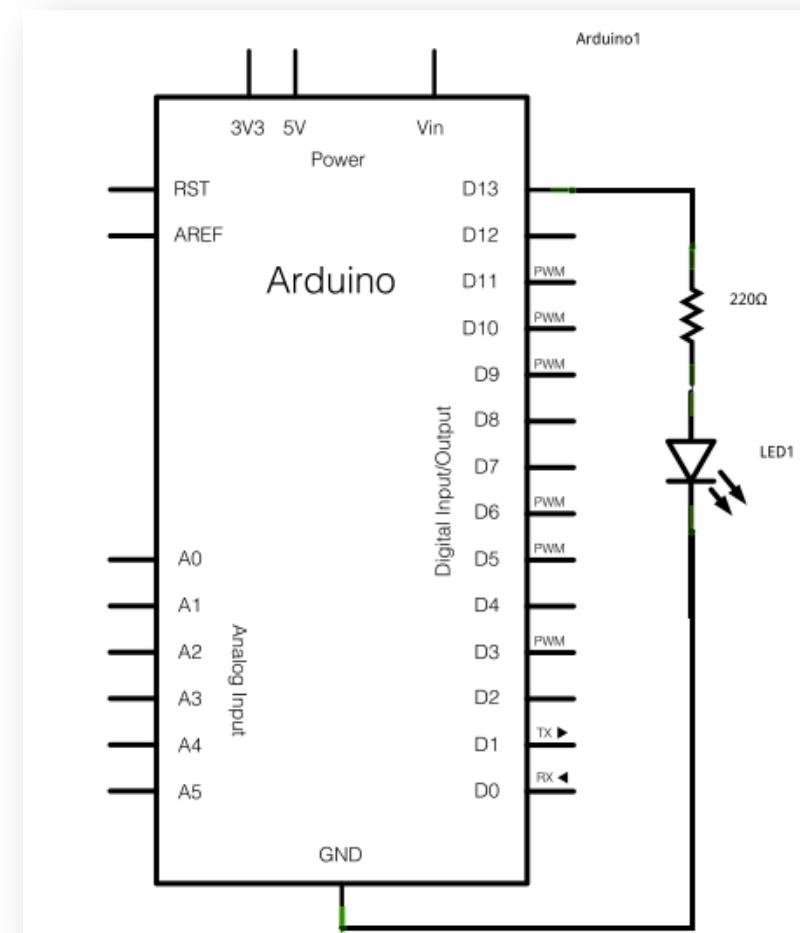
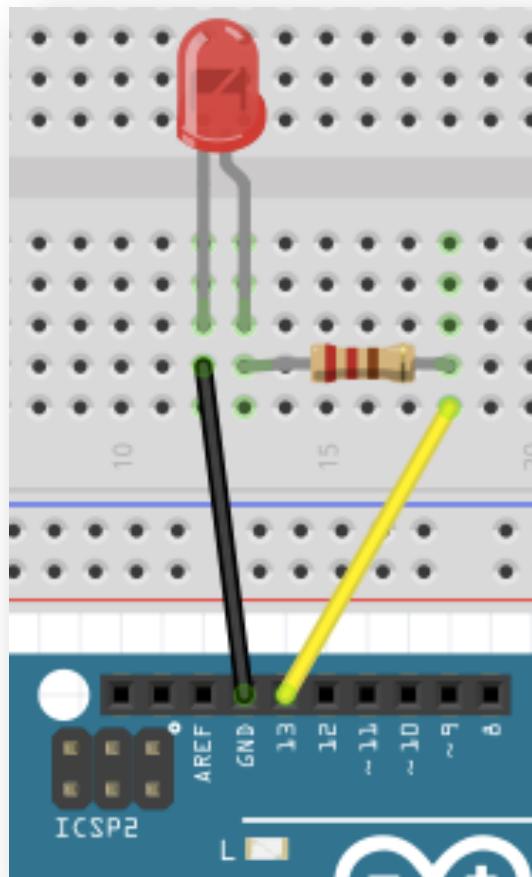
- PCB 보드를 만들기 전 프로토타입
- 납땜할 필요 없음
- 분리 및 재조립 가능
- 같은 열끼리 연결
- 좌우 세로 줄 전원 연결(버스영역)
- 중앙 5칸씩, 부품 연결(IC 영역)



Digital I/O

Arduino

❖ LED Blink 회로구성



Digital I/O

Arduino

❖ 스케치(소스코드) 구조

- C/C++ 언어
- 자체적인 Pre-processor 사용
 - 표준 C/C++과 다름
 - 함수 선언 이전에 호출 가능
 - 이진 표기 (0b 또는 0B)
- Setup()
 - 최초 1회 호출, 초기화 작업
- loop()
 - 전원이 켜져 있는 동안 실행될 기능
 - 영원히 반복
- 빌드 프로세스
 - main 함수를 갖는 중간 파일 생성
 - #include "arduino.h"
 - hardware/arduino/cores/arduino/
 - avr-gcc 컴파일
 - hardware/tools/avr/bin/
 - *.o 파일 링크 및 업로드

```
void setup(){
```

```
}
```

```
void loop(){
```

```
}
```

```
#include <arduino.h>
```

```
int main(void){
```

```
    init();
```

```
    setup();
```

```
    for(;;){
```

```
        loop();
```

```
}
```

```
    return 0;
```

```
}
```

Digital I/O

Arduino

❖ Data Types

Type	Bytes	Range
boolean	1	false(0), true(1)
byte	1	0 ~ 255
char	1	-128 ~ 127
int	2	-32768~32767
unsigned int	2	0 ~ 65535
long	4	-2147483648 ~ 2147483647
unsigned long	4	0 ~ 4294967295
float	4	3.4028235+38 ~ -3.4028235+38
double	4	float와 동일
String	문자열 Class	

❖ 입출력 기본 함수

- `pinMode(pin, mode)`
 - 지정된 핀을 입력 또는 출력으로 설정
 - pin : 설정하려는 핀의 번호
 - mode : INPUT, OUTPUT, INPUT_PULLUP
- `delay(ms)`
 - 지정된 시간 동안 프로그램을 일시 정지
 - ms : 일시 정지 하고 싶은 시간, milliseconds, unsigned long
- `digitalWrite(pin, value)`
 - 지정된 핀에 HIGH(5V) 또는 LOW(0V)를 출력
 - pin : 출력하려는 핀의 번호
 - value : HIGH 또는 LOW
- `digitalRead(pin)`
 - 지정된 핀에 값을 HIGH 또는 LOW로 입력(읽어 들임)
 - pin : 입력하려는 핀의 번호

Digital I/O

Arduino

❖ LED Blink 스케치

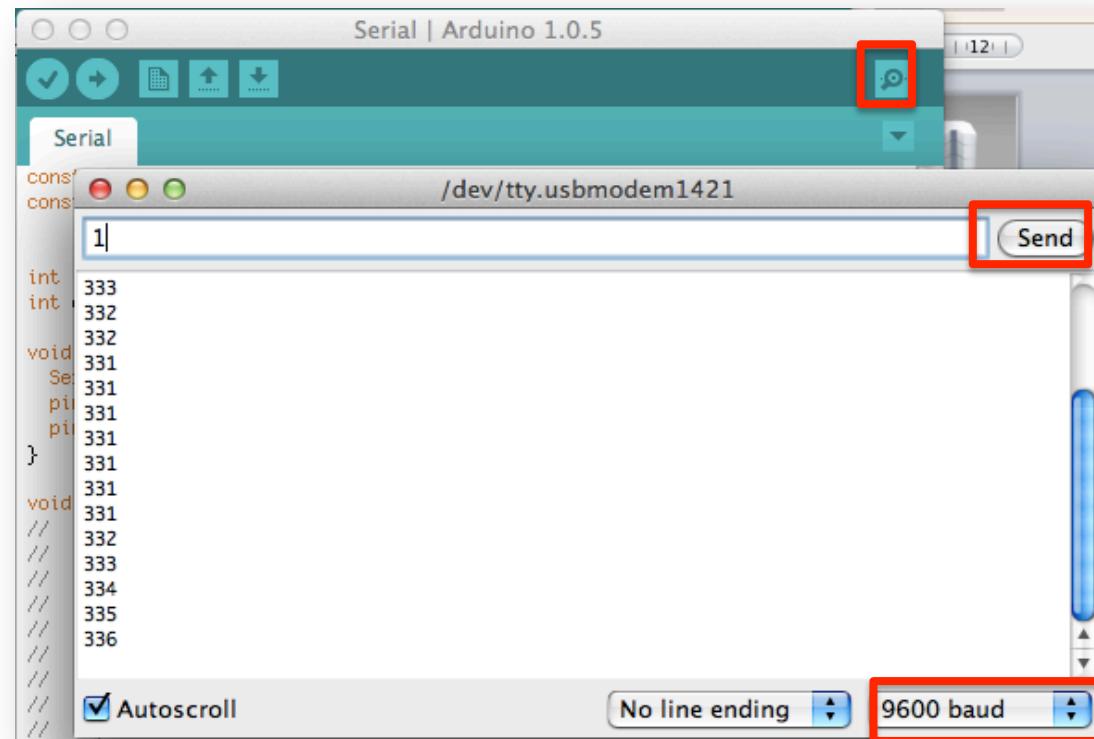
```
#define PIN_LED 13      //LED는 디지털 핀 13번에 있음

void setup(){
    pinMode(PIN_LED, OUTPUT); //LED있는 핀을 출력으로 설정
}

void loop(){
    digitalWrite(PIN_LED, HIGH); //LED를 켠다
    delay(1000);               //1초 동안 대기
    digitalWrite(PIN_LED, LOW); // LED를 끈다
    delay(1000);               // 1초 동안 대기
}
```

❖ 시리얼 모니터

- Arduino와 PC간의 Serial 통신 도구
- Tools > Serial Monitor 또는 단축 아이콘, Ctrl + Shift + M
- Baud(보오, 전송속도)를 스케치와 동일하게 선택



❖ Serial Input Output

- Serial.begin(baud)
 - baud(보오) 전송속도, 송수신측 같은 값을 사용
 - Serial.begin(9600)
- Serial.available()
 - 읽은 수 있는 바이트 수(버퍼에 저장되어 있는)
 - Serial.read() 호출 전에 If문으로 확인
- Serial.read()
 - 도착한 첫번째 바이트를 읽어 반환
 - 데이터가 없는 경우 -1 반환
- Serial.write()
 - 이진 데이터를 출력
- Serial.print();
 - ASCII Text를 출력
- Serial.println()
 - print() 함수에 개행문자 추가
- SerialEvent
 - Serial 데이터가 도착했을 때 호출됨

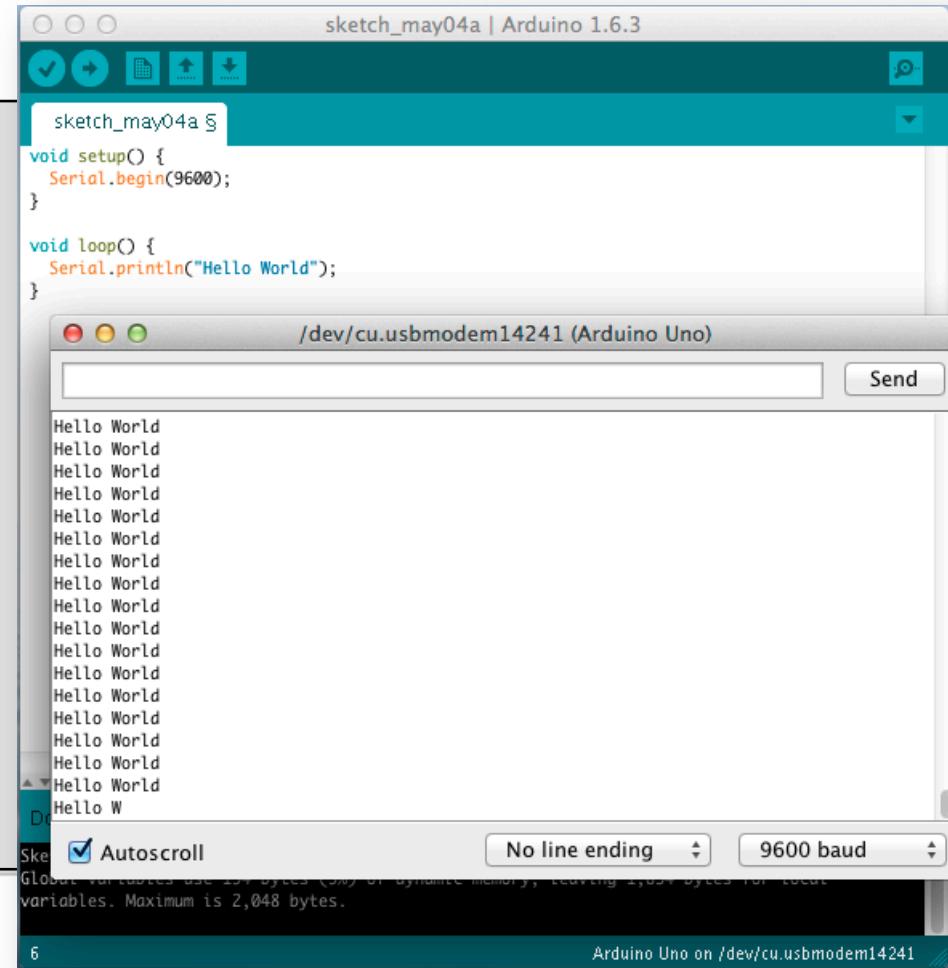
Digital I/O

Arduino

❖ 시리얼모니터 출력

- Hello World

```
void setup{
    Serial.begin(9600);
}
void loop(){
    Serial.println("Hello World");
}
```



Digital I/O

Arduino

❖ 시리얼모니터로 LED On/Off

- Loop에서 처리

```
#define LED 13
int val = 0;
void setup(){
    Serial.begin(9600);
    pinMode(LED, OUTPUT);
}
void loop(){
    if(Serial.available() > 0){
        val = Serial.read();
        if(val == 49){ // ASCII '1'
            digitalWrite(13, HIGH);
        }
        if(val == 48){ // ASCII '0'
            digitalWrite(13, LOW);
        }
    }
}
```



Digital I/O

Arduino

❖ 시리얼모니터로 LED On/Off

- SerialEvent에서 처리

```
#define PIN_LED 13
int val = 0;

void setup() {
    Serial.begin(9600);
    pinMode(PIN_LED, OUTPUT);
}

void loop(){
    if(val == 49){
        digitalWrite(13, HIGH);
    }else if(val == 48){
        digitalWrite(13, LOW);
    }
    delay(100);
}
void serialEvent() {
    val = Serial.read();
}
```

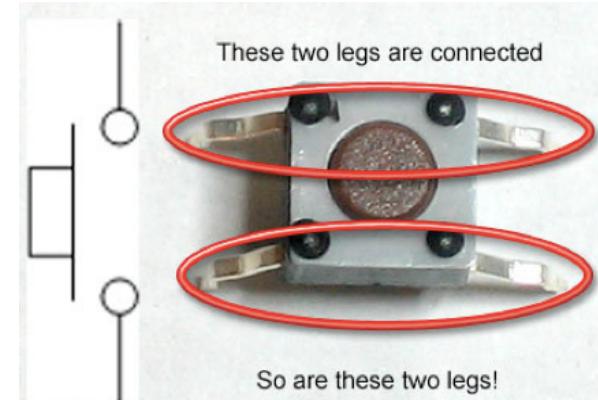


Digital I/O

Arduino

❖ 푸쉬 버튼 스위치 입력

- 버튼 스위치로 LED를 켜고 끄
- 버튼 스위치
 - 두개의 다리는 붙어 있다
 - 4개 중 2개만 연결해도 된다.
- 회로 연결
 - 13번 - LED - GND
 - 풀다운 저항
 - 5V - 버튼 - 10k 저항 - GND
 - 버튼 - Digital 7번
 - 풀업 저항
 - 5V - 10k저항 - 버튼 - Digital 7번
 - 버튼 - GND

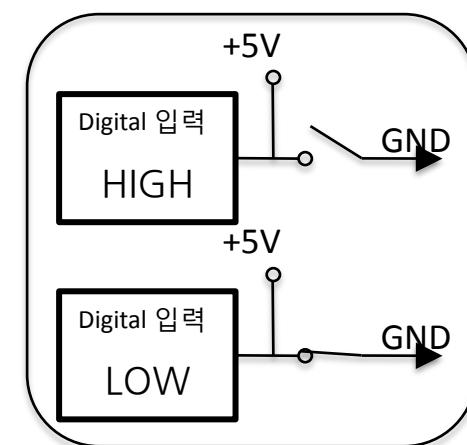
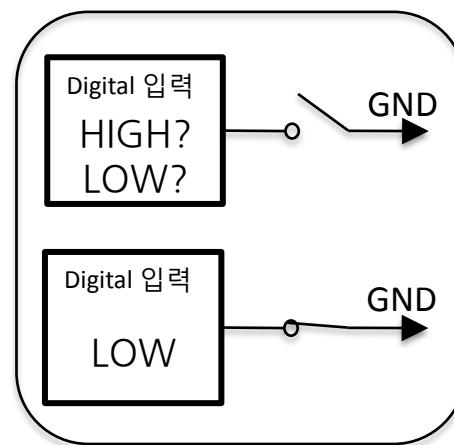
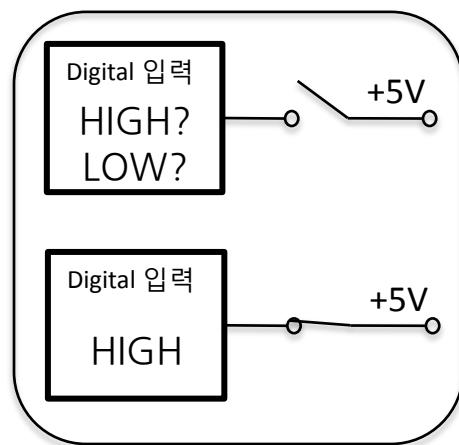


Digital I/O

Arduino

❖ 풀업(Pull Up), 풀다운(Pull Down) 저항

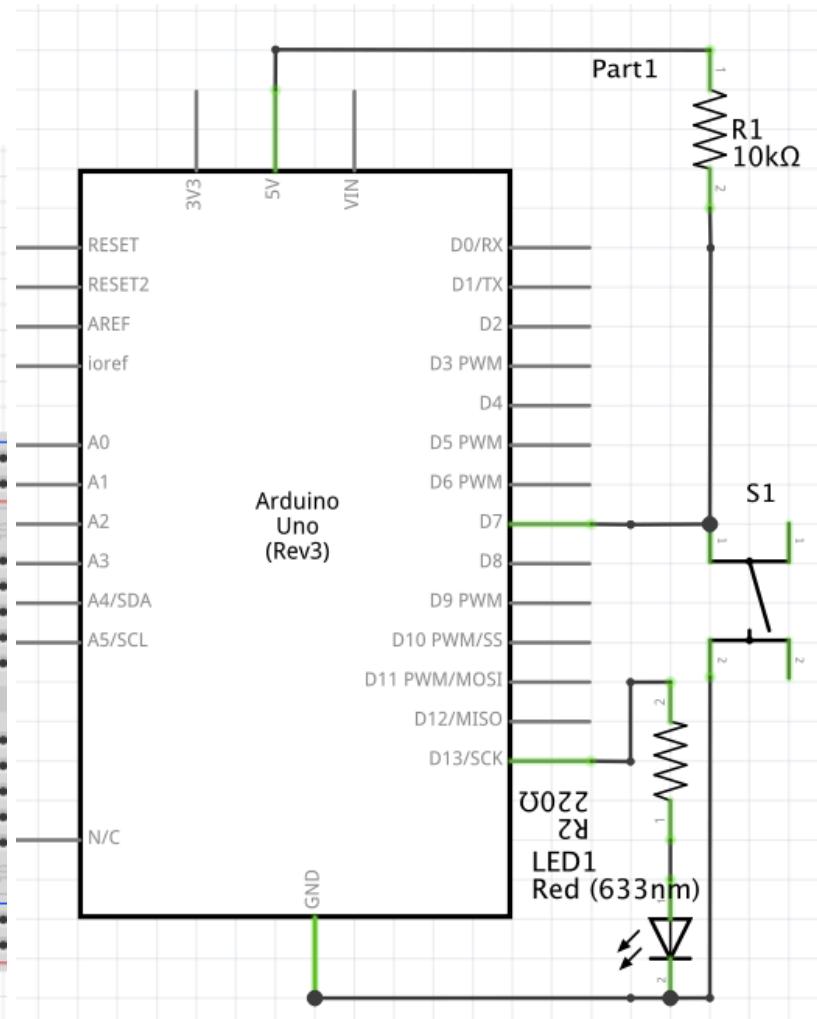
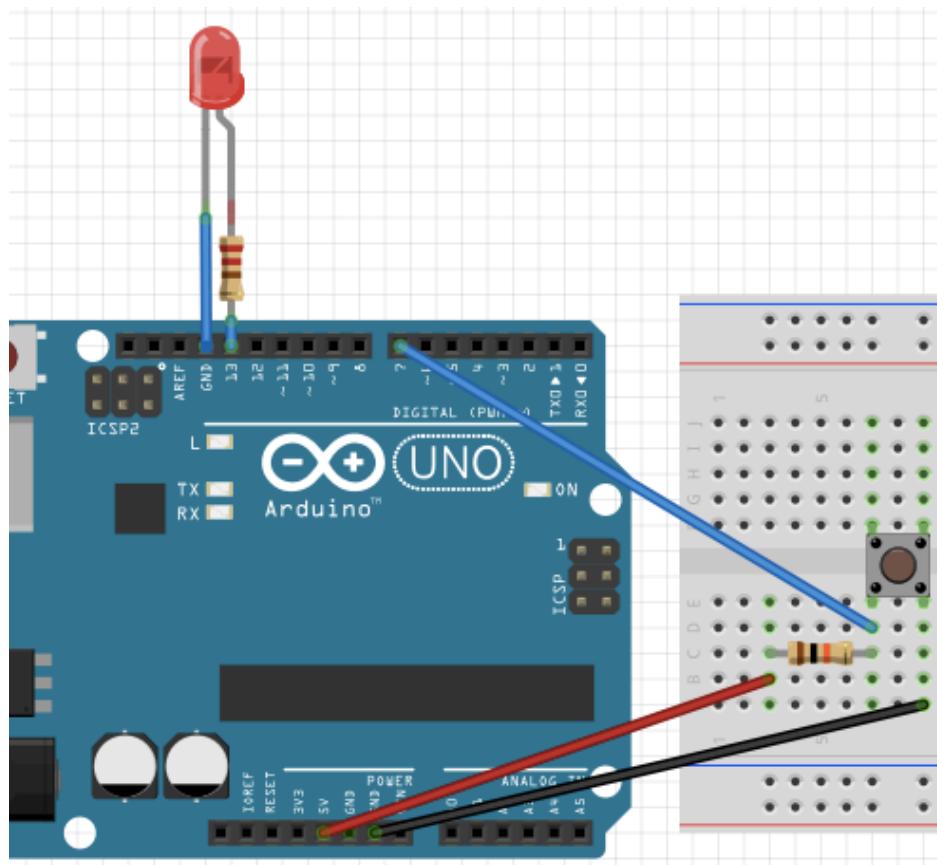
- 플로팅(Floating) 상태
 - 스위치가 열려있는 동안 어떤 상태인지 알 수 없는 상태
 - 주변 핀의 전압, 정전기 등 잡음에 취약
 - 스위치가 열려있는 동안 Vcc(5V) 또는 0V(GND)를 연결해서 해결
 - Vcc와 GND를 그대로 연결하면 단락되어 과전류 문제
 - 일반적으로 10KΩ 저항을 사용하여 해결
 - 저항을 전원(Vcc)에 연결하면 풀업(Pull Up), GND 연결하면 풀다운(Pull Down)



Digital I/O

Arduino

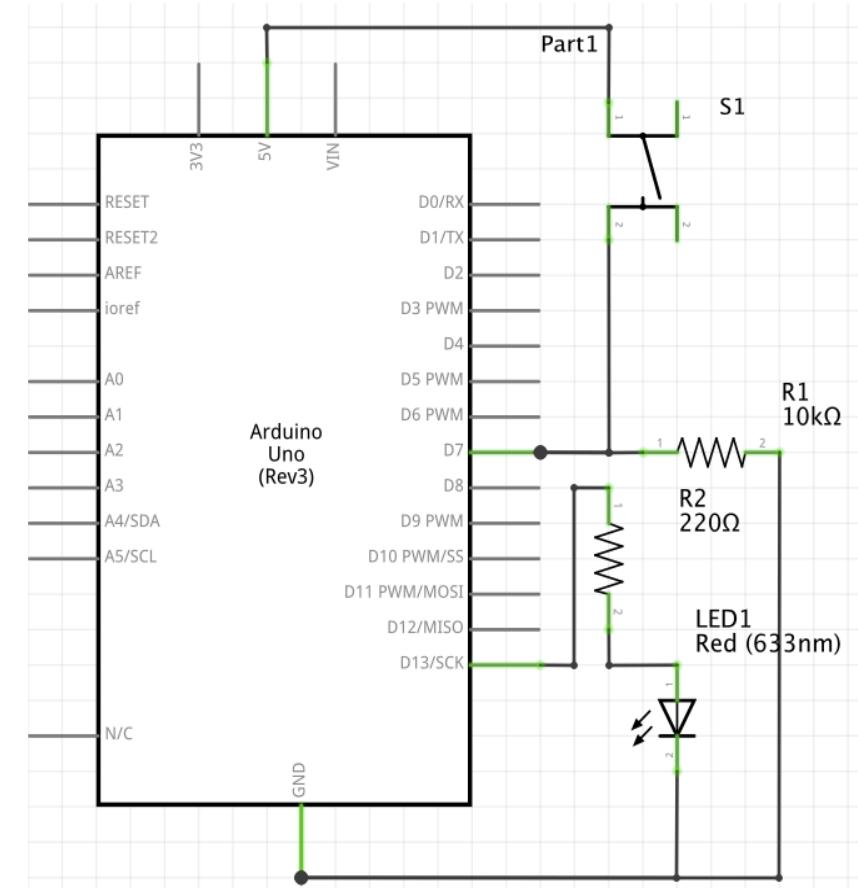
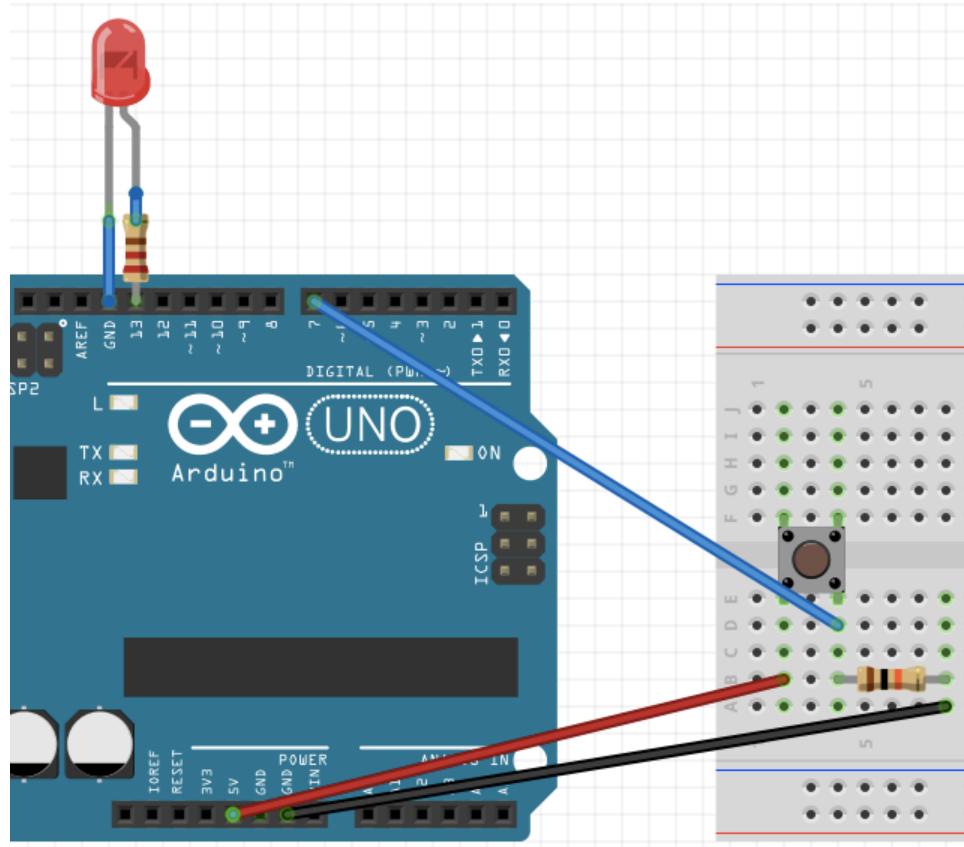
- ❖ 푸쉬버튼 스위치 회로(풀업 저항)



Digital I/O

Arduino

- ❖ 푸쉬버튼 스위치 회로(풀다운 저항)



Digital I/O

Arduino

❖ 푸쉬버튼 스위치 스케치

```
#define LED 13      //LED는 디지털 핀 13번에 있음
#define BUTTON 7 //버튼은 디지털 핀 7번에 있음

int val = 0;
void setup(){
    pinMode(LED, OUTPUT); //LED있는 핀을 출력으로 설정
    pinMode(BUTTON, INPUT); //BUTTON있는 핀을 입력으로 설정
}

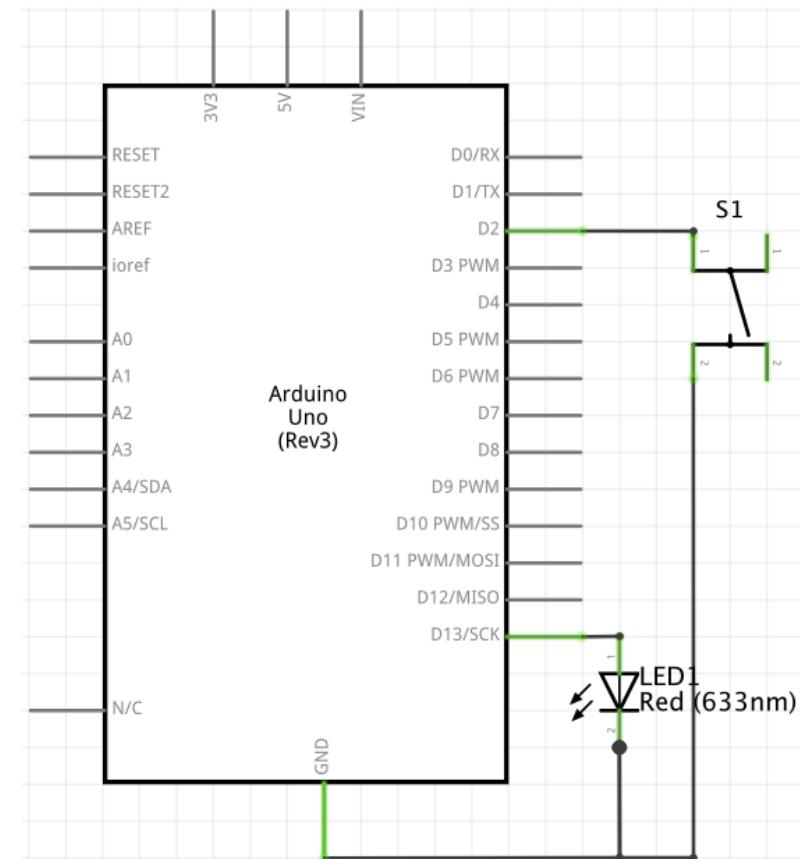
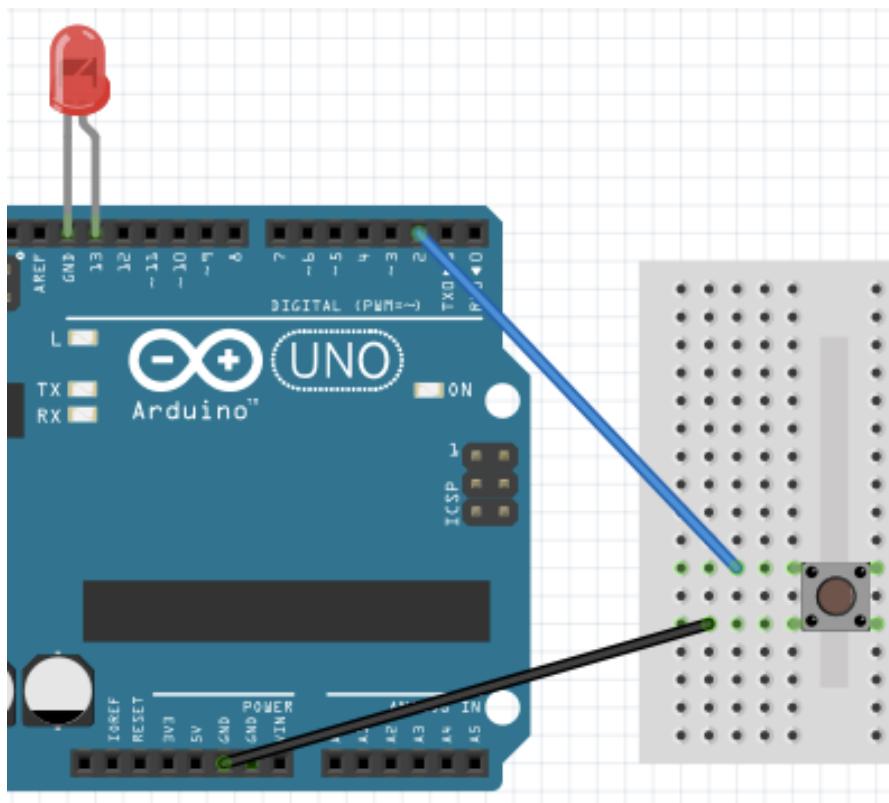
void loop(){
    val = digitalRead(BUTTON); // 버튼의 입력 값을 저장
    if(val == HIGH){           // 버튼이 눌림 상태인가?
        digitalWrite(LED, HIGH); //LED를 켠다
    }else{
        digitalWrite(LED, LOW); // LED를 끈다
    }
}
```

Digital I/O

Arduino

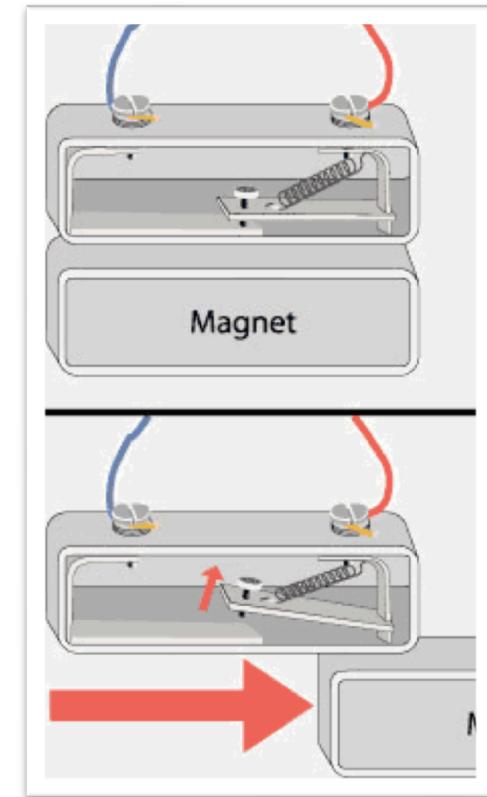
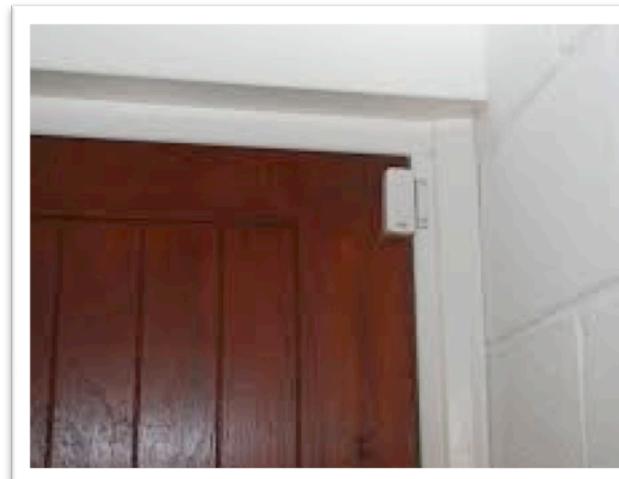
❖ 푸쉬버튼 회로(내부 풀업)

- 풀업저항을 MCU 내부적으로 제공
- pinMode(PIN, INPUT_PULLUP);
- 0,1번은 시리얼 통신과 동시 사용 불가



❖ Magnetic Door Switch

- 문열림 탐지
- 두개의 자석이 붙고 떨어짐에 따라 동작하는 스위치
- 필요 부품
 - 자석 도어 스위치
 - 10KΩ 저항
- Push Button 과 동일한 회로와 스케치 사용

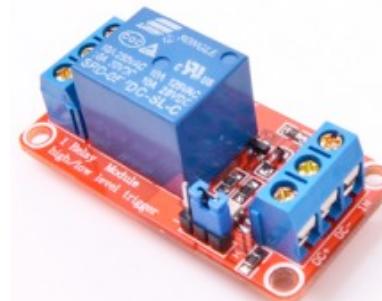


Digital I/O

Arduino

❖ 릴레이 스위치

- 220V 가전제품 켜고 끄기
- 보드의 5V 전원 제어 이외 가전제품
- 필요 부품
 - 릴레이 스위치
 - IN, 5V, GND
 - 무접점 반도체 릴레이
 - IN, GND
 - 220V 플러그 암/수



Digital I/O

Arduino

- ❖ 릴레이 스위치 회로구성



Digital I/O

Arduino

❖ 릴레이 스위치 스케치

- LED Blink와 동일

```
const int LED = 13;
int val = 0;
void setup(){
    Serial.begin(9600);
    pinMode(LED, OUTPUT);
}
void loop(){
    if(Serial.available() > 0){
        val = Serial.read();
        if(val == 49){ // ASCII '1'
            digitalWrite(13, HIGH);
        }
        if(val == 48){ // ASCII '0'
            digitalWrite(13, LOW);
        }
    }
}
```



Digital I/O

Arduino

❖ 동작 감지하기

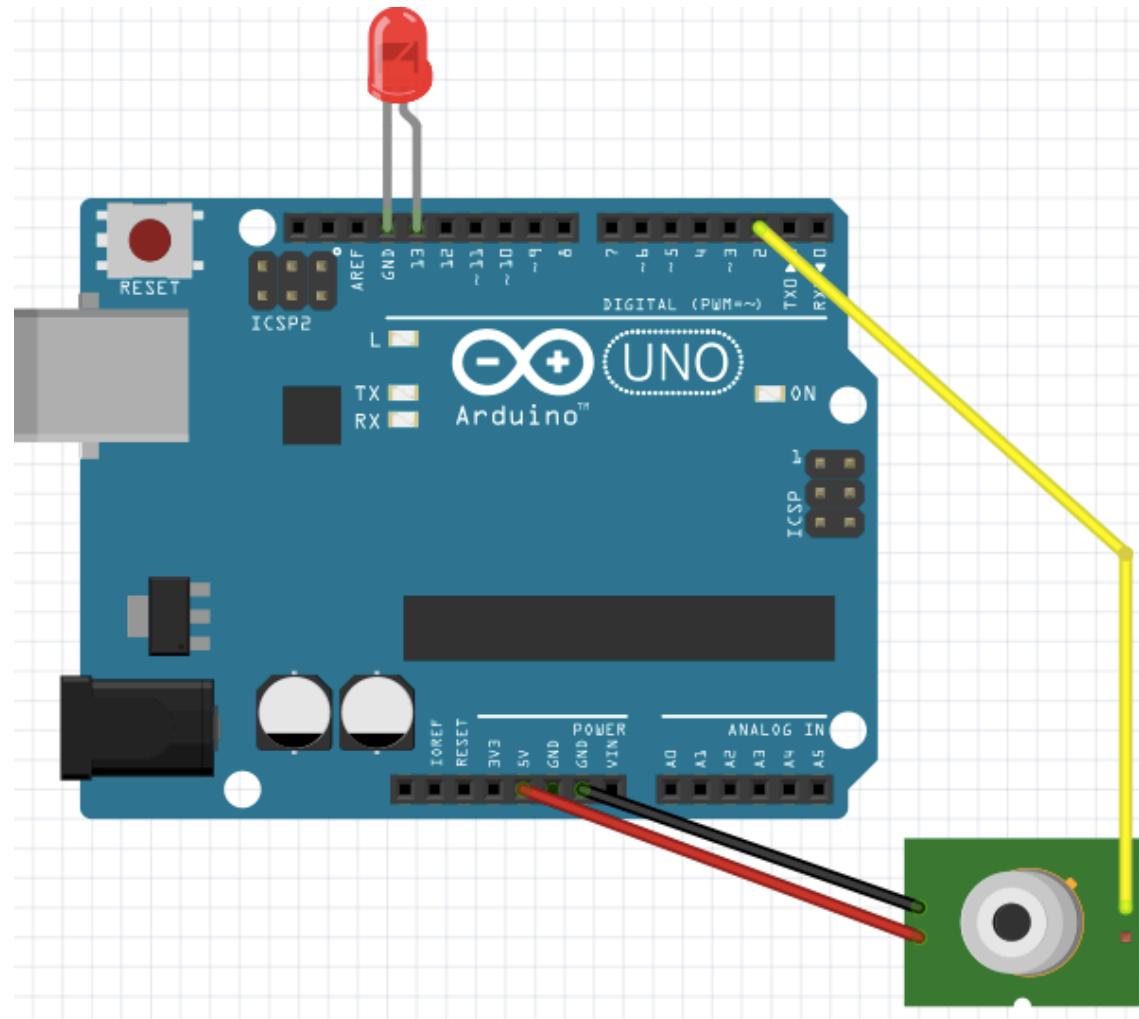
- 주변에 움직이는 것이 있는지 감지
- 사람이 움직이면 LED 켜짐
- 필요 부품
 - Passive Infrared, PIR) 센서
 - Out
 - GND
 - Vcc(5V)



Digital I/O

Arduino

❖ PIR 센서 회로구성



Digital I/O

Arduino

❖ PIR 센서 스케치

```
#define PIN_LED 13
#define PIN_PIR 2

void setup(){
    pinMode(PIN_LED, OUTPUT);
    pinMode(PIN_PIR, INPUT);
}

void loop(){
    int val = digitalRead(PIN_PIR);
    if(val == HIGH){
        digitalWrite(PIN_LED, HIGH);
    }else if(val == LOW){
        digitalWrite(PIN_LED, LOW);
    }
    delay(100);
}
```

세부목차

Arduino

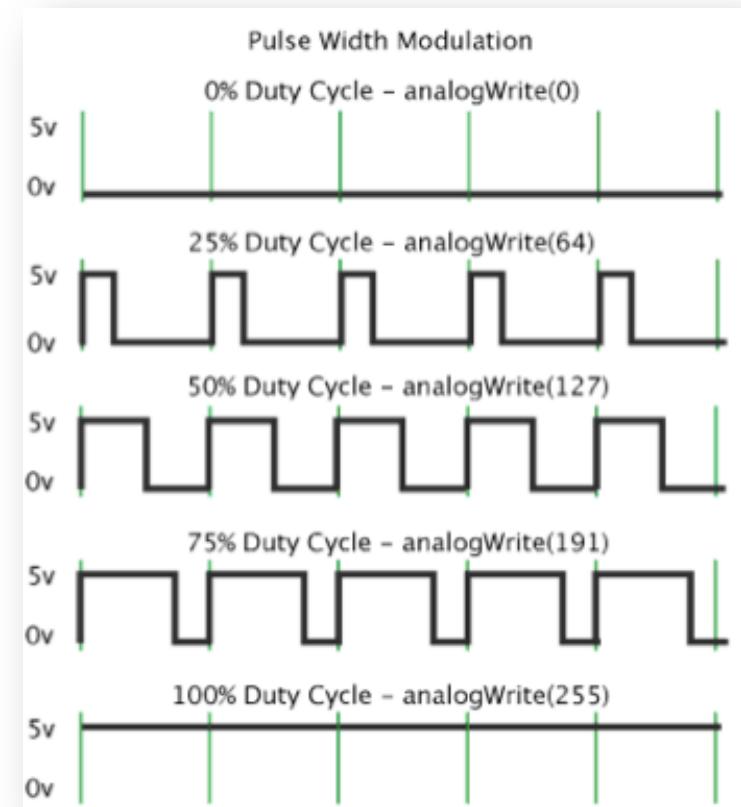
1. Introduction
2. 개발환경
3. Digital I/O
4. Analog I/O
5. 외부 라이브러리
6. 인터럽트와 타이머
7. Advanced I/O
8. 시리얼 통신
9. 아두이노 메모리

Analog I/O

Arduino

❖ Analog Out

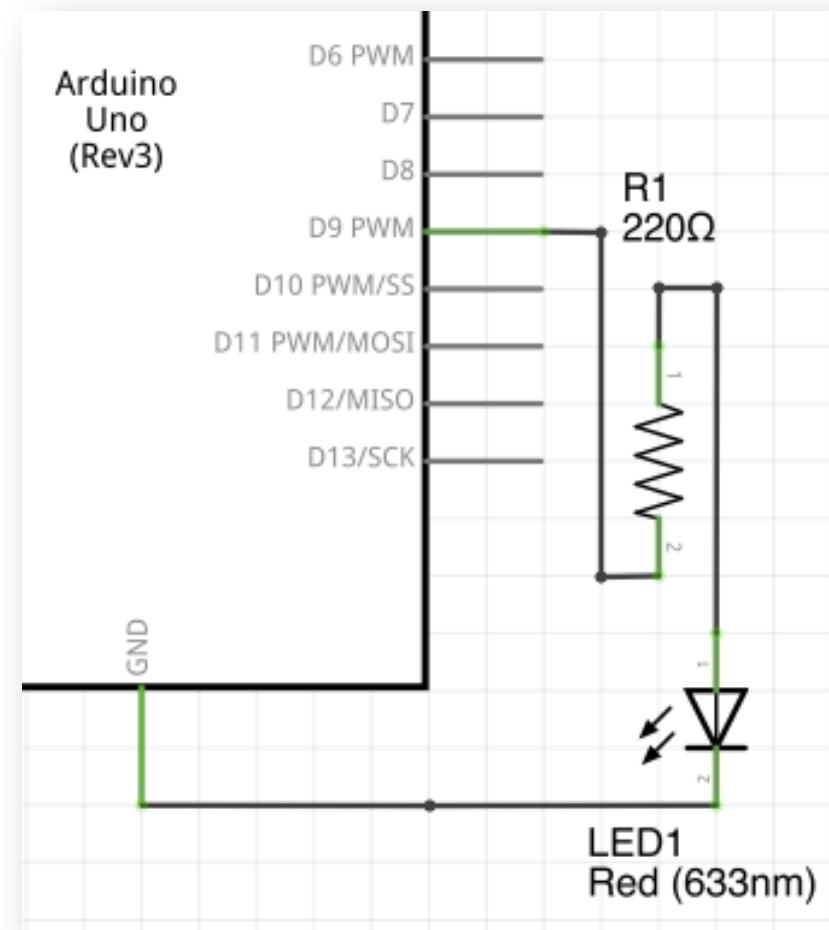
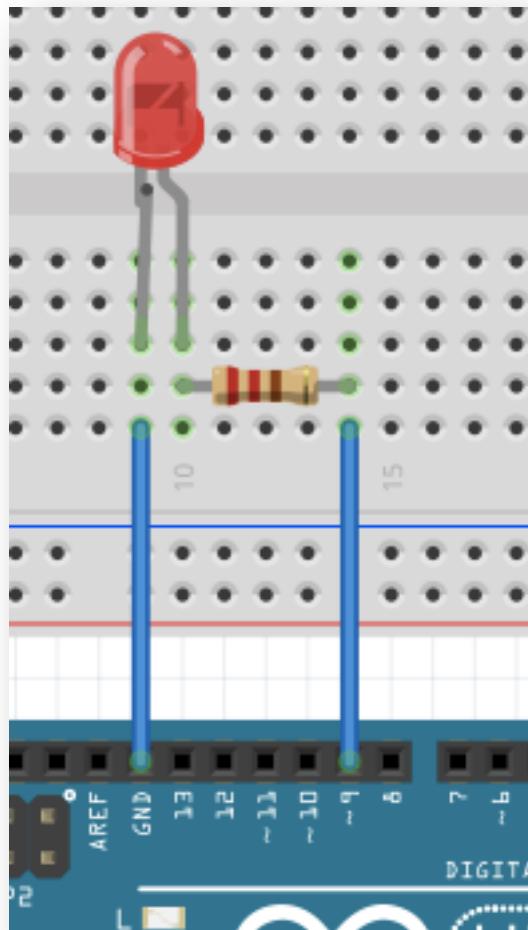
- PWM(Pulse Width Modulation) 펄스 폭 변조
 - PWM 지원 핀 : 물결표시(~) 3,5,6,9,10,11
 - Duty Cycle
 - 한 주기 내에서 HIGH 상태 시간 비율
 - 디지털 출력을 아날로그 처럼 흉내
 - 0~255(8Bit)
 - `analogWrite()`
- LED Fade
 - LED 서서히 꺼지고 켜짐
 - 필요 부품
 - 아두이노 보드
 - LED
 - 저항 220Ω
 - 회로 연결
 - 디지털 9번 - 저항 - LED - GND



Analog I/O

Arduino

❖ LED Fade 회로 구성



Analog I/O

Arduino

❖ LED Fade 스케치

```
#define LED 9

int i = 0;
void setup(){
    pinMode(LED, OUTPUT);
}

void loop(){
    for(i = 0; i<255; i++){
        analogWrite(LED, i);
        delay(10);
    }
    for(i = 255; i>0; i--){
        analogWrite(LED, i);
        delay(10);
    }
}
```

❖ Analog Input

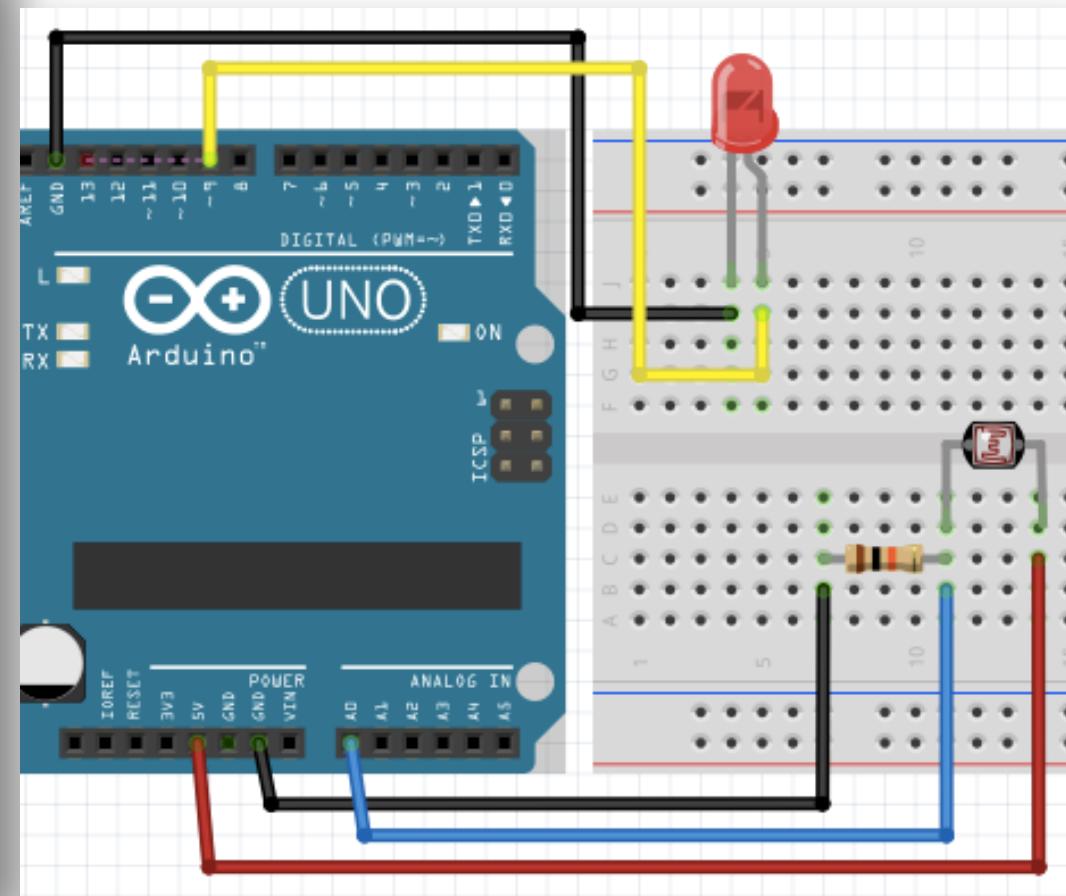
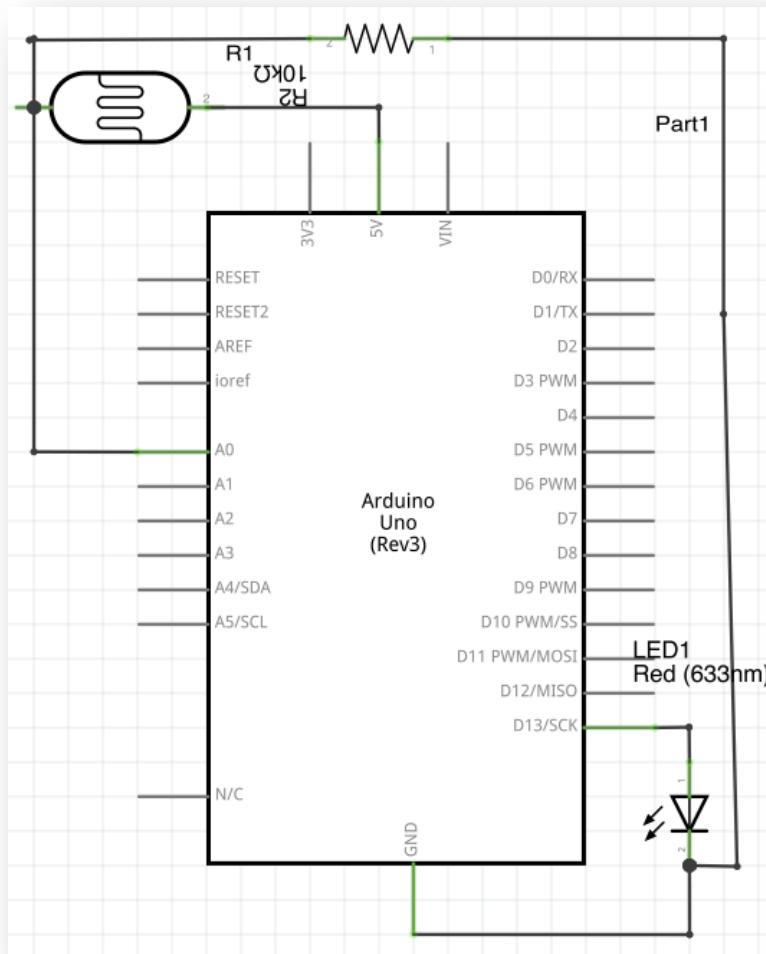
- ADC(Analog to Digital Converter) 내장
 - 아날로그-디지털 변환기
 - A0 ~ A5 (6개 핀)
 - 상수 A0 = 14, A1=15 순으로 선언
 - 10비트 해상도 : 0~1023
 - analogRead()
- 빛의 밝기에 따라 LED 밝기 조절
 - 필요 부품
 - LDR(Light Dependent Resistor)
 - 조광센서
 - 양음극 없음
 - 빛에 따라 저항값 변화, 가변저항
 - 10k Ω 저항
 - LED



Analog I/O

Arduino

❖ LDR(조도센서) 회로구성



Analog I/O

Arduino

❖ LDR(조도센서) 스케치

```
#define LED 9

int val = 0;
void setup(){
    pinMode(LED, OUTPUT);
}

void loop(){
    val = analogRead(0); // 0번 에서 값을 읽어옴
    analogWrite(LED, val/4); // 센서 밝기 값만큼으로 LED 켜기
    delay(10);
}
```

Analog I/O

Arduino

❖ Thermister

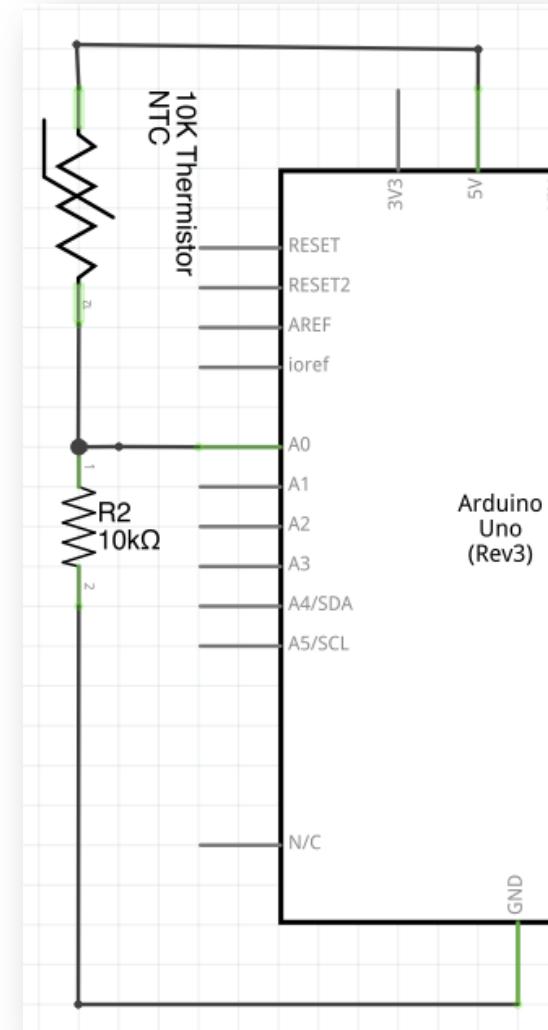
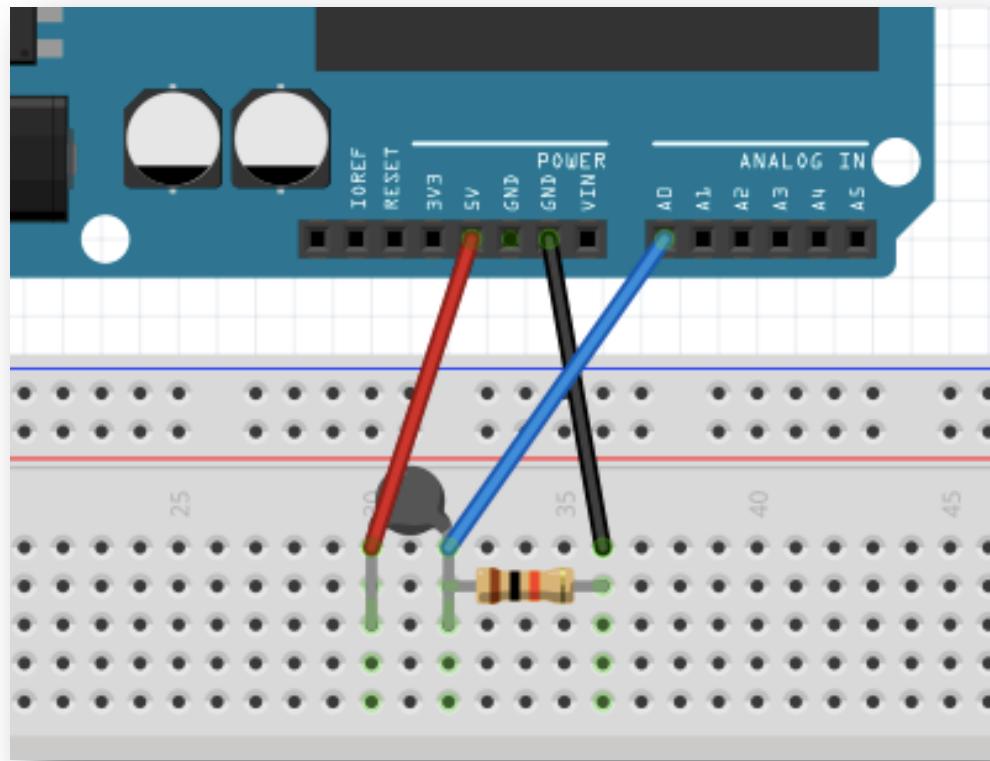
- 온도계 만들기
- 필요 부품
 - Thermistor(Thermally Sensitive Resistor)
 - 열가변 저항(NTC)
 - 온도 상승, 저항값 감소
 - 극 없음
 - Analog 포트
 - 10k Ω 저항
 - LED
- 온도 계산식
- 입력 전압 = $10K\Omega / \text{서미스터저항} + 10K\Omega$



Analog I/O

Arduino

❖ Thermister 회로구성



Analog I/O

Arduino

❖ Thermister 스케치

```
#include <math.h>

void setup(){
    Serial.begin(9600);
}

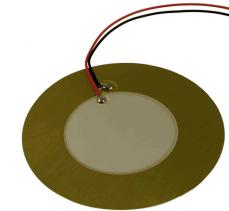
double thermister(int rawAdc){
    double temp;
    temp = log(((10240000/rawAdc) - 10000));
    temp = 1 / (0.00112914 + (0.000234125 * temp) +
        (0.0000000876741 * temp * temp * temp));
    temp = temp - 273.15;
    return temp;
}
void loop(){
    int rawAdc = analogRead(0);
    Serial.print(rawAdc);
    Serial.print("->");
    double temp = thermister(rawAdc);
    Serial.println(temp);
    delay(100);
}
```

Analog I/O

Arduino

❖ 진동, 충격 감지

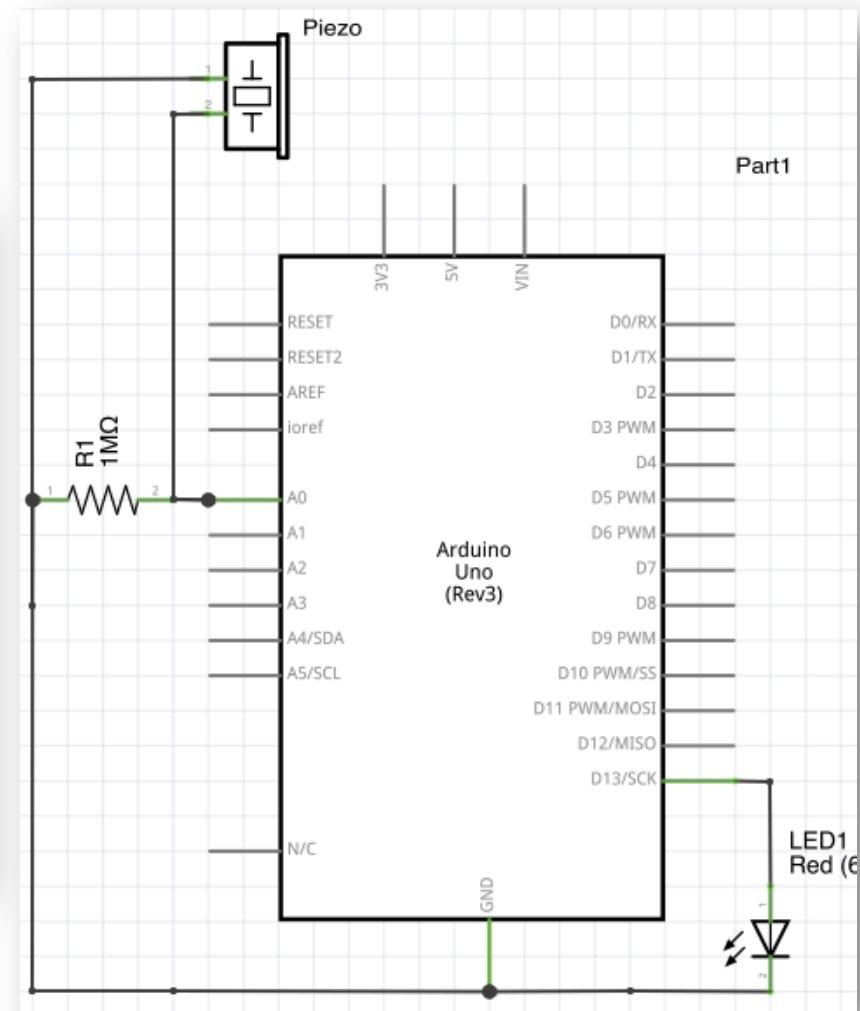
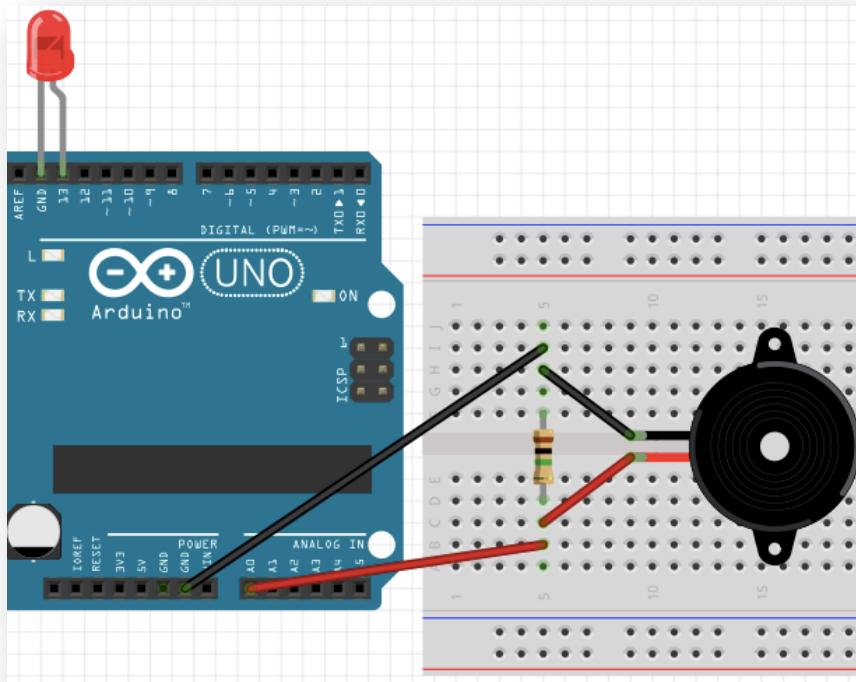
- Piezo 센서
 - 진동, 충격에 반응하여 전류 생산
 - 노크, 충격, 유리창 깨짐 등을 감지 가능
 - 필요 부품
 - Piezo Sensor
 - 1MΩ 저항
 - LED
 - 아두이노 보드



Analog I/O

Arduino

❖ Piezo 센서 회로 구성



Analog I/O

Arduino

❖ Piezo 스케치

```
#define PIN_LED 13
#define PIN_PIEZO A0
const int THRESHOLD = 100;

void setup(){
    pinMode(PIN_LED, OUTPUT);
    Serial.begin(9600);
}

void loop(){
    int val = analogRead(PIN_PIEZO);
    Serial.println(val);
    if(val >= THRESHOLD){
        digitalWrite(PIN_LED, HIGH);
    }else{
        digitalWrite(PIN_LED, LOW);
    }
    delay(100);
}
```

세부목차

Arduino

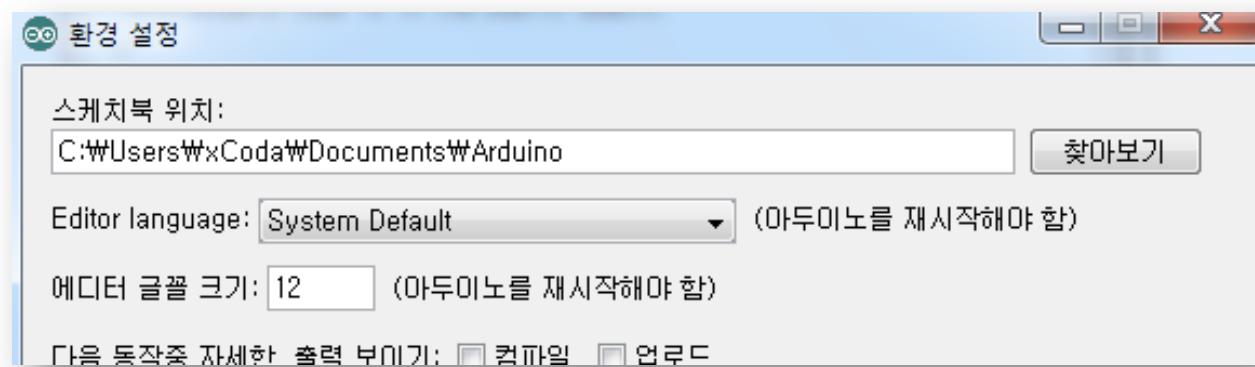
1. Introduction
2. 개발환경
3. Digital I/O
4. Analog I/O
5. 외부 라이브러리
6. 인터럽트와 타이머
7. Advanced I/O
8. 시리얼 통신
9. 아두이노 메모리

❖ 외부 라이브러리

- 아두이노 IDE에 기본 추가 되어 있지 않은 라이브러리
- 필요에 따라 제공받아 별도로 추가해야 한다.
- 자주 사용하는 기능은 라이브러리로 작성해서 재사용 가능
- libraries 폴더 내에 라이브러리 이름과 동일한 이름의 고유의 폴더

❖ 라이브러리 추가 방법

- 스케치북 위치
 - Windows : 파일 > 환경 설정
 - Mac : Arduino > 환경 설정
- libraries 디렉토리에 외부 라이브러리 디렉토리 파일 복사
 - 디렉토리이름에 특수 문자 사용 불가
- 아두이노 IDE 재 시작
- 스케치 > 라이브러리 가져오기 > 선택

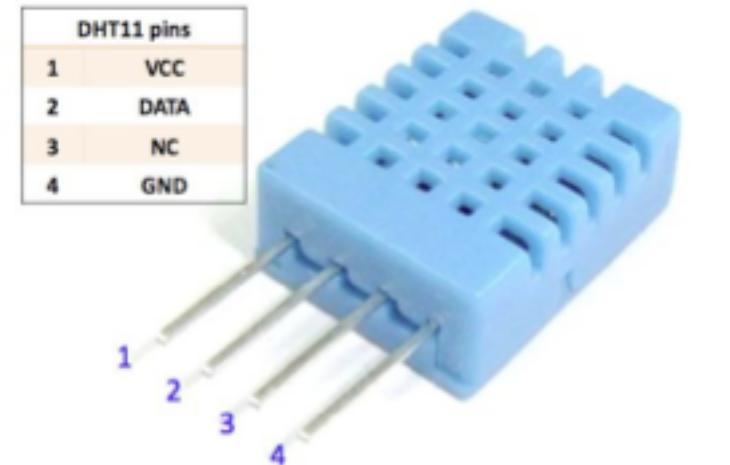


외부 라이브러리

Arduino

❖ DHT-11 온도/습도 모듈

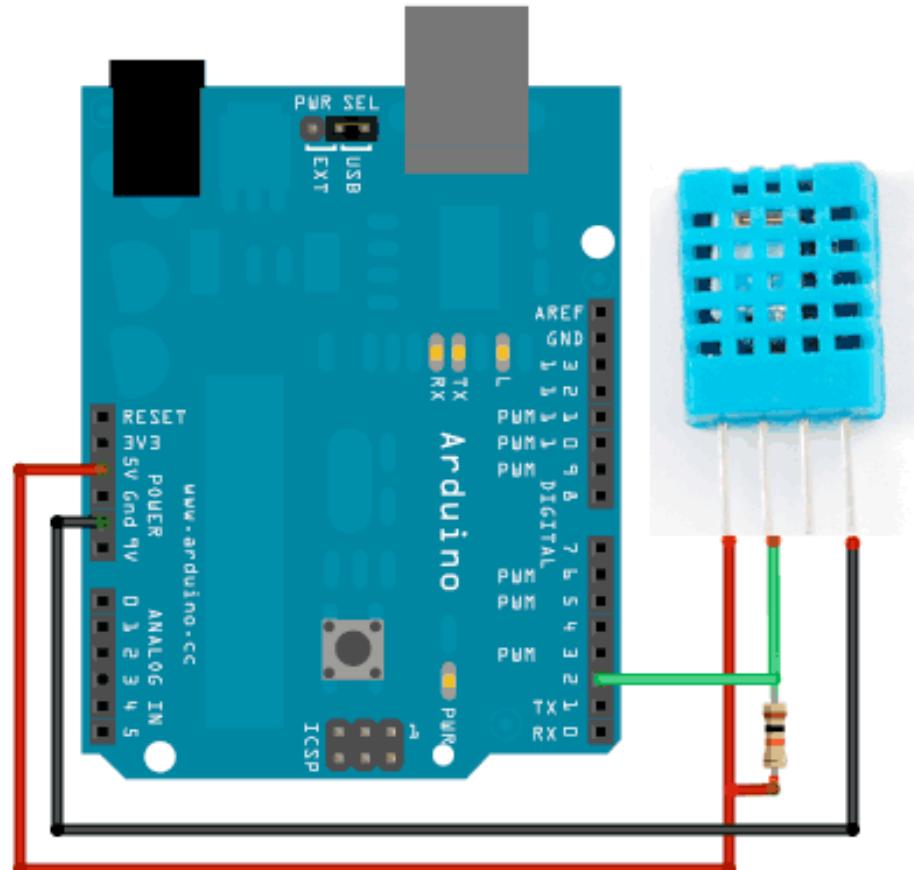
- 온도/습도 센서를 모듈로 구성
- 4핀
 - 1 : Vcc (3~5V)
 - 2 : Data Out
 - 10k 풀업저항 필요
 - 내부풀업(100k) 사용불가
 - 3 : 사용안함
 - 4 : GND
- 좀 더 성능이 좋은 DHT-22 도 있음
- <https://learn.adafruit.com/dht/overview>



외부 라이브러리

Arduino

- ❖ DHT-11 회로 구성

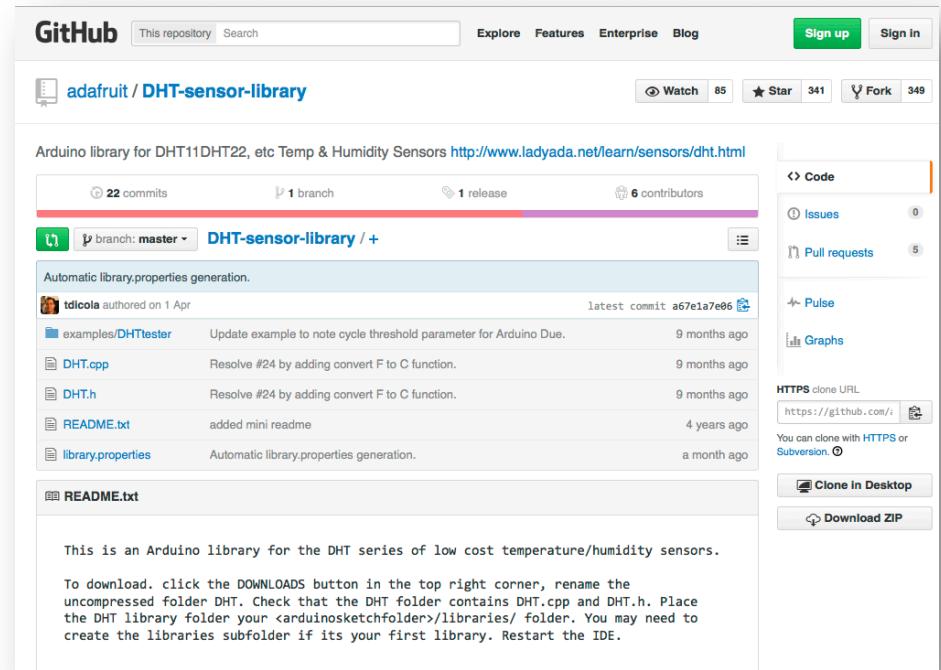


외부 라이브러리

Arduino

❖ DHT-11 라이브러리

- <https://github.com/adafruit/DHT-sensor-library>
- 압축 해제 후 폴더 이름을 "DHT"로 변경
- <아두이노스케치폴터>/libraries/에 복사
- IDE 재시작



외부 라이브러리

Arduino

❖ DHT-11 스케치 (다음에 계속)

```
#include "DHT.h"
#define DHTPIN 2      //Data 핀
#define DHTTYPE DHT11  // DHT 11

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  Serial.println("DHTxx test!");
  dht.begin();
}

void loop() {
  delay(2000);

  float h = dht.readHumidity(); // 습도
  float t = dht.readTemperature(); // 섭씨온도
  float f = dht.readTemperature(true); // 화씨온도
```

외부 라이브러리

Arduino

❖ DHT-11 스케치 (앞에서 계속)

```
if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
}

float hi = dht.computeHeatIndex(f, h); //열파지수

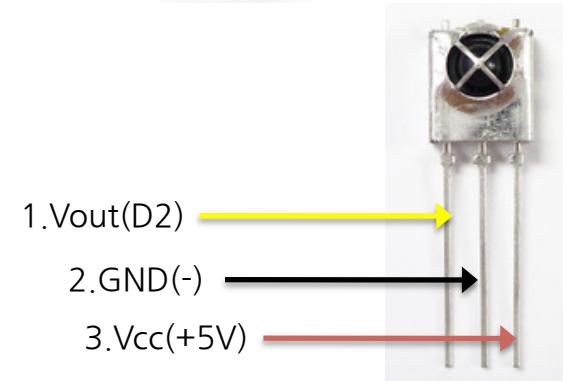
Serial.print("Humidity: ");
Serial.print(h);
Serial.print(" %\t");
Serial.print("Temperature: ");
Serial.print(t);
Serial.print(" *C ");
Serial.print(f);
Serial.print(" *F\t");
Serial.print("Heat index: ");
Serial.print(hi);
Serial.println(" *F");
}
```

외부 라이브러리

Arduino

❖ 적외선 센서

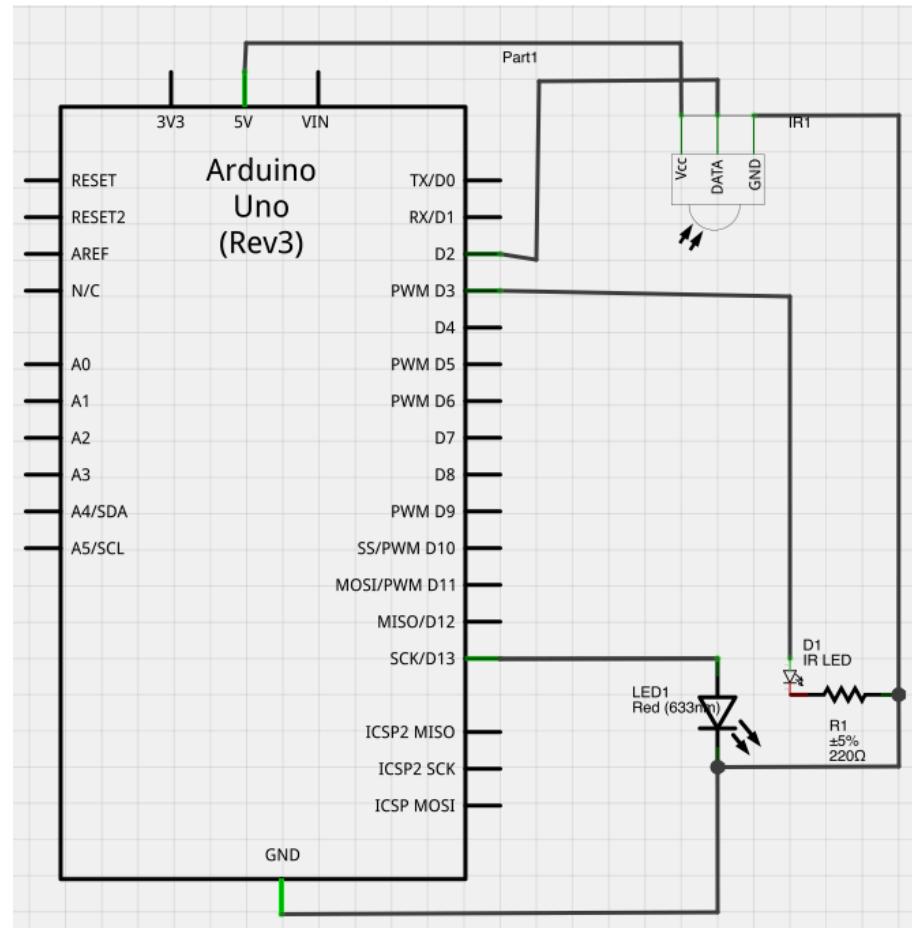
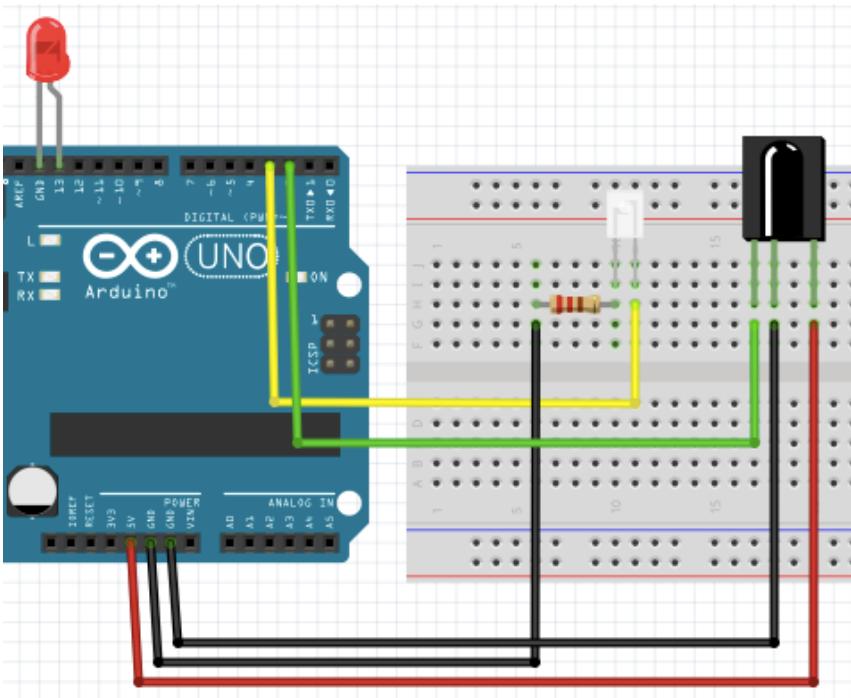
- 발신부와 수신부 사이의 장애물 감지
- 외부 침입 장치로 사용 가능
- 필요 부품
 - IR LED
 - 육안으로 켜진 상태 판별 불가
 - 디지털 카메라 뷰파인더로 판별
 - 고출력 회로 구성시 최대 5m
 - 220Ω 저항
 - IR 수광부
 - 핀1 : Vout
 - 핀2 : GND
 - 핀3 : Vcc
 - IR 송수신 Library
 - <http://www.righto.com/2009/08/multi-protocol-infrared-remote-library.html>
 - <https://github.com/shirriff/Arduino-IRremote>



외부 라이브러리

Arduino

❖ 적외선 센서 회로구성

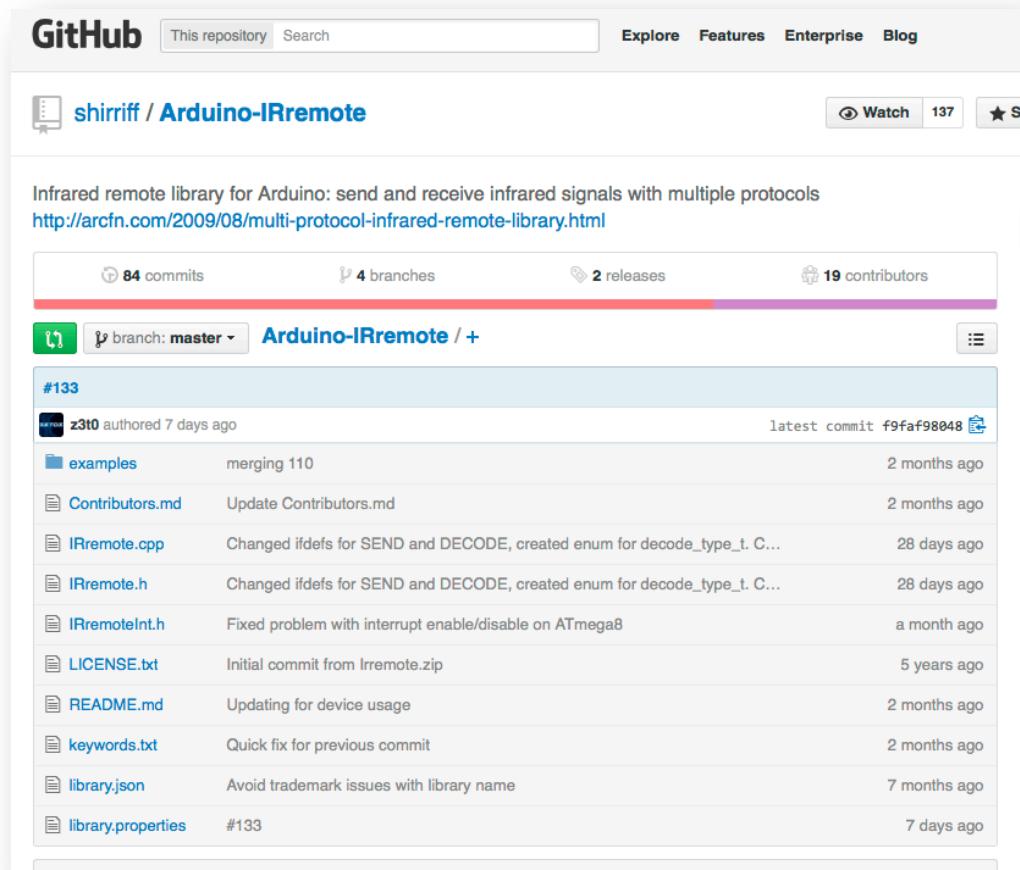


외부 라이브러리

Arduino

❖ IR 송수신 Library

- <http://www.righto.com/2009/08/multi-protocol-infrared-remote-library.html>
- <https://github.com/shirriff/Arduino-IRremote>



외부 라이브러리

Arduino

❖ 적외선 센서 스케치

```
#include <IRremote.h>

const int PIN_IRRCV = 2;
const int PIN_IRLED = 3;//Fixed in IRremote
const int PIN_LED = 13;
IRsend irsend;

void setup(){
    pinMode(PIN_IRRCV, INPUT);
    pinMode(PIN_LED, OUTPUT);
    irsend.enableIROut(38);
    irsend.mark(0);
}

boolean lightState = false;
unsigned long last = millis();

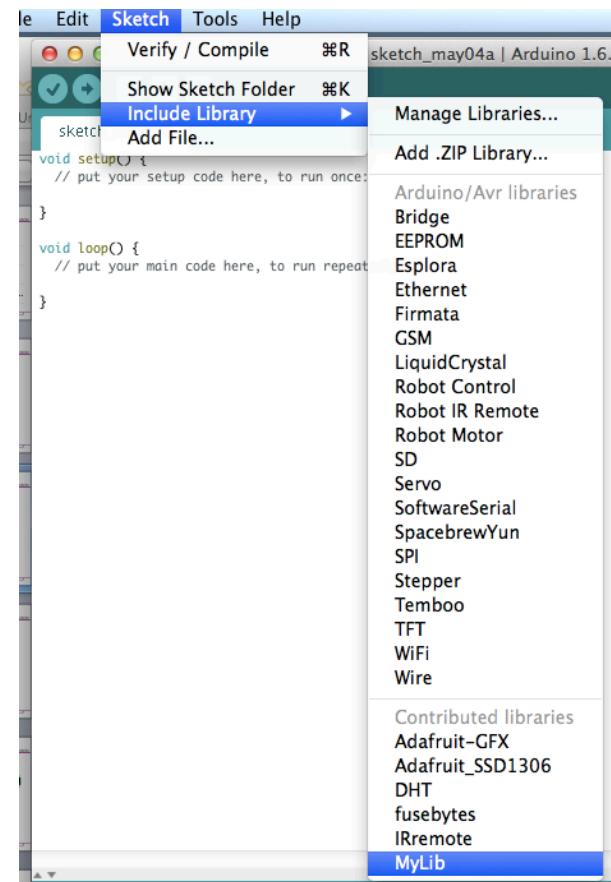
void loop(){
    int val = digitalRead(PIN_IRRCV);
    if(val == 0){
        if(millis() - last > 250){
            lightState = !lightState;
            digitalWrite(PIN_LED, lightState);
        }
        last = millis();
    }
    delay(100);
}
```

외부 라이브러리

Arduino

❖ 외부 라이브러리 만들기

- 자주 사용하는 기능을 라이브러리로 만들어 재 사용
- <arduino sketch>/libraries/에 라이브러리 이름의 폴더 생성
 - MyLib.h
 - MyLib.cpp
- 인사말을 전달하여 객체 생성
- 이름을 전달하여 인사말 출력



외부 라이브러리

Arduino

❖ 외부 라이브러리 만들기

- MyLib.h

```
#ifndef MY_LIB
#define MY_LIB

#include "Arduino.h"

class MyLib{
private:
    String _greeting;
public:
    MyLib(String greeting);
    void sayHello(String name);
};

#endif
```

외부 라이브러리

Arduino

❖ 외부 라이브러리 만들기

- MyLib.cpp

```
#include "Arduino.h"
#include "MyLib.h"

MyLib::MyLib(String greeting){
    _greeting = greeting;
}

void MyLib::sayHello(String name){
    Serial.println(_greeting + " " + name + "!!");
}
```

외부 라이브러리

Arduino

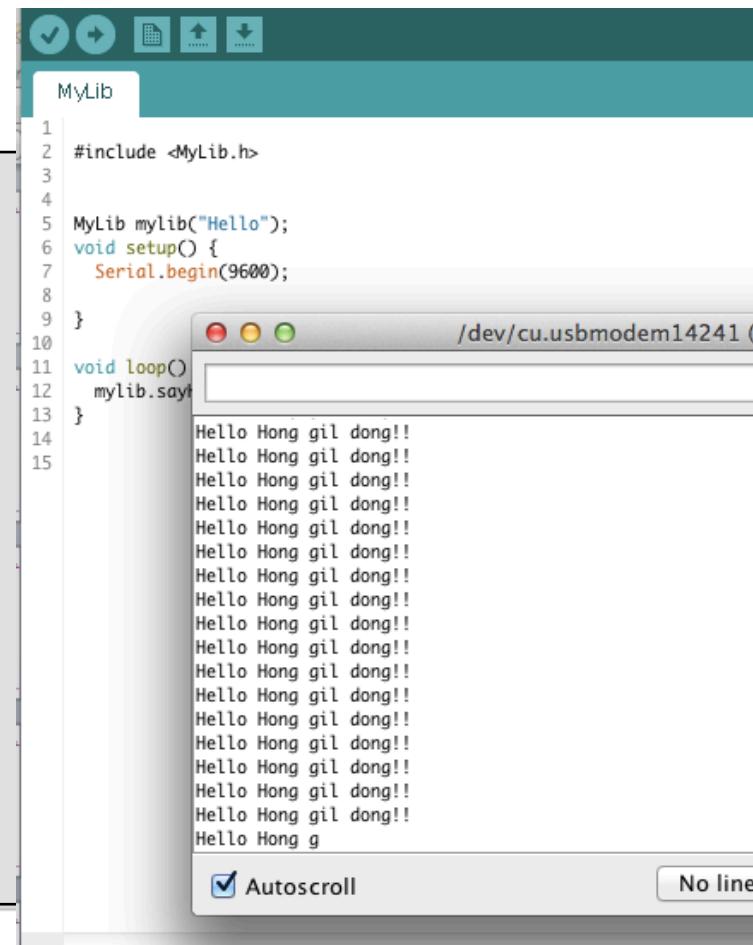
❖ 외부 라이브러리 만들기

- MyLib 사용 스케치

```
#include <MyLib.h>

MyLib mylib("Hello");
void setup() {
    Serial.begin(9600);
}

void loop() {
    mylib.sayHello("Hong gil dong");
}
```



세부목차

Arduino

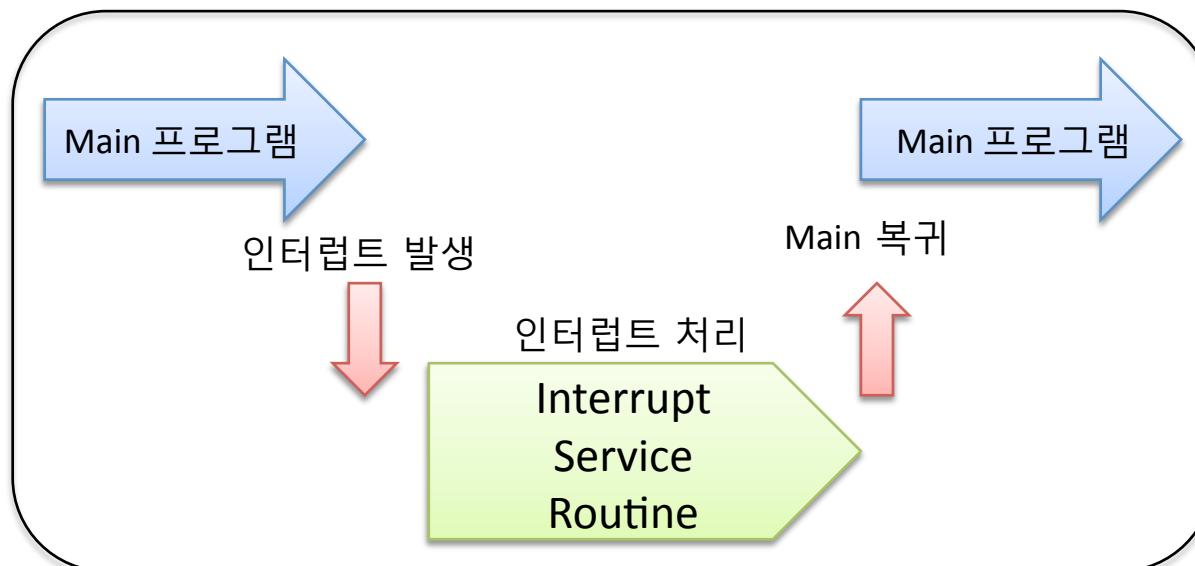
1. Introduction
2. 개발환경
3. Digital I/O
4. Analog I/O
5. 외부 라이브러리
6. 인터럽트와 타이머
7. Advanced I/O
8. 시리얼 통신
9. 아두이노 메모리

인터럽트와 타이머

Arduino

❖ 입력방식

- 폴링(Polling)
 - 주기적으로 핀의 상태 또는 값을 확인
 - delay() 함수를 이용하여 주기 조정
 - delay() 하는 동안 전체 프로그램이 중단
- 인터럽트(Interrupt)
 - 원하는 조건이 되면 지정된 함수(ISR)를 콜백
 - delay() 주기와 별개로 동작



인터럽트와 타이머

Arduino

❖ 인터럽트

- attachInterrupt(interrupt, *function, mode)
 - 인터럽트 발생하였을때 처리할 ISR(콜백 함수) 지정
 - interrupt : 인터럽트 번호
 - *function : ISR(Interrupt Service Routine)
 - 인터럽트 발생시 실행할 함수 포인터
 - 매개변수, 반환값 지정 불가능
 - ISR 내에서 변경하는 변수는 volatile로 선언
 - delay(), millis() 함수 사용 불가
 - Serial data 읽기 불가
 - 최대한 짧고 빠르게 수행하게 작성
 - mode : 인터럽트 발생 시점
- detachInterrupt(interrupt)
 - 인터럽트 처리 루틴 제거
 - interrupt : 인터럽트 번호
- interrupts()
 - 인터럽트 기능 활성화
 - AVR의 sei() 함수와 동일
- noInterrupts()
 - 인터럽트 기능 비활성화
 - AVR의 cli() 함수와 동일

인터럽트와 타이머

Arduino

❖ 인터럽트 번호와 핀

보드 (핀번호)	인터럽트 0	인터럽트 1	인터럽트 2	인터럽트 3	인터럽트 4	인터럽트 5
UNO	2	3				
Mega2560	2	3	21	20	19	18
Leonardo	3	2	0	1	7	

❖ 인터럽트 발생 시점

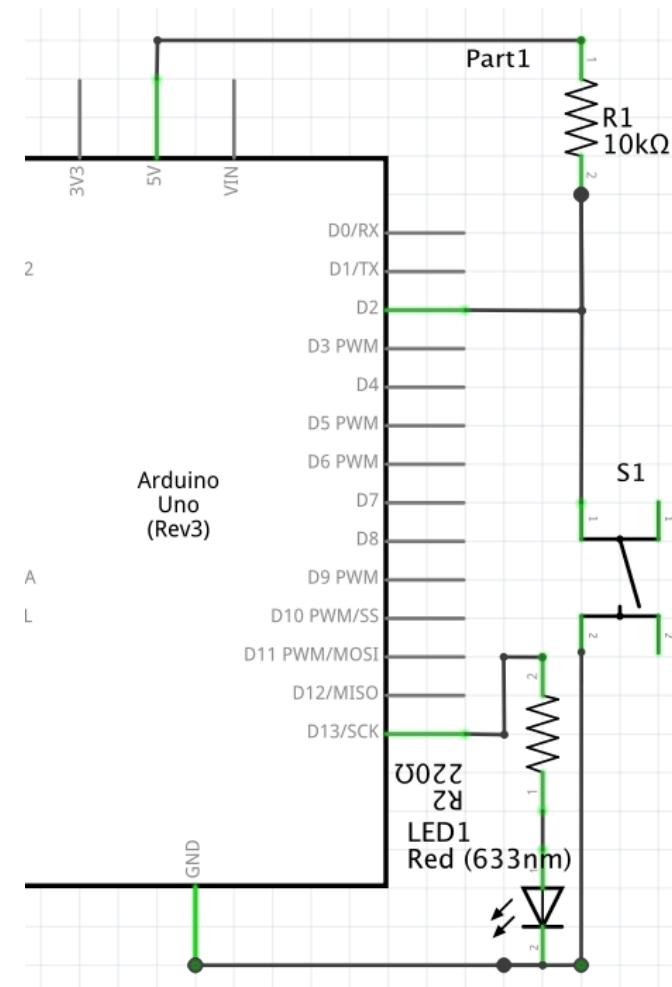
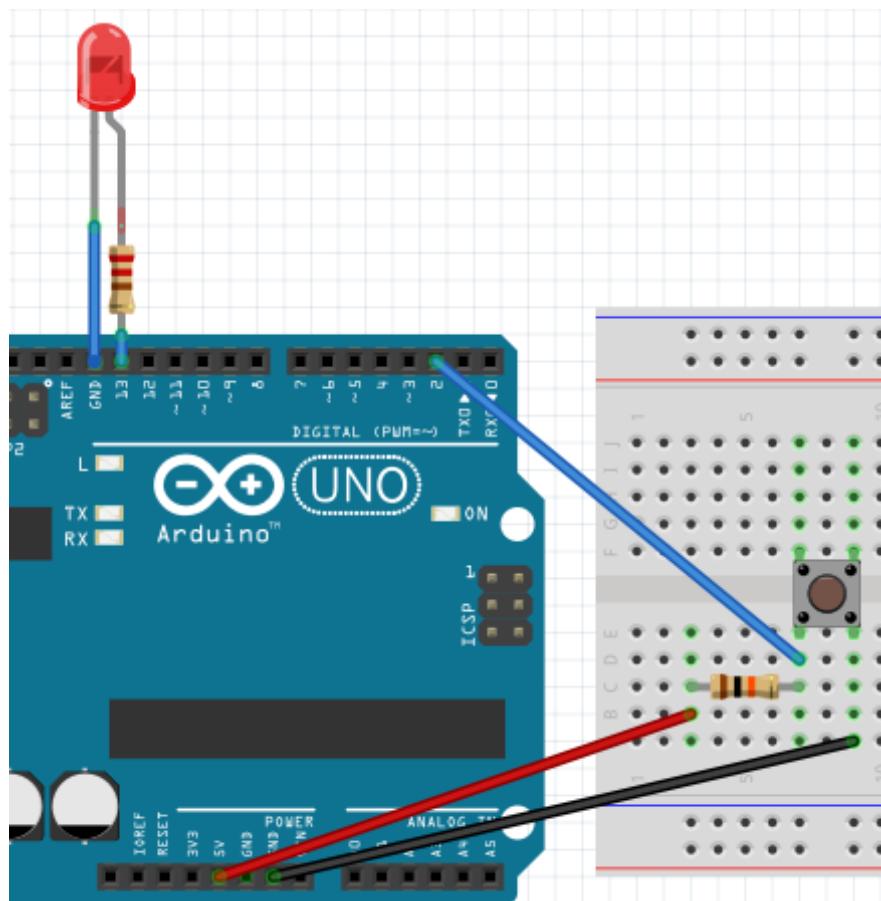
상수	인터럽트 발생 시점
LOW	입력값이 LOW일 때
CHANGE	입력값의 상태가 변화할 때 (RISING + FALLING)
RISING	입력값이 LOW에서 HIGH로 변할 때 (상승 엣지)
FALLING	입력값이 HIGH에서 LOW로 변할 때 (하강 엣지)
HIGH	입력값이 HIGH 일 때 (아두이노 두에 만 지원)

인터럽트와 타이머

Arduino

❖ 인터럽트를 이용한 스위치 입력

- 회로구성



인터럽트와 타이머

Arduino

- ❖ 인터럽트를 이용한 스위치 입력
 - 스케치

```
#define PIN_LED 13
volatile int state = LOW;

void setup() {
    pinMode(PIN_LED, OUTPUT);
    attachInterrupt(0, blink, CHANGE);
}

void loop() {
    digitalWrite(PIN_LED, state);
}

void blink(){
    state = !state;
}
```

❖ Time 함수

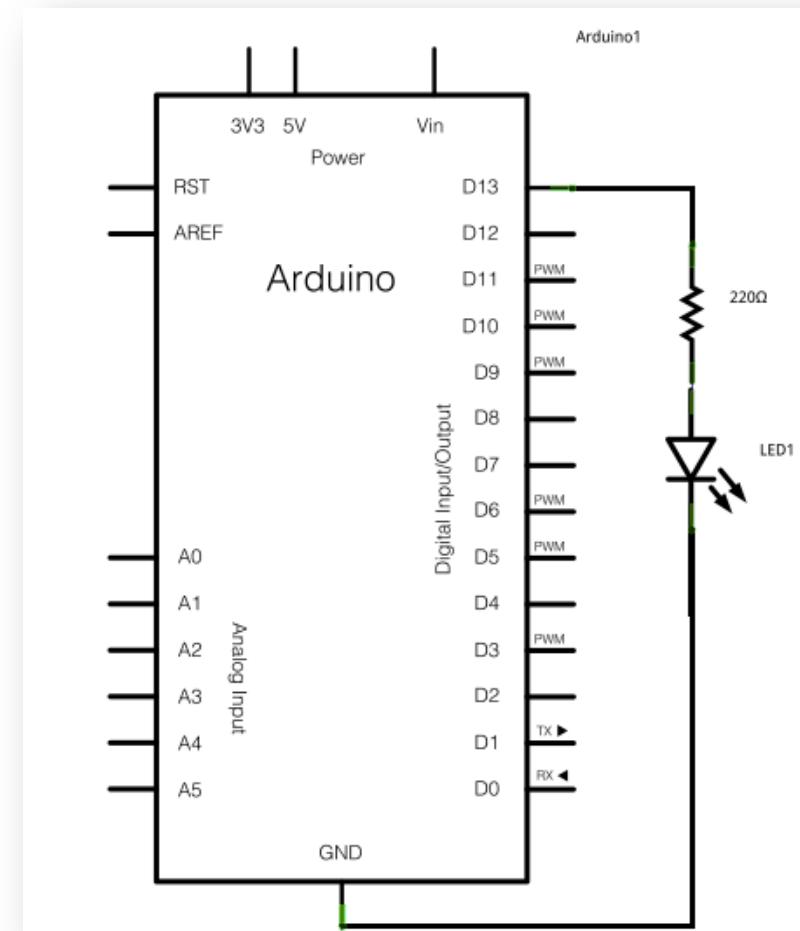
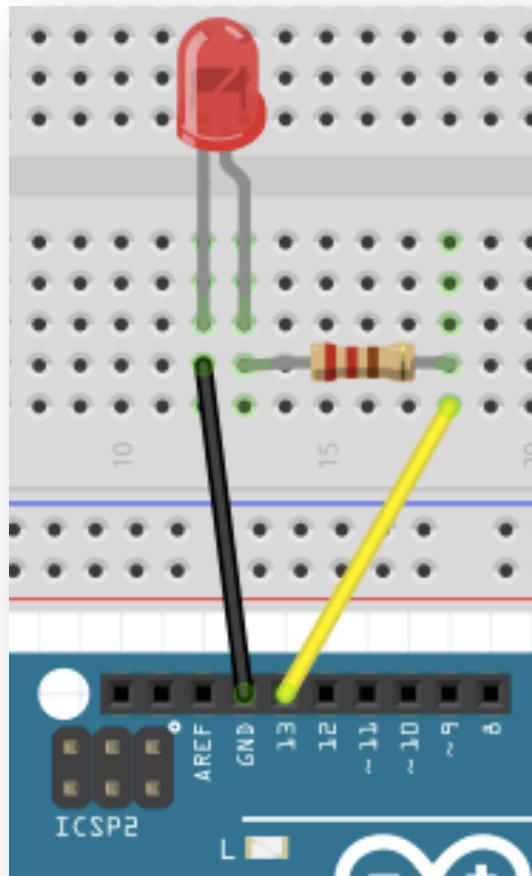
- millis() : unsigned long
 - 프로그램 시작된 이후의 경과 시간 반환
 - 1/1000초 (milliseconds)
 - 약 5일 경과후 오버플로, 0 초기화
- micros() : unsigned long
 - 프로그램 시작된 이후의 경과 시간 반환
 - 1/1000000초(microseconds)
 - 약 70분 경과 후 오버플로, 0 초기화
- delay(ms)
 - 지정한 시간 만큼 프로그램 실행 지연
 - milliseconds
 - 지연되는 동안 센서의 값을 읽지 못하는 문제점
 - 인터럽트 사용
 - millis() 이용한 경과시간 측정
- delayMicroseconds(us)
 - 지정한 시간 만큼 프로그램 실행 지연
 - Micro seconds
 - 최대 지연 시간은 16383 마이크로초, 초과시 delay() 사용

인터럽트와 타이머

Arduino

❖ Delay 함수 없이 LED Blink

- 회로구성



인터럽트와 타이머

Arduino

❖ Delay 함수 없이 LED Blink

- 스케치

```
#define PIN_LED 13

unsigned long lastMillis = 0;
long interval = 1000;
int state = LOW;
void setup() {
    pinMode(PIN_LED, OUTPUT);
}

void loop() {
    if(millis() - lastMillis > interval){
        lastMillis = millis();
        if(state == LOW){
            state = HIGH;
        }else{
            state = LOW;
        }
        digitalWrite(PIN_LED, state);
    }
}
```

❖ 타이머/카운터

- 시스템 클럭에 의해 동작
- 입력되는 펄스를 세는 장치-카운터
- 주기가 일정한 갯수-타이머
- delay(), millis(), PWM에서 사용
- 특정 시간 조건에 따라 ISR 실행
 - Time overflow : 타이머 카운트가 최대 값에 도달하는 경우
 - CTC(Clear timer on Compare match) : 정해진 값에 도달하는 경우
- 아두이노 우노(Atmega328) 3개의 timer
 - timer0
 - 8비트(256)
 - PWM 5, 6 핀
 - delay(), millis() 등 시간함수에서 사용
 - timer1
 - 16비트(65536)
 - PWM 9, 10 핀
 - Servo(서보모터 라이브러리)에서 사용
 - timer2
 - 8비트(256)
 - PWM 3, 11핀
 - tone() 함수에서 사용

인터럽트와 타이머

Arduino

❖ 타이머/카운터

- ISR(TIMER_XXX) 매크로 구현
- 인터럽트 비활성화
- 관련 레지스터를 설정
- 인터럽트 활성화

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

❖ 관련 레지스터

- TCCR (Timer/Counter Control Register)
 - TCCR1A, TCCR1B
 - 타이머 동작을 설정
- TCNT (Timer/Counter Register)
 - 타이머 값이 저장
- OCR(Output Compare Register)
 - 출력 비교 레지스터
- TIMSK (Timer/Counter Interrupt Mask Register)
 - 인터럽트 마스크 레지스터
- TIFR(Timer/Counter Interrupt Flag Register)
 - Pending 타이머 인터럽트 표시

인터럽트와 타이머

Arduino

❖ Timer를 이용한 LED Blink

- CTC 모드

```
#define ledPin 13
void setup(){
    pinMode(ledPin, OUTPUT);

    noInterrupts();                      // disable all interrupts
    TCCR1A = 0;
    TCCR1B = 0;
    TCNT1  = 0;

    OCR1A = 31250;                      // compare match register 16MHz/256/2Hz
    TCCR1B |= (1 << WGM12);           // CTC mode
    TCCR1B |= (1 << CS12);            // 256 prescaler
    TIMSK1 |= (1 << OCIE1A);          // enable timer compare interrupt
    interrupts();                        // enable all interrupts
}

ISR(TIMER1_COMPA_vect){             // timer compare interrupt service routine
    digitalWrite(ledPin, digitalRead(ledPin) ^ 1); // toggle LED pin
}

void loop(){}
```

❖ Timer를 이용한 LED Blink

- timer overflow 모드

```
#define ledPin 13
void setup(){
    pinMode(ledPin, OUTPUT);

    noInterrupts();                  // disable all interrupts
    TCCR1A = 0;
    TCCR1B = 0;

    TCNT1 = 34286;                 // preload timer 65536-16MHz/256/2Hz
    TCCR1B |= (1 << CS12);        // 256 prescaler
    TIMSK1 |= (1 << TOIE1);       // enable timer overflow interrupt
    interrupts();                   // enable all interrupts
}

ISR(TIMER1_OVF_vect){           // interrupt service routine
    TCNT1 = 34286;               // preload timer
    digitalWrite(ledPin, digitalRead(ledPin) ^ 1);
}

void loop(){}
```

❖ Metro 라이브러리

- millis() 함수를 이용한 라이브러리
- <http://playground.arduino.cc/Code/Metro>
- 주요함수
 - Metro(interval_ms[, autoreset])
 - 생성자
 - interval_ms : 경과시간
 - autoreset : 경과시간 초과시 자동 리셋 여부
 - interval(interval_ms)
 - 새로운 경과시간 설정
 - check() : char
 - 경과시간 만료 여부 (0 or 1) 반환
 - reset()
 - 객체 초기화 후 타이머 재시작

❖ MsTimer2 라이브러리

- 하드웨어 timer를 이용한 라이브러리
- <http://playground.arduino.cc/Main/MsTimer2>

인터럽트와 타이머

Arduino

❖ Metro 라이브러리

- #### ■ 실습 스케치

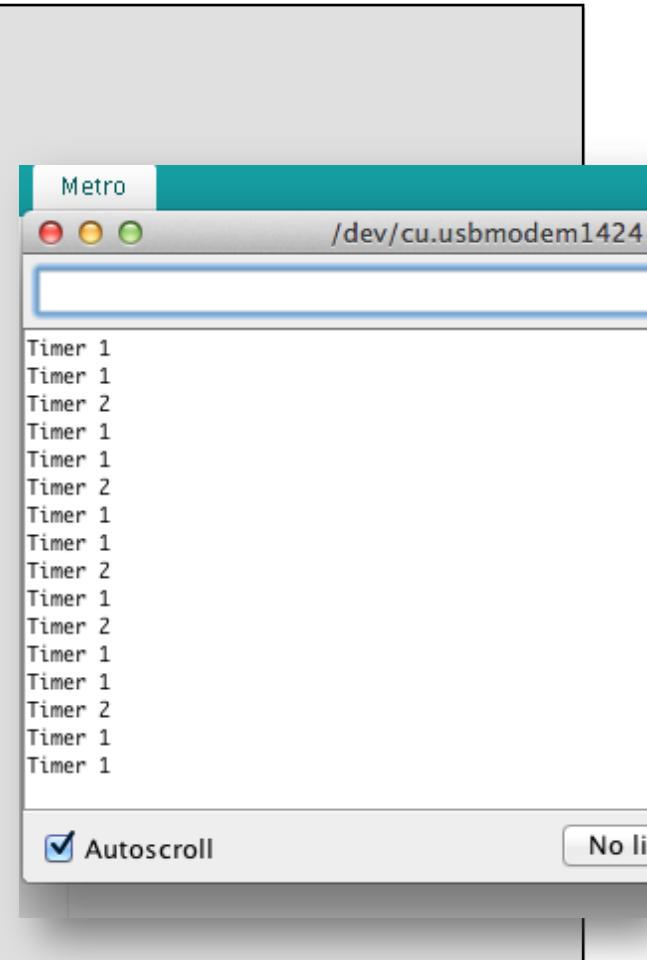
```
#include <Metro.h>
Metro metro0 = Metro(1000);
Metro metro1 = Metro(2000);

void setup(){
    Serial.begin(9600);
}

void loop(){

    if (metro0.check() == 1) {
        Serial.println("Timer 1");
    }

    if (metro1.check() == 1) {
        Serial.println("Timer 2");
    }
}
```



❖ MsTimer2 라이브러리

- 하드웨어 timer를 이용한 라이브러리
- <http://playground.arduino.cc/Main/MsTimer2>
- timer2를 사용
 - PWM 3, 11핀, analogWrite() 사용불가
 - 동시에 여러 타이머 사용 불가
- 주요함수
 - set(ms, *function)
 - 타이머 설정
 - ms : 경과시간 간격
 - *function : ISR 함수
 - start()
 - 타이머 인터럽트 활성화
 - stop()
 - 타이머 인터럽트 비활성화

인터럽트와 타이머

Arduino

❖ MsTimer2 라이브러리

- 실습 스케치

```
#include <MsTimer2.h>

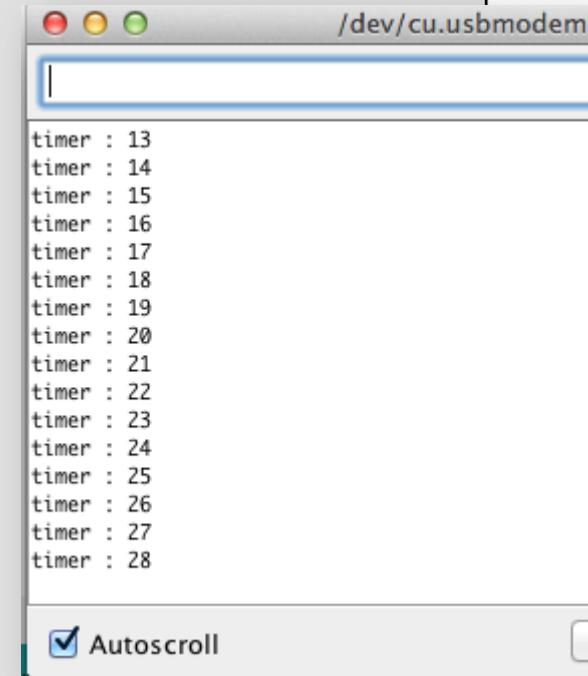
int count = 0;

void callback(){
    Serial.println("timer : " + String(count++));
}

void setup(){
    Serial.begin(9600);

    MsTimer2::set(1000, callback);
    MsTimer2::start();
}

void loop(){}
```



세부목차

Arduino

1. Introduction
2. 개발환경
3. Digital I/O
4. Analog I/O
5. 외부 라이브러리
6. 인터럽트와 타이머
7. Advanced I/O
8. 시리얼 통신
9. 아두이노 메모리

❖ 소리 출력

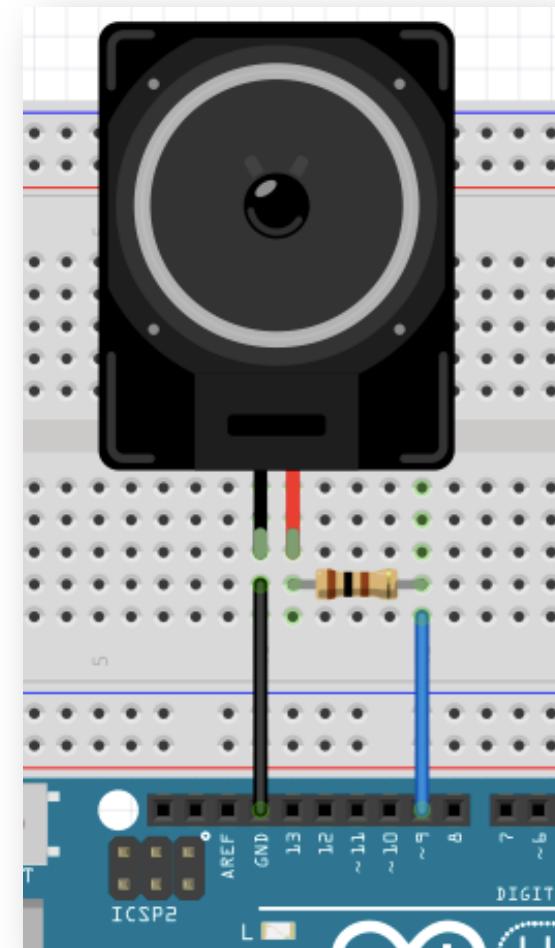
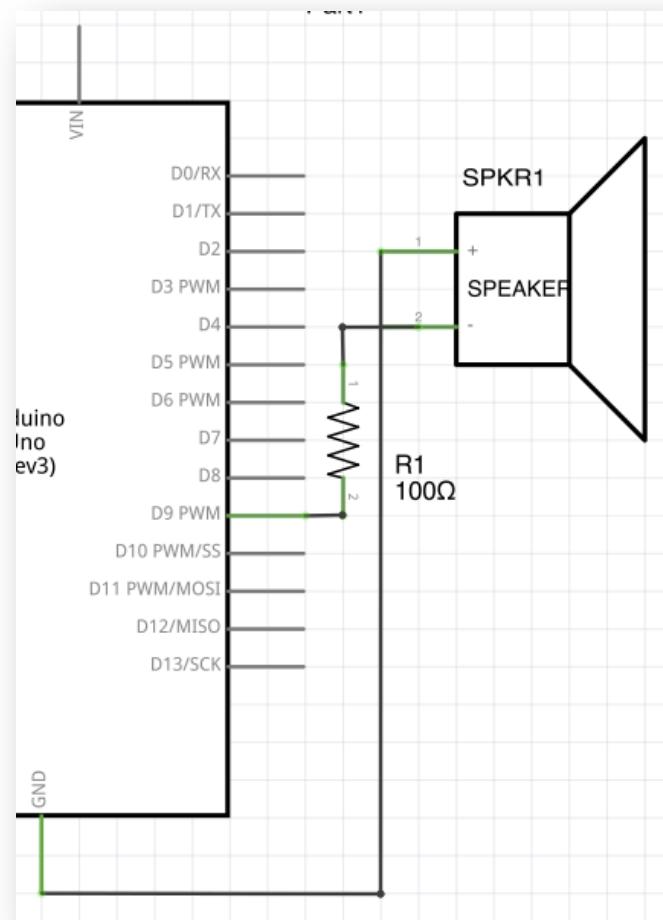
- Digital output
- 필요 부품
 - 부저 스피커 또는 Piezo
 - 가변저항(볼륨조절)
- 원시 함수
 - digitalWrite()
 - 소리를 내거나 끔
 - analogWirte()
 - 소리의 세기를 지정
 - 음 피치
 - 소리 출력 지연 시간 조정
 - delay(), delayMicroseconds()
- 고급 함수
 - tone(pin, frequency, duration)
 - 지정된 시간 동안 해당 주파수의 소리 출력
 - 시간 지정 없으면 noTone() 호출시 까지 출력
 - 타이머 사용으로 PWM 3,11 번 핀 동시 사용 불가
 - noTone(pin)
 - tone()에 의한 소리를 멈춘다.



Advanced I/O

Arduino

❖ 신호음 재생 회로 구성



Advanced I/O

Arduino

❖ 소리 출력

- digitalWrite 함수

```
#define PIN_SPEAKER 9

void setup () {
    pinMode (PIN_SPEAKER, OUTPUT);
}
void loop () {
    digitalWrite (PIN_SPEAKER, HIGH);
    delay (500);
    digitalWrite (PIN_SPEAKER, LOW);
    delay(500);
}
```

Advanced I/O

Arduino

❖ 소리 출력

- analogWrite 함수

```
#define PIN_SPEAKER 9

void setup () {
    pinMode (PIN_SPEAKER, OUTPUT);
}
void loop () {
    for(int i=100; i<256; i++){
        analogWrite (PIN_SPEAKER, i);
        delay (500);
        analogWrite (PIN_SPEAKER, 0);
        delay(500);
    }
}
```

Advanced I/O

Arduino

❖ 피치 출력

- 출력 사이의 지연 시간 조정

```
#define PIN_SPEAKER 9
void setup (){
    pinMode (PIN_SPEAKER, OUTPUT) ;
}

void loop (){
    for (int i = 0; i <100; i++){
        digitalWrite (PIN_SPEAKER, HIGH) ;
        delay (1) ;
        digitalWrite (PIN_SPEAKER, LOW) ;
        delay (1) ;
    }
    delay(1000);
    for (int i = 0; i <100; i++){
        digitalWrite (PIN_SPEAKER, HIGH) ;
        delayMicroseconds(1912/2);
        digitalWrite (PIN_SPEAKER, LOW) ;
        delayMicroseconds(1912/2);
    }
    delay(1000);
}
```

Advanced I/O

Arduino

❖ 경보음 출력

- tone() 함수

```
#define PIN_SPEAKER 9

void setup(){
}

void loop(){
    tone(PIN_SPEAKER, 392, 500);
    delay(500);
    tone(PIN_SPEAKER, 523, 500);
    delay(500);
}
```

Advanced I/O

Arduino

❖ 반짝반짝 작은별 스케치

```
#define PIN_SPEAKER 9

char noteNames[] = {'C', 'D', 'E', 'F', 'G', 'A', 'B'};
unsigned int frequencies[] = {262,294, 330, 349, 392, 440, 494};
const byte noteCount = sizeof(noteNames);
char score[] = "CCGGAAG FFEEDDC GGFFeed GGFFeed CCGGAAG FFEEDDC";
const byte scoreLen = sizeof(score);
void setup(){}
void loop(){
    for(int i=0; i<scoreLen; i++){
        int duration = 333;
        playNote(score[i], duration);
    }
    delay(4000);
}
void playNote(char note, int duration){
    for(int i=0; i < noteCount; i++){
        if(noteNames[i] == note){
            tone(PIN_SPEAKER, frequencies[i], duration);
        }
    }
    delay(duration);
}
```

❖ SPI 통신

- SPI 라이브러리의 소프트웨어 구현
- shiftOut(dataPin, clockPin, bitOrder, value) : void
 - dataPin : 데이터 출력 핀
 - clockPin : 클럭 펄스 핀
 - bitOrder : 비트 출력 순서, MSB or LSB
 - value : 출력 데이터 값(byte)
- shiftIn(dataPin, clockPin, bitOrder) : uint8_t
 - dataPin : 데이터 입력 핀
 - clockPin : 클럭 펄스 핀
 - bitOrder : 비트 입력 순서, MSB or LSB
 - 반환 : 입력 데이터 값(byte)

❖ 기타

- pluseln(pin, value, timeout) : unsigned long
 - pin : 입력 핀
 - value : 펄스 유형, HIGH or LOW
 - timeout(optional) : 펄스 시작을 기다리는 시간, 기본 1초
 - 지정한 핀으로 부터 HIGH 또는 LOW 값이 변화 될때 까지의 시간을 반환

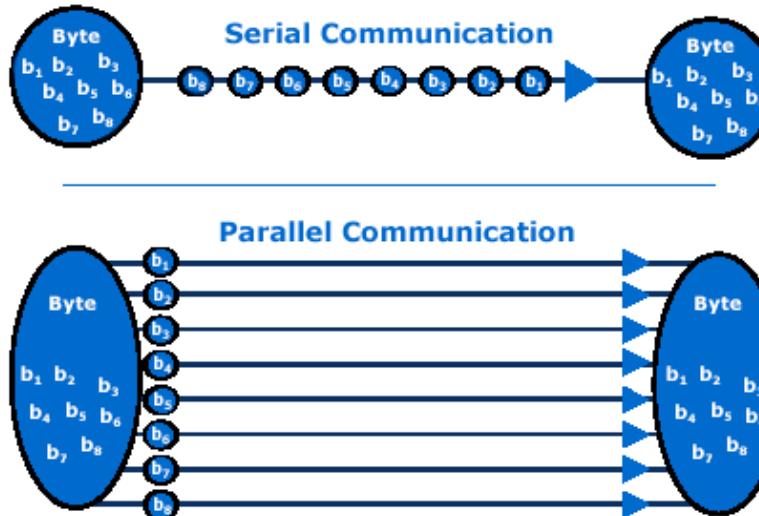
세부목차

Arduino

1. Introduction
2. 개발환경
3. Digital I/O
4. Analog I/O
5. 외부 라이브러리
6. 인터럽트와 타이머
7. Advanced I/O
8. 시리얼 통신
9. 아두이노 메모리

❖ 직렬통신 Vs 병렬통신

- 직렬(Serial) 통신
 - 여러 비트를 순차적으로 전송
 - 속도가 느리다
 - 통신 회선은 1개
- 병렬(Parallel) 통신
 - 동시에 여러 비트를 전송
 - 속도가 빠르다
 - 통신 회선은 전송 비트 수 만큼 필요



시리얼 통신

Arduino

❖ 직렬통신

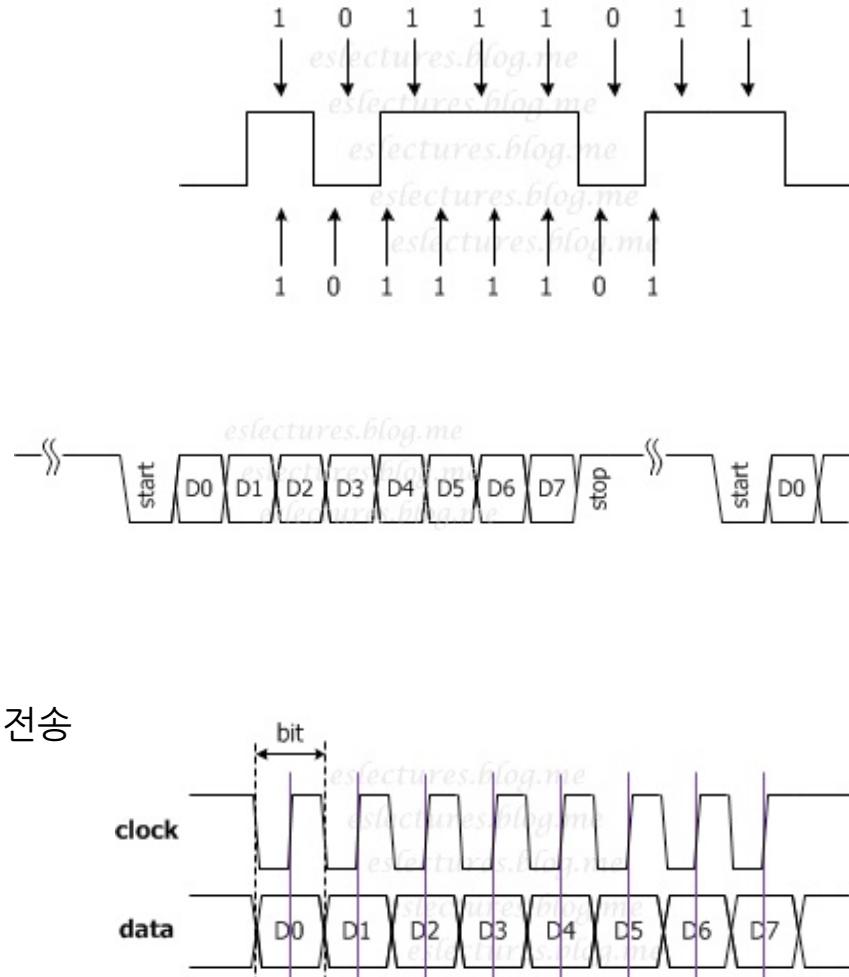
- 데이터를 한 비트씩 순차적으로 전송
- 어느 시점을 한 비트인지 구분 할 방법 필요
- 송수신자 간 비트 구분 시점에 대한 방식

❖ 비동기적(Asynchronous) 직렬통신

- 데이터 구분 주기를 서로 약속
- 클럭 신호를 따로 보내지 않음
- 양단간 통신속도가 맞지 않으면 통신 불능
- 시작비트와 정지비트가 추가로 필요
- 1:1 통신만 가능
- RS-232(UART) 통신 프로토콜이 대표적

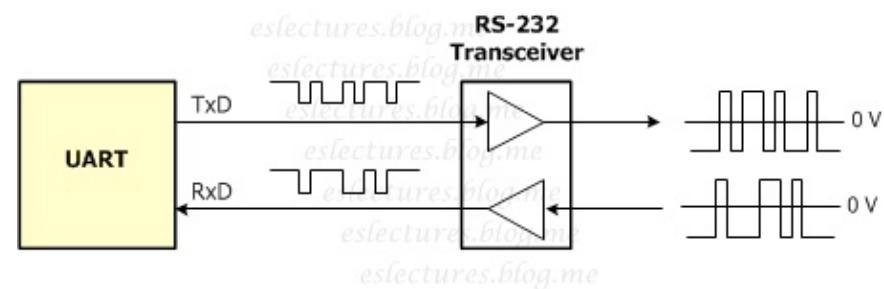
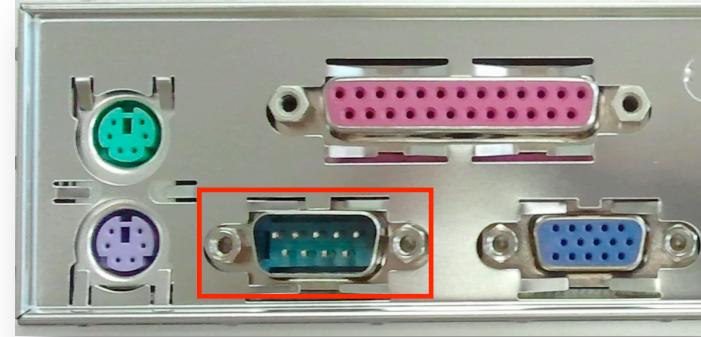
❖ 동기적(Synchronous) 직렬통신

- 데이터 신호와 비트 구분신호(Clock)를 별도로 전송
- 클럭에 마춰서 데이터 신호 인식
- 양단간 속도 약속 불필요
- 최고 속도 제한
- 1:N 통신 가능,
- Master/Slave 관계, Master가 클럭 주도
- I²C, SPI 통신 프로토콜이 대표적



❖ UART/RS-232

- UART(Universal Asynchronous Receiver Transmitter)
 - 비동기 통신을 위한 기계장치
 - 대부분의 MCU 하드웨어 기능 내장
 - 클럭과 시작/정지 비트를 소프트웨어로도 구현 가능
 - 통신속도 지정 : Baud Rate
 - 2개의 데이터 회선 사용
 - TxD(송신), RxD(수신)
- RS-232
 - 미국 EIA(Electronic Industries Association)
 - UART에 대한 전기적, 기계적(커넥터) 특성에 대한 표준
 - PC에서의 시리얼 포트를 대표
- UART/RS-232 통신
 - MCU UART TTL(Transistor to Transistor Logic) : 0~5V
 - RS232 : -12V ~ +12V
 - RS232 - TTL 변환칩
 - MAX232 / Maxim사
 - USB - TTL 변환칩
 - FT232R / FTDI Chip사
 - CP2102 / Silicon Lab사



❖ 아두이노 UART

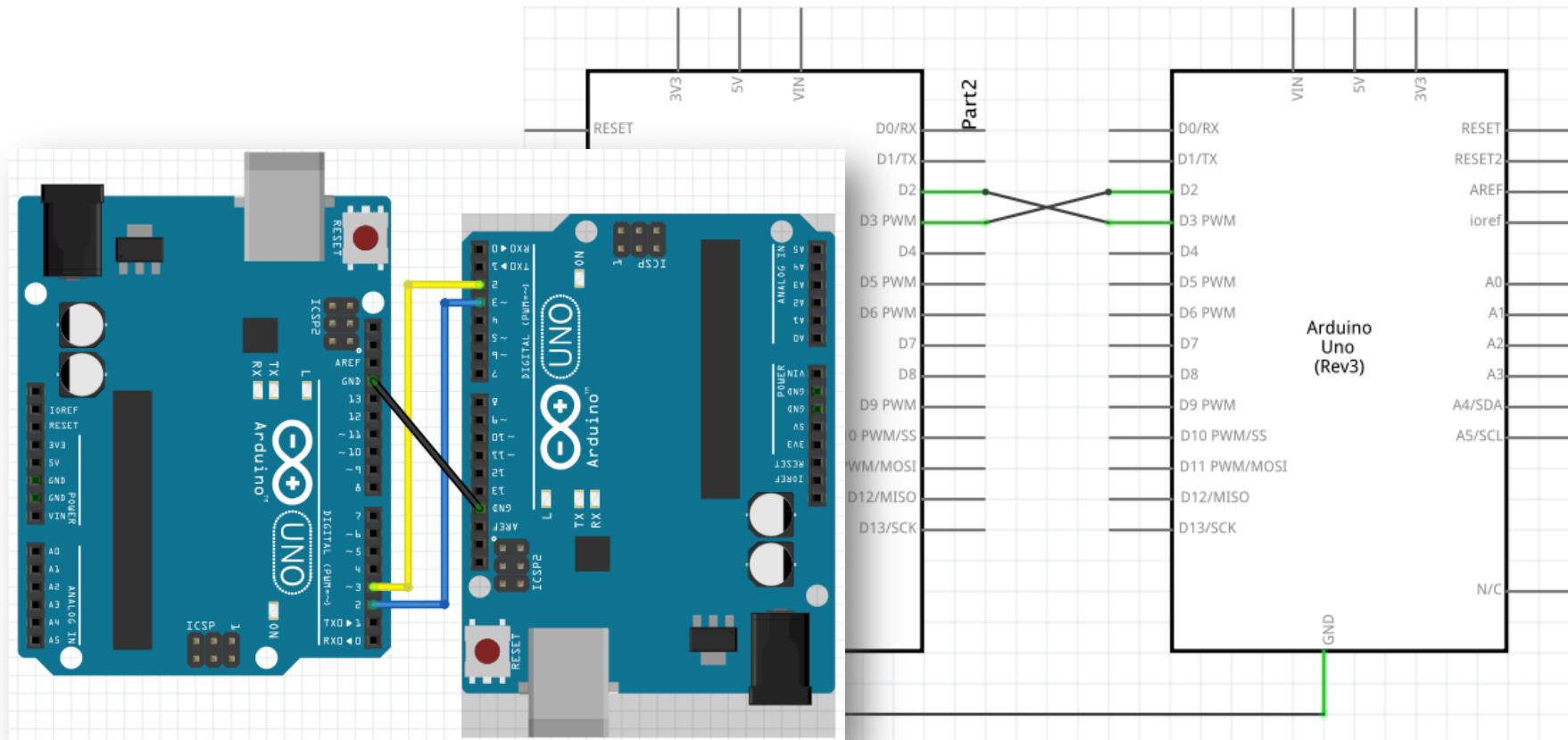
- Serial 클래스
 - 하드웨어 UART
 - 64Bytes 버퍼
 - USB-TTL 모듈 내장
 - RxD : 디지털 0번핀
 - TxD : 디지털 1번핀
 - 통신 중에 디지털 0,1 핀 사용 불가
- SoftwareSerial 클래스
 - 소프트웨어적 구현, 핀 사용 제약 없음
 - 기본 라이브러리 <SoftwareSerial.h>
 - 최대 통신 속도 115200bps 제약
 - 주요함수
 - SoftwareSerial(rx_pin, tx_pin [, inverse])
 - 생성자
 - begin(speed)
 - available()
 - read()
 - write()
 - print(), println()

시리얼 통신

Arduino

❖ 아두이노 2대 UART 통신

- 회로구성



시리얼 통신

Arduino

❖ 아두이노 2대 UART 통신

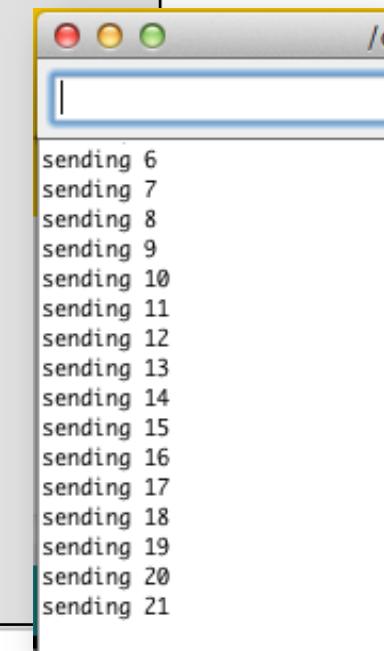
- 송신 측 스케치

```
#include <SoftwareSerial.h>

#define RX  2
#define TX  3

SoftwareSerial serial(RX, TX);
int count=0;
void setup() {
  Serial.begin(9600);
  serial.begin(9600);
  Serial.println("ready to send.");
}

void loop() {
  serial.println("Hello there!" + String(count++));
  Serial.println("sending " + String(count));
  delay(1000);
}
```



시리얼 통신

Arduino

❖ 아두이노 2대 UART 통신

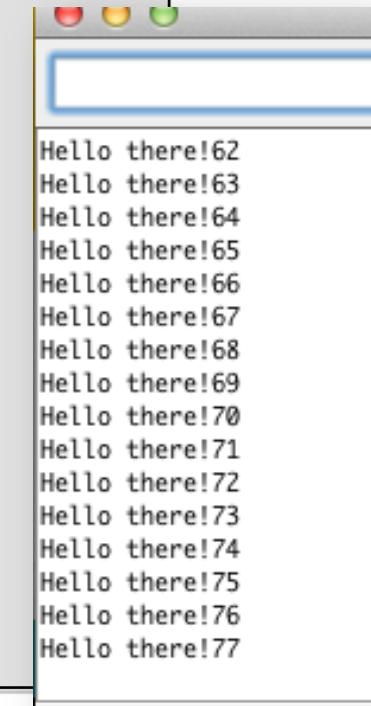
- 수신 측 스케치

```
#include <SoftwareSerial.h>

#define RX  2
#define TX  3

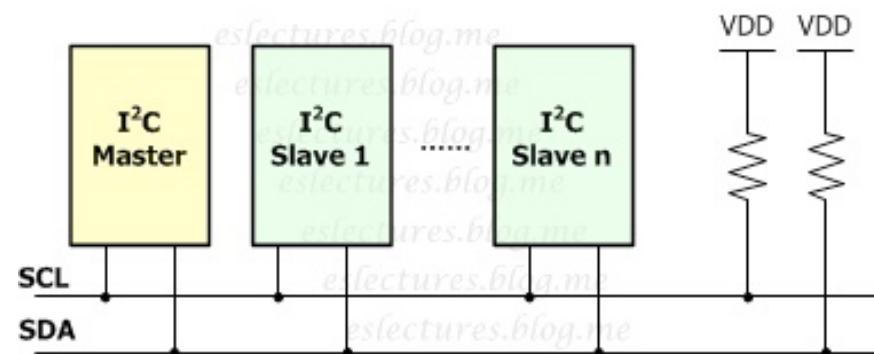
SoftwareSerial serial(RX, TX);
int count=0;
void setup() {
  Serial.begin(9600);
  serial.begin(9600);
  Serial.println("ready to receive.");
}

void loop() {
  if(serial.available()){
    Serial.write(serial.read());
  }
}
```



❖ I²C(Inter Integrated Circuit) 통신

- 동기적 직렬통신
- MCU와 주변장치간의 통신을 위한 프로토콜
- Philips 사 개발
- 통신 회선 : 2회선(TWI : Two Wire Interface)
 - SCL(Serial Clock)
 - SDA(Serial Data)
 - 각 회선 풀업저항 필요
- 실제 통신 회선은 1개(SDA)
 - 반이중 통신만 가능
 - 전송 속도 느림
- 하나의 버스에 여러장치 연결 가능
 - Slave 장치 고유의 주소(7비트)
 - 최대 128개(2^7) 슬레이브 장치
 - 장치 추가에 따른 회선 추가 없음



❖ 아두이노 I²C

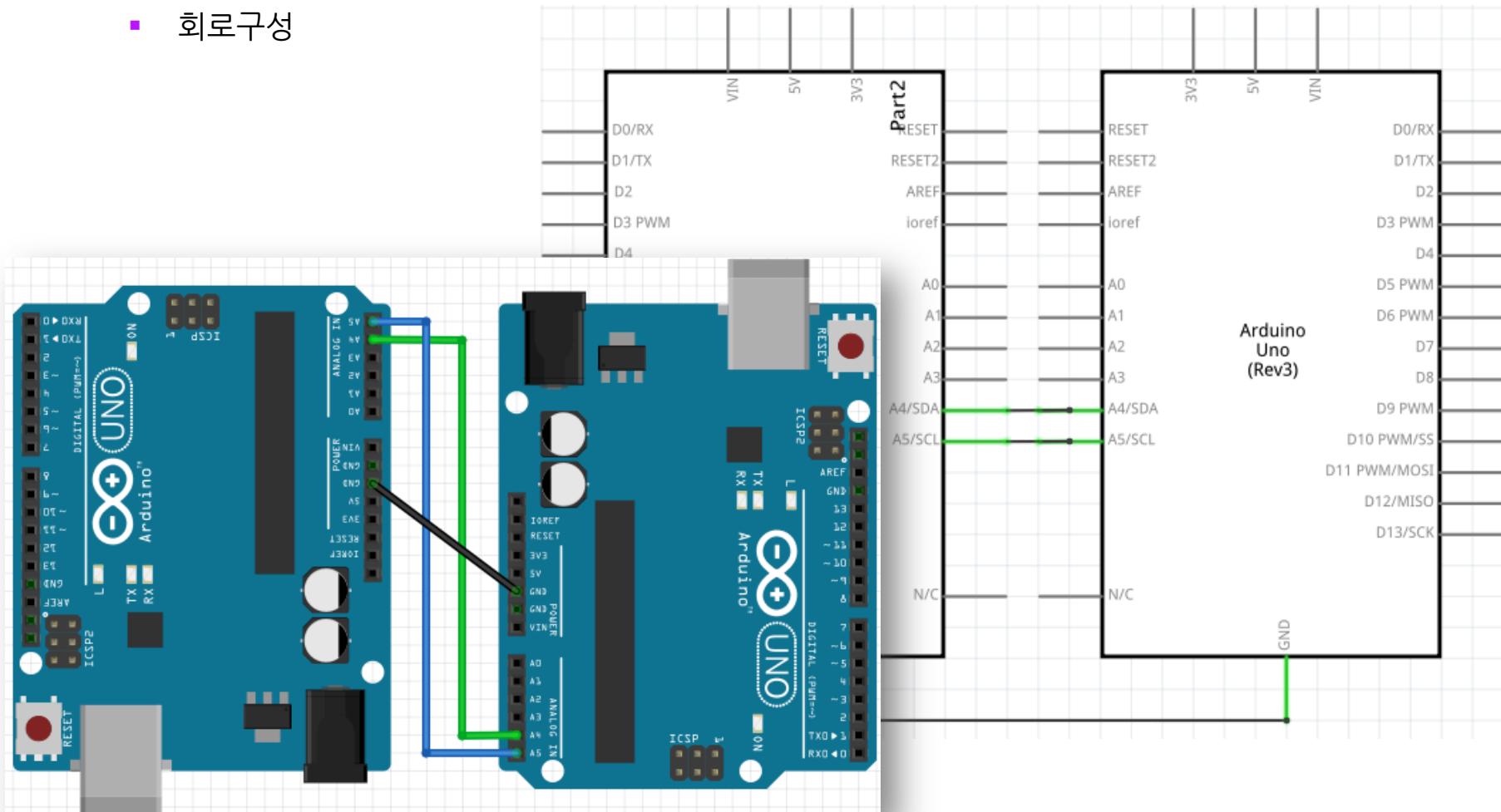
- 13번 핀 위쪽 끝에 SDA, SCL 핀
- Analog 4번(SDA), 5번(SCL) 연결
- 일반적으로 아두이노가 Master
- 주변장치가 Slave
- 기본 라이브러리
 - <Wire.h>
 - begin() : 마스터로서 시작
 - begin(address) : 슬레이브로서 시작
 - address : 슬레이브 주소
 - beginTransmission(address) : 슬레이브에게 전송시작
 - write() : 전송
 - endTransmission() : 전송 중지
 - available() : 버퍼 확인
 - read() : 1 바이트 읽기
 - requestFrom(address, quantity) : 전송 요청
 - onRequest(*function) : request event
 - onReceive(*function) : transmission event

시리얼 통신

Arduino

❖ 아두이노 2대간 I²C 통신

- 회로구성



- ❖ 아두이노 2대간 I²C 통신
 - Master Writer 스케치

```
#include <Wire.h>

void setup(){
    Wire.begin();
}

byte x = 0;

void loop(){
    Wire.beginTransmission(4); //slave 주소는 4
    Wire.write("x is ");
    Wire.write(x);
    Wire.endTransmission();

    x++;
    delay(500);
}
```

시리얼 통신

Arduino

❖ 아두이노 2대간 I²C 통신

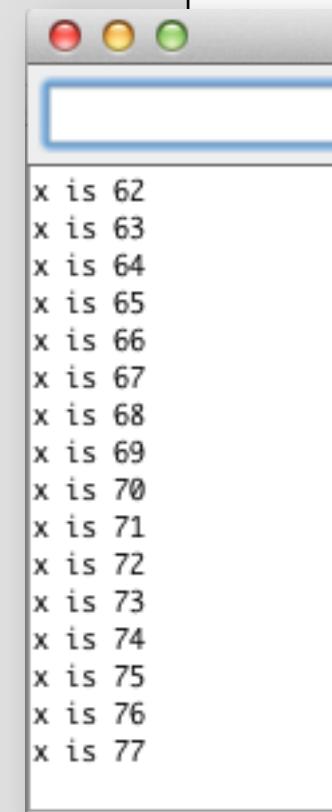
- Slave Receiver 스케치

```
#include <Wire.h>

void setup(){
    Wire.begin(4); //slave 주소는 4
    Wire.onReceive(receiveEvent);
    Serial.begin(9600);
}

void loop(){
    delay(100);
}

void receiveEvent(int howMany){
    while(1 < Wire.available()){
        char c = Wire.read();
        Serial.print(c);
    }
    int x = Wire.read();
    Serial.println(x);
}
```



시리얼 통신

Arduino

- ❖ 아두이노 2대간 I²C 통신
 - Master Reader 스케치

```
#include <Wire.h>

void setup(){
    Wire.begin();
    Serial.begin(9600);
}

void loop(){
    Wire.requestFrom(2, 6); // slave 주소는 2
    while(Wire.available()){
        char c = Wire.read();
        Serial.print(c);
    }
    Serial.println();
    delay(500);
}
```



- ❖ 아두이노 2대간 I²C 통신

- Slave Sender 스케치

```
#include <Wire.h>

void setup(){
    Wire.begin(2); // slave 주소는 2
    Wire.onRequest(requestEvent);
}

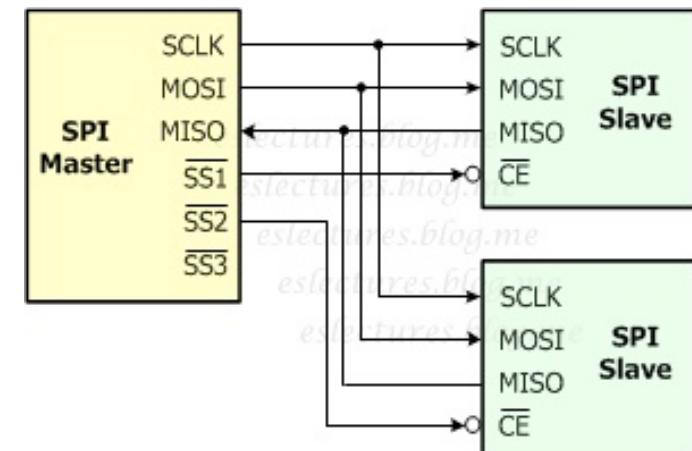
void loop(){
    delay(100);
}

void requestEvent(){
    Wire.write("hello!");
}
```

❖ SPI(Serial Peripheral Interconnect) 통신

- 동기적 직렬통신
- 모토롤라에 의해 개발
- 통신 회선 : 4회선
 - SCK, SCLK(Serial Clock) : 마스터가 클럭 전송
 - MOSI(Master Output Slave Input) : 마스터 데이터 전송
 - MISO(Master Input Slave Output) : 마스터 데이터 수신
 - SS(Slave Select) : 마스터가 슬레이브 선택
 - 슬레이브 장치마다 고유의 1회선 추가
 - 선택한 슬레이브에만 '0', 나머지는 '1'
- 실제 통신 회선은 2개(MOSI, MISO)
- 전이중 통신, 전송 속도 빠름
- SPI Mode
 - 시작 비트(0,1), 클럭 비트(0,1) 설정

SPI Mode	CPOL(Clock Polarity)	CPHA(Clock Phase)
0	0	0
1	0	1
2	1	0
3	1	1

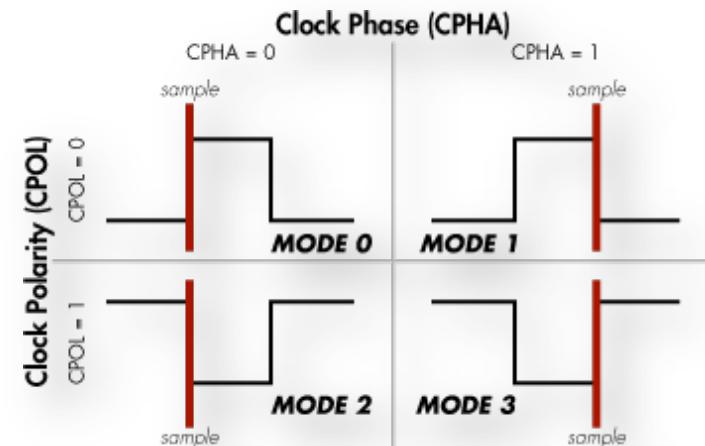
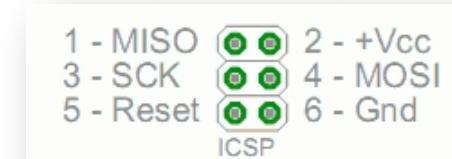


시리얼 통신

Arduino

❖ 아두이노 SPI

- 연결핀
 - MOSI : 11
 - MISO : 12
 - SCLK : 13
 - SS : 10
 - SS 추가는 디지털 핀 중에 하나 사용
 - 슬레이브인 경우 고정(LOW 인가)
 - ICSP 헤더핀에도 연결
- 기본 라이브러리
 - <SPI.h> Master 만 지원
 - begin() : 초기화
 - end() : 통신 종료
 - setBitOrder(bitOrder) : 전송순서 설정
 - LSBFIRST
 - MSBFIRST
 - setClockDivider(rate) : 전송속도 분주율, 기본4Mhz(SPI_CLOCK_DIV4)
 - SPI_CLOCK_DIV2, 4, 8, 16, 32, 64, 128
 - 시스템 클럭 / N
 - setDataMode : SPI 모드 설정
 - SPI_MODE0, 1, 2, 3
 - transfer(data) : data 송/수신

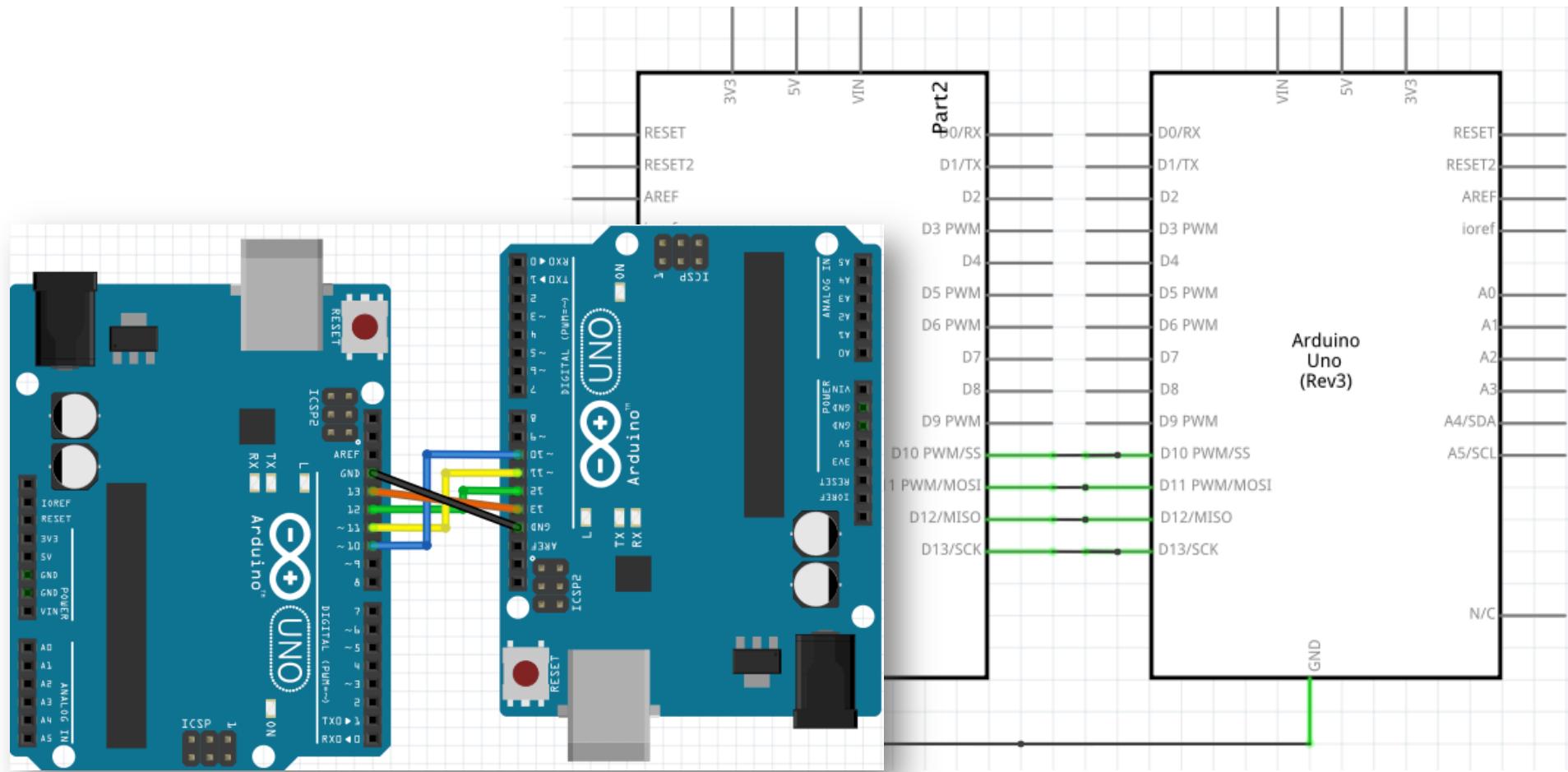


시리얼 통신

Arduino

❖ 아두이노 SPI 통신

▪ 회로구성



❖ 아두이노 SPI 통신

- Master 스케치

```
#include <SPI.h>
void setup() {
    SPI.begin();
    digitalWrite(SS, HIGH);
    SPI.setClockDivider(SPI_CLOCK_DIV16);
}

void loop() {
    const char *p = "Hello World\n";
    digitalWrite(SS, LOW);
    for(int i=0; i<strlen(p); i++){
        SPI.transfer(p[i]);
    }
    digitalWrite(SS, HIGH);
    delay(1000);
}
```

시리얼 통신

Arduino

❖ 아두이노 SPI 통신

- Slave 스케치

```
#include <SPI.h>
char buff[100];

volatile byte pos;
volatile boolean process_it;

void setup() {
  Serial.begin(9600);
  pinMode(MISO, OUTPUT);
  SPI.setClockDivider(SPI_CLOCK_DIV16);

  SPCR |= 1<<(SPE);
  SPCR &= ~(1<<(MSTR));
  SPCR |= 1<<(SPIE);
  pos = 0;
  process_it = false;

}
```

```
ISR(SPI_STC_vect){
  byte c = SPDR;
  if(pos < sizeof(buff)){
    buff[pos++] = c;
    if(c == '\n'){
      process_it = true;
    }
  }
}

void loop() {
  if(process_it){
    buff[pos] = 0;
    Serial.print(buff);
    pos=0;
    process_it = false;
  }
}
```



세부목차

Arduino

1. Introduction
2. 개발환경
3. Digital I/O
4. Analog I/O
5. 외부 라이브러리
6. 인터럽트와 타이머
7. Advanced I/O
8. 시리얼 통신
9. **아두이노 메모리**

아두이노 메모리

Arduino

❖ 메모리 종류

- Flash Memory
 - 부트로더, 프로그램 영역
 - 스케치를 업로드하면 저장하는 공간
 - 16비트 단위 메모리 주소
 - 0x0000 ~ 0x3FFF
 - 아두이노 우노(Atmega328) : 32KBytes
- SRAM
 - 프로그램 실행 중에 사용하는 영역,
 - 각종 변수, 버퍼
 - 아두이노 우노(Atmega328) : 2KBytes
- EEPROM
 - 영구적으로 저장할 데이터
 - 하드 디스크와 같은 역할
 - 아두이노 우노(Atmega328) : 1KByte

❖ PROGMEM

- SRAM 부족 해결
 - 메모리 할당을 SRAM이 아닌 Flash에 지시
 - 긴 문자열이나 배열에 주로 사용
 - 읽기 위해서는 전용함수 필요
 - pgm_read_byte()
 - 실행 중 쓰기 불가
- PROGMEM 키워드
 - 변수 선언에 사용
 - const로 선언해야 함
 - const data_type variableName[] PROGMEM = "...";
- F() 매크로
 - 함수 외부에서는 사용 불가
 - Serial.println(F("F macro"));

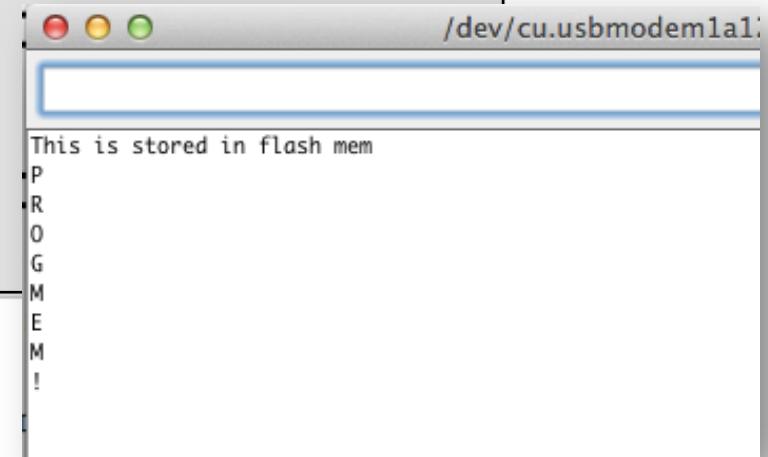
❖ PROGMEM 스케치 예제

```
const char str[] PROGMEM = "PROGMEM!";

void setup() {
    Serial.begin(9600);
    Serial.println(F("This is stored in flash memory."));

    for(int i=0; i<strlen(str); i++){
        Serial.println((char)pgm_read_byte(str + i));
    }
}

void loop() {
```



아두이노 메모리

Arduino

❖ EEPROM 라이브러리

- EEPROM 클래스, <EEPROM.h>
 - write(address, value)
 - address : 0 ~ 1023 (1KBytes)
 - value : 0~255 (byte), 1Byte 보다 큰 데이터는 나누어 저장
 - 3.3ms 소요(write 후에 지연필요)
 - 100,000회 쓰고 지우기 가능
 - read(address)
 - 지정된 주소의 값 읽기(1Byte)
 - update(address, value)
 - 지정된 주소에 전달된 값이 다른 경우에만 쓰기
 - 쓰기 횟수 절약을 위해 write() 보다 효과적
 - put(address, data)
 - 모든 데이터 타입의 데이터(primitive 또는 struct) 쓰기
 - 내부적으로 update() 함수 사용
 - get()
 - 모든 데이터 타입의 데이터 읽기

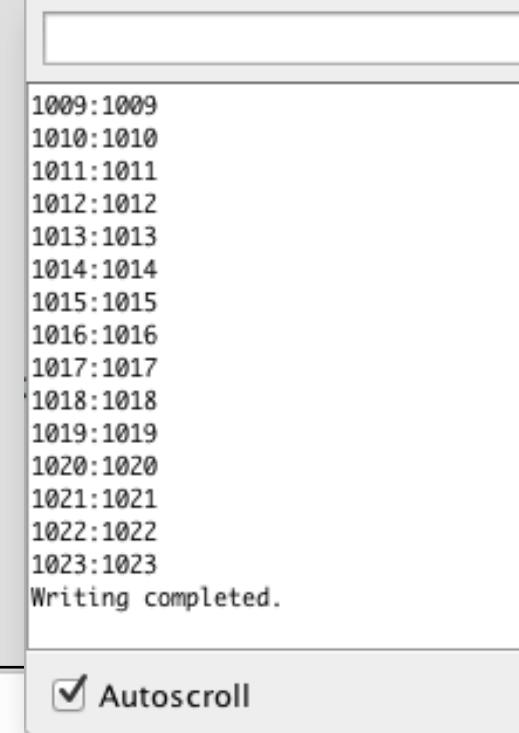
❖ EEPROM

- Write

```
#include <EEPROM.h>

void setup()
{
    Serial.begin(9600);
    Serial.println("EEPROM writing.");
    for(int i=0; i<1024; i++){
//        EEPROM.write(i, i);
        EEPROM.update(i, i);
        delay(10);
        Serial.println(String(i) + ":" + String(i));
    }
    Serial.println("Writing completed.");
}

void loop(){}
}
```



아두이노 메모리

Arduino

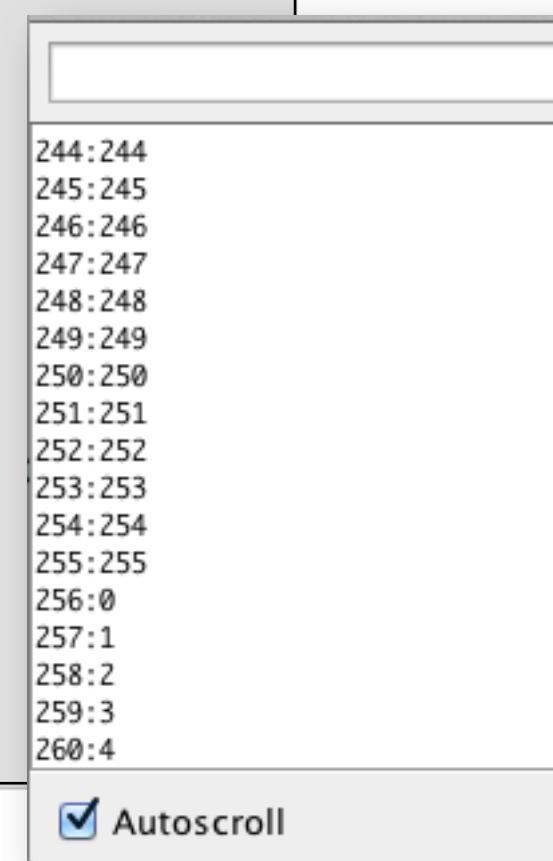
❖ EEPROM

- Read

```
#include <EEPROM.h>

void setup()
{
    Serial.begin(9600);
    Serial.println("EEPROM reading.");
    for(int i=0; i<1024; i++){
        int value = EEPROM.read(i);
        Serial.println(String(i) + ":" + String(value));
    }
}

void loop(){}
```



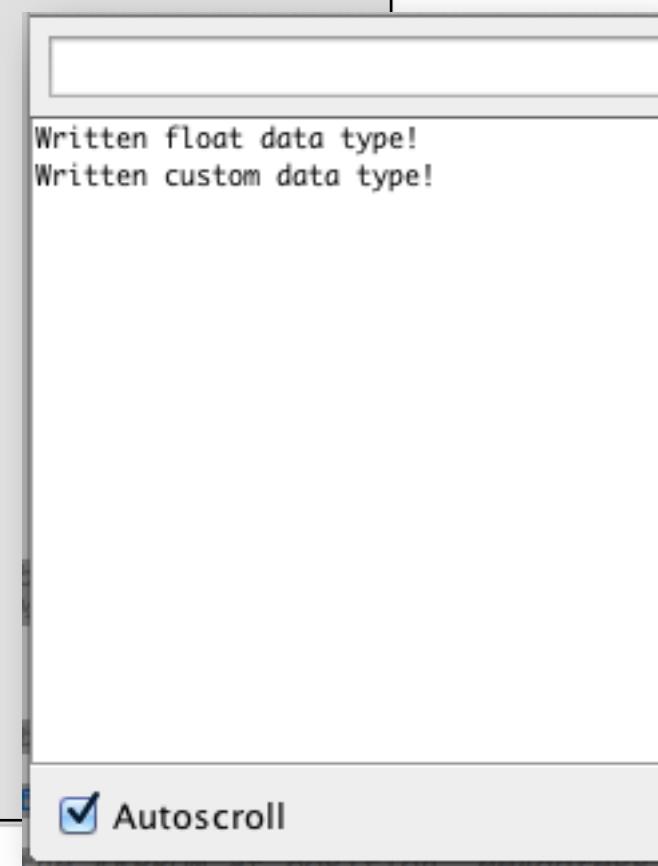
❖ EEPROM

- Put

```
#include <EEPROM.h>

struct MyObject{
    byte age;
    char name[10];
};

void setup(){
    Serial.begin(9600);
    float f = 3.14f;
    int eeAddress = 0;
    EEPROM.put( eeAddress, f );
    Serial.println("Written float data type!");
    MyObject customVar = { 25, "Lee" };
    eeAddress += sizeof(float);
    EEPROM.put( eeAddress, customVar );
    Serial.println( "Written custom data type!" );
}
void loop(){ /* Empty loop */ }
```



❖ EEPROM

- Get

```
#include <EEPROM.h>
struct MyObject{
    byte age;
    char name[10];
};
void setup(){
    float f = 0.00f;
    int eeAddress = 0;
    Serial.begin( 9600 );
    Serial.print( "Read float from EEPROM: " );
    EEPROM.get( eeAddress, f );
    Serial.println( f );
    eeAddress = sizeof(float);
    MyObject customVar;
    EEPROM.get( eeAddress, customVar );

    Serial.println("Read custom object from EEPROM: ");
    Serial.println( customVar.age );
    Serial.println( customVar.name );
}
void loop(){ /* Empty loop */ }
```

