

# **종합 설계를 위한 아두이노 실습 가이드**

**전북대학교 공과대학 전자공학부  
백 흥 기 교수**

# 목 차

## 1. 아두이노 소개

1.1 아두이노란?	(1-1)
1.2 아두이노 통합환경 (IDE)	(1-4)
1.3 아두이노 스케치 프로그램	(1-9)
1.4 아두이노 라이브러리	(1-16)

## 2. 아두이노 프로그래밍

2.1 아두이노 프로그램	(2-1)
2.2 아두이노 변수	(2-2)
2.3 아두이노 제어 문	(2-5)
2.4 아두이노 연산자	(2-8)
2.5 아두이노 프로그램과 메모리	(2-10)

## 3. 아두이노 입/출력

3.1 디지털 출력	(3-2)
3.2 디지털 입력	(3-7)
3.3 아날로그 입력	(3-20)
3.4 아날로그 출력	(3-27)
3.5 아두이노 인터럽트	(3-30)

## 4. 시리얼 (serial) 통신

4.1 시리얼 통신	(4-1)
4.2 아두이노에서 데이터 송신	(4-5)
4.3 아두이노에서 데이터 수신	(4-12)
4.4 이진 데이터 송수신	(4-17)

## 5. 모터 제어

5.1 모터 종류	(5-1)
5.2 서보 모터 제어	(5-2)
5.3 직류 모터 제어	(5-7)

5.4 스텔퍼 모터 제어 -----	(5-18)
<b>6. 스피커 제어</b>	
6.1 스피커 제어 -----	(6-1)
6.2 음악 연주 -----	(6-8)
<b>7. 초음파 센서</b>	
<b>8. 블루투스 통신</b>	
8.1 아두이노 블루투스 통신 -----	(8-1)
8.2 아두이노와 스마트 폰 사이의 블루투스 통신 -----	(8-10)
8.3 아두이노 사이의 블루투스 통신 -----	(8-32)
<b>9. 디스플레이 제어</b>	
9.1 7-세그먼트 디스플레이 -----	(9-1)
9.2 4 자리 7-세그먼트 디스플레이 -----	(9-11)
9.3 LCD 디스플레이 -----	(9-21)
9.4 Matrix LED 디스플레이 -----	(9-27)
<b>10. WIFI 통신</b>	
8.7 아두이노를 이용한 7-세그먼트 업/다운 회로 -----	(10-)
8.8 아두이노를 이용한 음악 레벨 미터 -----	(10-)
8.9 아두이노를 이용한 스펙트럼 표시 회로 -----	(10-)
8.10 심전도 측정 회로 -----	(10-)
<b>11. 스펙트럼 분석</b>	
11.1 MSGEQ7 을 이용한 스펙트럼 분석 -----	(11-1)
11.2 FFT 를 이용한 스펙트럼 분석 -----	(11-14)
<b>12. SD 카드 제어</b>	
12.1 SPI (serial peripheral interface) 통신 -----	(12-1)
12.2 SD 카드 모듈 -----	(12-4)

# 1. 아두이노 소개

## 1.1 아두이노(Arduino)란?

### ✚ 아두이노란?

- ✓ 오픈 소스를 기반으로 한 단일 보드 마이크로 컨트롤러로 완성된 보드와 관련 개발 도구 및 환경 – 위키백과에서
- ✓ [https://www.youtube.com/watch?time\\_continue=35&v=UoBXOOdLXY](https://www.youtube.com/watch?time_continue=35&v=UoBXOOdLXY)

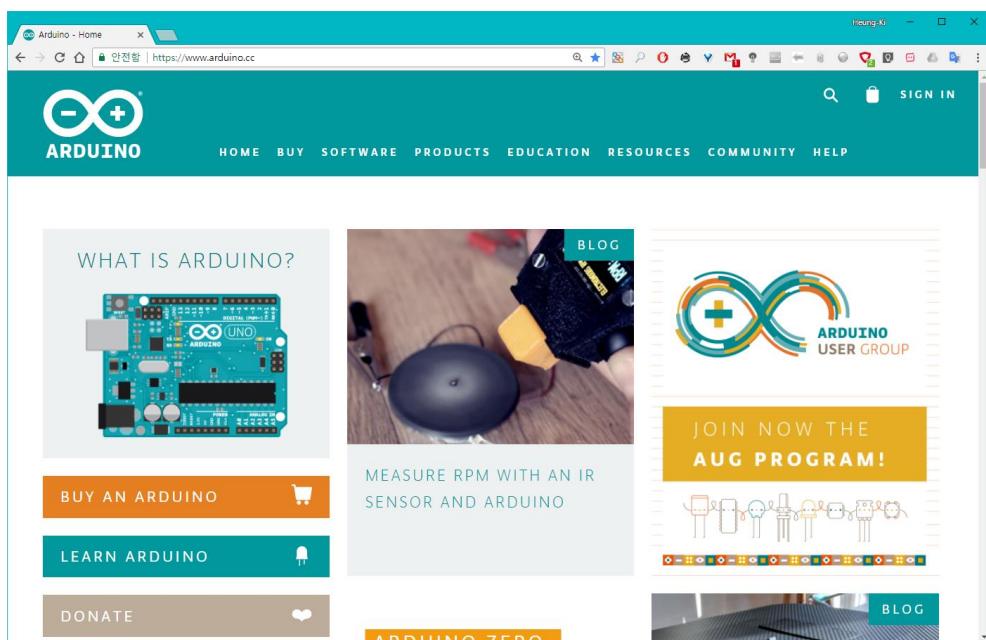
WHAT IS ARDUINO?

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects.

[Learn more about Arduino](#)

### ✚ 아두이노 홈페이지

- ✓ <https://www.arduino.cc/>



## ✚ 아두이노 제품군

ENTRY LEVEL	UNO	LEONARDO	101	ESPLORA	MICRO	NANO	MINI	MKR2UNO ADAPTER
	STARTER KIT	LCD SCREEN						
ENHANCED FEATURES	MEGA	ZERO	DUE	MEGA ADK	MO	MO PRO	MKR ZERO	MOTOR SHIELD
	USB HOST SHIELD	PROTO SHIELD		MKR PROTO SHIELD	4 RELAYS SHIELD			
	MEGA PROTO SHIELD	MKR RELAY PROTO SHIELD		ISP	USB2SERIAL MICRO			
	USB2SERIAL CONVERTER							
INTERNET OF THINGS	YUN	ETHERNET	TIAN	INDUSTRIAL 101	LEONARDO ETH	MKR FOX 1200		
	MKR WAN 1300	MKR GSM 1400	MKR1000	YUN MINI	YUN SHIELD	WIRELESS SD SHIELD		
	WIRELESS PROTO SHIELD	ETHERNET SHIELD V2	GSM SHIELD V2		MKR IoT BUNDLE			

## ✚ 아두이노 우노(Uno)

- ✓ 아두이노 기본 보드



## ✚ 아두이노 우노 스펙

- ✓ 마이크로 컨트롤러 : ATmega328P
- ✓ 클럭 주파수 : 16 MHz
- ✓ 동작 전압 : 5 V
  - ✗ USB로 전원 공급 (5 V)
  - ✗ 외부에서 공급 가능 (7 ~ 12 V)
- ✓ 입력/출력 핀
  - ✗ 디지털 입력/출력 핀 : 14 개 (PWM : 6 개)
  - ✗ 아날로그 입력/출력 핀 : 6 개

✓ 메모리

✗ Flash 메모리 (32 k)

- 프로그램이 업로드 되는 공간
- 비 휘발성 메모리

✗ SRAM 메모리 (2 k)

- 프로그램이 실행 중에 읽고 쓸 수 있는 휘발성 메모리
- 스케치 프로그램의 변수가 저장되는 공간
- 용량이 작다.

✗ EEPROM 메모리 (1 k)

- 프로그램이 실행 중에 읽고 쓸 수 있는 비 휘발성 메모리
- SRAM에 비해 속도가 느리다.

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13

✚ 아두이노 제품 비교

- ✓ <https://www.arduino.cc/en/Products/Compare>

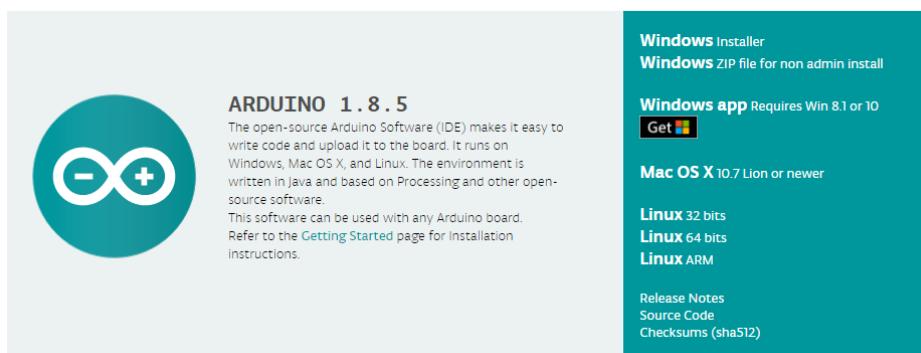
## 1.2 아두이노 통합 환경 (IDE)

### ✚ 아두이노 통합 환경 (IDE)

Access the Online IDE



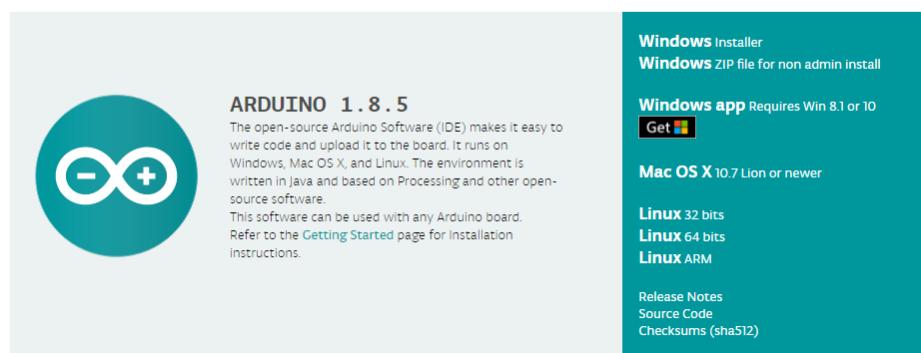
Download the Arduino IDE



### ✚ 아두이노 통합 환경 다운로드 및 설치

- ✓ Arduino-1.8.5-windows.zip
- ✓ 압축 해제 후 C: 드라이브에 넣는다.

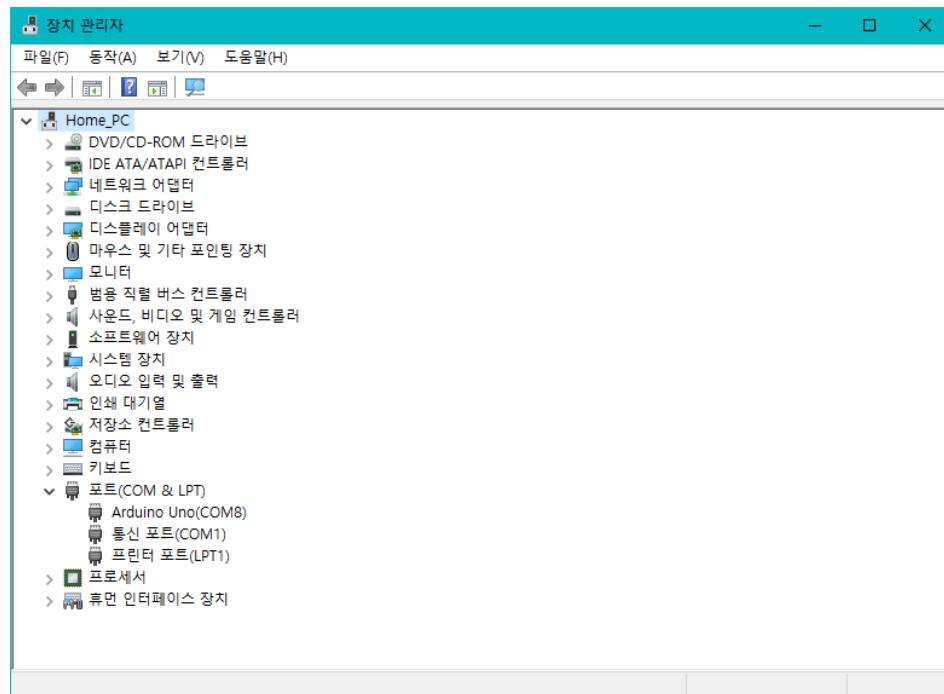
Download the Arduino IDE



### ✚ 장치 관리자

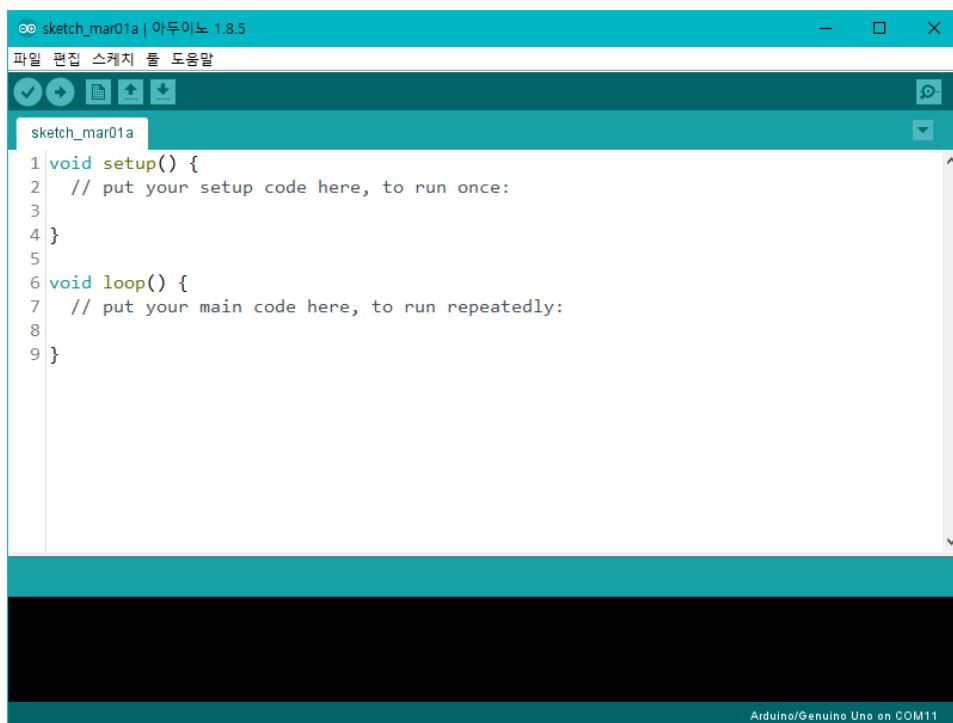
- ✓ 아두이노와 컴퓨터를 USB 케이블로 연결한 상태에서 장치 관리자를 열면 포트에 Arduino Uno 포트가 생긴다.

- ✓ 포트 번호는 상황에 따라 달라질 수 있다.



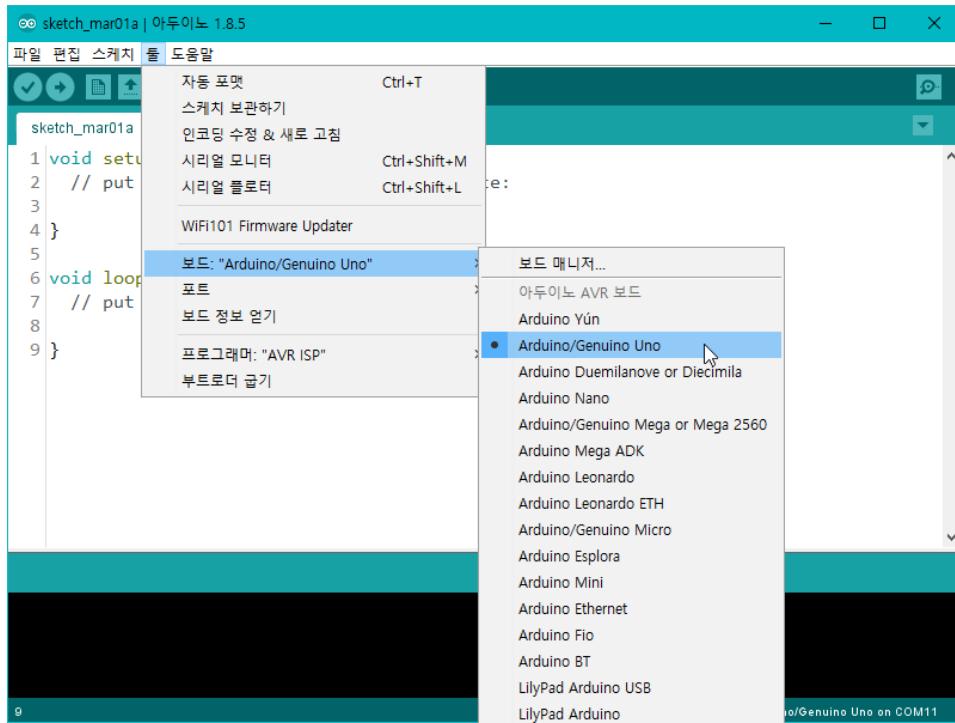
### ✚ 아두이노 프로그램: 스케치 (sketch)

- ✓ 아두이노 프로그램을 처음 실행하면 아래와 같은 창이 열린다.
- ✓ 처음 아두이노 프로그램의 파일 이름은 파일이 생성된 월일을 바탕으로 만들어진다. (예: sketch\_mar01a)



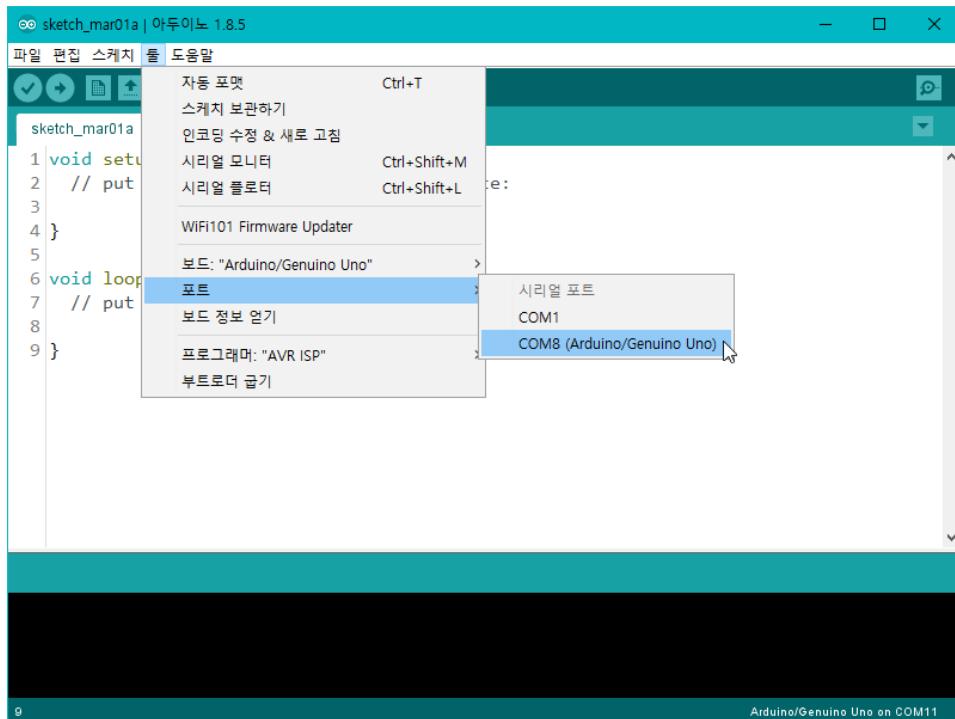
### ✚ 아두이노 보드 설정

- ✓ 메뉴 [툴 - 보드]를 선택한 후 원하는 보드를 선택한다.



#### ■ COM 포트 설정

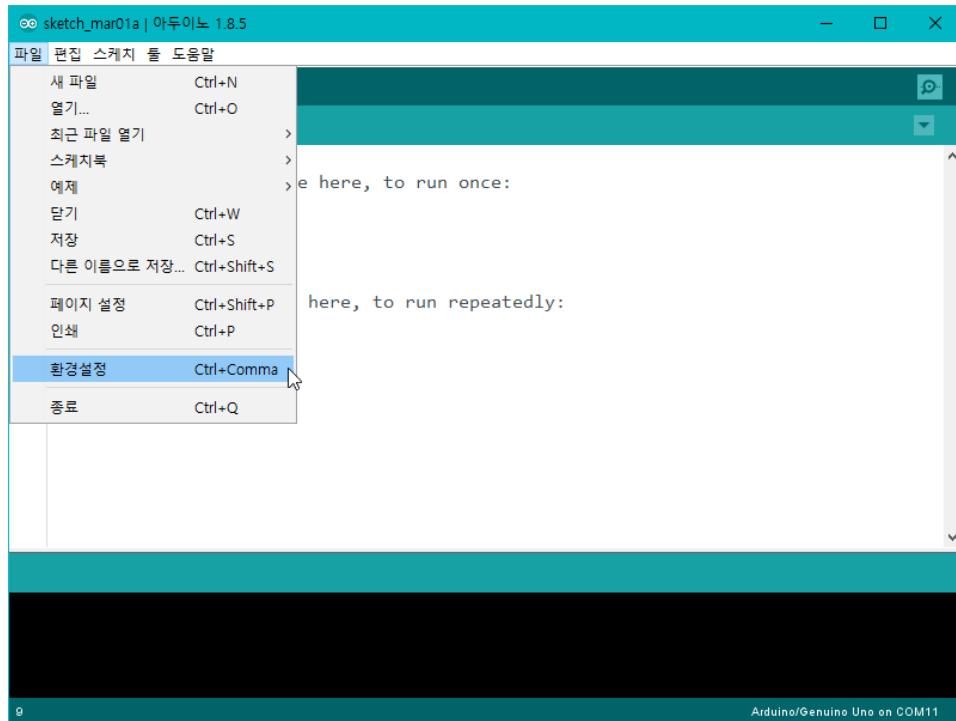
- ✓ 메뉴 [툴 - 포트]를 선택한 후 Arduino/Genuino Uno 를 선택한다.



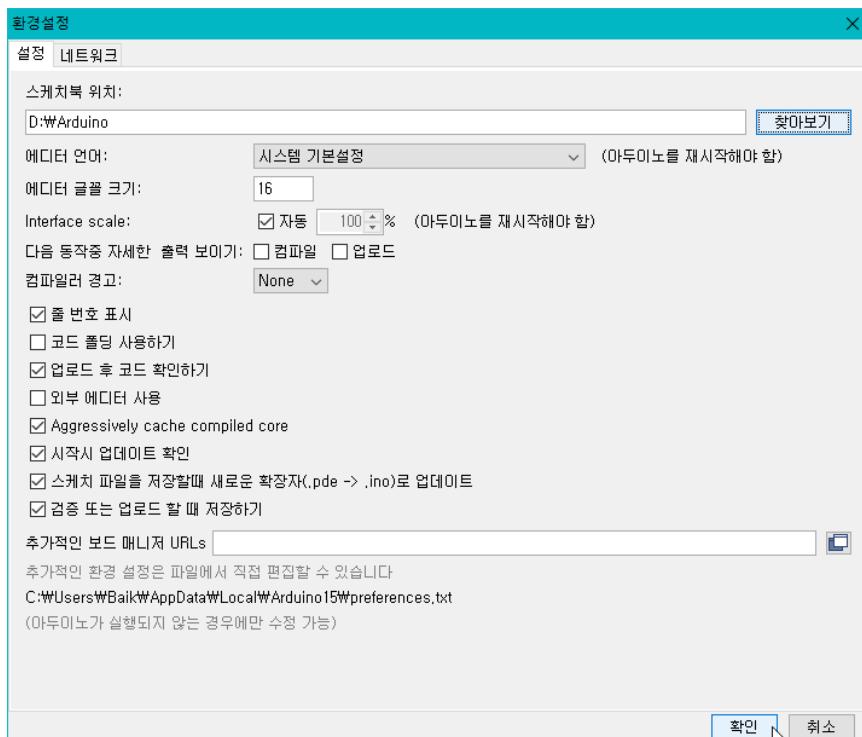
#### ■ 아두이노 환경 설정

- ✓ 아두이노 프로그램이 저장되거나 불러올 수 있는 폴더, 편집기의 폰트 종류와 크기 등을 설정할 수 있다.

- ✓ 메뉴 [파일 – 환경설정]을 선택하여 환경 설정 창을 연다.



## ■ 환경 설정 창

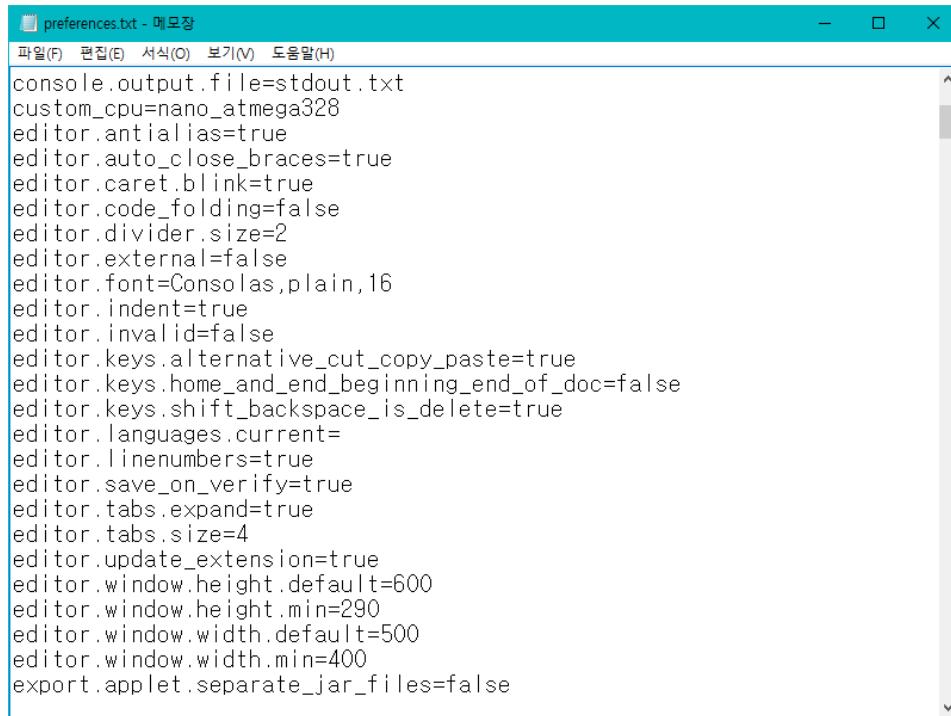


- ✓ 스케치북 폴더 설정

- ✗ 아두이노 프로그램이 저장되거나 불러올 수 있는 폴더를 설정한다.  
✗ D:\Arduino

- ✓ 편집기 글꼴 및 크기 설정

- ✖ Preferences.txt를 클릭하면 파일을 수정할 수 있다.



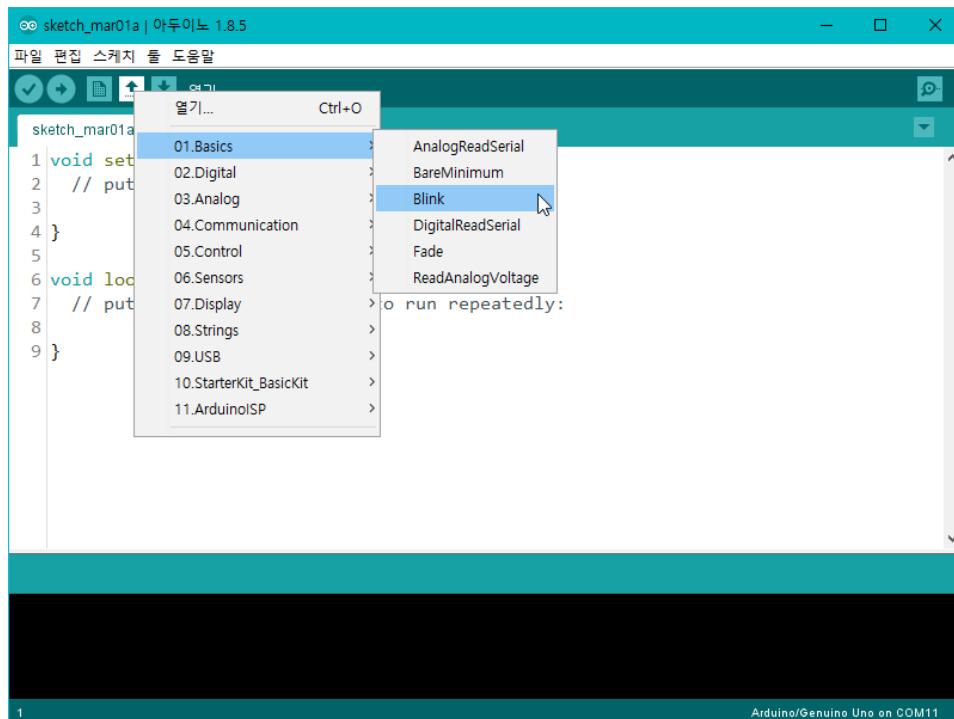
```
preferences.txt - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
console.output.file=stdout.txt
custom_cpu=nano_atmega328
editor.antiAlias=true
editor.autoCloseBraces=true
editor.caret.blink=true
editor.codeFolding=false
editor.divider.size=2
editor.external=false
editor.font=Consolas,plain,16
editor.indent=true
editor.invalid=false
editor.keys.alternativeCutCopyPaste=true
editor.keys.homeAndEndBeginningEndOfDoc=false
editor.keys.shiftBackspaceIsDelete=true
editor.languages.current=
editor.lineNumbers=true
editor.saveOnVerify=true
editor.tabs.expand=true
editor.tabs.size=4
editor.updateExtension=true
editor.window.height.default=600
editor.window.height.min=290
editor.window.width.default=500
editor.window.width.min=400
export.applet.separateJarFiles=false
```

- ✖ 아두이노 스케치 프로그램이 열려 있는 상태에서는 Preferences.txt를 수정하여 저장해도 아두이노 스케치 프로그램을 닫을 때 원래대로 환경을 저장하므로 모든 아두이노 프로그램을 닫은 상태에서 Preferences.txt를 저장해야 수정된 내용이 적용된다.

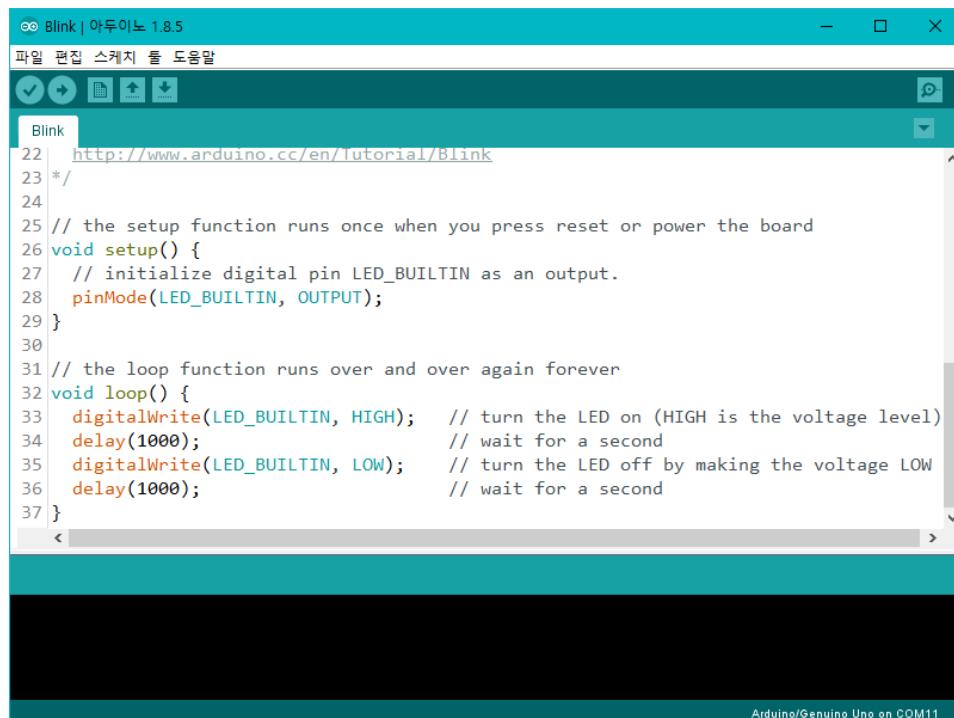
## 1.3 아두이노 스케치 프로그램

### ➊ 예제 프로그램 불러 오기

- ✓ 메뉴 [파일 – 예제 - 01.Basics – Blink]를 선택하여 Blink 파일을 연다.



- ✓ Blink 프로그램은 별도의 하드웨어가 필요 없는 LED 점멸 프로그램으로  
아두이노 우노의 13 번 핀에 연결된 LED 를 점멸 시킨다.



### ➋ 프로그램 컴파일

- ✓ 메뉴 [스케치 – 확인/컴파일]을 선택하거나, 확인 아이콘을 클릭한다.
- ✓ 단축 키 : Ctr+R

```

Blink | 아두이노 1.8.5
파일 편집 스케치 둘 도움말
Blink
22 | http://www.arduino.cc/en/Tutorial/Blink
23 */
24
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27   // initialize digital pin LED_BUILTIN as an output.
28   pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33   digitalWrite(LED_BUILTIN, HIGH);    // turn the LED on (HIGH is the voltage level)
34   delay(1000);                      // wait for a second
35   digitalWrite(LED_BUILTIN, LOW);     // turn the LED off by making the voltage LOW
36   delay(1000);                      // wait for a second
37 }
  
```

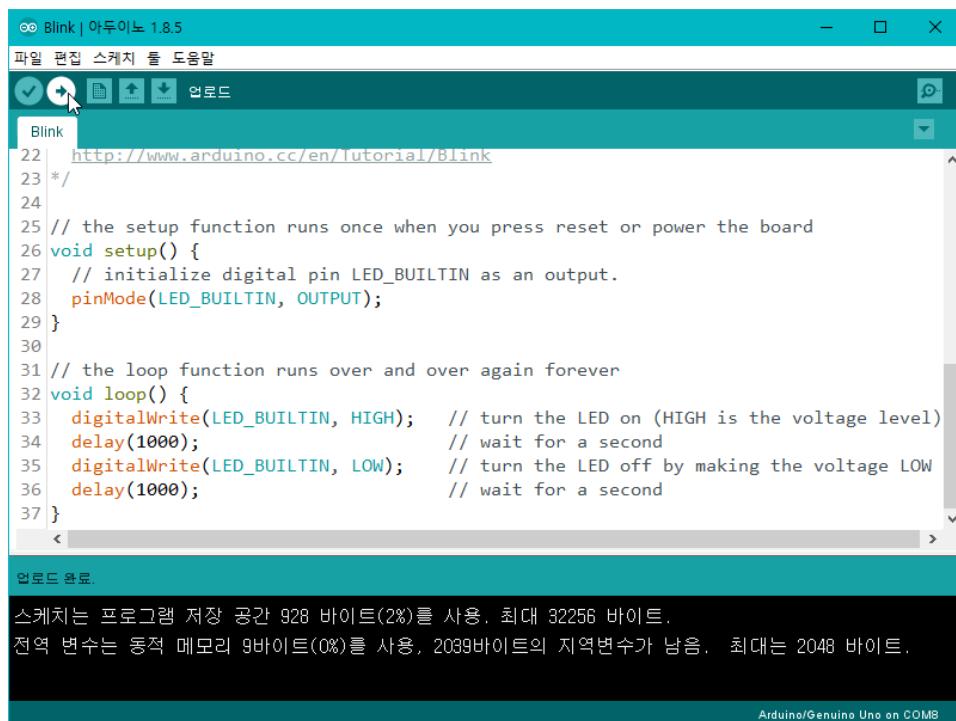
컴파일 완료.  
스케치는 프로그램 저장 공간 928 바이트(2%)를 사용, 최대 32256 바이트.  
전역 변수는 동적 메모리 9바이트(0%)를 사용, 2039바이트의 지역변수가 남음. 최대는 2048 바이트.

Arduino/Genuino Uno on COM11

- ✓ 에러가 없이 컴파일이 완료되면 '컴파일 완료'라는 메시지가 나온다.

#### 프로그램 업로드

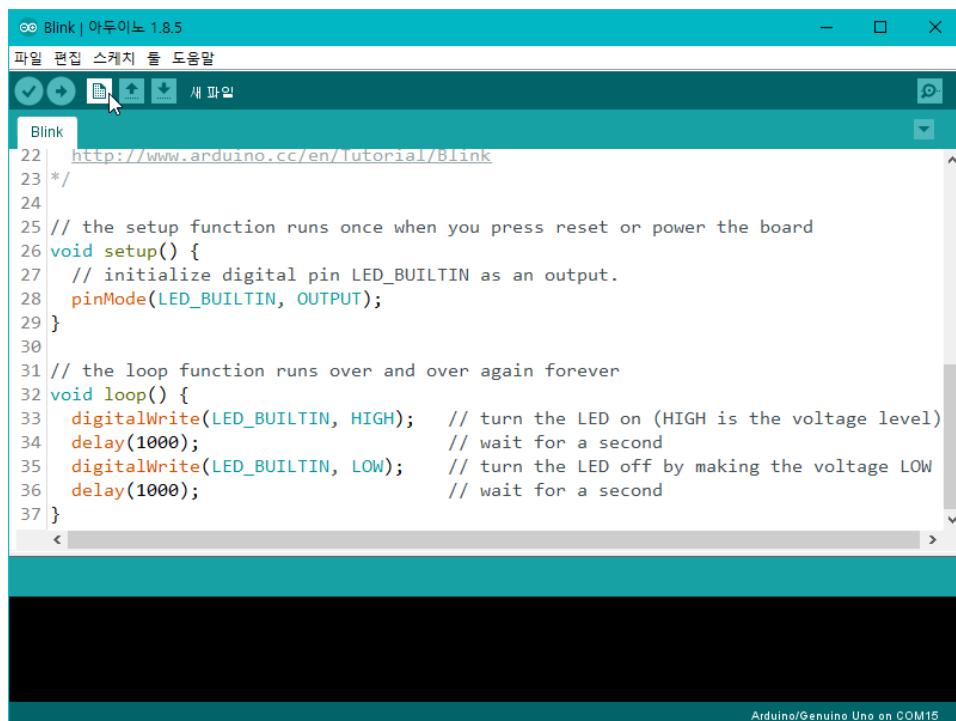
- ✓ USB 케이블을 연결한다.
- ✓ 메뉴 [스케치 – 업로드]를 선택하거나, 업로드 아이콘을 클릭한다.
- ✓ 단축 키 : Ctr+U



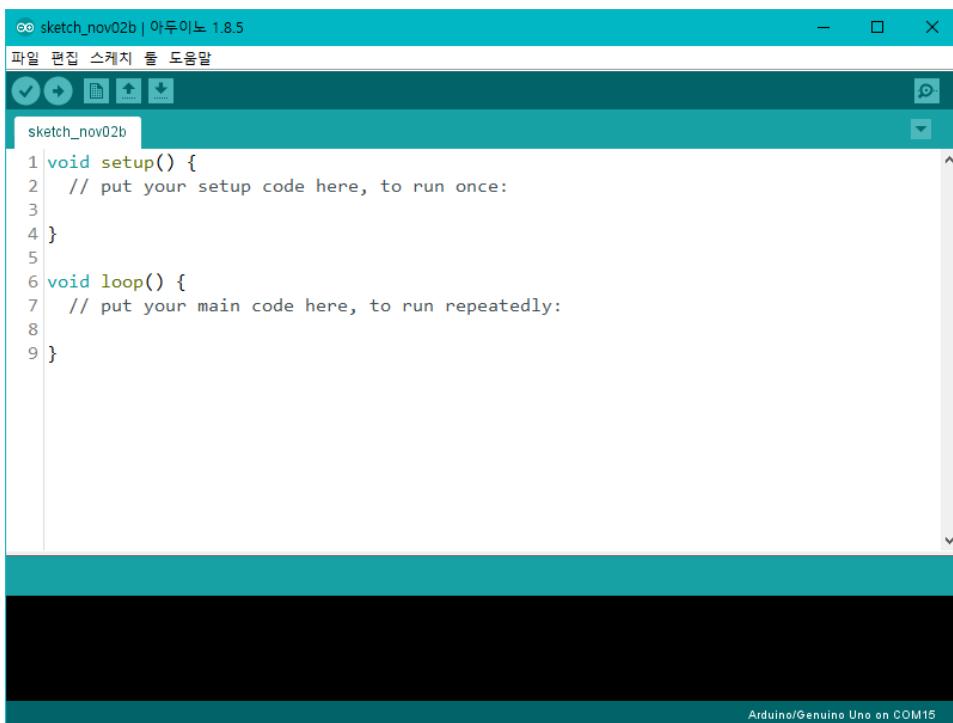
✓ 에러없이 업로드가 완료되면 '업로드 완료' 메시지가 나온다.

#### ➊ 새 프로그램 만들기

- ✓ 메뉴 [파일 – 새 파일]을 선택하거나, 새 파일 아이콘을 클릭한다.
- ✓ 단축 키 : Ctr+N

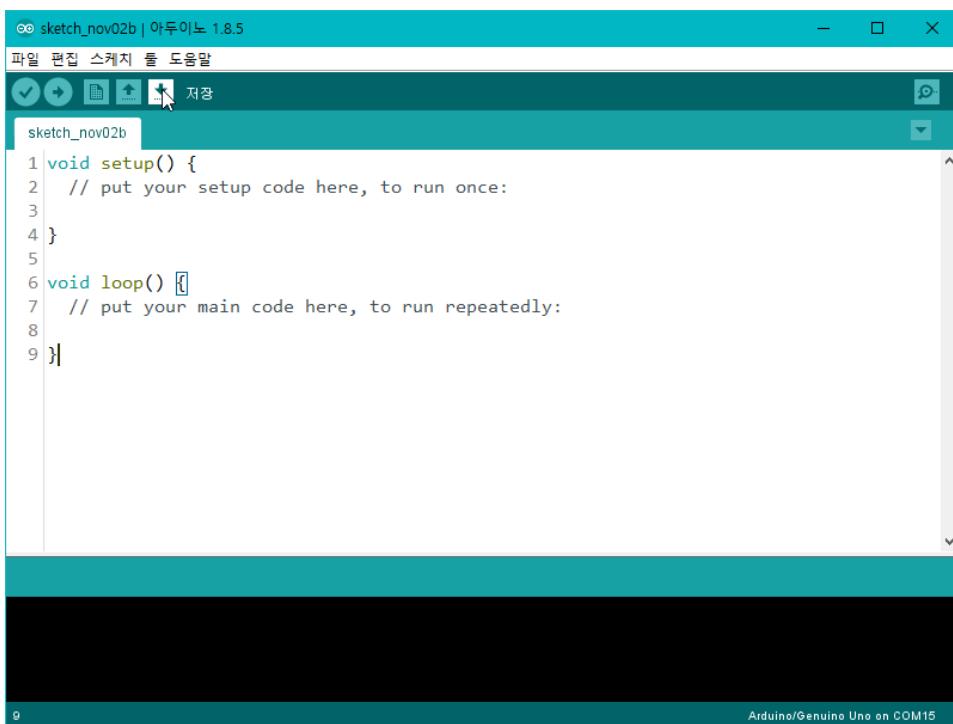


✓ 새로운 스케치 창이 열린다.

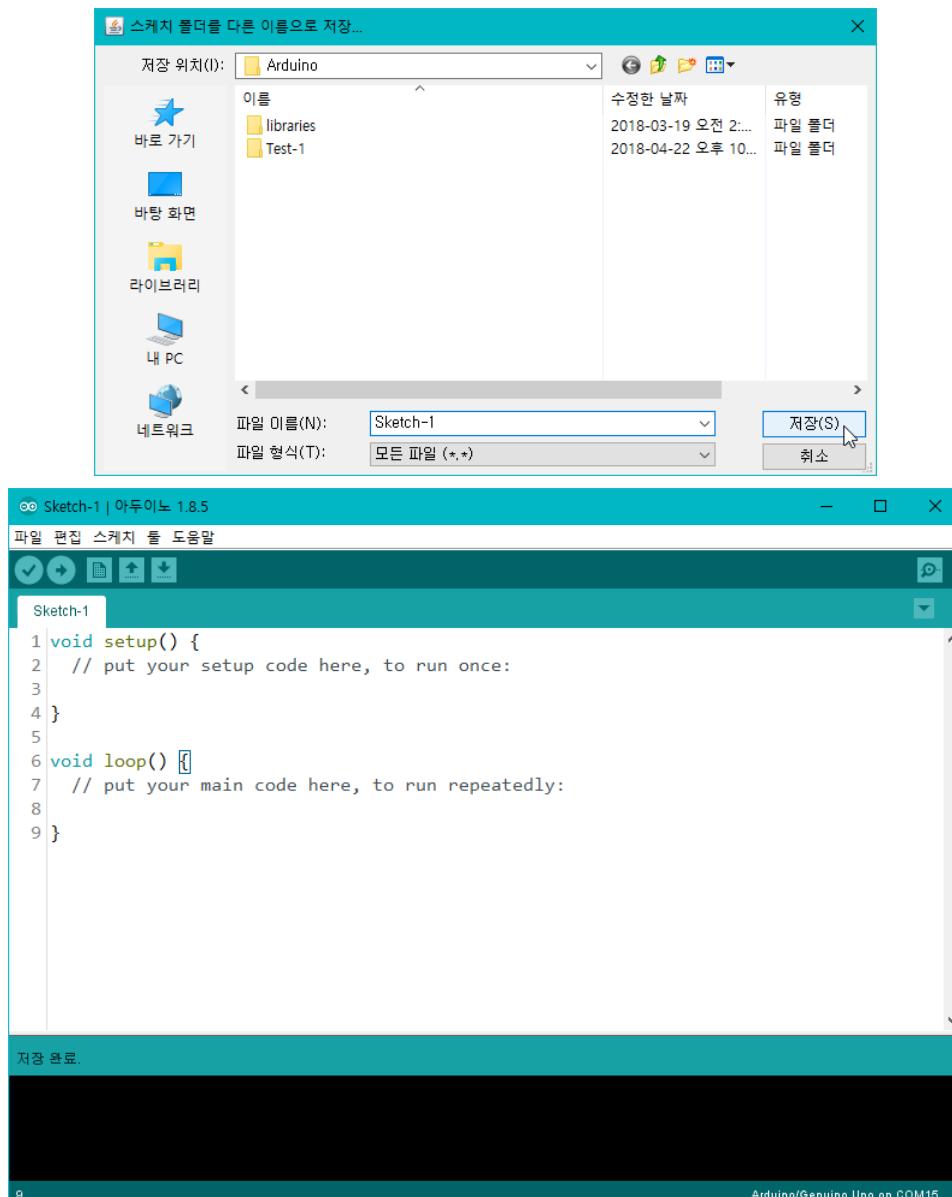


### ▶ 프로그램 저장하기

- ✓ 메뉴 [파일 – 저장]을 선택하거나, 저장 아이콘을 클릭한다.
- ✓ 단축 키 : Ctr+S



- ✓ 저장할 파일 이름을 입력하고 저장한다.



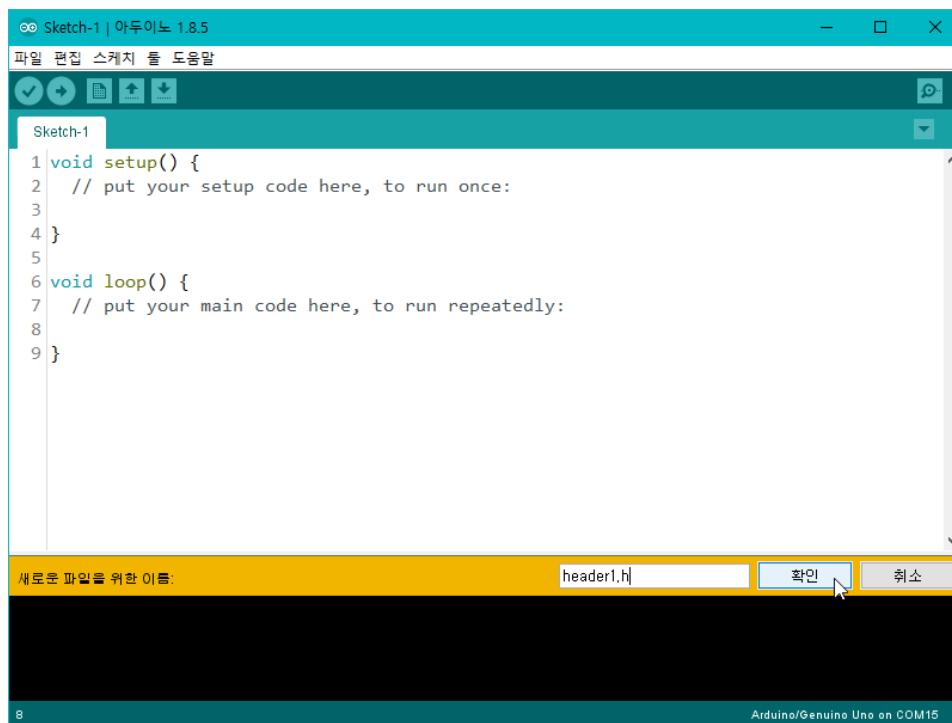
- ✓ 프로그램의 이름이 변경되었음을 알 수 있다.

#### 새 탭 만들기

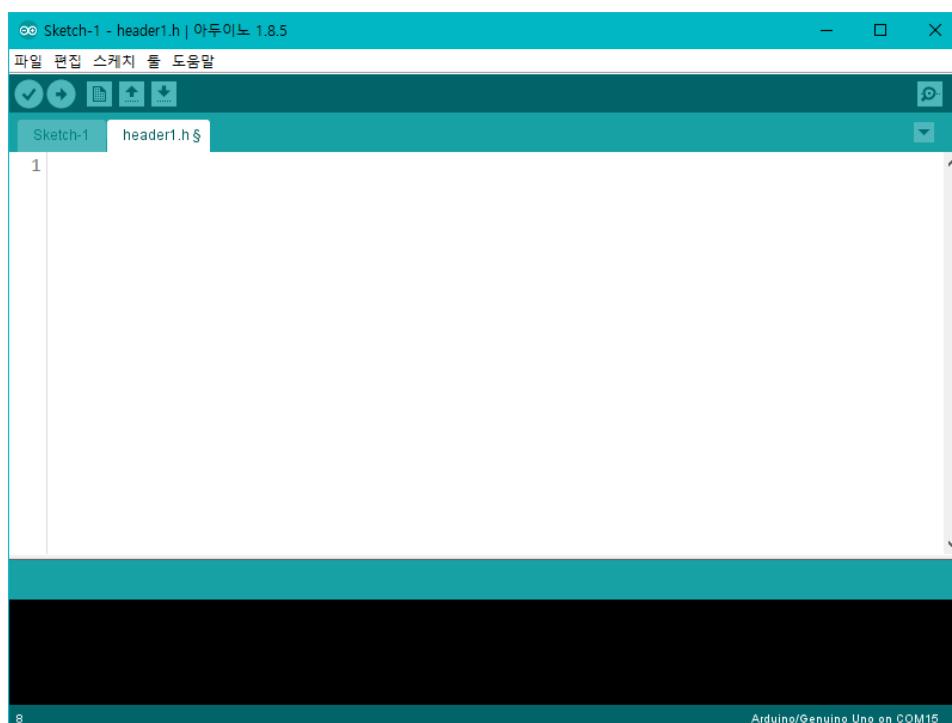
- ✓ 스케치 프로그램과 관련있는 헤더 파일 등을 작성할 때 사용한다.
- ✓ 오른쪽 상단의 탭 아이콘을 클릭하여 새 탭을 선택한다.
- ✓ 단축 키 : Ctr+Shift+N



- ✓ 파일 이름을 입력하고 저장 버튼을 클릭하여 저장한다.



- ✓ 프로그램을 작성하기 위한 탭이 열린다.



## 1.4 아두이노 라이브러리

### ✚ 아두이노 라이브러리란?

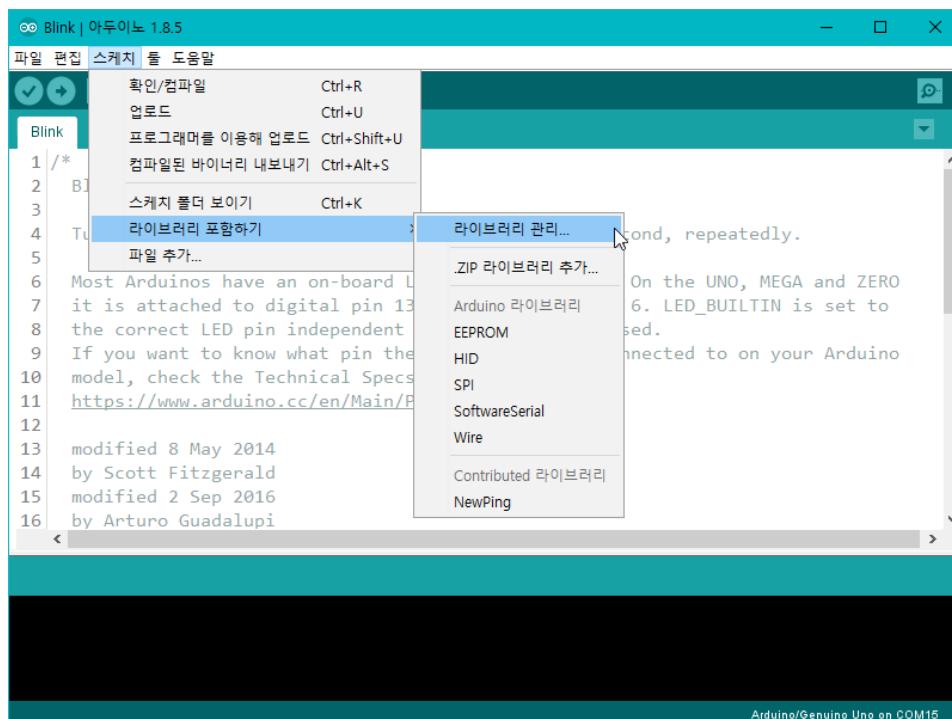
- ✓ 아두이노 환경은 대부분의 프로그래밍 플랫폼과 마찬가지로 라이브러리 사용을 통해 확장될 수 있으며 라이브러리는 스케치에서 사용하기 위한 추가 기능을 제공한다. (예: 하드웨어 작업 또는 데이터 조작 등)
- ✓ 아두이노 라이브러리는 C/C++의 표준 함수 일부분과 AVR의 내장 모듈 및 각종 부가 장치를 제어하는 함수들로 구성되어 있다.
- ✓ 아두이노 IDE에는 여러 라이브러리가 설치되어 있지만 다운로드하거나 직접 만들 수도 있습니다.

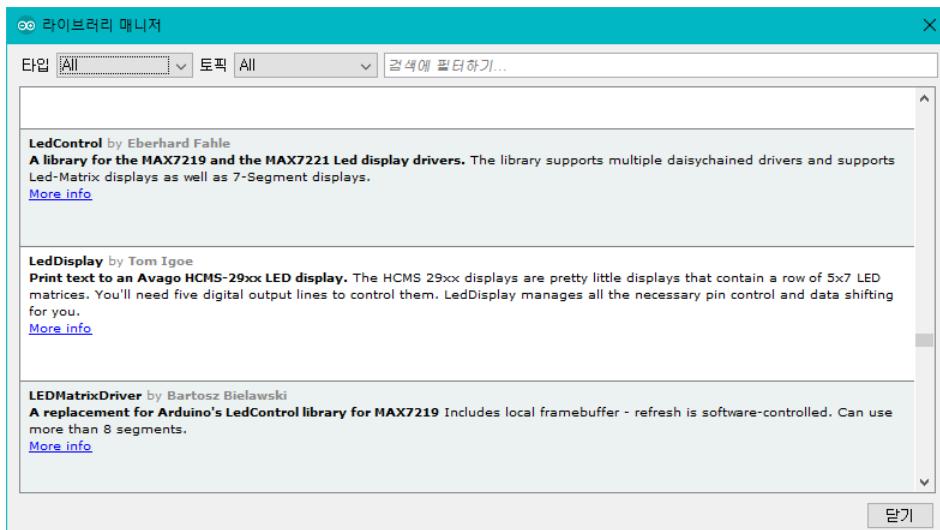
### ✚ 아두이노 라이브러리 설치 방법

- ✓ IDE에서 자체적으로 제공하는 라이브러리를 설치하거나, 인터넷 등에서 검색하여 다운로드한 후 설치할 수 있다. (다운로드 후 설치 방법은 2 가지 방법이 있다.)

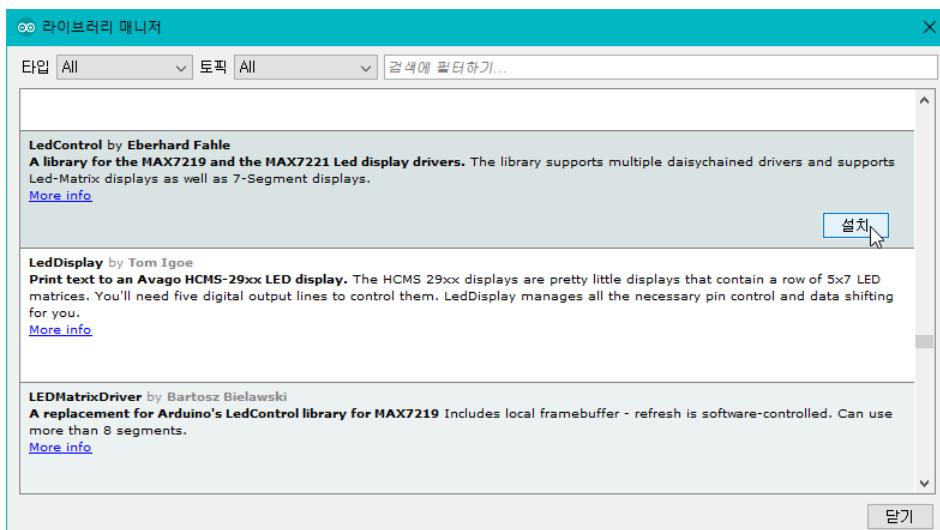
### ✚ IDE에서 자체적으로 제공하는 라이브러리 설치하기.

- ✓ 메뉴 [스케치 – 라이브러리 포함하기 – 라이브러리 관리...]를 클릭하여 라이브러리 매니저 창을 연다.

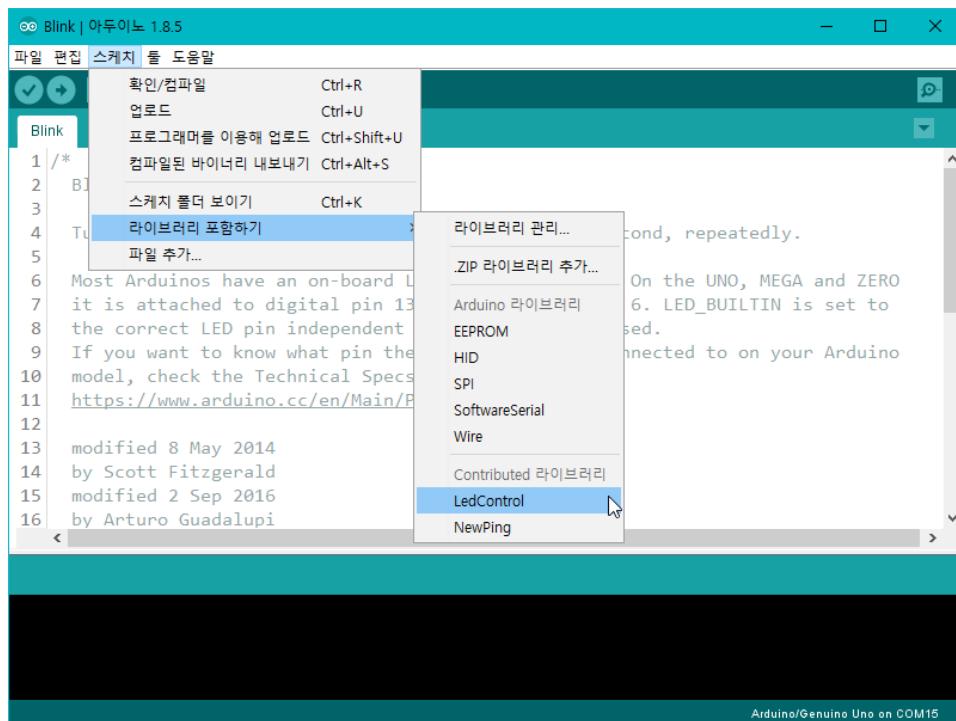




- ✓ 설치하고자 하는 라이브러리를 찾아 클릭하면 설치 버튼이 나타난다.
- ✓ 설치 버튼을 클릭하여 라이브러리를 설치한다. (LedDisplay 라이브러리를 설치하였다.)

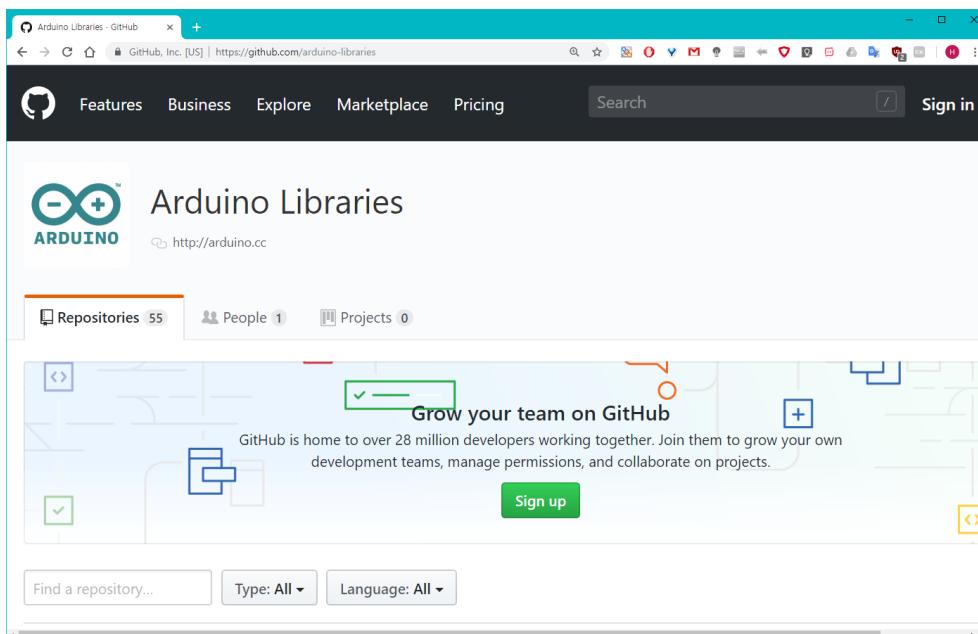


- ✓ 메뉴 [스케치 – 라이브러리 포함하기]에 마우스를 놓으면 설치된 LedDisplay 라이브러리를 볼 수 있다.

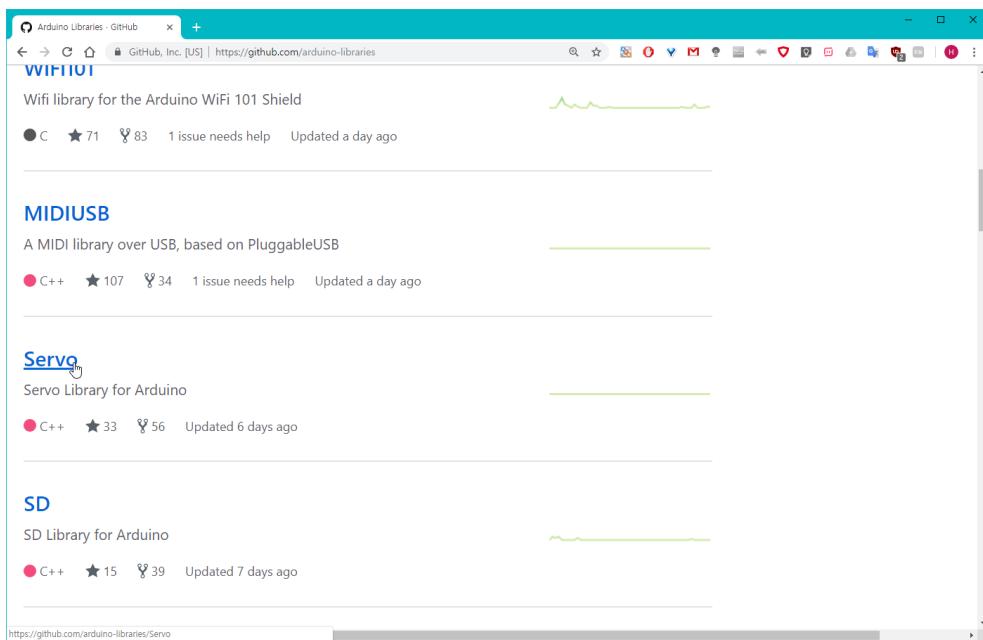


#### ➊ 라이브러리를 인터넷 등에서 다운로드하기.

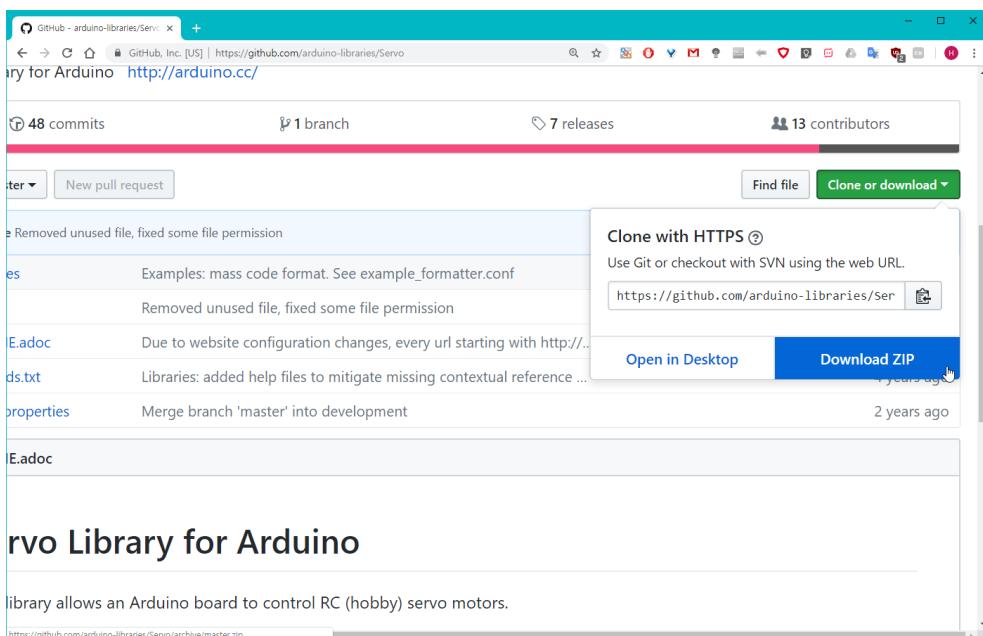
- ✓ 라이브러리는 구글에서 검색하여 다운로드하거나 GitHub에서 다운로드 한다. (<https://github.com/arduino-libraries>)



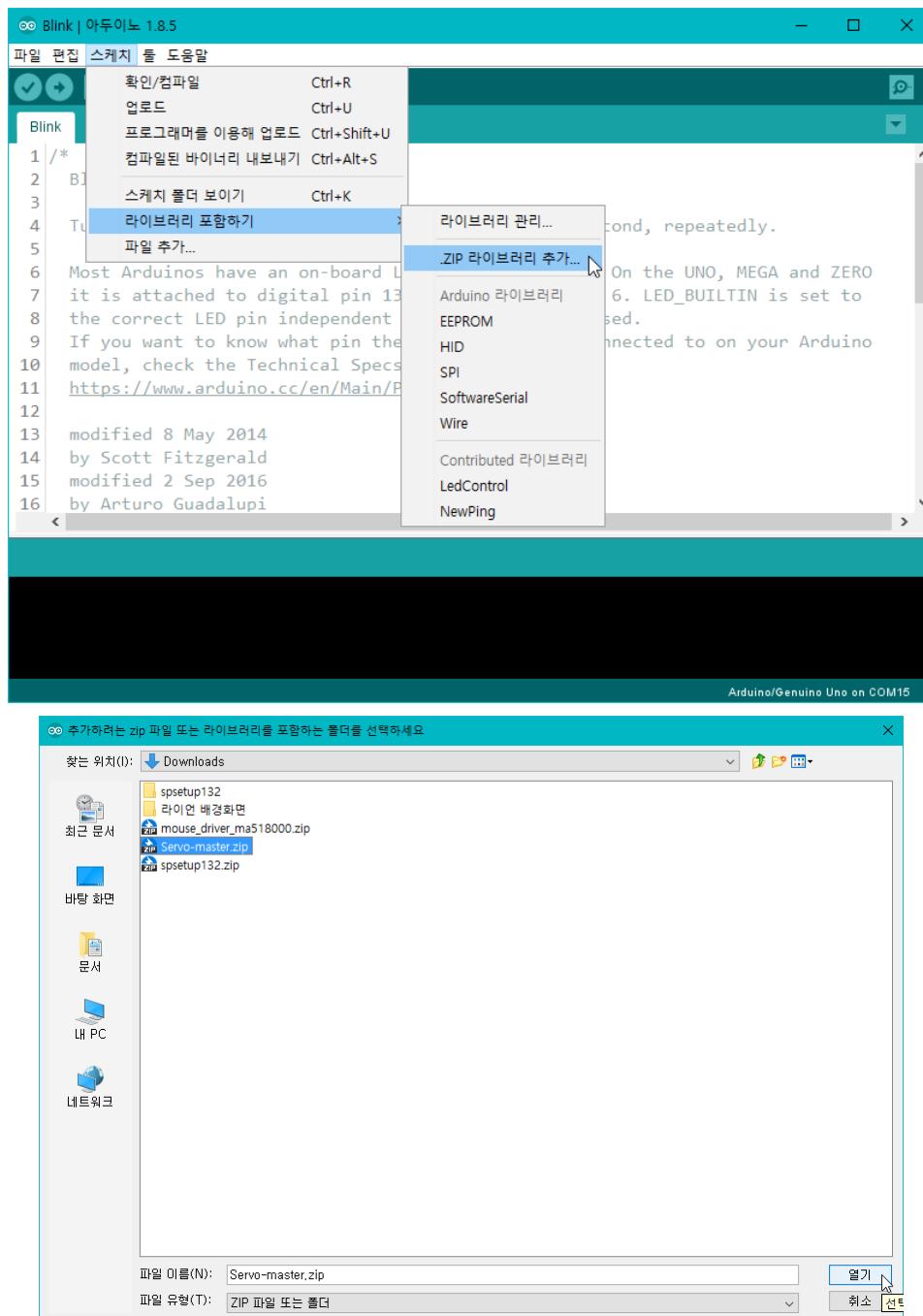
- ✓ 원하는 라이브러리를 찾는다. (예: Servo 라이브러리)



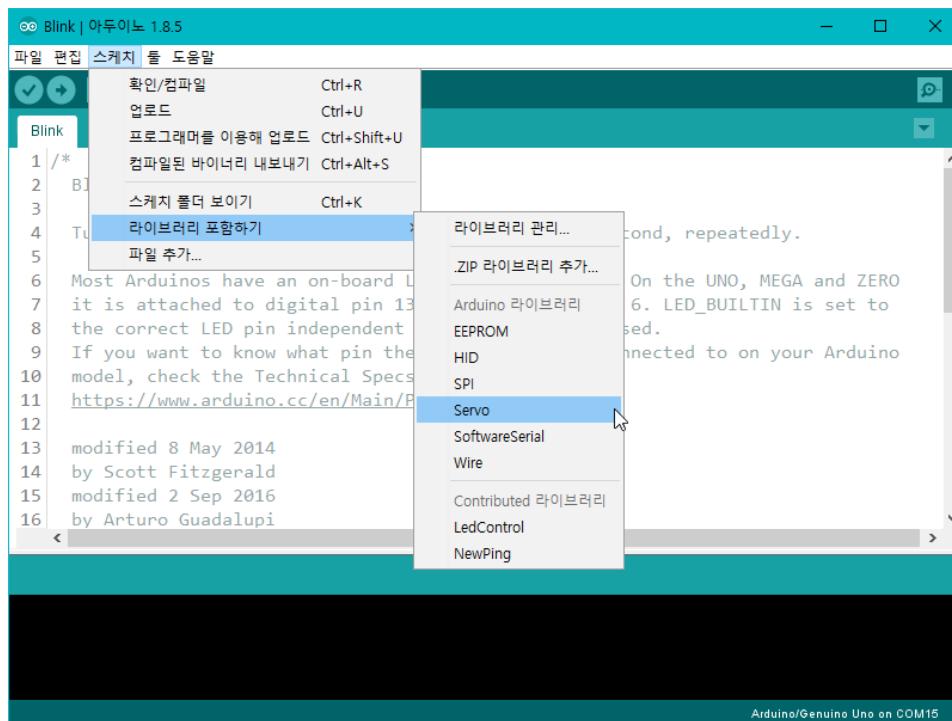
- ✓ Clone or download 버튼을 클릭한 후 Download ZIP 버튼을 클릭하면 라이브러리가 다운로드된다. (Servo-master.zip)



- ✓ 일반적으로 라이브러리는 zip 파일 형태로 배포된다.
- ➊ 라이브러리 설치 방법 1
  - ✓ 메뉴 [스케치 – 라이브러리 포함하기 – ZIP 라이브러리 추가...]를 클릭하고 다운로드한 라이브러리를 선택한다.

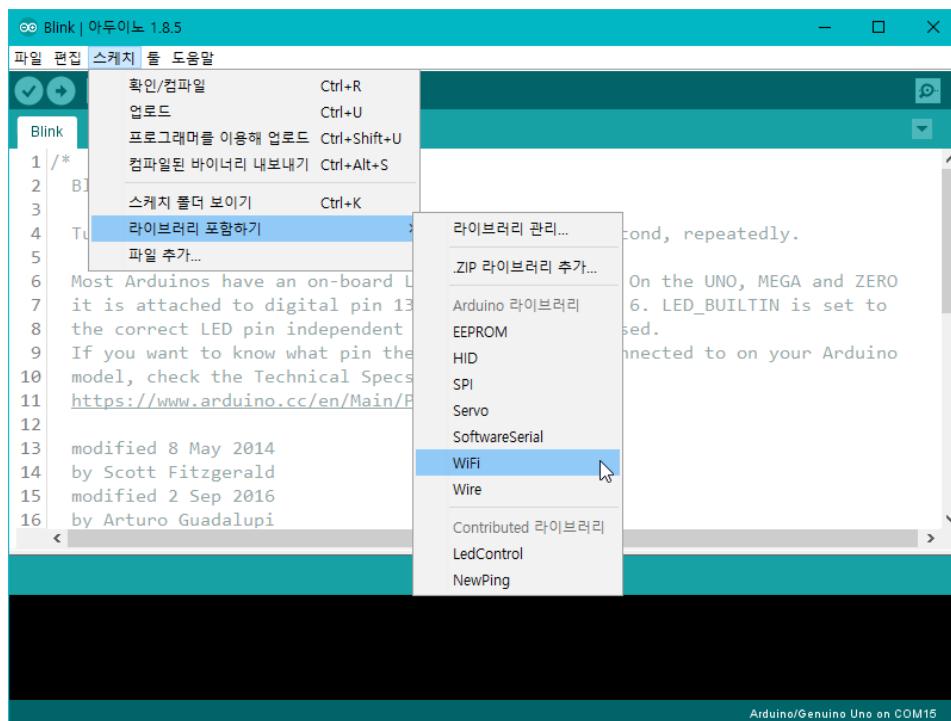


- ✓ 메뉴 [스케치 – 라이브러리 포함하기]에 마우스를 놓으면 설치된 Servo 라이브러리를 볼 수 있다.



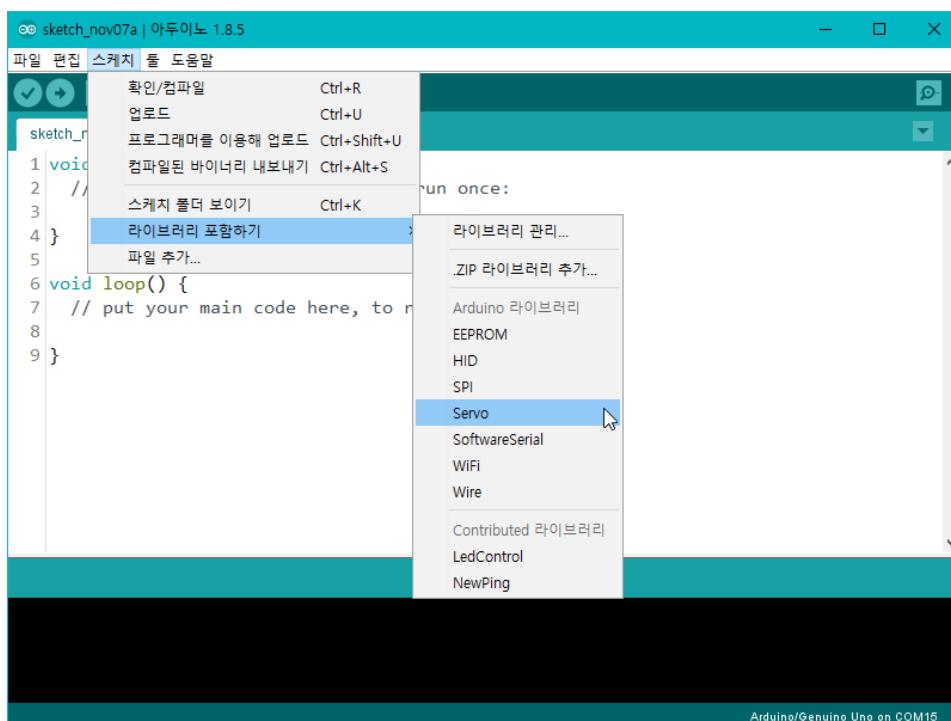
## ➊ 라이브러리 설치 방법 2

- ✓ GitHub에서 WiFi 라이브러리를 다운로드 한다. (WiFi-master.zip)
- ✓ 다운로드한 압축 파일 (WiFi-master.zip)을 압축 해제한 후 폴더 전체 (WiFi-mater)를 아두이노 실행 파일이 있는 폴더 안의 libraries 폴더에 넣는다. (아두이노 아이콘을 마우스 오른쪽 버튼 클릭 후 속성을 선택하면 아두이노 실행 파일이 있는 위치를 알 수 있다.)
- ✓ 메뉴 [스케치 – 라이브러리 포함하기]에 마우스를 놓으면 설치된 WiFi 라이브러리를 볼 수 있다.

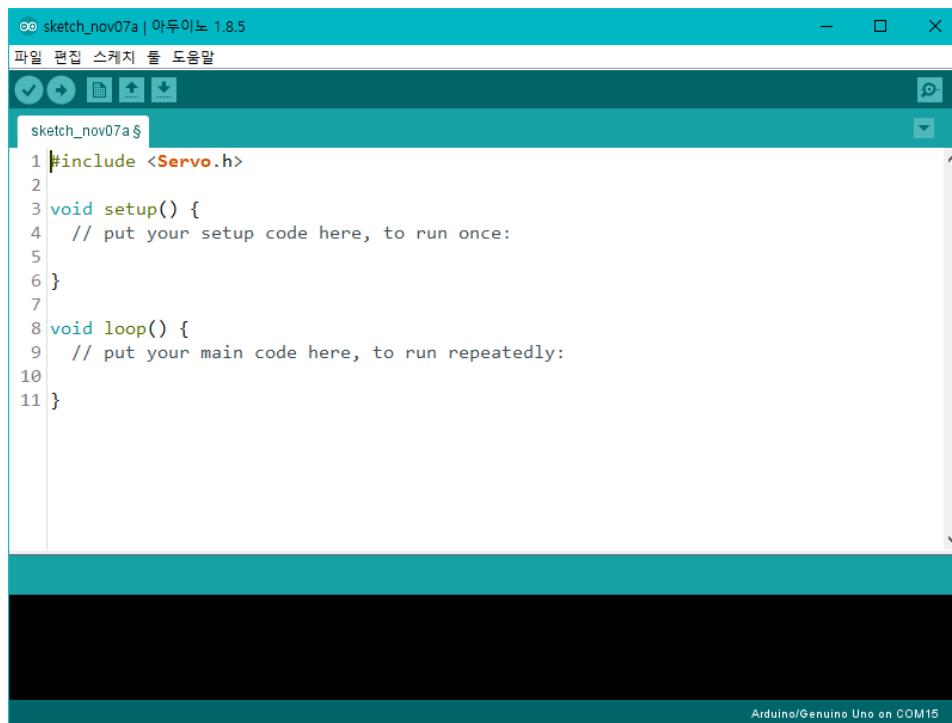


### ▣ 아두이노 라이브러리의 사용 방법 (예: Servo 라이브러리)

- ✓ 메뉴 [스케치 – 라이브러리 포함하기 – Servo]를 클릭하면 헤더 파일이 삽입된다.



- ✓ 헤더 파일이 삽입된 스케치 프로그램



The screenshot shows the Arduino IDE interface. The title bar reads "sketch\_nov07a | 아두이노 1.8.5". The menu bar includes "파일", "편집", "스케치", "툴", and "도움말". Below the menu is a toolbar with icons for file operations. The main area contains the following C++ code:

```
1 #include <Servo.h>
2
3 void setup() {
4     // put your setup code here, to run once:
5 }
6
7 void loop() {
8     // put your main code here, to run repeatedly:
9 }
10
11 }
```

In the status bar at the bottom, it says "Arduino/Genuino Uno on COM15".

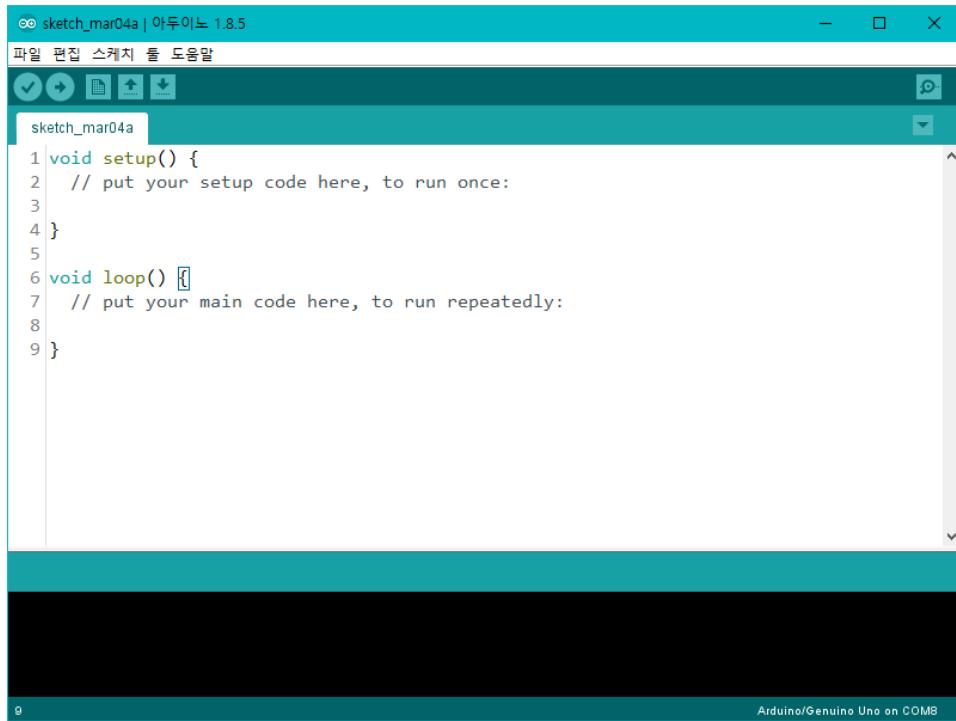
- ✓ 메뉴를 사용하지 않고 직접 헤더 파일 이름(Servo.h)을 다음과 같이 직접 삽입해도 된다.

```
#include <Servo.h>
```

## 2. 아두이노 프로그래밍

### 2.1 아두이노 프로그램

#### ✚ 아두이노 스케치(sketch) 프로그램



The screenshot shows the Arduino IDE interface. The title bar reads "sketch\_mar04a | 아두이노 1.8.5". The menu bar includes "파일", "편집", "스케치", "풀", and "도움말". Below the menu is a toolbar with icons for file operations. The main area displays the code for "sketch\_mar04a":

```
1 void setup() {  
2     // put your setup code here, to run once:  
3 }  
4  
5 void loop() {  
6     // put your main code here, to run repeatedly:  
7 }  
8  
9 }
```

In the bottom right corner, it says "Arduino/Genuino Uno on COM8".

- ✓ **void setup()**
  - ✗ 프로그램이 시작될 때 한 번만 실행된다.
- ✓ **void loop()**
  - ✗ 프로그램이 실행되면 반복적으로 실행된다.
- ✓ **주석 문**
  - ✗ `// ~~~~~`
    - 한 줄을 주석 문으로 처리한다.
  - ✗ `/* ~~~~~ */`
    - 여러 줄을 주석 문으로 처리할 때 사용한다.

## 2.2 아두이노 변수

### ■ 변수 분류

분류	선언문	바이트	범위
정수형	<code>int</code>	2	$-2^{15} \sim 2^{15}-1$
	<code>unsigned int</code>	2	$0 \sim 2^{16}-1$
	<code>long</code>	4	$-2^{31} \sim 2^{31}-1$
	<code>unsigned long</code>	4	$0 \sim 2^{16}-1$
부동 소수형	<code>float</code>	4	$-3.4028235E38 \sim 3.4028235E38$
	<code>double</code>	4	float와 같음
부울 대수형	<code>boolean</code>	1	false(0), true(1)
문자형	<code>char</code>	1	$-128 \sim 127$
	<code>byte</code>	1	$0 \sim 255$
	<code>void</code>		리턴 값 없는 함수에 사용

### ■ 정수형 분류

분류	선언문	바이트	범위
정수형	<code>int8_t</code>	1	$-2^7 \sim 2^7-1$
	<code>uint8_t</code>	1	$0 \sim 2^8-1$
	<code>int16_t</code>	2	$-2^{16} \sim 2^{16}-1$
	<code>uint16_t</code>	2	$0 \sim 2^{16}-1$
	<code>int32_t</code>	4	$-2^{31} \sim 2^{31}-1$
	<code>uint32_t</code>	3	$0 \sim 2^{32}-1$
	<code>int64_t</code>	8	$-2^{63} \sim 2^{63}-1$
	<code>uint64_t</code>	8	$0 \sim 2^{64}-1$

### ■ 아두이노 배열

- ✓ 배열의 인덱스는 0부터 시작한다.

### ■ 배열 선언 방법

```
int inputPin[4];
```

- ✗ 배열의 크기를 지정하고 모두 0으로 초기화 한다.

```
int inputPin[] = {2, 3, 4, 5};
```

- ✗ 주어진 값을 갖는 배열 생성한다.

### ■ 문자열 선언 방법

`Char myName[] = "Baik Heung-Ki";`

- ✖ 문자열은 마지막에 '□0'을 포함한다. (위 배열의 크기는 14)

`String myName = "Baik Heung-Ki";`

- ✖ 다양한 문자열 처리를 위해 String 객체(object)를 사용한다.

#### ✚ 아두이노 상수

`false`

- ✖ 논리 거짓
- ✖ 0으로 정의한다.

`true`

- ✖ 논리 참
- ✖ 0이 아닌 정수는 모두 참이다.

`HIGH`

- ✖ 논리 상태 1 (5 V)
- ✖ 전원 전압이 5 V인 보드에서는 3.0 V 이상인 전압은 HIGH로 인식한다.
- ✖ 전원 전압이 3.3 V인 보드에서는 2.0 V 이상인 전압은 HIGH로 인식한다.

`LOW`

- ✖ 논리 상태 0 (0 V)
- ✖ 전원 전압이 5 V인 보드에서는 1.5 V 이하인 전압은 LOW로 인식한다.
- ✖ 전원 전압이 3.3 V인 보드에서는 1.0 V 이하인 전압은 LOW로 인식한다.

`LED_BUILTIN`

- ✖ 내장된 LED가 연결된 핀 번호 (우노의 경우 디지털 13 번 핀)

#### ✚ 변수 범위 (scope)

##### ✓ 전역 변수

- ✖ 프로그램의 모든 함수에서 볼 수 있는 변수
- ✖ 함수 (`setup()`, `loop()`, 기타 정의된 함수) 밖에서 선언된 변수는 전역 변수이다.

##### ✓ 지역 변수

- ✖ 선언된 함수 안에서만 볼 수 있는 변수
- ✖ 프로그램이 커지면 오직 한 함수만이 접근할 수 있는 지역 변수를 만

드는 것이 좋다.

▶ 변수 한정자 (qualifier)

**const**

- ✖️ 변수를 읽기 전용의 상수로 만든다.
- ✖️ const 변수에 값을 할당하면 컴파일 에러가 생긴다.  
`const float PI = 3.14159;`
  - PI는 항상 3.14159의 값을 갖는 실수형 상수이다.

**#define**

- ✖️ 상수를 변수 이름으로 정의한다.
- ✖️ 상수를 정의할 때 #define 대신 const를 사용하는게 좋다.  
`#define PI 3.14159`
  - PI를 3.14159로 정의한다.

**static**

- ✖️ 지역 변수에 사용한다.
- ✖️ 일반 지역 변수는 함수가 불릴 때마다 만들어지고 함수가 끝나면 없어지지만 static 변수는 함수가 처음 불릴 때 한 번만 만들어지고 초기화되며 함수가 끝나더라도 없어지지 않는다.

**volatile**

- ✖️ 컴파일러가 변수를 저장 레지스터가 아니라 RAM으로부터 변수를 로드하도록 지시한다.
- ✖️ 인터럽트 서비스 루틴에서만 사용된다.

## 2.3 아두이노 제어 문

### ✚ for 문

#### ✓ 문법

```
for (initialization; condition; increment) {  
    statements  
}  
  
✓ for 문 예제
```

---

```
int sum = 0;  
for (int i = 1; i <= 10; i++) {  
    sum = sum + i;  
}
```

---

### ✚ if 문

#### ✓ 문법

```
if (condition) {  
    statements  
}  
  
else if (condition) {  
    statements  
}  
  
else {  
    statements  
}  
  
✓ if 문 예제
```

---

```
int signx;  
if (x > 0) {  
    signx = 1;  
}  
else if (x < 0) {  
    signx = -1;  
}  
else {  
    signx = 0;  
}
```

---

✚ switch case 문

- ✓ 문법

```
switch (var) {
    case label1:
        statements
        break;
    case label2:
        statements
        break;
    default:
        statements
        break;
}
```

✗ 허용 가능한 변수 (var) : 정수, 문자

- ✓ switch case 문 예제

---

```
Char var = 'a';
switch (var) {
    case 'a':
        str = "aaaa";
        break;
    case 'b':
        str = "bbbb";
        break;
    default:
        str = "cccc";
        break;
}
```

---

✚ while 문

- ✓ 문법

```
while (expression) {
    statements
}
```

- ✓ while 문 예제

---

```
int sum = 0;
int x = 1;
```

```
while (x <= 10) {  
    sum = sum + x;  
    x++;  
}
```

#### ✚ do... while 문

- ✓ 문법

```
do {  
    statements  
} while (expression);
```

- ✓ do... while 문 예제

```
int sum = 0;  
int x = 0;  
do {  
    sum = sum + x;  
    x++;  
} while (x <= 10);
```

#### ✚ break 문

- ✓ do, for, while loop에서 빠져 나올 때 사용
- ✓ break 문 예제

```
sum = 0;  
x = 1;  
while (true) {  
    sum += x;  
    x++;  
    if (x > 10) {  
        break;  
    }  
}
```

## 2.4 아두이노 연산자

### ▣ 관계 연산자

연산자	기능	예	결과
<code>==</code>	같다	<code>2 == 3</code>	<code>false</code>
<code>!=</code>	다르다	<code>2 != 3</code>	<code>true</code>
<code>&gt;</code>	크다	<code>2 &gt; 3</code>	<code>false</code>
<code>&lt;</code>	작다	<code>2 &lt; 3</code>	<code>true</code>
<code>&gt;=</code>	크거나 같다	<code>2 &gt;= 3</code>	<code>false</code>
<code>&lt;=</code>	작거나 같다	<code>2 &lt;= 3</code>	<code>true</code>

### ▣ 논리 연산자

연산자	기능	예	결과
<code>&amp;&amp;</code>	논리 곱 (and)	<code>true &amp;&amp; false</code>	<code>false</code>
<code>  </code>	논리 합 (or)	<code>true    false</code>	<code>true</code>
<code>!</code>	부정 (not)	<code>! true</code>	<code>false</code>

### ▣ 비트 연산자

연산자	기능	예	결과
<code>&amp;</code>	비트 곱	<code>3 &amp; 1 (11 &amp; 01)</code>	<code>1 (01)</code>
<code> </code>	비트 합	<code>3   1 (11   01)</code>	<code>3 (11)</code>
<code>^</code>	배타적 비트 합	<code>3 ^ 1 (11 ^ 01)</code>	<code>2 (10)</code>
<code>~</code>	비트 부정	<code>~ 1 (~ 00000001)</code>	<code>254 (10000000)</code>

- ✖ 주어진 숫자를 이진수로 바꾸고 비트 끼리 연산한다.
- ✖ 비트 부정 결과는 데이터의 비트 수에 따라 달라진다.

 복합 연산자

연산자	예	등가식	결과
<code>+=</code>	<code>x += 2</code>	<code>x = x + 2</code>	<code>x</code> 에 2를 더한다
<code>-=</code>	<code>x -= 2</code>	<code>x = x - 2</code>	<code>x</code> 에 2를 뺀다
<code>*=</code>	<code>x *= 2</code>	<code>x = x * 2</code>	<code>x</code> 에 2를 곱한다
<code>/=</code>	<code>x /= 2</code>	<code>x = x / 2</code>	<code>x</code> 에 2를 나눈다
<code>&gt;&gt;=</code>	<code>x &gt;&gt;= 2</code>	<code>x = x &gt;&gt; 2</code>	<code>x</code> 를 우측으로 2비트 이동
<code>&lt;&lt;=</code>	<code>x &lt;&lt;= 2</code>	<code>x = x &lt;&lt; 2</code>	<code>x</code> 를 좌측으로 2비트 이동
<code>&amp;=</code>	<code>x &amp;= 2</code>	<code>x = x &amp; 2</code>	<code>x</code> 와 2를 비트 곱 연산
<code> =</code>	<code>x  = 2</code>	<code>x = x   2</code>	<code>x</code> 와 2를 비트 합 연산
<code>^=</code>	<code>x ^= 2</code>	<code>x = x ^ 2</code>	<code>x</code> 와 2를 배타적 비트 합 연산
<code>++</code>	<code>x++</code>	<code>x = x + 1</code>	<code>x</code> 를 1만큼 증가시킨다
<code>--</code>	<code>x--</code>	<code>x = x - 1</code>	<code>x</code> 를 1만큼 감소시킨다

## 2.5 아두이노 프로그램과 메모리

### ✚ 아두이노 Uno 메모리

- ✓ 플래시 (flash) 메모리
  - ✗ 용량 : 32 kbyte
  - ✗ 프로그램이 업로드 되는 공간
- ✓ SRAM 메모리
  - ✗ 용량 : 2 kbyte
  - ✗ 스캐치 프로그램의 변수가 저장되는 공간
- ✓ EEPROM 메모리
  - ✗ 용량 : 1 kbyte

### ✚ 아두이노 프로그램 작성 시 주의 사항

- ✓ SRAM 메모리가 플래시 메모리에 비해 상대적으로 작다.
- ✓ 프로그램이 크고 변수가 많아지면 SRAM 부족 현상이 발생할 수 있다.
- ✓ 메모리 부족이 문제가 될 경우 SRAM 대신 플래시 메모리를 사용해야 한다.

### ✚ PROGMEM

- ✓ pgmspace.h 에 정의된 데이터 형태에 사용 가능하다.
- ✓ 선언된 데이터를 SRAM 대신 Flash 메모리에 저장한다.
- ✓ 스캐치 프로그램 실행 중 변경되지 않을 고정된 변수에만 사용 가능하다.
- ✓ 스캐치 프로그램 실행 중 변경되는 변수는 SRAM 메모리를 사용한다.
- ✓ 주로 긴 배열 데이터나 긴 문자열에 사용한다.
- ✓ PROGMEM 을 사용하기 위해서는 변수가 광역 변수로 정의되거나 static 키워드로 정의되어야 한다.

### ✚ PROGMEM 사용법

- ✓ 문법

```
const datatype variableName[] PROGMEM = { ... }
const PROGMEM datatype variableName[] = { ... }
const datatype PROGMEM variableName[] = { ... }
```

- ✗ PROGMEM은 주로 배열 변수에 사용한다.

---

 ✓ PROGMEM 예제
 

---

```
// 헤더 파일
#include <avr/pgmspace.h>

// 부호가 없는 2 바이트 (16 비트) 정수 배열을 flash 메모리에 할당한다.
const PROGMEM uint16_t charSet[] = { 65000, 32796, 16843, 10, 11234 };
// 문자열 변수를 flash 메모리에 할당한다.
const char signMessage[] PROGMEM = {"I AM PREDATOR, UNSEEN COMBATANT.
CREATED BY THE UNITED STATES DEPART"};

unsigned int displayInt;
int k;
char myChar;

void setup() {
    Serial.begin(9600);
    // 시리얼 포트가 연결되기를 기다린다.
    while (!Serial) ;
    // 2 바이트 정수 열을 읽고 나타낸다.
    for (k = 0; k < 5; k++) {
        displayInt = pgm_read_word_near(charSet + k);
        Serial.println(displayInt);
    }
    Serial.println();
    // 문자열을 읽고 나타낸다.
    for (k = 0; k < strlen_P(signMessage); k++) {
        myChar = pgm_read_byte_near(signMessage + k);
        Serial.print(myChar);
    }
    Serial.println();
}

void loop() {
```

---

- ✖ int 배열과 char 배열이 PROGMEM 키워드로 선언되었다.
- ✖ pgm\_read\_word\_near(addr)와 pgm\_read\_byte\_near은 pgmspace.h에 정의된 매크로로서 변수의 주소로 변수 값을 읽는다. (입력 변수 형식이 포인터이다.)

`#define pgm_read_byte(addr) (*(const unsigned char *) (addr))`

```
#define pgm_read_word(addr) (*(const unsigned short *)(addr))
#define pgm_read_byte_near(addr) pgm_read_byte(addr)
#define pgm_read_word_near(addr) pgm_read_word(addr)
```

✓ 시리얼 모니터 출력



✚ 관련 사이트

- ✓ <https://www.arduino.cc/reference/ko/language/variables/utilities/progmem/>
- ✓ <https://www.arduino.cc/reference/en/language/variables/utilities/progmem/>

### 3. 아두이노 입/출력

#### ▣ 아두이노 우노 입/출력 핀

- ✓ 디지털 입/출력 핀 : 0 ~ 13
- ✓ 아날로그 입력 핀 : A0 ~ A5
- ✓ 아날로그 출력 핀 : 3, 5, 6, 9, 10, 11 (~ 표시)



### 3.1 디지털 출력

#### 관련 함수

```
while (expression) {
pinMode(pin, OUTPUT);
    * 아두이노 핀(pin)을 디지털 출력 모드로 설정한다.

digitalWrite(pin, HIGH);
    * 아두이노 핀(pin)에 디지털 HIGH (5 V)를 출력한다.

digitalWrite(pin, LOW);
    * 아두이노 핀(pin)에 디지털 LOW (0 V)를 출력한다.
```

#### 참고 사항

- ✓ 디지털 입/출력 모드 설정은 일반적으로 디지털 핀 (0 ~ 13)에 대해 한다.
- ✓ 아날로그 입력 핀 (A0 ~ A5)도 모드 설정을 통해 디지털 입/출력 모드로 사용할 수 있다.

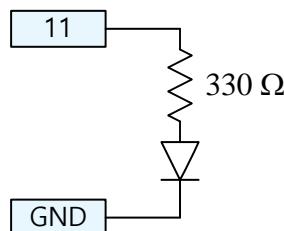
#### 예제 3-1-1

#### 목표

- ✓ LED 1 개를 1 Hz 의 속도로 점멸시킨다.

#### 회로도

- ✓ LED 는 저항 (수  $100\Omega$ )과 같이 연결해야 한다. 저항값에 따라 LED 의 밝기가 결정된다.
- ✓ 디지털 11 번 핀과 접지 사이에 저항과 LED 를 직렬로 연결한다.



#### 스케치 프로그램

```
// Blink LED at 1 Hz

// 디지털 11 번 핀에 LED를 연결한다.
const int ledPin = 11;
const int delayTime = 500;
```

```

void setup() {
    // 디지털 11 번 핀을 디지털 출력으로 설정한다.
    pinMode(ledPin, OUTPUT);
}

void loop() {
    // 디지털 11 번 핀에 HIGH를 내보낸다. LED가 켜진다.
    digitalWrite(ledPin, HIGH);
    // 500 μs를 유지한다. (500 μs동안 LED가 켜진다.)
    delay(delayTime);
    // 디지털 11 번 핀에 LOW를 내보낸다. LED가 꺼진다.
    digitalWrite(ledPin, LOW);
    // 500 μs를 유지한다. (500 μs동안 LED가 꺼진다.)
    delay(delayTime);
}

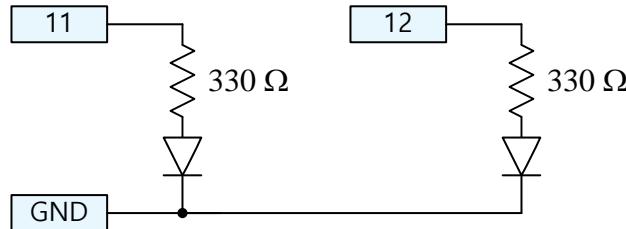
```

**예제 3-1-2****▣ 목표**

- ✓ LED 2 개를 1 Hz 의 속도로 점멸시킨다.

**▣ 회로도**

- ✓ 디지털 11 번 핀과 접지 사이에 저항과 LED 를 직렬 연결하고, 12 번 핀과 접지 사이에 저항과 LED 를 직렬 연결한다.

**▣ 스케치 프로그램**

```

// Blink 2 LED at 1 Hz

// 디지털 11번과 12번에 LED를 연결한다.
const int ledPin1 = 11;
const int ledPin2 = 12;
const int delayTime = 500;

void setup() {
    pinMode(ledPin1, OUTPUT);

```

```

    pinMode(ledPin2, OUTPUT);
}

void loop() {
    // 디지털 11번 핀과 12번 핀에 HIGH를 내 보내 LED를 켠다.
    digitalWrite(ledPin1, HIGH);
    digitalWrite(ledPin2, HIGH);
    delay(delayTime);
    // 디지털 11번 핀과 12번 핀에 LOW를 내 보내 LED를 끈다.
    digitalWrite(ledPin1, LOW);
    digitalWrite(ledPin2, LOW);
    delay(delayTime);
}

```

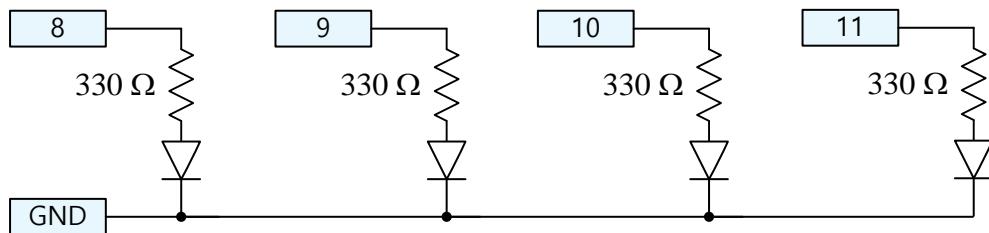
### 예제 3-1-3

#### ▣ 목표

- ✓ LED 4 개를 1 Hz 의 속도로 순서대로 점멸시킨다.

#### ▣ 회로도

- ✓ 디지털 8, 9, 10, 11 번 핀과 접지 사이에 저항과 LED 를 직렬 연결한다.



#### ▣ 스케치 프로그램

```

// Blink 4 LED at 1 Hz

const int delayTime = 1000;
// LED 연결핀
int ledPin[] = {8, 9, 10, 11}

void setup() {
    // 4 개의 LED 연결 핀을 출력 모드로 설정하고 LED를 모두 끈다.
    for (int i = 0; i < 4; i++) {
        pinMode(ledPin[i], OUTPUT);
        digitalWrite(ledPin[i], LOW);
    }
}

```

```

void loop() {
    // 4 개의 LED를 순서대로 켰다가 끈다.
    for (int i = 0; i < 4; i++) {
        digitalWrite(ledPin[i], HIGH);
        delay(delayTime);
        digitalWrite(ledPin[i], LOW);
    }
}

```

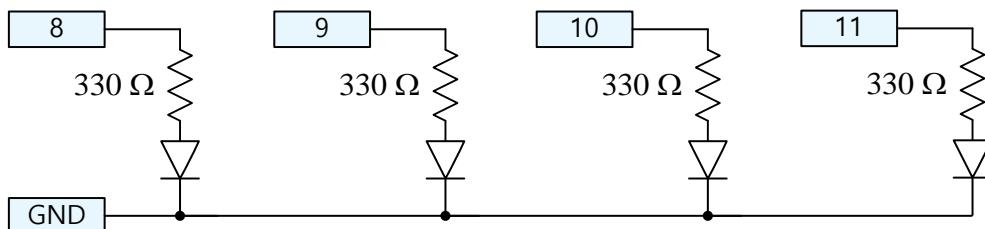
### 예제 3-1-4

#### ▣ 목표

- ✓ 500  $\mu$ s 마다 랜덤한 값을 발생시키고 이 값에 따라 4 개의 LED 를 값의 크기에 따라 점멸시킨다. (bar 모드)

#### ▣ 회로도

- ✓ 디지털 8, 9, 10, 11 번 핀과 접지 사이에 저항과 LED 를 직렬 연결한다.



#### ▣ 스케치 프로그램

```

// Random blink 4 LED at 2 Hz

const int delayTime = 500;
// LED 연결핀
int ledPin[] = {8, 9, 10, 11};

void setup() {
    // 4 개의 LED 연결 핀을 출력 모드로 설정하고 LED를 모두 끈다.
    for (int i = 0; i < 4; i++) {
        pinMode(ledPin[i], OUTPUT);
        digitalWrite(ledPin[i], LOW);
    }
}

void loop() {
    // 0 ~ 4 사이의 랜덤 값(정수)을 발생시킨다.
    int ri = random(5);

```

```
// 발생한 랜덤 값에 해당하는 LED 까지만 켠다.  
for (int i = 0; i < ri; i++) {  
    digitalWrite(ledPin[i], HIGH);  
}  
delay(delayTime);  
turnOff_allLed();  
  
// 모든 LED를 끄는 함수  
void turnOff_allLed() {  
    for (int i = 0; i < 4; i++) {  
        digitalWrite(ledPin[i], LOW);  
    }  
}
```

## 연습 문제

### ➊ 연습 문제 3-1-1

- ✓ 4 개의 LED 를 1 Hz 의 속도로 좌우로 점멸하는 회로를 구성하고 프로그램을 작성하라. (dot 모드)

### ➋ 연습 문제 3-1-2

- ✓ 4 개의 LED 를 1 Hz 의 속도로 좌우로 점멸하는 회로를 구성하고 프로그램을 작성하라. (bar 모드)

### ➌ 연습 문제 3-1-3

- ✓ 500 μs 마다 랜덤한 값을 발생시키고 이 값에 따라 4 개의 LED 를 값의 크기에 따라 점멸하는 회로를 구성하고 프로그램을 작성하라. (dot 모드)

## 3.2 디지털 입력

### 관련 함수

```
pinMode(pin, INPUT);
pinMode(pin, INPUT_PULLUP);
```

- ✗ 아두이노 핀(pin)을 디지털 입력 모드로 설정한다.

```
digitalRead(pin);
```

- ✗ 아두이노 핀(pin)에 가해진 디지털 값을 읽는다.
- ✗ 전원 전압이 5V인 보드에서는 3.0V 이상을 HIGH로 읽고 1.5V 이하를 LOW로 읽는다.
- ✗ 전원 전압이 3.3V인 보드에서 2.0V 이상을 HIGH로 읽고 1.0V 이하를 LOW로 읽는다.

### 입력 모드의 종류

#### ✓ INPUT 모드

- ✗ 아두이노 내부의 풀-업 저항을 비활성화 한다.

#### ✓ INPUT\_PULLUP 모드

- ✗ 아두이노 내부의 풀-업 저항을 활성화 한다.

### 아두이노 핀에 0과 1을 인가하는 방법

#### ✓ 풀-다운 저항을 사용하는 방법

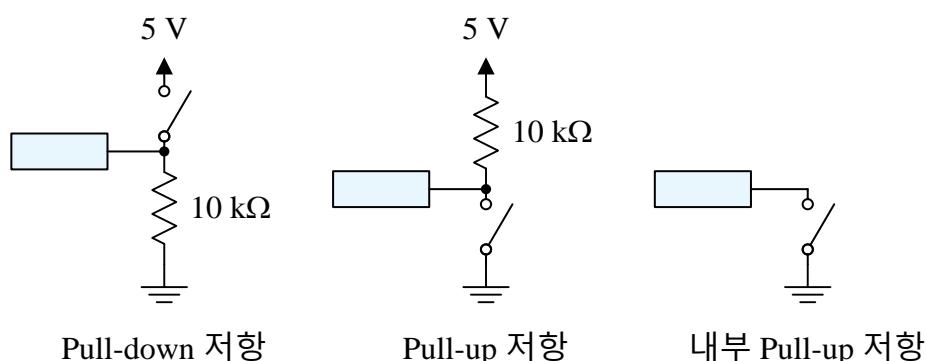
- ✗ 스위치가 ON이면 5V, 스위치가 OFF이면 0V가 인가된다.

#### ✓ 풀-업 저항을 사용하는 방법

- ✗ 스위치가 ON이면 0V, 스위치가 OFF이면 5V가 인가된다.

#### ✓ 내부 풀-업 저항을 사용하는 경우

- ✗ 스위치가 ON이면 0V, 스위치가 OFF이면 5V가 인가된다.

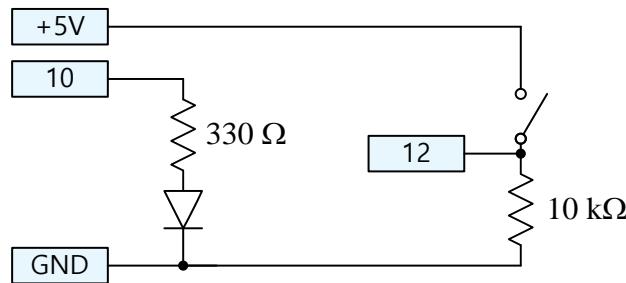


**예제 3-2-1****▣ 목표**

- ✓ 풀-다운 저항을 사용하여 스위치가 눌릴 때 LED 가 켜지도록 한다.

**▣ 회로도**

- ✓ 디지털 10 번 핀에 저항과 LED 를 직렬 연결하고, 디지털 12 번 핀에 스위치를 연결한다.

**▣ 스케치 프로그램**

```

// Turn on LED if switch is pressed
// Switch mode is pulldown

const int btnPin = 12;
const int ledPin = 10;

void setup() {
    pinMode(btnPin, INPUT);
    pinMode(ledPin, OUTPUT);
}

void loop() {
    // 스위치(버튼) 상태를 이용해 LED를 On/Off 한다.
    int btnState = digitalRead(btnPin);
    digitalWrite(ledPin, btnState);
}

```

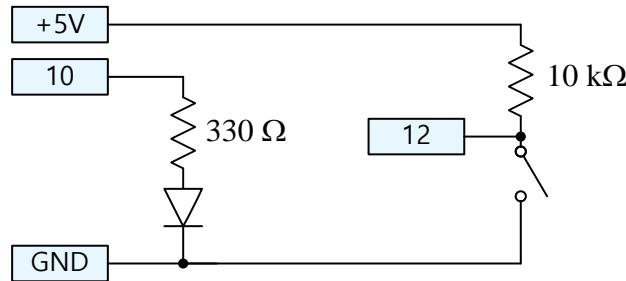
- ✓ 풀-다운 저항을 사용할 경우 스위치를 누르면 버튼 상태가 HIGH 가 되고, 스위치를 떼면 버튼 상태가 LOW 가 되므로 버튼 상태를 디지털 출력으로 LED 에 내 보내면 된다.

**예제 3-2-2****▣ 목표**

- ✓ 풀-업 저항을 사용하여 스위치가 눌릴 때 LED 가 켜지도록 한다.

#### ▣ 회로도

- ✓ 디지털 10 번 핀에 저항과 LED 를 직렬 연결하고, 디지털 12 번 핀에 스위치를 연결한다.



#### ▣ 스케치 프로그램

```
// Turn on LED if switch is pressed
// Switch mode is pullup

const int btnPin = 12;
const int ledPin = 10;

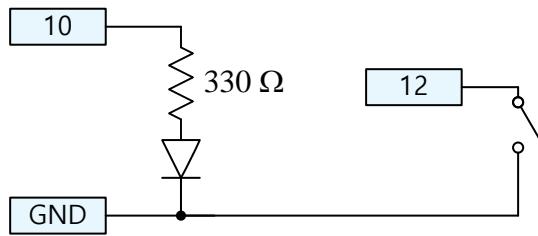
void setup() {
    pinMode(btnPin, INPUT);
    pinMode(ledPin, OUTPUT);
}

void loop() {
    // 스위치 상태를 이용해 LED를 On/Off 한다.
    int btnState = digitalRead(btnPin);
    digitalWrite(ledPin, !btnState);
}
```

- ✓ 풀-업 저항을 사용할 경우 스위치를 누르면 버튼 상태가 LOW 가 되고, 스위치를 떼면 버튼 상태가 HIGH 가 되므로 버튼 상태의 반대 상태를 디지털 출력으로 LED 에 내 보내면 된다.

#### ▣ 내부 풀-업 저항을 사용할 경우

- ✓ 회로도
  - ✗ 디지털 10번 핀에 저항과 LED를 직렬 연결하고, 디지털 12번 핀에 스위치를 연결한다.



✗ 내부 풀-업 저항을 사용하므로 풀-업 저항  $10\text{ k}\Omega$ 을 제거할 수 있다.

✓ 스케치 프로그램

```
// Turn on LED if switch is pressed
// Switch mode is internal pullup

const int btnPin = 12;
const int ledPin = 10;

void setup() {
    pinMode(btnPin, INPUT_PULLUP);
    pinMode(ledPin, OUTPUT);
}

void loop() {
    // 스위치 상태를 이용해 LED를 On/Off 한다.
    int btnState = digitalRead(btnPin);
    digitalWrite(ledPin, !btnState);
}
```

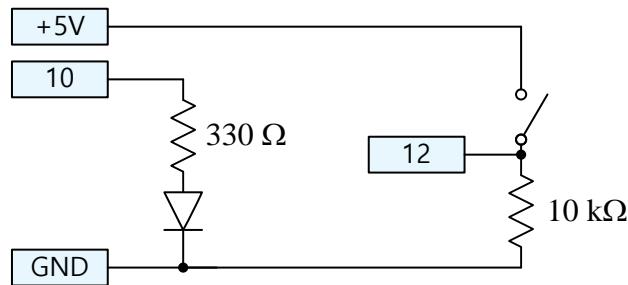
**예제|| 3-2-3**

✚ 목표

✓ 풀-다운 저항을 사용하여 스위치가 눌릴 때마다 LED 가 점멸하도록 한다.

✚ 회로도

✓ 디지털 10 번 핀에 저항과 LED 를 직렬 연결하고, 디지털 12 번 핀에 스위치를 연결한다.



 스케치 프로그램

```
// Turn on and off LED (toggling) if switch is pressed
// Switch mode is pulldown

const int btnPin = 12;
const int ledPin = 10;
int ledState = 0;
int lastBtnState = 0;

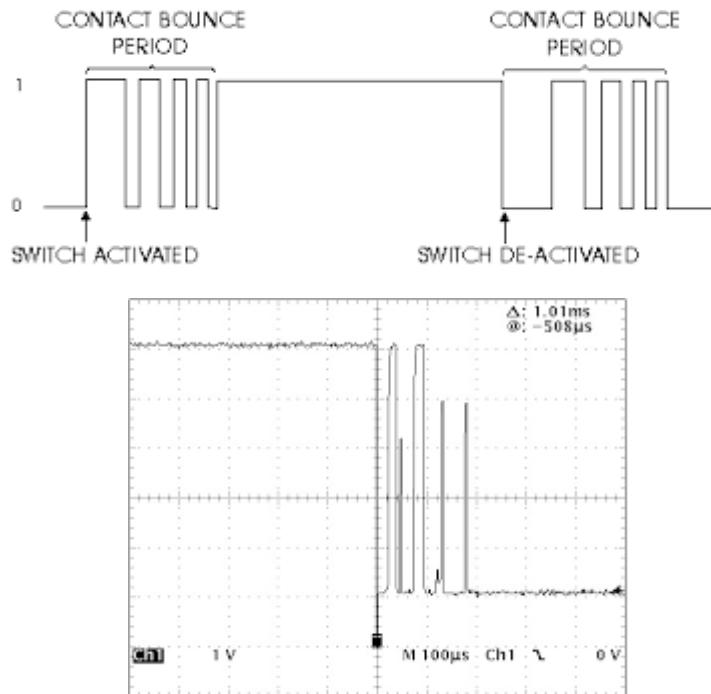
void setup() {
    pinMode(btnPin,INPUT);
    pinMode(ledPin,OUTPUT);
}

void loop() {
    // 스위치가 눌러지면 LED를 On/Off 한다.
    if (pressBtn(btnPin) == 1) {
        ledState = !ledState;
        digitalWrite(ledPin,ledState);
    }
}

// 스위치가 눌려졌는지 감지하는 함수
int pressBtn(int bPin) {
    int btnState = digitalRead(bPin);
    // 스위치 상태가 전 상태와 같으면 0을 돌려준다.
    if (btnState == lastBtnState) {
        return 0;
    }
    // 스위치 상태가 전 상태와 다르면 스위치 상태를 전 상태에 저장하고
    // 스위치가 눌려지면 상태가 1이 되고, 스위치를 떼면 상태가 0이 되므로
    // 스위치가 눌려졌을 때 (상태가 1일 때) 1을 돌려 준다.
    else {
        lastBtnState = btnState;
        if (btnState == 1) {
            return 1;
        }
        else {
            return 0;
        }
    }
}
```

✚ 스위치의 bouncing 효과

- ✓ 스위치가 눌리면 기계적인 특성으로 아주 짧은 시간 동안 스위치가 ON/OFF 하는 현상



✚ Debouncing – 스위치의 bouncing 효과 제거

- ✓ 하드웨어 방법
- ✓ 소프트웨어 방법

✗ 스케치 프로그램 파일 – 예제 – 02 Digital – Debounce 참조

- 스위치가 눌리고 일정 시간 (스위치의 기계적인 상태가 안정화 되는데 걸리는 시간)이 지난 후의 상태를 조사하여 스위치의 눌림 여부를 판단한다.

✗ Bounce2.h 라이브러리 사용

- <http://playground.arduino.cc/Code/Bounce>

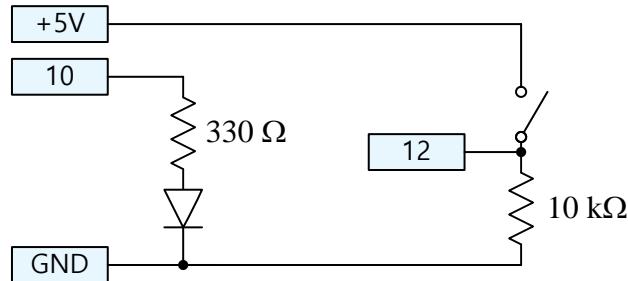
**예제 3-2-4**

✚ 목표

- ✓ 풀-다운 저항을 사용하여 스위치가 눌릴 때마다 LED 가 점멸하도록 한다.
- ✓ 소프트웨어 방법으로 debouncing 한다.

✚ 회로도

- ✓ 디지털 10 번 핀에 저항과 LED 를 직렬 연결하고, 디지털 12 번 핀에 스위치를 연결한다.



### ▶ 스케치 프로그램

```
// Turn on and off LED (toggling) if switch is pressed
// Switch mode is pulldown and debouncing

const int btnPin = 12;
const int ledPin = 10;
int ledState = LOW;
int btnState;
int lastBtnState = LOW;
// 스위치가 눌리는 시간 (μs)을 저장하는 변수
unsigned long lastDbTime = 0;
// bouncing 효과가 없어질 때까지 걸리는 시간 (μs)
unsigned long dbDelay = 50;

void setup() {
    pinMode(btnPin, INPUT);
    pinMode(ledPin, OUTPUT);
    digitalWrite(ledPin, ledState);
}

void loop() {
    // 스위치 상태를 읽는다.
    int reading = digitalRead(btnPin);
    // 읽은 스위치 상태가 전의 상태와 다르면 debouncing timer를 초기화 한다.
    if (reading != lastBtnState) {
        lastDbTime = millis();
    }
    // 스위치가 새로 눌러진 시간과 과거 눌렸던 시간을 비교하여 주어진
    // 시간 보다 크면 스위치가 실제 눌러졌다고 판단한다.
    if ((millis() - lastDbTime) > dbDelay) {
        // 읽은 스위치 상태가 저장된 스위치 상태와 다르면 읽은 스위치 상태를
        // 저장한다.
    }
}
```

```

        if (reading != btnState) {
            btnState = reading;
            // 스위치가 HIGH로 바뀌었을 때만 (눌리면 HIGH) LED 상태를 바꾼다.
            if (btnState == HIGH) {
                ledState = !ledState;
            }
        }
    }
    // LED 상태에 따라 LED를 on/off하고 읽은 스위치 상태를 저장한다.
    digitalWrite(ledPin, ledState);
    lastBtnState = reading;
}

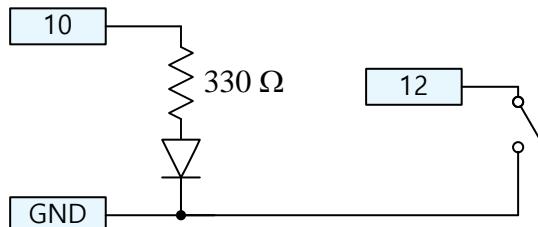
```

**예제 3-2-5****▣ 목표**

- ✓ 내부 풀-업 저항을 사용하여 스위치가 눌릴 때마다 LED 가 점멸하도록 한다.
- ✓ 소프트웨어 방법으로 debouncing 한다.

**▣ 회로도**

- ✓ 디지털 10 번 핀에 저항과 LED 를 직렬 연결하고, 디지털 12 번 핀에 스위치를 연결한다.

**▣ 스케치 프로그램**

```

// Turn on and off LED (toggling) if switch is pressed
// Switch mode is internal pullup and debouncing

const int btnPin = 12;
const int ledPin = 10;
int ledState = LOW;
int btnState;
int lastBtnState = LOW;
// 스위치가 눌리는 시간 (μs)을 저장하는 변수
unsigned long lastDbTime = 0;

```

```

// bouncing 효과가 없어질 때까지 걸리는 시간 (μs)
unsigned long dbDelay = 50;

void setup() {
    pinMode(btnPin, INPUT_PULLUP);
    pinMode(ledPin, OUTPUT);
    digitalWrite(ledPin, ledState);
}

void loop() {
    // 스위치 상태를 읽는다.
    int reading = digitalRead(btnPin);
    // 읽은 스위치 상태가 전의 상태와 다르면 debouncing timer를 초기화 한다.
    if (reading != lastBtnState) {
        lastDbTime = millis();
    }
    // 스위치가 새로 눌러진 시간과 과거 눌렸던 시간을 비교하여 주어진
    // 시간 보다 크면 스위치가 실제 눌러졌다고 판단한다.
    if ((millis() - lastDbTime) > dbDelay) {
        // 읽은 스위치 상태가 저장된 스위치 상태와 다르면 읽은 스위치 상태를
        // 저장한다.
        if (reading != btnState) {
            btnState = reading;
            // 스위치가 LOW로 바뀌었을 때만 (눌리면 LOW) LED 상태를 바꾼다.
            if (btnState == LOW) {
                ledState = !ledState;
            }
        }
    }
    // LED 상태에 따라 LED를 on/off하고 읽은 스위치 상태를 저장한다.
    digitalWrite(ledPin, ledState);
    lastBtnState = reading;
}

```

- ✓ pinMode() 함수에 INPUT\_PULLUP 을 사용한 것을 제외하면 예제 3-2-4 의 스케치 프로그램과 같다.

#### ▣ Bounce2 라이브러리

[Bounce2.h](#)

- ✓ Debouncing 라이브러리
- ✓ <https://github.com/thomasfredericks/Bounce2>

#### ▣ 관련 함수

```

Bounce();
    × Bounce 객체

.attach(pin);
    × 스위치를 아두이노와 연결한다.

    × pin : 스위치가 연결되는 아두이노 핀 번호

.interval(time);
    × Debouncing 시간을 설정한다.

    × time : debouncing 시간 (ms)

.update();
    × 객체를 업데이트한다.

    × loop()에 객체 당 한 번만 실행한다.

.read();
    × 업데이트된 핀 상태를 읽는다.

.fell();
    × 핀의 상태가 HIGH에서 LOW로 변하면 true를 돌려 준다.

.rose();
    × 핀의 상태가 LOW에서 HIGH로 변하면 true를 돌려 준다.

```

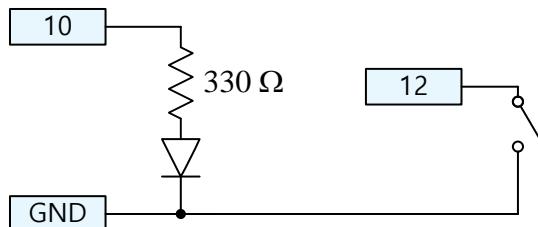
### 예제 3-2-6

#### ▣ 목표

- ✓ 내부 풀-업 저항을 사용하여 스위치가 눌릴 때마다 LED 가 점멸하도록 한다.
- ✓ Bounce2.h 라이브러리를 사용하여 debouncing 한다.

#### ▣ 회로도

- ✓ 디지털 10 번 핀에 저항과 LED 를 직렬 연결하고, 디지털 12 번 핀에 스위치를 연결한다.



#### ▣ 스케치 프로그램

---

✓ Bounce2.h 라이브러리 사용

---

```
// Turn on and off LED (toggling) if switch is pressed
// Switch mode is internal pullup and debouncing
// Library : Bounce2.h

#include <Bounce2.h>

const int btnPin = 12;
const int ledPin = 10;
// Bounce 객체 선언
Bounce Db = Bounce();
int ledState = LOW;

void setup() {
    pinMode(btnPin, INPUT_PULLUP);
    pinMode(ledPin, OUTPUT);
    digitalWrite(ledPin, ledState);
    // 아두이노 12 번 핀을 스위치에 연결한다.
    Db.attach(btnPin);
    // debouncing 시간을 20 ms로 설정한다.
    Db.interval(20);
}

void loop() {
    Db.update();
    // 스위치 상태가 HIGH에서 LOW로 바뀌면 LED 상태를 바꾼다.
    if (Db.fell()) {
        ledState = !ledState;
        digitalWrite(ledPin, ledState);
    }
}
```

---

- ✓ 회로도에서 스위치가 눌리면 스위치 상태가 HIGH에서 LOW로 변하기 때문에 Bounce.fell() 함수를 사용한다.

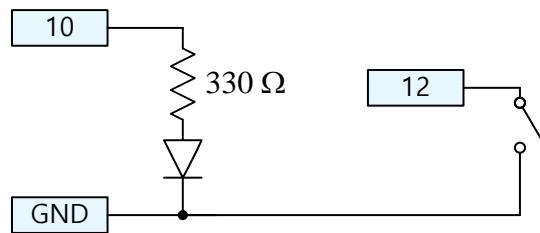
**예제 3-2-7**

 **목표**

- ✓ 내부 풀-업 저항을 사용하여 스위치를 뗄 때마다 LED 가 점멸하도록 한다.
- ✓ Bounce2.h 라이브러리를 사용하여 debouncing 한다.

 **회로도**

- ✓ 디지털 10 번 핀에 저항과 LED 를 직렬 연결하고, 디지털 12 번 핀에 스위치를 연결한다.



### 스케치 프로그램

- ✓ Bounce2.h 라이브러리 사용

```
// Turn on and off LED (toggling) if switch is depressed
// Switch mode is internal pullup and debouncing
// Library : Bounce2.h
```

```
#include <Bounce2.h>

const int btnPin = 12;
const int ledPin = 10;
// Bounce 객체 선언
Bounce Db = Bounce();
int ledState = LOW;

void setup() {
    pinMode(btnPin, INPUT_PULLUP);
    pinMode(ledPin, OUTPUT);
    digitalWrite(ledPin, ledState);
    // 아두이노 12 번 핀을 스위치에 연결한다.
    Db.attach(btnPin);
    // debouncing 시간을 20 ms로 설정한다.
    Db.interval(20);
}

void loop() {
    Db.update();
    // 스위치 상태가 LOW에서 HIGH로 바뀌면 LED 상태를 바꾼다.
    if (Db.rose()) {
        ledState = !ledState;
        digitalWrite(ledPin, ledState);
    }
}
```

- ✓ 회로도에서 스위치를 떼면 스위치 상태가 LOW에서 HIGH로 변하기 때문에 `Bounce.rose()` 함수를 사용한다.

## 연습 문제

### 연습 문제 3-2-1

- ✓ 풀-다운 저항을 사용하여 스위치가 눌릴 때마다 4개의 LED가 순서대로 점멸하는 회로를 구성하고 프로그램을 작성하라.

### 3.3 아날로그 입력

#### ▣ 아날로그-디지털 변환 (ADC)

- ✓ 기준 전압과 비교하여 아날로그 값을 10 bit(1024 단계)로 AD 변환한다.
- ✓ 기준 전압이 5 V 일 때 해상도 :  $5/1024=4.88 \text{ mV}$

#### ▣ 관련 함수

`analogReference(type);`

- ✗ 아날로그 입력에 필요한 기준 전압을 설정한다.

`analogRead(pin);`

- ✗ 아두이노 핀 (pin)에 가해진 아날로그 값을 읽는다.

`map(value, fromLow, fromHigh, toLow, toHigh)`

- ✗ 주어진 범위 안의 값을 또 다른 주어진 범위의 값으로 변환시킨다.

#### ▣ 기준 전압 설정

`analogReference(type)`

- ✗ DEFAULT : 5 V (5 V 보드), 3 V (3 V 보드)
- ✗ INTERNAL : 1.1 V (ATmega 168, ATmega328), 3.56 V (ATmega8)
- ✗ INTERNAL1V1 : 1.1 V (Mega only)
- ✗ INTERNAL2V56 : 2.56 V (Mega only)
- ✗ EXTERNAL : AREF 핀에 가해진 전압 (0~5 V)

#### ▣ 아날로그 입력 값 읽기

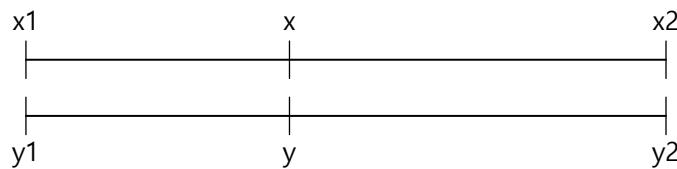
`analogRead(pin);`

- ✗ 0 ~ 1023 사이의 정수 값을 돌려 준다.
- ✗ 아날로그 값을 읽는데 대략 100  $\mu\text{s}$  걸린다. (최대 샘플링 속도는 대략 10 kHz)

#### ▣ 읽은 값 변경하기

`y = map(x, x1, x2, y1, y2)`

- ✓ x1 ~ x2 의 범위를 y1 ~ y2 의 범위로 대응시켜 x 값을 변환한다.



$$y = y_1 + \frac{y_2 - y_1}{x_2 - x_1} (x - x_1)$$

- ✓ x 가 x1 ~ x2 의 범위 내에 있을 필요 없다.
- ✓ 내부적으로 정수 연산을 하고 정수 값을 돌려 준다. (소수점 이하 버림)

```
float y = map(x, 0, 10000, 0, 10);
```

- x = 1234 ⇒ y = 1.00
- x = 12345 ⇒ y = 12.00

```
float y = map(x, 0, 10000, 0, 10000) / 1000.0;
```

- x = 1234 ⇒ y = 1.234
- x = 12345 ⇒ y = 12.345
- 자세한 값으로 변환하려면 큰 범위를 잡고 결과를 적당한 값으로 나누어 준다.

```
float y = map(x, 0, 10, 0, 100);
```

- x = 1.2 ⇒ y = 10.0
- x는 0~10 사이의 정수 1로 바뀌고 y는 1에 대응하는 10이 된다.

```
y = map(x, 0, 10, 10, 0);
```

- 0 ~ 10 사이의 정수를 10 ~ 0의 정수로 바꾼다.

- ✓ map() 함수는 내부적으로 정수 연산을 하기 때문에 변환에 주의해야 한다.

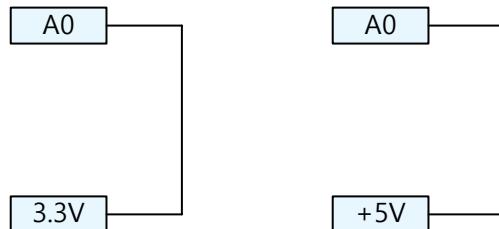
### 예제 3-3-1

#### ▣ 목표

- ✓ 아두이노에서 출력되는 3.3 V 와 5 V 전압을 측정한 후 시리얼 모니터에 나타낸다.

#### ▣ 회로도

- ✓ 회로 1 : 아날로그 입력 A0 핀에 3.3 V 전압을 인가한다.
- ✓ 회로 2 : 아날로그 입력 A0 핀에 5 V 전압을 인가한다.



#### ▣ 스케치 프로그램

```
// Measure given voltage

const int inPin = A0;
int delayTime = 500;

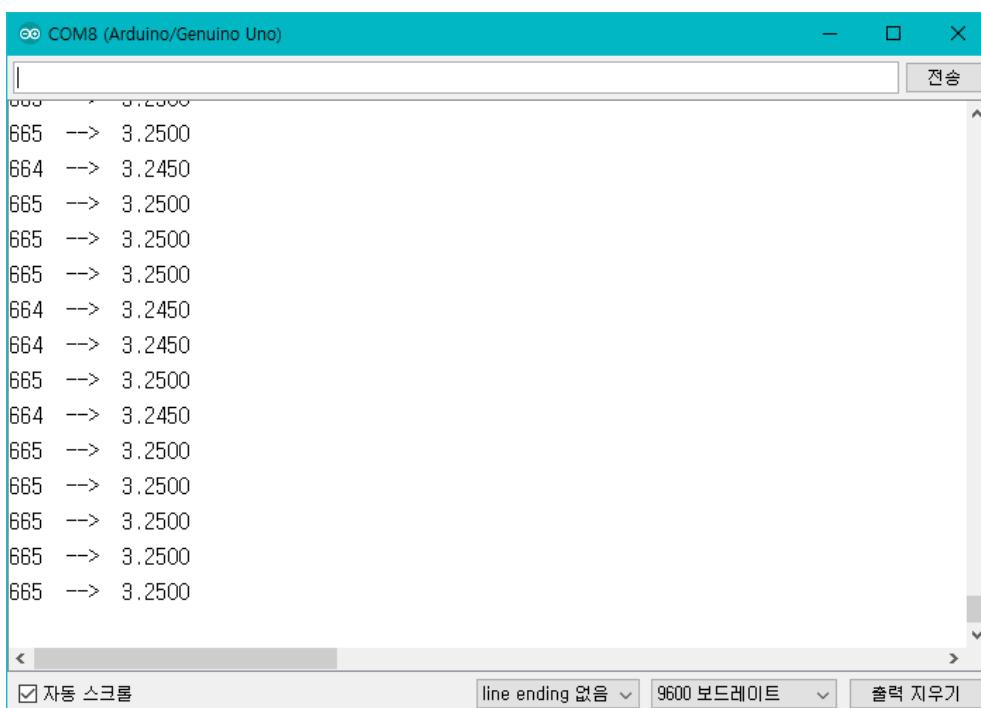
void setup() {
    // 시리얼 통신의 속도를 9600으로 설정한다.
    Serial.begin(9600);
}

void loop() {
    // 아날로그 A0 핀에서 값을 읽는다.
    int inValue = analogRead(inPin);
    // 읽은 값을 0~5 사이의 값으로 변환한다.
    float inVtg = map(inValue, 0, 1023, 0, 5000) / 1000.0;
    // 읽은 값과 변환한 값을 시리얼 모니터에 나타낸다.
    Serial.print(inValue);
    Serial.print(" --> ");
    Serial.println(inVtg, 4);
    delay(delayTime);
}
```

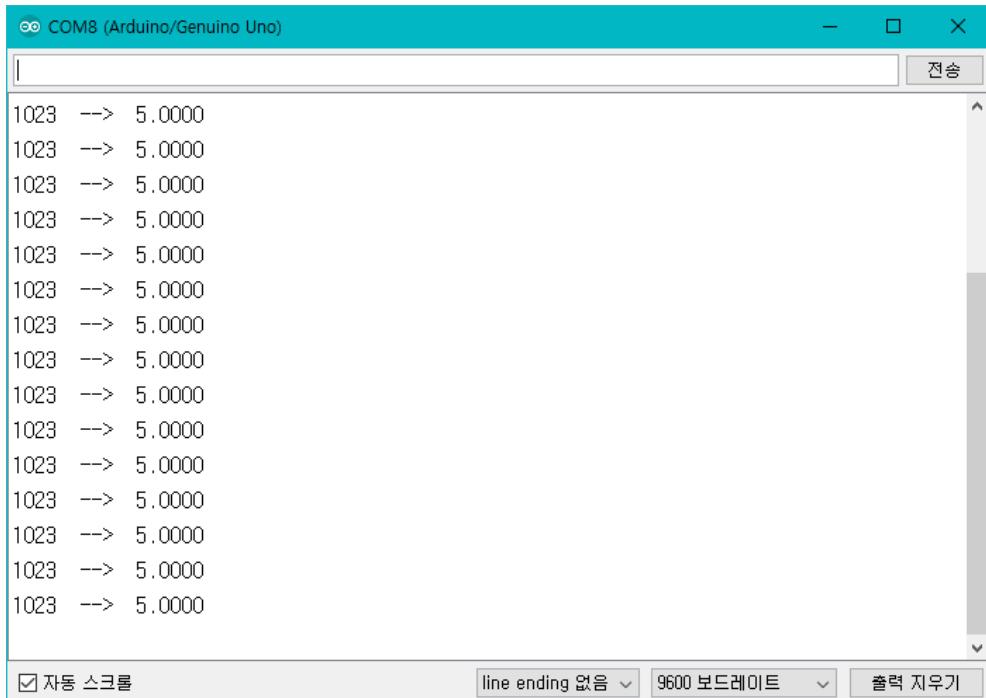
✓ 주의 사항

➊ 시리얼 모니터 출력

✓ 3.3 V 를 연결할 경우



✓ 5 V를 연결할 경우



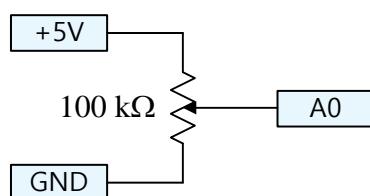
### 예제 3-3-2

#### ▣ 목표

- ✓ 가변 저항 값을 읽어 시리얼 모니터에 나타낸다.
- ✓ 가변 저항을 변화시켜 가면서 저항 값의 변화를 확인한다.

#### ▣ 회로도

- ✓ 가변 저항의 양단에 전압 5 V를 인가하고 가변 단자의 전압을 아날로그 입력 A0 핀에 인가한다.



#### ▣ 스케치 프로그램

```

// Measure the resistance of variable resistor

const int inPin = A0;
int delayTime = 100;

void setup() {
    // 시리얼 통신의 속도를 9600으로 설정한다.

```

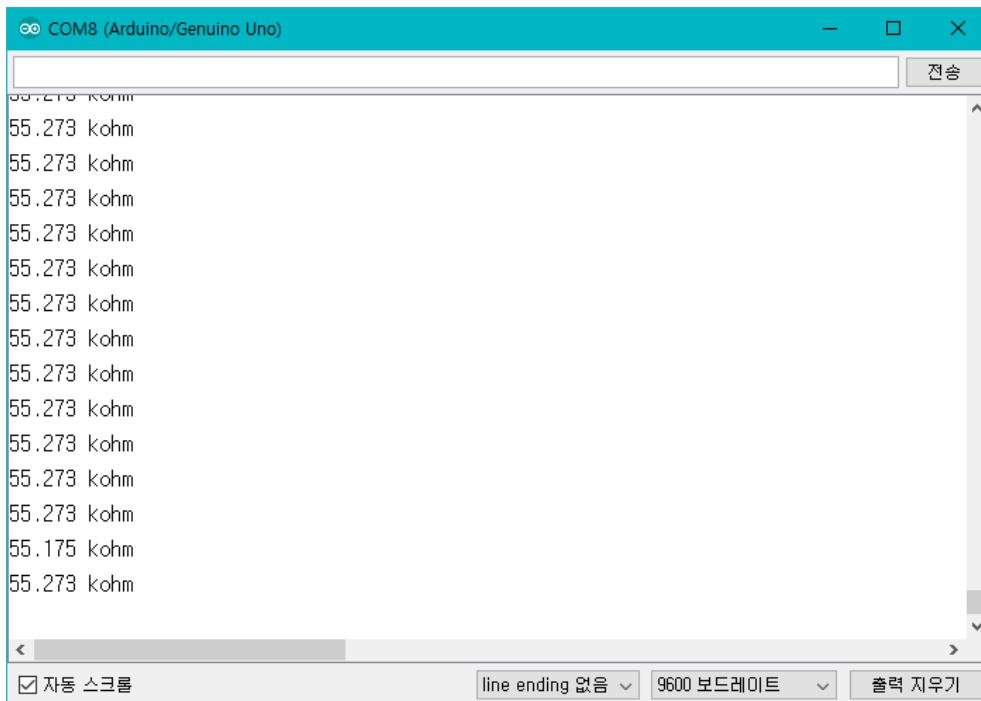
```

Serial.begin(9600);
}

void loop() {
    // 아날로그 A0 핀에서 값을 읽는다.
    int inValue = analogRead(inPin);
    // 읽은 값을 0~100 사이의 값으로 변환한다.
    float rVal = map(inValue, 0, 1023, 0, 100000) / 1000.0;
    // 변환한 값을 소수 3자리로 단위와 함께 시리얼 모니터에 나타낸다.
    Serial.print(rVal, 3);
    Serial.println(" kohm");
    delay(delayTime);
}

```

### ✚ 시리얼 모니터



- ✚ `analogRead` 함수 실행에 100  $\mu\text{s}$  정도의 시간이 소요되지만 시리얼 통신을 같이 사용하면 샘플링 주기가 증가한다. (소요되는 시간이 증가한다.)

- ✓ 시리얼 통신 속도가 9600 일 때 최소 샘플링 주기는 대략 6236  $\mu\text{s}$  이다.
- ✓ 시리얼 통신 속도가 115200 일 때 최소 샘플링 주기는 대략 420  $\mu\text{s}$  이다.
- ✓ 시리얼 통신 속도가 2000000 일 때 최소 샘플링 주기는 대략 270  $\mu\text{s}$  이다.

### ✚ 일정한 주기로 샘플링 하는 방법 (데이터를 읽는 방법)

- ✓ 문법 1

```
currTime = micros();
```

```

if ((currTime - lastTime) >= sampTime) {
    lastTime = currTime;
    x = analogRead(inPin);
}

✖ micros() : 현재의 시간을 얻는 함수, 단위 μs
✖ currTime : 현재의 시간, 단위 μs
✖ lastTime : 전에 데이터를 읽었을 때의 시간, 단위 μs
✖ sampTime : 샘플링 주기, 단위 μs
✖ 전에 데이터를 읽었을 때와 현재의 시간 차가 sampTime보다 작으면
    아무 일도 하지 않고 클 때만 데이터를 읽는다.

```

✓ 문법 2

```

do {
    currTime = micros();
} while ((currTime - lastTime) < sampTime);
lastTime = currTime;
x = analogRead(inPin);

```

✖ 문법 2는 샘플링하는 동안 아무 작업도 하지 않지만 문법 1은 if 문 안에서만 샘플링하고 if 문 밖에서 다른 작업을 하기 때문에 적절히 선택해야 한다.

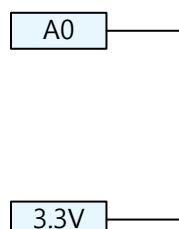
### 예제 3-3-3

목표

- ✓ 아두이노에서 출력되는 3.3 V 전압을 샘플링 주파수 1 kHz로 샘플링하여 측정한 후 시리얼 모니터에 나타낸다.

회로도

- ✓ 아날로그 입력 A0 핀에 3.3 V 전압을 인가한다.



스케치 프로그램

```

// Sampling frequency of analogRead function

const int inPin = A0;
const int sampTime = 1000;
long lastSampTime;
int sTime;

void setup() {
    // 시리얼 통신의 속도를 2000000으로 설정한다.
    Serial.begin(2000000);
    // 현재의 시간을 마지막 샘플링 시간에 저장한다.
    lastSampTime = micros();
}

void loop() {
    // 마지막 샘플링 시간과 현재의 시간 차가 sampTime보다 작으면 아무 일도
    // 하지 않고 클 때만 데이터를 읽고 읽은 값을 시리얼 모니터에 나타낸다.
    if ((micros() - lastSampTime) >= sampTime) {
        lastSampTime = micros();
        int x = analogRead(inPin);
        Serial.println(x);
    }
}

```

- ✓ 샘플링 주파수를 크게 하려면 가능한 시리얼 통신 속도를 크게 한다.
- ✓ 샘플링 주파수가 1 kHz 이므로 샘플링 주기는  $1/1000=1000 \mu\text{s}$  이다.
- ✓ 샘플링을 한 후  $1000 \mu\text{s}$  가 지나기 전에는 아무런 일도 하지 않고  $1000 \mu\text{s}$  가 지난 후에 다시 샘플링을 한다.

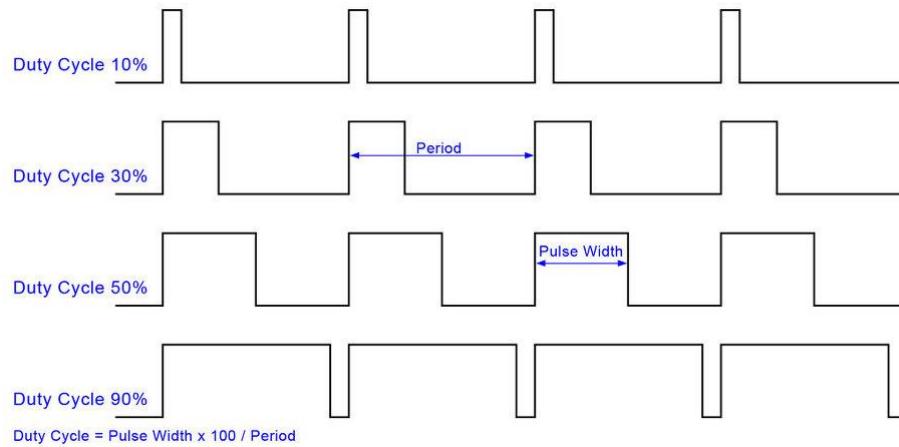
## 연습 문제

- ➊ 연습 문제 3-3-1
  - ✓ 가변 저항 값에 비례하여 4 개의 LED 를 켜는 회로를 구성하고 프로그램을 작성하라.
- ➋ 연습 문제 3-3-2
  - ✓ 함수 발생기로 피크 값이 5 V 이고 주파수가 1 Hz 인 삼각파를 발생시키고 이를 아두이노로 받아 4 개의 LED 를 크기에 따라 점멸하는 회로를 구성하고 프로그램을 작성하라.

## 3.4 아날로그 출력

### ■ PWM (pulse width modulation)

- ✓ Duty cycle



### ■ 관련 함수

`analogWrite(pin, value);`

- ✗ 아두이노 핀(3, 5, 6, 9, 10, 11 번 핀)에 PWM 파형을 출력한다.
- ✗ value 값으로 duty cycle을 제어한다.
- ✗ value : 0~255 사이의 정수
  - value = 0 : 항상 off, 평균 전압 0 V
  - value = 255 : 항상 on, 평균 전압 5 V
- ✗ PWM 신호의 주파수
  - 대부분의 핀 : 490 Hz
  - 아두이노 우노의 5 번 핀, 6 번 핀 : 980 Hz
- ✗ `analogWrite()` 함수를 부르기 전에 `pinMode()` 함수를 부를 필요 없다.

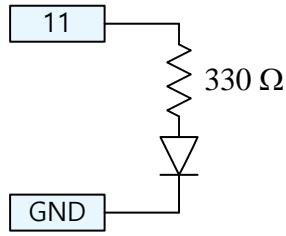
### 예제 3-4-1

### ■ 목표

- ✓ LED 의 밝기를 대략 0.5 Hz 의 속도로 미세하게 밝아졌다가 어두워지게 한다.

### ■ 회로도

- ✓ 디지털 11 번 핀에 저항과 LED 를 직렬 연결한다.



### 스케치 프로그램

```
// Control bright of LED

const int ledPin = 11;
int delayTime;

void setup() {
    // LED 밝기 상태가 유지되는 시간 설정
    delayTime = 2 * 1024 / 256;
}

void loop() {
    // LED의 밝기를 256 레벨로 설정하고 밝기를 증가시킨다.
    for (int n = 0; n < 256; n++) {
        analogWrite(ledPin, n);
        delay(delayTime);
    }
    // LED의 밝기를 256 레벨로 설정하고 밝기를 감소시킨다.
    for (int n = 255; n >= 0; n--) {
        analogWrite(ledPin, n);
        delay(delayTime);
    }
}
```

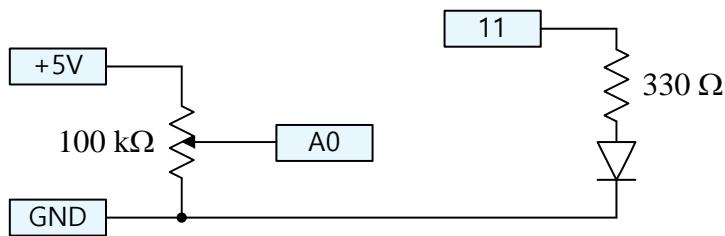
### 예제 3-4-2

#### 목표

- ✓ 가변 저항을 이용하여 LED의 밝기를 조정한다.

#### 회로도

- ✓ 가변 저항의 가변 단자의 전압을 아날로그 입력 A0 핀에 인가하고 디지털 11 번 핀에 저항과 LED를 직렬 연결한다.



### 悒 스케치 프로그램

```
// Control bright of LED using the variable resistor

const int inPin = A0;
const int outPin = 11;

void setup() {
}

void loop() {
    // 아날로그 A0 핀으로 저항 값에 따른 전압을 읽는다.
    int inVal = analogRead(inPin);
    // 읽은 전압을 0~255 사이의 값으로 변환한다.
    int outVal = map(inVal, 0, 1023, 0, 255);
    // 11 번 핀에 PWM 파형을 내 보낸다.
    analogWrite(outPin, outVal);
}
```

### 연습 문제

#### 悒 연습 문제 3-4-1

- ✓ 함수 발생기로 피크 값이 5 V이고 주파수가 1 Hz 인 삼각파를 발생시키고 이를 아두이노로 받아 1 개의 LED 의 밝기를 변화시키는 회로를 구성하고 프로그램을 작성하라.

## 3.5 아두이노 인터럽트

### 3.5.1 아두이노 인터럽트

#### ▣ 인터럽트란?

- ✓ 프로그램 실행 중 어떤 조건이 만족되면 자동으로 특정 프로그램이 실행되도록 하는 프로세스

#### ▣ 아두이노 인터럽트

- ✓ loop() 안의 프로그램이 실행되는 도중 인터럽트가 발생하면 인터럽트로 지정된 함수가 실행된다.
- ✓ 아두이노에서 인터럽트는 특정 핀의 상태가 변할 때 인터럽트가 실행된다.
- ✓ 아두이노 제품에 따라 인터럽트 핀이 다르다.
  - ✗ 아두이노 우노 – 디지털 2 번, 3 번 핀 (2 개)
  - ✗ 아두이노 메가 – 디지털 2 번, 3 번, 18 번, 19 번, 20번, 21 번 (6 개)
  - ✗ 아두이노 마이크로 – 디지털 0 번, 1 번, 2 번, 3 번, 7 번 (5 개)
- ✓ <https://www.arduino.cc/reference/en/language/functions/interrupts/interrupts/>

/

#### ▣ 관련 함수

**interrupts();**

- ✗ 인터럽트를 가능하게 한다.

**noInterrupts();**

- ✗ 인터럽트를 불가능하게 한다.

**attachInterrupt(digitalPinToInterrupt(pin), ISR, mode);**

- ✗ 주어진 핀에 의한 인터럽트를 가능하게 한다.

✗ pin : 인터럽트 핀 번호

✗ ISR : 인터럽트 서비스 루틴 (interrupt service routine), 인터럽트가 발생하면 실행되는 함수

✗ mode : 인터럽트가 발생 (trigger) 되는 상태를 정의한 모드

- LOW : 인터럽트 핀의 상태가 LOW일 때 발생한다. (LOW일 동안 반복적으로 발생한다.)

- CHANGE : 인터럽트 핀의 상태가 바뀔 때 발생한다.

- RISING : 인터럽트 핀의 상태가 LOW에서 HIGH로 바뀔 때 발생한다.
- FALLING : 인터럽트 핀의 상태가 HIGH에서 LOW로 바뀔 때 발생한다.

`detachInterrupts(digitalPinToInterrupt(pin));`

- ✖ 주어진 핀에 의한 인터럽트를 불가능하게 한다.

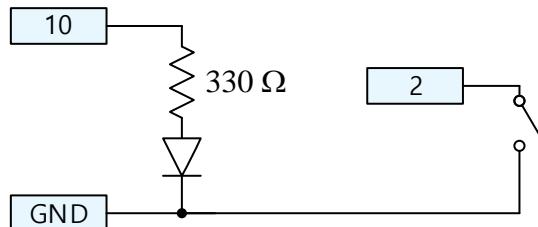
### 예제 3-5-1

#### ➊ 목표

- ✓ 내부 풀-업 저항을 사용하여 스위치가 눌릴 때 LED 가 켜지도록 한다.
- ✓ 스위치 상태에 따라 인터럽트를 사용한다.

#### ➋ 회로도

- ✓ 디지털 10 번 핀에 저항과 LED 를 직렬 연결하고, 디지털 2 번 핀에 스위치를 연결한다.



#### ➌ 스케치 프로그램

```

// Turn on LED if switch is pressed
// Switch mode is internal pullup
// Use interrupt

const byte ledPin = 10;
// 인터럽트 핀
const byte interPin = 2;
volatile byte ledState = LOW;

void setup() {
    pinMode(ledPin, OUTPUT);
    // 인터럽트 핀 (2번 핀)을 내부 풀-업 저항을 이용한 입력 핀으로 설정한다.
    pinMode(interPin, INPUT_PULLUP);
    // 2 번 핀에서 인터럽트를 가능하게 한다.
    // 2 번 핀의 상태가 변하면 인터럽트가 발생하고
    // 인터럽트가 발생하면 blinkLed() 함수가 실행된다.
}

```

```

        attachInterrupt(digitalPinToInterrupt(interPin), blinkLed, CHANGE);
    }

void loop() {
    // 스위치 상태를 이용해 LED를 On/Off 한다.
    digitalWrite(ledPin, ledState);
}

// 인터럽트 서비스 루틴, LED의 상태를 바꾼다.
void blinkLed() {
    ledState = !ledState;
}

```

 주의 사항

- ✓ ledState 변수는 인터럽트 서비스 루틴 안에서 계속적으로 변하는 전역 변수이므로 volatile 변수로 선언해야 한다.

 동작 원리

- ✓ 처음 LED 상태가 LOW (off)이고 스위치 상태도 LOW 이므로 스위치를 누르면 스위치 상태가 LOW에서 HIGH로 바뀌고 인터럽트가 발생하여 LED 상태도 LOW에서 HIGH로 바뀐다.
- ✓ 스위치를 떼면 스위치 상태가 HIGH에서 LOW로 바뀌고 인터럽트가 발생하여 LED 상태도 HIGH에서 LOW로 바뀐다.

### 3.5.2 타이머 (timer) 인터럽트

 일정 시간마다 특정한 작업을 하는 방법

- ✓ delay() 함수를 사용하는 방법
  - ✗ 기다리는 동안 다른 작업을 할 수 없다.
- ✓ 타이머 인터럽트를 사용하는 방법
  - ✗ 기다리는 동안 다른 작업을 할 수 있다.

 타이머 인터럽트 라이브러리

**MsTimer2.h**

- ✓ 설정한 시간마다 인터럽트를 수행한다.

 관련 함수

**MsTimer2::set(unsigned long ms, void(\*f)());**

- ✖ 타이머 인터럽트를 사용할 때 필요한 타이머와 인터럽트 서비스 루틴 (ISR)을 설정한다.

- ✖ ms : 타이머 시간, 인터럽트가 발생하는 시간 간격, 단위 ms

**MsTimer2::start();**

- ✖ 타이머 인터럽트를 시작한다.

**MsTimer2::stop();**

- ✖ 타이머 인터럽트를 멈춘다.

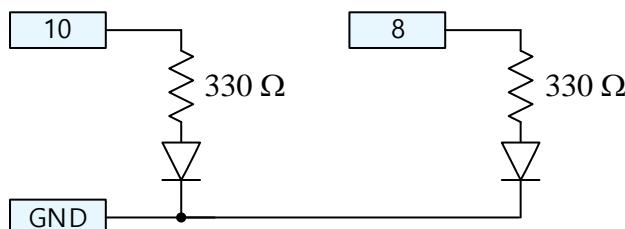
### 예제 3-5-2

#### ▣ 목표

- ✓ LED 1 과 LDE 2 를 각각 300 ms 와 1000 ms 간격으로 점멸시킨다.
- ✓ LED 1 은 delay() 함수를 이용하고 LED 2 는 타이머 인터럽트를 사용한다.

#### ▣ 회로도

- ✓ LED 는 저항(수 100Ω)과 같이 연결해야 한다. 저항값에 따라 LED 의 밝기가 결정된다.
- ✓ 디지털 11 번 핀과 접지 사이, 8 번 핀과 접지 사이에 저항과 LED 를 직렬로 연결한다.



#### ▣ 스케치 프로그램

```
// Blink LED at 1 Hz using timer interrupt
// Library : MsTimer2.h
```

```
#include <MsTimer2.h>

// LED 연결 핀
const int led1Pin = 8;
const int led2Pin = 10;
// LED 점멸 시간
const int delayTime1 = 300;
const int delayTime2 = 1000;
```

```
int led1State = 0;
int led2State = 0;

void setup() {
    pinMode(led1Pin, OUTPUT);
    pinMode(led2Pin, OUTPUT);
    // 타이머 인터럽트를 설정한다.
    MsTimer2::set(delayTime1, blinkLed1);
    // 타이머 인터럽트를 시작한다.
    MsTimer2::start();
}

void loop() {
    // LED 2를 1000 ms 간격으로 점멸시킨다.
    led2State = !led2State;
    digitalWrite(led2Pin, led2State);
    delay(delayTime2);
}

// LED 1을 점멸시키는 인터럽트 서비스 루틴
void blinkLed1() {
    led1State = !led1State;
    digitalWrite(led1Pin, led1State);
}
```

## 4. 시리얼 통신

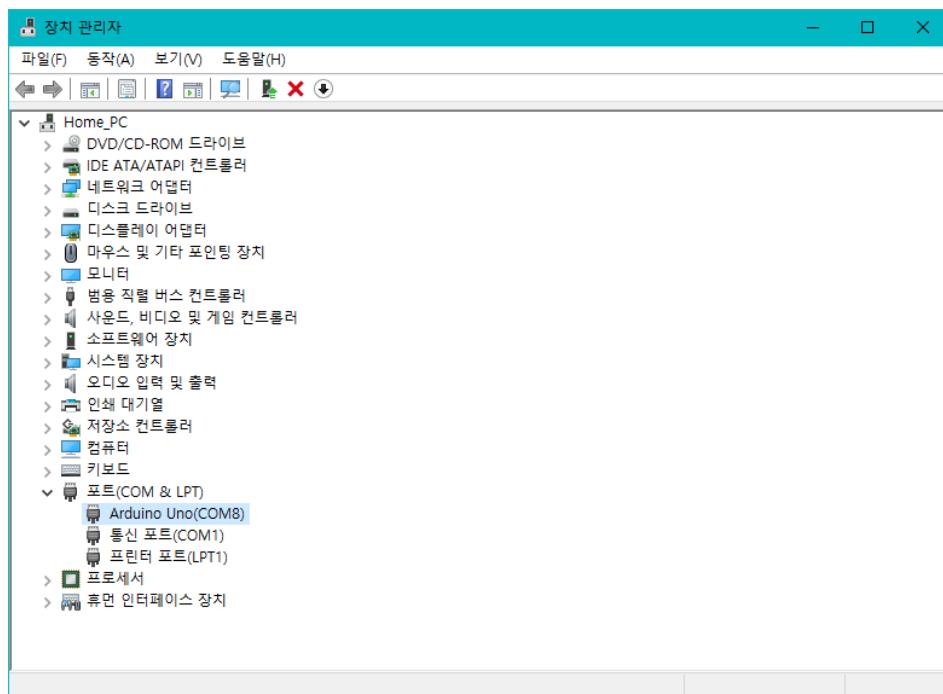
### 4.1 시리얼 통신

#### ✚ 시리얼 (serial) 통신이란?

- ✓ 아두이노와 컴퓨터, 아두이노와 다른 시리얼 장치간의 통신을 위해 사용하는 일반적인 통신 수단
- ✓ 아두이노 프로그램의 디버깅에도 사용된다.
- ✓ TX/RX 핀 (디지털 1/0 번 핀)을 통해 통신한다.

#### ✚ 시리얼 통신 포트 설정

- ✓ 장치 관리자에서 확인할 수 있다.

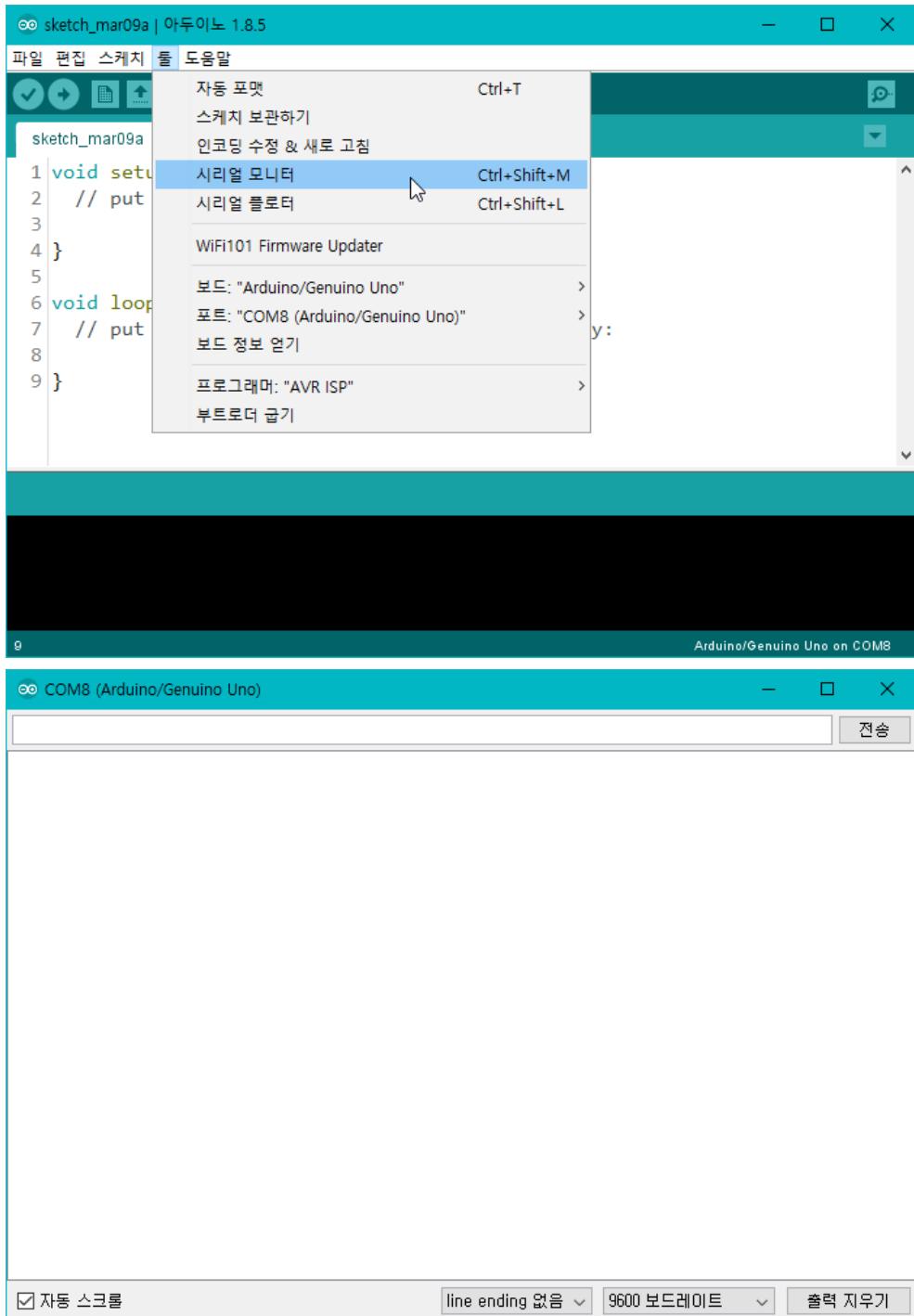


#### ✚ 시리얼 통신 속도 설정

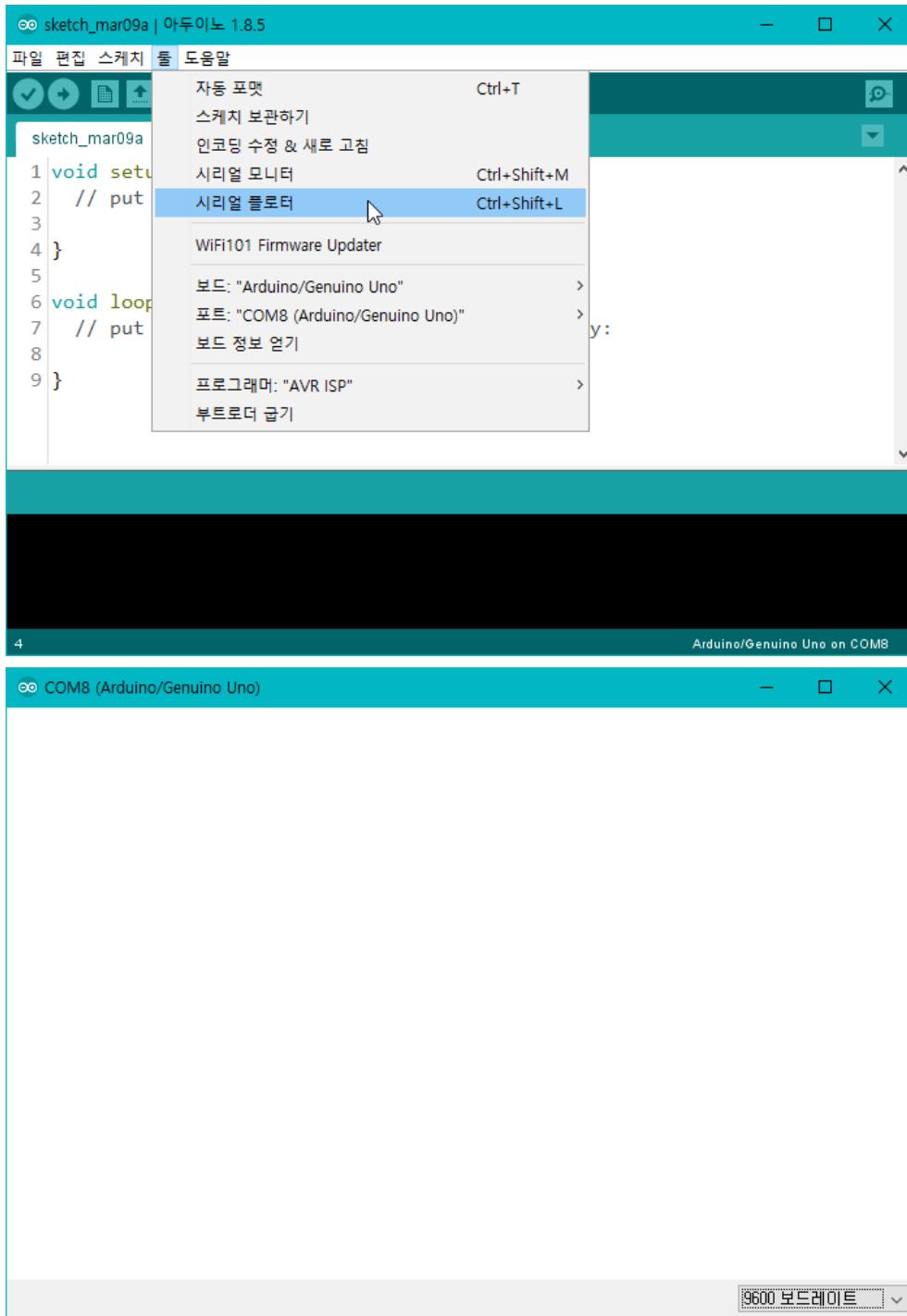
- ✓ 시리얼 모니터에서 가능한 통신 속도 – 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 74880, 115200, 230400, 250000, 500000, 1000000, 2000000
- ✓ `serial.begin()` 함수를 사용하여 결정하며 일반적으로 9600 을 사용한다.

#### ✚ 아두이노 시리얼 모니터

- ✓ 아두이노에 시리얼 통신을 통해 데이터를 보내거나 데이터를 받아 표시하는 창
- ✓ 메뉴 툴 – 시리얼 모니터를 선택한다. (단축키 : Ctrl+Shift+M)



- ✓ 프로그램과 시리얼 모니터의 통신 속도를 일치시켜야 한다.
- ✚ 아두이노 시리얼 플로터
  - ✓ 아두이노에서 시리얼 통신을 통해 보낸 데이터를 그리는 창
  - ✓ 메뉴 툴 – 시리얼 플로터를 선택한다. (단축키 : Ctrl+Shift+L)



- ✓ 시리얼 모니터와 마찬가지로 프로그램과 시리얼 플로터의 통신 속도를 일치시켜야 한다.

#### SerialPlot

- ✓ 아두이노에 시리얼 통신을 통해 데이터를 보내거나 데이터를 받아 그려주는 프로그램
- ✓ 시리얼 포트 선택 가능
- ✓ 수신 데이터 설정 가능

- ✓ 파형 snapshot 가능
  - ✗ 이진 수 – uint8, uint16, uint32, int8, int16, int32, float
  - ✗ ASCII 수 – 숫자 분리 (쉼마, 빈칸, 탭, 기타 문자) 선택 가능
  - ✗ 사용자 정의 수 – 시작 프레임 설정 가능, uint8, uint16, uint32, int8, int16, int32, float



## 4.2 아두이노에서 데이터 송신

- ✚ 문자열 및 텍스트 데이터 송신

- ✚ 관련 함수

`Serial.begin(speed)`

- ✖ 시리얼 통신 속도 speed로 시리얼 통신을 시작한다.

`Serial.print(val);`

- ✖ val을 시리얼 포트에 쓴다. (송신)

`Serial.print(data, format);`

- ✖ val을 format 형식으로 시리얼 포트에 쓴다.

`Serial.println(val);`

- ✖ val과 carriage return 문자(\r) 또는 newline 문자(\n)을 시리얼 포트에 쓴다.

- ✓ <https://www.arduino.cc/reference/en/language/functions/communication/serial/>

### 예제 4-2-1

- ✚ 목표

- ✓ 시리얼 통신을 이용하여 컴퓨터에 문자열 및 데이터 송신한다.
- ✓ 아두이노 시리얼 모니터로 송신된 데이터 확인한다.

- ✚ 회로도

- ✓ 없음

- ✚ 검토 사항

- ✓ 실수를 보낼 경우 소수점의 위치는 어떻게 되는가?

- ✚ 스케치 프로그램

---

```
// Send data to PC by serial communication
```

```
int x = 1;
float y = 2;

void setup() {
    // 시리얼 통신의 속도를 9600으로 설정한다.
    Serial.begin(9600);
}
```

```

void loop() {
    // 문장과 x, y 값을 시리얼 모니터에 나타낸다.
    Serial.println("Hello Arduino.");
    Serial.print("x = ");
    Serial.print(x);
    Serial.print(", y = ");
    Serial.println(y);
}

```

### ➊ 시리얼 모니터



### 예제 4-2-2

#### ➊ 목표

- ✓ 사인파 데이터를 생성하고 시리얼 통신을 이용하여 컴퓨터에 데이터 송신한다.
- ✓ 아두이노 시리얼 모니터로 송신된 데이터를 확인한다.
- ✓ 아두이노 시리얼 플로터와 SerialPlot 프로그램을 이용하여 송신된 데이터를 그린다.

#### ➋ 회로도

- ✓ 없음

#### ➌ 스케치 프로그램

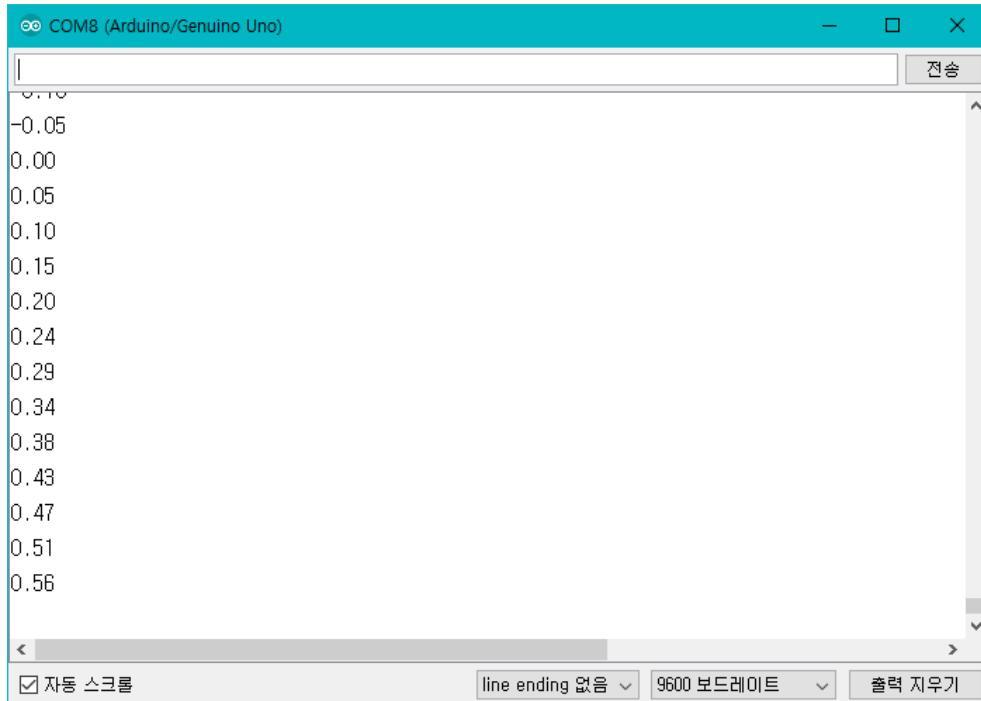
```
// Send sine wave data to PC by serial communication

const int nx = 256;
float w0, x;

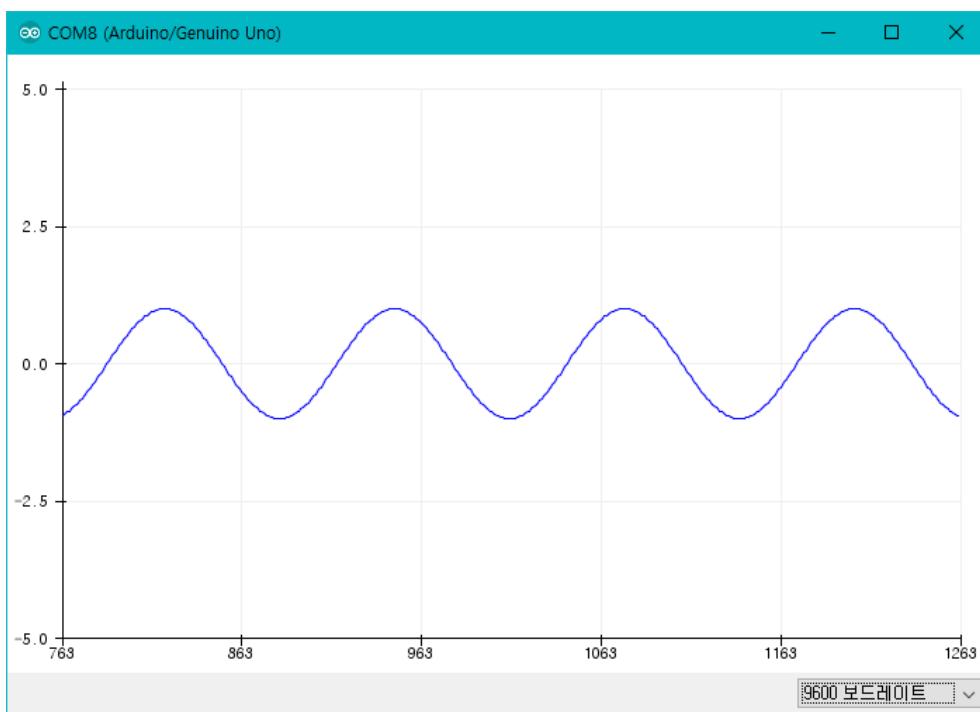
void setup() {
    // 시리얼 통신의 속도를 9600으로 설정한다.
    Serial.begin(9600);
    w0 = 4.0 * PI / nx;
}

void loop() {
    // 사인파를 만들고 시리얼 모니터에 나타낸다.
    for (int n = 0; n < nx; n++) {
        x = sin(w0 * n);
        Serial.println(x);
    }
}
```

#### ➊ 시리얼 모니터

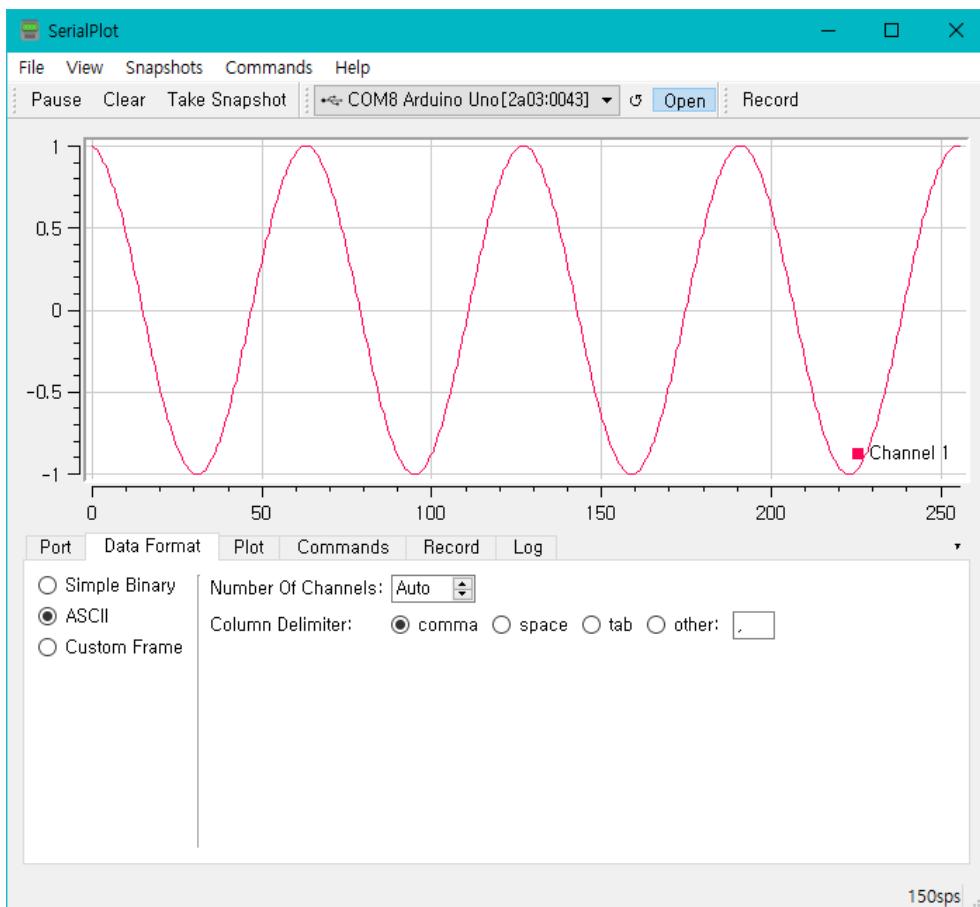


#### ➋ 시리얼 플로터



### SerialPlot

- ✓ Data Format 으로 ASCII 를 선택한다.



### 예제 4-2-3

## 목표

- ✓ 피크 값이 1 인 사인파와 피크 값이 0.1 인 잡음을 생성하고 사인파, 잡음, 잡음이 섞인 사인파를 송신한다.
- ✓ 아두이노 시리얼 모니터로 송신된 데이터를 확인한다.
- ✓ 아두이노 시리얼 플로터와 SerialPlot 프로그램을 이용하여 송신된 데이터를 데이터 별로 그린다.

## 회로도

- ✓ 없음

## 스케치 프로그램

```
// Send several data to PC by serial communication

const int nx = 256;
float w0, xs, xn, x;

void setup() {
    // 시리얼 통신의 속도를 9600으로 설정한다.
    Serial.begin(9600);
    w0 = 4.0 * PI / nx;
}

void loop() {
    // 사인파와 잡음을 만들고 시리얼 모니터에 나타낸다.
    for (int n = 0; n < nx; n++) {
        // 사인파
        xs = 100 * sin(w0 * n);
        // 잡음
        xn = random(-1000,1000) / 100.0;
        // 잡음이 합쳐진 사인파
        x = xs + xn;
        // 3 개의 데이터를 콤마로 분리하고 시리얼 모니터에 나타낸다.
        Serial.print(",");
        Serial.print(xn);
        Serial.print(",");
        Serial.println(x);
    }
}
```

## 시리얼 모니터

```

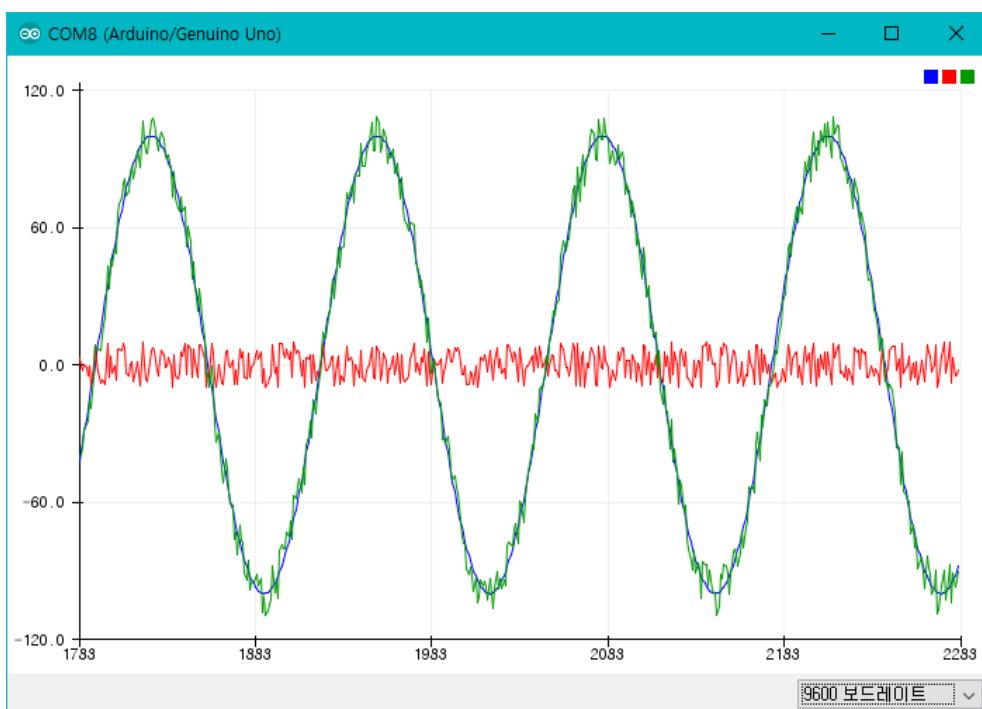
COM8 (Arduino/Genuino Uno)

9.80, -5.20, 4.60
14.67, -7.97, 6.70
19.51, 9.00, 28.51
24.30, -4.80, 19.50
29.03, -4.67, 24.36
33.69, 2.58, 36.27
38.27, 0.03, 38.30
42.76, -3.24, 39.52
47.14, -0.50, 46.64
51.41, 9.34, 60.75
55.56, -2.20, 53.36
59.57, 8.79, 68.36
63.44, 8.92, 71.76
67.16, -4.26, 62.90
70.71, 5.49, 76.20
74.10, -4.5

```

자동 스크롤 line ending 없음 ▾ 9600 보드레이트 ▾ 출력 지우기

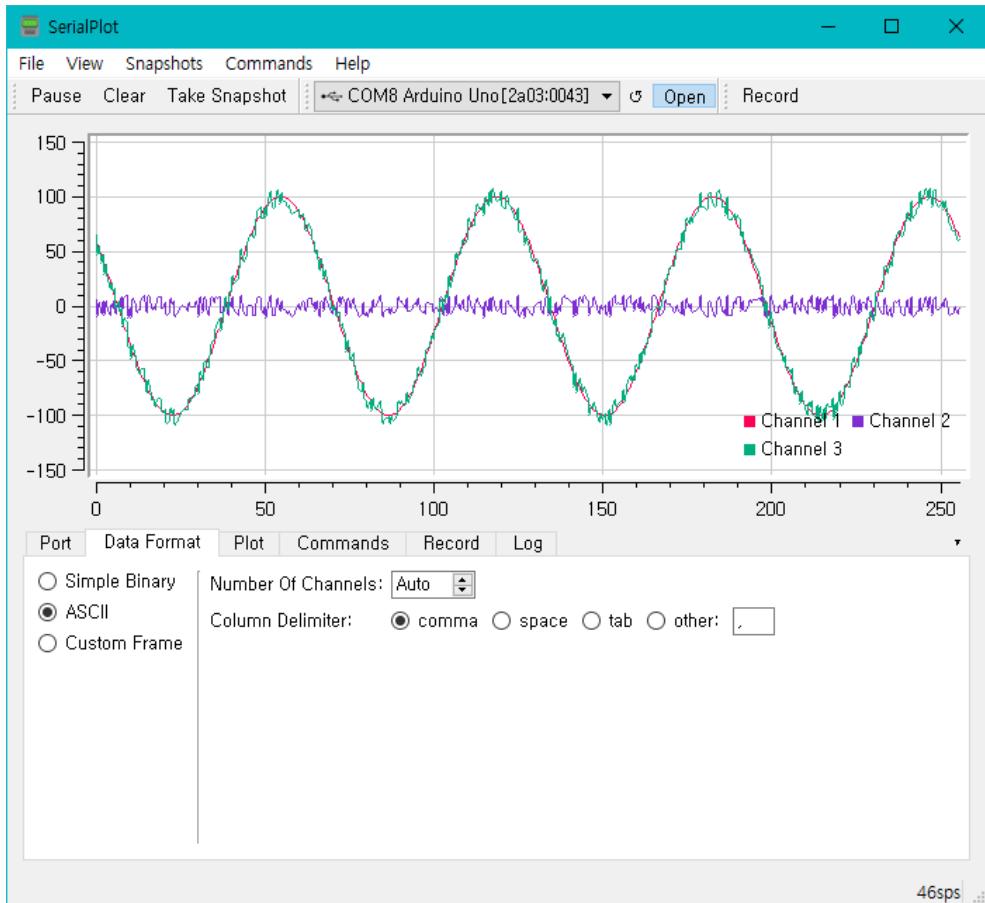
### ➊ 시리얼 플로터



- ✓ 한 줄에 여러 개의 데이터를 컴마로 분리하면 각각 별도로 그래프를 그릴 수 있다.

### ➋ SerialPlot

- ✓ Data Format 으로 ASCII 를 선택한다.



- ✓ 시리얼 플로터와 마찬가지로 한 줄에 여러 개의 데이터를 컴마, 스페이스, 탭 등으로 분리하면 각각 별도로 그래프를 그릴 수 있다.

### 연습 문제

#### 연습 문제 4-2-1

- ✓ 사인파와 잡음을 생성하고 사인파, 잡음, 잡음이 섞인 사인파를 송신하고 송신된 데이터를 아두이노 시리얼 플로터와 SerialPlot 을 이용하여 그려라. 사인파의 피크값과 SNR(신호 대 잡음 비)은 시리얼 모니터에서 입력한 값으로 하라.

## 4.3 아두이노에서 데이터 수신

- 시리얼 모니터에서 아두이노에 데이터를 전송하면 아두이노에서 수신할 수 있다.

- 관련 함수

`Serial.available();`

- ✖ 시리얼 포트의 버퍼에 있는 바이트 (byte) 수를 돌려 준다.
- ✖ 값이 0 이면 (false) 시리얼 포트의 버퍼에 데이터가 없다.
- ✖ 값이 0이 아니면 (true) 시리얼 포트 버퍼에 읽을 데이터가 있다.

`Serial.read();`

- ✖ 시리얼 포트의 버퍼에 있는 데이터의 첫 번째 바이트를 읽는다.

`Serial.readString();`

- ✖ Stream 클래스의 함수
- ✖ 시리얼 포트의 버퍼에 있는 문자 열을 읽는다.

`Serial.toInt();`

- ✖ 읽은 문자열을 정수로 변환한다.

`Serial.toFloat();`

- ✖ 읽은 문자열을 실수로 변환한다.

### 예제 4-3-1

- 목표

- ✓ 시리얼 통신을 통해 문자를 받고 받은 문자를 시리얼 통신을 통해 송신한다.

- 회로도

- ✓ 없음

- 검토 사항

- ✓ 문자 1 개를 입력하거나 문자 여러 개를 입력한 후 엔터를 치고 결과를 관찰하라.

- 스케치 프로그램

---

```
// Receive data from PC by serial communication
```

```
void setup() {
```

```

// 시리얼 통신의 속도를 9600으로 설정한다.
Serial.begin(9600);

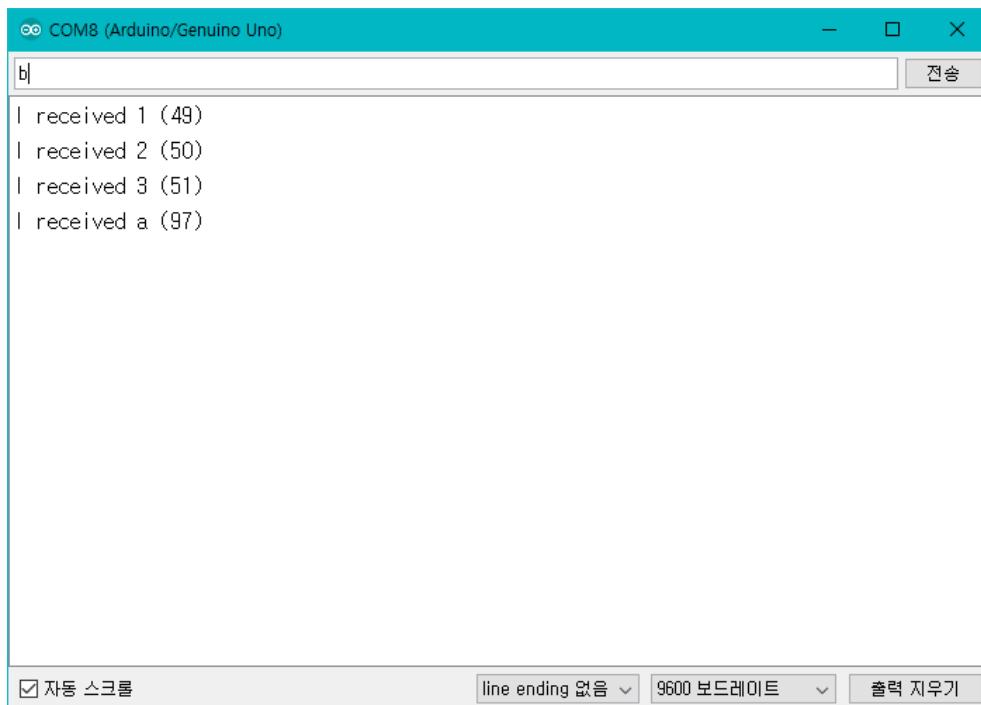
}

void loop() {
    // 시리얼 버퍼에 데이터가 있으면 데이터를 읽는다.
    if (Serial.available()) {
        char myChar = Serial.read();
        Serial.print("I received ");
        // 읽은 문자를 시리얼 모니터에 나타낸다.
        Serial.print(myChar);
        Serial.print(" (");

        // 읽은 문자의 ASCII 코드 값을 나타낸다.
        Serial.print(myChar, DEC);
        Serial.println(")");
    }
}

```

## ➊ 시리얼 모니터



## 예제 4-3-2

### ➊ 목표

- ✓ 시리얼 통신을 통해 문자열을 받고 받은 문자를 시리얼 통신을 통해 송신한다.

### ➋ 회로도

✓ 없음

### ▣ 스케치 프로그램

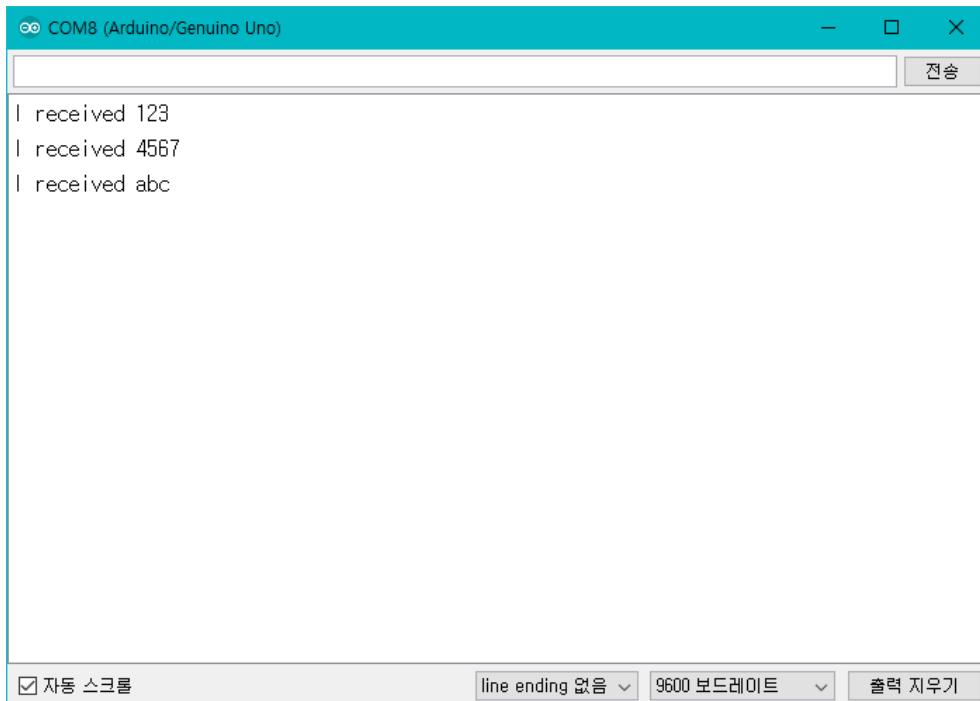
```
// Receive data from PC by serial communication

String myString;

void setup() {
    // 시리얼 통신의 속도를 9600으로 설정한다.
    Serial.begin(9600);
}

void loop() {
    // 시리얼 버퍼에 데이터가 있으면 데이터를 읽는다.
    if (Serial.available()) {
        // 문자열을 읽고 시리얼 모니터에 나타낸다.
        myString = Serial.readString();
        Serial.print("I received ");
        Serial.println(myString);
    }
}
```

### ▣ 시리얼 모니터



### 예제 4-3-3

### ▣ 목표

- ✓ 시리얼 통신을 통해 문자열을 받고 받은 문자를 시리얼 통신을 통해 송신한다.
- ✓ 받은 문자열을 숫자 (정수, 실수)로 변환한다.

 회로도

- ✓ 없음

 스케치 프로그램

```
// Receive data from PC by serial communication

String myString;

void setup() {
    // 시리얼 통신의 속도를 9600으로 설정한다.
    Serial.begin(9600);
}

void loop() {
    // 시리얼 버퍼에 데이터가 있으면 데이터를 읽는다.
    if (Serial.available()) {
        // 문자열을 읽고 시리얼 모니터에 나타낸다.
        myString = Serial.readString();
        Serial.print("I received ");
        Serial.println(myString);
        // 문자열을 정수로 변환하고 시리얼 모니터에 나타낸다.
        int myInt = myString.toInt();
        Serial.println(myInt);
        // 문자열을 실수로 변환하고 시리얼 모니터에 나타낸다.
        float myFloat = myString.toFloat();
        Serial.println(myFloat);
    }
}
```

 시리얼 모니터



#### 참고 사항

- ✓ 실수 문자열을 String.toInt()로 변환하면 정수가 된다.

#### 연습 문제

##### 연습 문제 4-3-1

- ✓ 아두이노 시리얼 모니터에서 0 을 입력하면 LED 가 꺼지고, 0 이외의 숫자를 입력하면 LED 가 켜지는 회로를 구성하고 프로그램을 작성하라.

##### 연습 문제 4-3-2

- ✓ 아두이노 시리얼 모니터에서 msec 단위로 숫자(양수)를 입력 받아 받은 시간만큼 LED 를 점멸하는 회로를 구성하고 프로그램을 작성하라. 단 초기값은 1000 msec 이다.

## 4.4 이진 (binary) 데이터 송신과 수신

### ✚ 이진 데이터 송신과 수신

- ✓ CR (carriage return) 등이 없어 데이터 끝을 알 수 없다.
- ✓ 데이터 유형을 확실히 알아야 한다. – 송신과 수신 측에서 데이터 유형 (크기)를 일치시켜야 한다.
- ✓ 프로그램에 따라 데이터 크기가 다르므로 주의해야 한다.
  - ✗ 정수의 경우 아두이노에서는 2 바이트이지만 대부분의 플랫폼에서는 4 바이트이다.

### ✚ 관련 함수

`Serial.write(val)`

- ✗ val을 1 byte로 송신한다.

`Serial.write(str)`

- ✗ 문자열을 byte 배열로 송신 한다.

`Serial.write(buf, len)`

- ✗ buf 변수의 데이터를 len 바이트만큼 송신한다.

`Serial.read();`

- ✗ 시리얼 포트의 버퍼에 있는 데이터의 첫 번째 바이트를 읽는다.

`Serial.readByte(buf, len);`

- ✗ 시리얼 버퍼에 있는 len 개 바이트를 읽어 buf 변수에 저장한다.
- ✗ buf : char[], byte[]
- ✗ len : 정수

### 예제 4-4-1

### ✚ 목표

- ✓ 사인파 데이터를 생성하고 시리얼 통신을 이용하여 컴퓨터에 이진 정수 데이터로 송신한다.
- ✓ SerialPlot 프로그램을 이용하여 송신된 데이터를 그린다.

### ✚ 회로도

- ✓ 없음

### ✚ 스케치 프로그램

```
// Generate Data and Send Data by Serial Communication
// Data type : binary integer

const int nx = 256;
float w0;

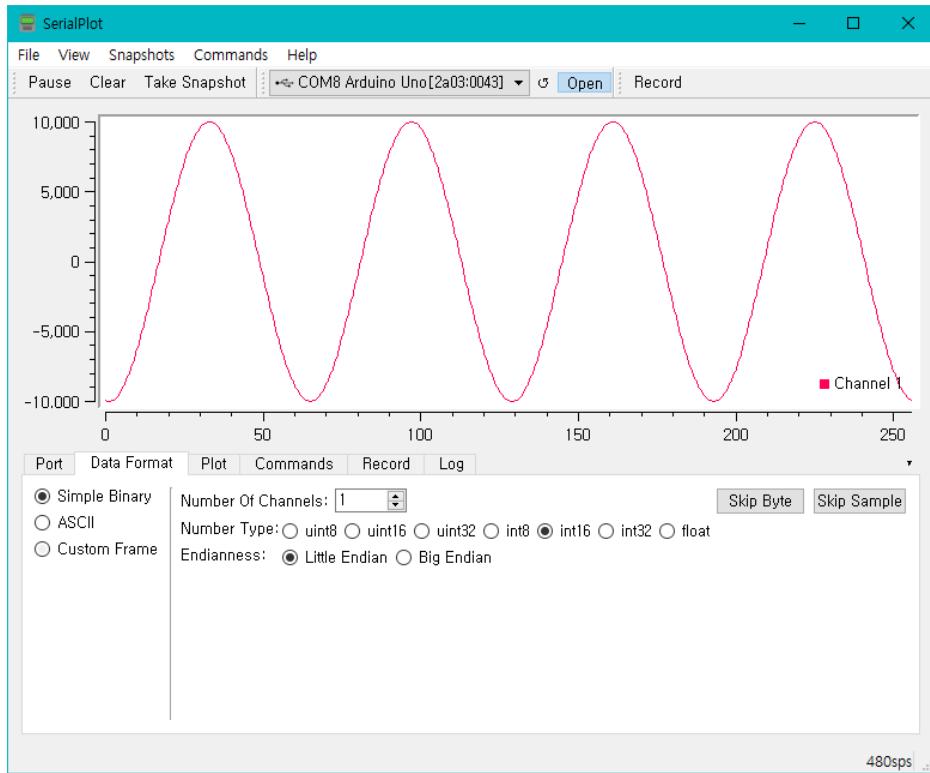
void setup() {
    // 시리얼 통신의 속도를 9600으로 설정한다.
    Serial.begin(9600);
    w0 = 4.0 * PI / nx;
}

void loop() {
    for (int n = 0; n < nx; n++) {
        // 사인파를 생성한다.
        float x = sin(w0 * n);
        // 정수로 변환한다.
        int xi = x * 10000;
        // 2 바이트 정수로 송신한다.
        sendBinary(xi);
    }
}

// 2 바이트로 16 비트 정수를 보내는 함수
void sendBinary(int x) {
    // 하위 바이트를 보낸다.
    Serial.write(lowByte(x));
    // 상위 바이트를 보낸다.
    Serial.write(highByte(x));
}
```

#### SerialPlot

- ✓ Data Format 탭에서 Simple Binary 를 선택하고 Number Type 에서 Int16 을 선택한다.



### 문제점

- ✓ 데이터를 2 바이트씩 읽어 정수를 만들기 때문에 데이터를 읽는 시점에 따라 틀린 값을 만들 수 있다.

### 해결 방법

- ✓ 방법 1 : SerialPlot 프로그램의 Data Format 탭의 Skip Byte 버튼을 눌러 바이트 읽는 시점을 바꾸면 올바른 데이터 값을 얻을 수 있다.
- ✓ 방법 2 : 데이터 앞에 헤더 바이트를 추가한다.
  - ✗ SerialPlot 프로그램의 Data Format 탭에서 Custom Frame을 선택하고 Frame Start에 적당한 헤더 바이트를 추가한 후 Fixed Size를 선택하고 4로 고친 후 Int16을 선택한다.

### 예제 4-4-2

#### 목표

- ✓ 사인파 데이터를 생성하고 시리얼 통신을 이용하여 컴퓨터에 이진 실수 데이터로 송신한다.
- ✓ 헤더 바이트로 0xFF를 추가한다.
- ✓ SerialPlot 프로그램을 이용하여 송신된 데이터를 그린다.

#### 회로도

- ✓ 없음

#### 스케치 프로그램

```
// Generate Data and Send Data by Serial Communication
// Data type : binary float with header

const int nx = 256;
// 헤더 문자
const byte header = 0xff;
float w0;

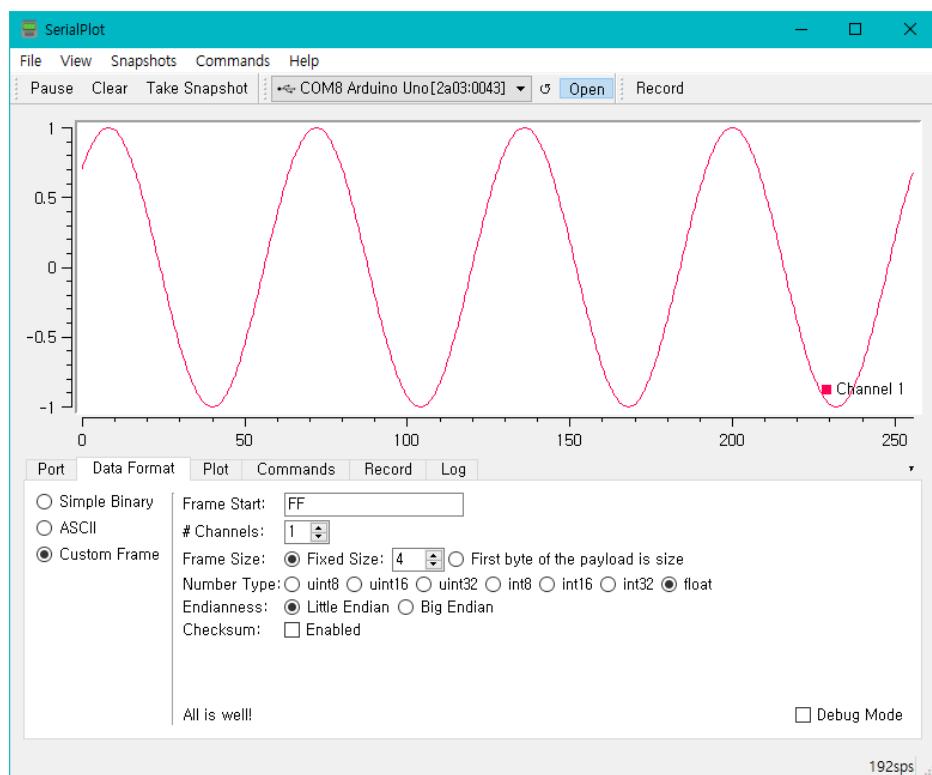
void setup() {
    // 시리얼 통신의 속도를 9600으로 설정한다.
    Serial.begin(9600);
    w0 = 4.0 * PI / nx;
}

void loop() {
    // 사인파를 생성하고 4 바이트 실수로 송신한다.
    for (int n = 0; n < nx; n++) {
        float x = sin(w0 * n);
        sendBinary(x);
    }
}

// 4 바이트로 32비트 실수를 보내는 함수
void sendBinary(float x) {
    // 헤더 문자를 송신한다.
    Serial.write(header);
    // 4 바이트 데이터를 송신한다.
    byte *px = (byte *) &x;
    Serial.write(px, 4);
}
```

#### SerialPlot

- ✓ Data Format 탭에서 Custom Frame 을 선택하고, Frame Start 에 FF(헤더 바이트)를 입력한다.



## 5. 모터 제어

### 5.1 모터의 종류

#### ■ 모터의 종류

##### ✓ 서보 모터

- ✗ PWM 신호로 회전을 제어한다.
- ✗ 0 ~ 180도 회전 각 조절 가능하고, 정밀한 위치 제어가 가능하다.

##### ✓ DC 모터

- ✗ 연속적인 회전이 필요할 때 사용한다. (선풍기, RC 카)
- ✗ 직류 전압으로 동작한다. (전압의 극성을 바꾸면 회전 방향이 변경된다.)

##### ✓ 스텝퍼 모터

- ✗ 회전 각을 스텝으로 나누어 정밀한 제어가 가능하다.



## 5.2 서보 모터 제어

### ■ 서보 모터

- ✓ 적색 (5 V), 흑색 (GND), 오렌지색 (PWM 제어)



### ■ 라이브러리

**Servo.h**

- ✓ 0 ~ 180 도 위치 제어 (1 도 간격으로 제어 가능)
- ✓ 12 개의 모터 제어 가능 (Mega 는 48 개)
- ✓ <https://www.arduino.cc/en/Reference/Servo>

### ■ 관련 함수

**.attach(pin);**

- \* 서보 모터의 제어 핀을 아두이노 핀에 연결한다.

**.write(int angle)**

- \* angle : 회전 각도, 0 ~ 180 사이의 정수

**.writeMicroseconds(int value)**

- \* value : 펄스의 지속 시간 ( $\mu$ s), 1000 ~ 2000 사이의 정수, 최대 시계 반 대 방향 (1000), 최대 시계 방향 (2000), 중앙 (1500)

**.read()**

- \* 서보 모터의 현재의 각도를 읽는다.

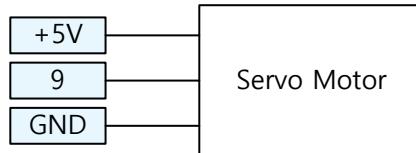
### 예제 5-2-1

### ■ 목표

- ✓ 0~180 도 사이를 왕복으로 회전한다.

회로도

- ✓ 디지털 9 번 핀에 모터 제어 선을 연결한다.



스케치 프로그램

- ✓ Servo.h 라이브러리 사용

---

```

// Control Servo Motor
// library : Servo.h

#include <Servo.h>

const int motorPin = 9;
const int delayTime = 10;
Servo myServo;

void setup() {
    // 서보 모터를 아두이노에 연결한다.
    myServo.attach(motorPin);
}

void loop() {
    // 10 µs 마다 0도에서 179도까지 1도씩 증가시키면서 서보 모터를 회전한다.
    for (int pos = 0; pos < 180; pos += 1) {
        myServo.write(pos);
        delay(delayTime);
    }
    // 10 µs 마다 180도에서 1도까지 1도씩 감소시키면서 서보 모터를 회전한다.
    for (int pos = 180; pos > 0; pos -= 1) {
        myServo.write(pos);
        delay(delayTime);
    }
}

```

---

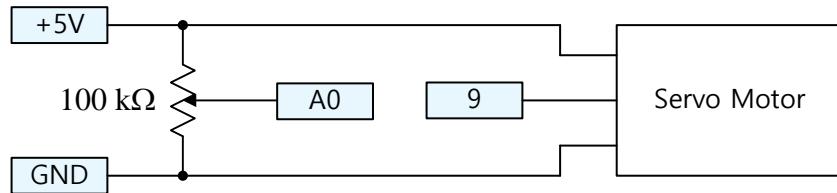
**예제 5-2-2**

목표

- ✓ 가변 저항의 값에 따라 서보 모터의 회전 각을 제어한다.

회로도

- ✓ 디지털 9 번 핀에 모터 제어 선을 연결한다.



#### ▶ 스케치 프로그램

- ✓ Servo.h 라이브러리 사용

```
// Control Servo Motor by Variable Resistor
// library : Servo.h

#include <Servo.h>

const int motorPin = 9;
const int vrPin = A0;
const int delayTime = 10;
Servo myServo;

void setup() {
    myServo.attach(motorPin);
    // 5 V를 10 비트 (1024 레벨)의 데이터로 읽는다.
    analogReference(DEFAULT);
}

void loop() {
    int potR = analogRead(vrPin);
    // 읽은 1024 레벨을 0~180 도 값으로 변환한다.
    int posM = map(potR, 0, 1023, 0, 180);
    myServo.write(posM);

}
```

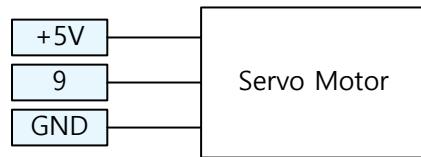
#### 예제 5-2-3

#### ▶ 목표

- ✓ 아두이노 시리얼 모니터에서 회전 각을 입력하면 시리얼 통신을 통해 입력한 각만큼 서보 모터가 회전하도록 한다. 0~180 을 벗어나는 숫자를 입력하면 무시하라.

#### ▶ 회로도

- ✓ 디지털 9 번 핀에 모터 제어 선 연결



### ✚ 스케치 프로그램

- ✓ Servo.h 라이브러리 사용

---

```

// Servo motor control by serial communication
// library : Servo.h

#include <Servo.h>

const int motorPin = 9;
Servo mySM;
String myStr = "";

void setup() {
    Serial.begin(9600);
    Serial.println("Enter the angle");
    mySM.attach(motorPin);
}

void loop() {
    // 시리얼 모니터에 각도를 입력하면 문자열로 읽은 후 정수로 변환한다.
    if (Serial.available()) {
        myStr = Serial.readString();
        int angle = myStr.toInt();
        // 읽은 값이 0~180 사이의 값이면 모터를 회전시키고 시리얼 모니터에
        // 나타내고, 범위를 벗어나면 무시한다.
        if (angle >= 0 && angle <= 180) {
            mySM.write(angle);
            Serial.print("Rotate ");
            Serial.print(angle);
            Serial.println(" degree");
        }
    }
}
  
```

---

### 연습 문제

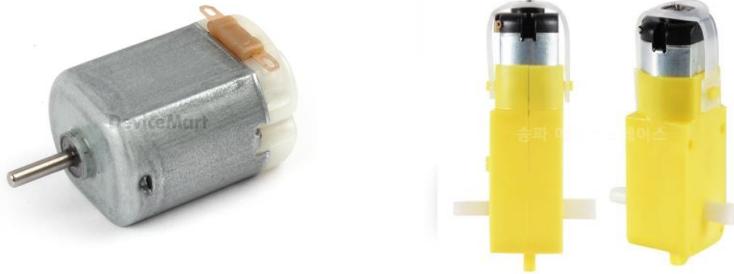
#### ✚ 연습 문제 5-2-1

- ✓ 스마트 폰에 0 도, 90 도, 180 도 3 개의 버튼을 만들고 이를 누를 경우 블루투스 통신을 통해 서보 모터가 해당 각도만큼 회전하는 회로를 구성하고 프로그램을 작성하라.

### 5.3 직류 모터 제어

#### ✚ 직류 (DC) 모터

- ✓ 직류 전압으로 모터 회전 속도를 제어한다.
- ✓ 아두이노에서는 PWM 신호로 모터 회전 속도를 제어한다.



#### ✚ 관련 함수

`analogWrite(pin, speed);`

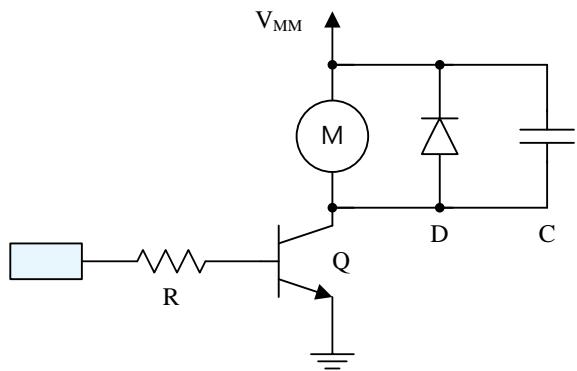
- ✗ pin : 모터에 연결되는 디지털 PWM 핀
- ✗ speed : 0~255 사이의 정수, 0이면 정지, 값이 클수록 속도가 빠르다.

#### ✚ 주의 사항

- ✓ 디지털 단자의 최대 출력 전류가 20 mA 이므로 모터에 흐르는 전류가 20 mA 보다 작아야 한다.
- ✓ 모터에 흐르는 전류가 20 mA 보다 크면 별도의 전원을 사용하고 트랜ジ스터 회로를 추가해야 한다.
- ✓ 우노 보드의 5 V 단자의 최대 전류 – USB (450 mA), power jack (650 mA)
- ✓ 모터의 회전 방향을 제어하려면 별도의 모터 드라이버 모듈이 필요 – L298N H-Bridge 모터 드라이버 모듈

#### ✚ 트랜지스터를 사용한 DC 모터 제어

- ✓ 모터가 한 방향으로만 회전한다.
- ✓ 다이오드 : 역방향 기전력으로 발생하는 역 전류로부터 트랜지스터와 아두이노 보드를 보호한다.
- ✓ 커패시터 : 잡음을 제거한다. (옵션)
- ✓  $V_{MM}$  : 모터에 맞는 별도 전압 (큰 전류를 흘릴 수 있는 전압)
- ✓ TR : 모터에 흐르는 전류를 감당할 수 있는 TR



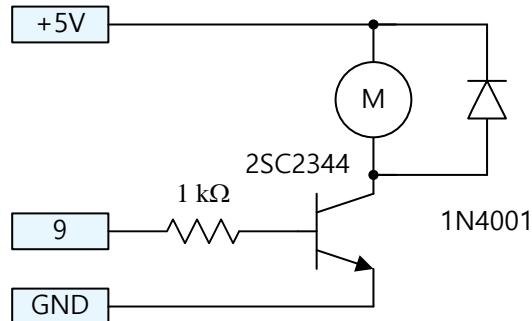
### 예제 5-3-1

#### ▣ 목표

- ✓ 아두이노로 직류 모터 속도를 제어한다. (6 단계로 제어하고 단계별 1초를 유지한다.)

#### ▣ 회로도

- ✓ 트랜지스터 베이스 단자를 저항을 통해 디지털 9 번 핀에 연결한다.



- ✓ 주의 사항 : 주어진 모터에 5 V 가 가해질 때 모터에 흐르는 전류는 180 mA 정도 흐르므로 큰 전류가 흐르는 파워 트랜지스터를 사용해야 한다.

#### ▣ 스케치 프로그램

```
// Control DC Motor

const int motorPin = 9;
const int delayTime = 1000;
int spdMotor[] = {0, 50, 100, 150, 200, 255};

void setup() {
}

void loop() {
    // 주어진 6 개의 각도로 모터를 회전시킨다.
    for (int i = 0; i < 6; i++) {
```

```

        analogWrite(motorPin, spdMotor[i]);
        delay(delayTime);
    }
}

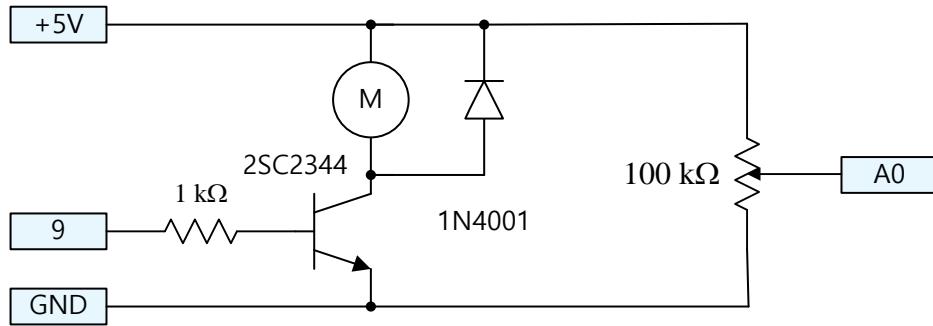
```

**예제 5-3-2****▣ 목표**

- ✓ 가변 저항을 사용하여 아두이노로 직류 모터 속도를 제어한다

**▣ 회로도**

- ✓ 트랜지스터 베이스 단자를 저항을 통해 디지털 9 번 핀에 연결하고, 가변 저항의 가변 단자를 아날로그 A0에 연결한다.



- ✓ 주의 사항 : 주어진 모터에 5 V 가 가해질 때 모터에 흐르는 전류는 180 mA 정도 흐르므로 큰 전류가 흐르는 파워 트랜지스터를 사용해야 한다.

**▣ 스케치 프로그램**

```

// Control DC Motor Speed using variable resistor

const int mPin = 9;;
const int rPin = A0;
const int delayTime = 20;

void setup() {
}

void loop() {
    // 가변 저항 값을 읽는다.
    int potR = analogRead(rPin);
    // 가변 저항 값(1024레벨)을 모터 속도 값(256 레벨)으로 변환한다.
    int mSpd = (potR, 0, 1023, 0, 255);
    // 변환된 값만큼 모터를 회전시킨다.
    analogWrite(mPin, mSpd);
}

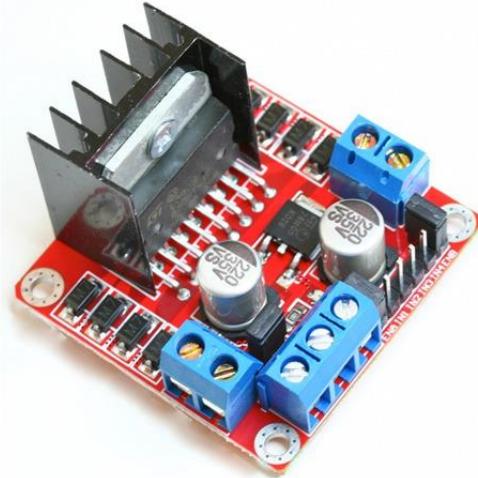
```

```

    delay(delayTime);
}

```

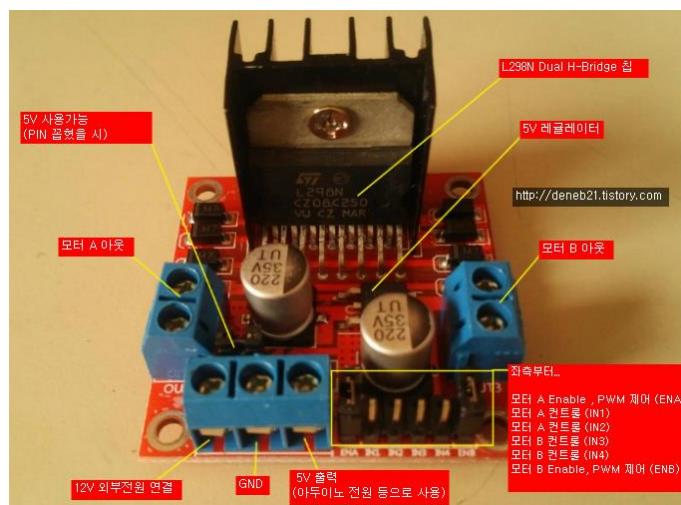
### ✚ L298N H-Bridge 모터 드라이버 모듈



#### ✓ 특징

- ✖ 2 개의 DC 모터 제어, 1 개의 스텝 모터 제어 가능
- ✖ 모터 A, B의 정 회전, 역 회전 제어 가능
- ✖ 모터 A, B의 속도 제어 가능
- ✖ 아두이노에 전원 (5 V) 공급 가능
- ✖ 모터를 구동하기 위한 외부 전원 (12 V) 입력 단자

#### ✓ 단자와 핀



- ✖ 전원 단자 (3)

- 12 V 외부 전원 입력 단자 : +12V

- 접지 단자 : GND
- 5 V 출력 단자 : +5V
- ✖ 모터 A 출력 단자 (2) : OUT1, OUT2
- ✖ 모터 B 출력 단자 (2) : OUT3, OUT4
- ✖ 모터 제어 단자 (6)
  - 모터 A 제어 핀 : IN1, IN2
  - 모터 B 제어 핀 : IN3, IN4
  - 모터 A PWM 제어 핀 : ENA
  - 모터 B PWM 제어 핀 : ENB
- ✓ 모터의 회전 방향 제어 (좌측 : 모터 A, 우측 : 모터 B)

모터 A			모터 B		
	IN1	IN2		IN3	IN4
정 회전	HIGH	LOW	정 회전	HIGH	LOW
역 회전	LOW	HIGH	역 회전	LOW	HIGH
정지	LOW	LOW	정지	LOW	LOW

IN1	IN2	IN3	IN4	모터 A	모터 B	
HIGH	LOW	HIGH	LOW	정 회전	정 회전	직진
LOW	HIGH	LOW	HIGH	역 회전	역 회전	후퇴
LOW	LOW	HIGH	HIGH	정지	정 회전	좌측 턴
HIGH	HIGH	LOW	LOW	정 회전	정지	우측 턴
LOW	HIGH	HIGH	LOW	역 회전	정 회전	좌 회전
HIGH	LOW	LOW	HIGH	정 회전	역 회전	우 회전
LOW	LOW	LOW	LOW	정지	정지	정지

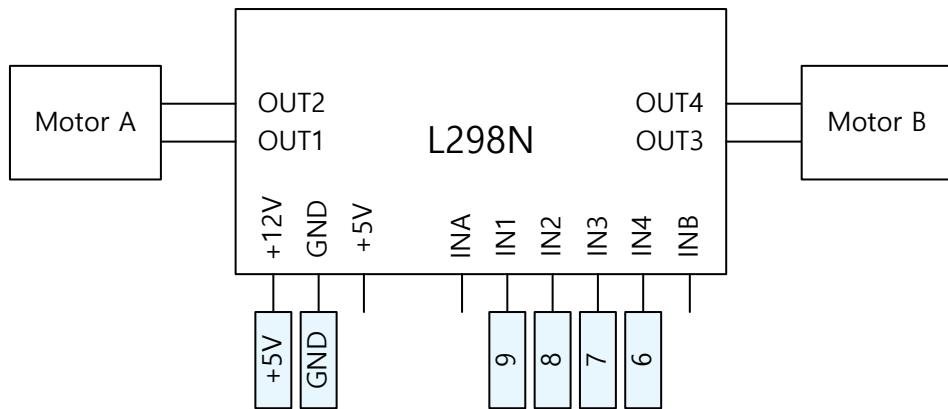
### 예제 5-3-3

#### ▣ 목표

- ✓ 아두이노로 2 개의 직류 모터 회전 방향을 제어한다.
- ✓ 2 개의 직류 모터의 회전 방향을 제어하여 전진, 후퇴, 좌측 턴, 우측 턴, 좌 회전, 우 회전, 정지를 실행할 수 있게 한다.

#### ▣ 회로도

- ✓ INA 와 INB 는 5 V 에 점퍼 핀으로 연결한다.



### ▶ 스케치 프로그램

```
// DC Motor Control using L298N Driver

const int in1Pin = 9, in2Pin = 8;
const int in3Pin = 7, in4Pin = 6;
const int delayTime = 3000;

void setup() {
    pinMode(in1Pin, OUTPUT);
    pinMode(in2Pin, OUTPUT);
    pinMode(in3Pin, OUTPUT);
    pinMode(in4Pin, OUTPUT);
}

// 자연 시간만큼 직진, 후진, 좌측 턴, 우측 턴, 좌 회전, 우 회전, 정지한다.
void loop() {
    goForward();    delay(delayTime);
    goBack();       delay(delayTime);
    turnLeft();     delay(delayTime);
    turnRight();    delay(delayTime);
    rotateLeft();   delay(delayTime);
    rotateRight();  delay(delayTime);
    stopMotor();    delay(delayTime);
}

// 직진 함수 (모터 A 정 회전, 모터 B 정 회전)
void goForward() {
    motorA_Go();   motorB_Go();
}

// 후진 함수 (모터 A 역 회전, 모터 B 역 회전)
void goBack() {
    motorA_Back(); motorB_Back();
}
```

```
}

// 정지 함수 (모터 A 정지. 모터 B 정지)
void stopMotor() {
    motorA_Stop(); motorB_Stop();
}

// 좌측 턴 (모터 A 정지. 모터 B 정 회전)
void turnLeft() {
    motorA_Stop(); motorB_Go();
}

// 우측 턴 (모터 A 정 회전. 모터 B 정지)
void turnRight() {
    motorA_Go(); motorB_Stop();
}

// 좌 회전 (모터 A 역 회전. 모터 B 정 회전)
void rotateLeft() {
    motorA_Back(); motorB_Go();
}

// 우 회전 (모터 A 정 회전. 모터 B 역 회전)
void rotateRight() {
    motorA_Go(); motorB_Back();
}

// 모터 A 정 회전 (모터 A 정지. 모터 B 정 회전)
void motorA_Go() {
    digitalWrite(in1Pin, HIGH);
    digitalWrite(in2Pin, LOW);
}

// 모터 A 역 회전
void motorA_Back() {
    digitalWrite(in1Pin, LOW);
    digitalWrite(in2Pin, HIGH);
}

// 모터 A 정지
void motorA_Stop() {
    digitalWrite(in1Pin, LOW);
    digitalWrite(in2Pin, LOW);
}
```

```

// 모터 B 정 회전
void motorB_Go() {
    digitalWrite(in3Pin, HIGH);
    digitalWrite(in4Pin, LOW);
}

// 모터 B 역 회전
void motorB_Back() {
    digitalWrite(in3Pin, LOW);
    digitalWrite(in4Pin, HIGH);
}

// 모터 B 정지
void motorB_Stop() {
    digitalWrite(in3Pin, LOW);
    digitalWrite(in4Pin, LOW);
}

```

#### ▣ 모터의 회전 속도 제어

- ✓ 모터 A의 회전 속도 제어
  - ✗ ENA 점퍼 핀을 제거하고 ENA 핀에 analogWrite 핀을 연결한다.
- ✓ 모터 B의 회전 속도 제어
  - ✗ ENB 점퍼 핀을 제거하고 ENB 핀에 analogWrite 핀을 연결한다.
- ✓ ENA 와 ENB 의 점퍼 핀을 사용하면 5 V 에 연결되어 모터가 최대 속도로 동작한다.

#### 예제 5-3-4

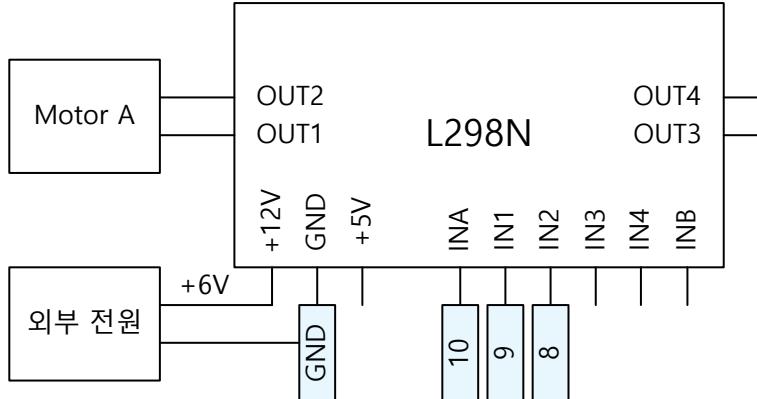
##### ▣ 목표

- ✓ 시리얼 모니터에 -255~255 사이의 숫자를 입력하여 양수이면 정 회전, 음수이면 역 회전, 숫자의 크기에 따라 속도를 제어한다.

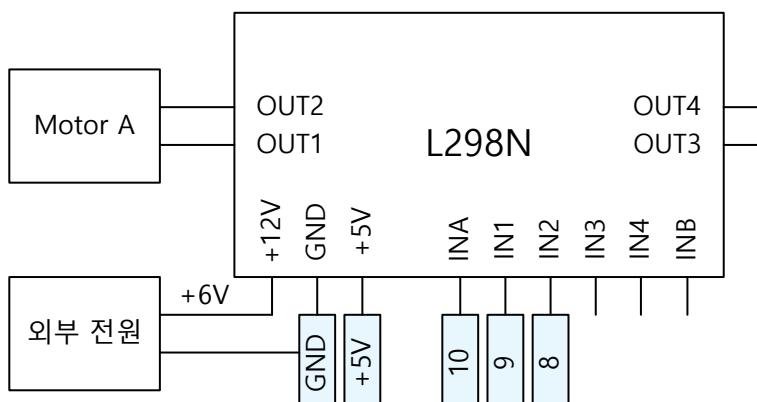
##### ▣ 회로도

- ✓ 모터 A 부분만 사용한다.
- ✓ 모터의 회전 속도를 제어하기 위해 ENA 와 5 V 에 연결된 점퍼 핀을 제거하고 ENA 에 아두이노 PWM 단자를 연결한다.

- ✓ 모터 드라이버의 전원에 6 V (건전지 4 개) 또는 그 이상의 전압을 연결한다.
- ✓ 아두이노의 전원 전압 5 V를 USB로 공급할 경우 회로도



- ✓ 아두이노의 전원 전압 5 V를 모터 드라이버로부터 공급할 경우 회로도



- ✓ 모터 드라이버의 전원 전압으로 9 V 배터리를 사용하는 경우 전류 용량이 작아 사용 시간이 짧아 가능하면 사용하지 않는게 좋다.

### 스케치 프로그램

```
// Control DC Motor speed using L298N Driver

// 모터 연결 핀
const int in1Pin = 9;
const int in2Pin = 8;
const int enaPin = 10;
String myString;

void setup() {
    Serial.begin(9600);
    pinMode(in1Pin, OUTPUT);
```

```
pinMode(in2Pin, OUTPUT);
// 모터를 정지시킨다.
digitalWrite(in1Pin, LOW);
digitalWrite(in2Pin, LOW);
}

void loop() {
    if (Serial.available()) {
        // 시리얼 모니터에서 문자열을 읽는다.
        myString = Serial.readString();
        Serial.println(myString);
        // 읽은 문자열을 정수 값으로 변환한다.
        int motorSpeed = myString.toInt();
        Serial.println(motorSpeed);
        // 변환한 값이 양수이면 모터를 주어진 속도로 정 회전시킨다.
        if (motorSpeed > 0) {
            digitalWrite(in1Pin, HIGH);
            digitalWrite(in2Pin, LOW);
            analogWrite(enaPin, motorSpeed);
        }
        // 변환한 값이 음수이면 모터를 주어진 속도로 역 회전시킨다.
        else if (motorSpeed < 0) {
            digitalWrite(in1Pin, LOW);
            digitalWrite(in2Pin, HIGH);
            analogWrite(enaPin, abs(motorSpeed));
        }
        // 변환한 값이 0이면 모터를 정지시킨다.
        else {
            digitalWrite(in1Pin, LOW);
            digitalWrite(in2Pin, LOW);
        }
    }
}
```

---

## 연습 문제

▣ 연습 문제 5-3-1

- ✓ 모터 드라이버를 사용하지 않고 시리얼 모니터에서 0 ~ 255 사이의 숫자를 입력하여 직류 모터의 회전 속도를 제어하는 회로를 구성하고 프로그램을 작성하라.

▣ 연습 문제 5-3-2

- ✓ 스마트 폰에 직진, 후퇴, 정지, 좌 회전, 우 회전 버튼을 만들고 이를 누를 경우 블루투스 통신을 통해 직류 모터가 해당 행동을 하는 회로를 구성하고 프로그램을 작성하라. (L298N 모터 드라이버를 사용하라.)

## 5.4 스텝 모터 제어

### ▣ 스텝 모터

- ✓ 28BYJ-48 스텝 모터 (unipolar 모터)
- ✓ ULN2003 드라이버 모듈로 제어한다.



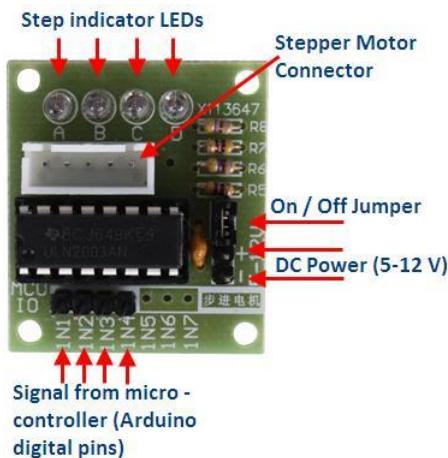
### ▣ 28BYJ-48 스텝 모터

- ✓ Unipolar 모터
- ✓ 동작 전압 : 5 V
- ✓ 동작 전류 : 대략 250 mA
- ✓ 4 상 모터
- ✓ 기어 비 : 64
- ✓ 스텝 각도 : 8 스텝 모드 (5.625 도, 64 스텝), 4 스텝 모드 (11.25 도, 32 스텝)
- ✓ 한 바퀴 스텝 수 : 8 스텝 모드 ( $64 \times 64 = 4096$ ), 4 스텝 모드 ( $32 \times 64 = 2048$ )
- ✓ 아두이노 Stepper 라이브러리 : 4 스텝 모드로 동작한다.



### ▣ ULN2003 드라이버 모듈

- ✓ 전원 단자 : 5~12 V, GND
- ✓ 제어 단자 : IN1, IN2, IN3, IN4



#### ■ 라이브러리

**Stepper.h**

- ✓ 단극 (unipolar), 양극 (bipolar) 스텝 모터를 제어한다.
- ✓ <https://www.arduino.cc/en/Reference/Stepper>
- ✓ <https://github.com/arduino/Arduino/tree/master/libraries/Stepper>

#### ■ 관련 함수

**Stepper(int numStep, int in1Pin, int in3Pin, int in2Pin, int in4Pin);**

- ✗ numStep : 360 도 회전하는데 필요한 스텝 수, 28BYJ-48의 경우 2048
- ✗ in1Pin ~ in4Pin : 드라이버 모듈 핀 in1 ~ in4에 연결할 아두이노 디지털 핀 (순서 조심: in1, in3, in2, in4)

**.step(int numStep)**

- ✗ 스텝 모터를 numStep 스텝만큼 회전
- ✗ 음수이면 역 회전

**.setSpeed(long motorSpeed)**

- ✗ motorSpeed : 분당 모터 회전 수 (rpm)

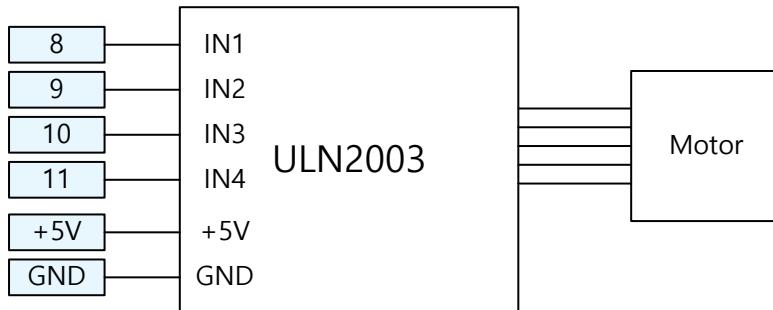
#### 예제 5-4-1

#### ■ 목표

- ✓ 0~360 도 사이를 왕복으로 회전한다.
- ✓ 순방향 속도 5 rpm, 역방향 속도 10 rpm

회로도

- ✓ 디지털 핀 8, 9, 10, 11 번에 드라이버 모듈 IN1, IN3, IN2, IN4 을 연결한다.



스케치 프로그램

- ✓ Stepper.h 라이브러리 사용

---

```

// Stepper Motor Control using ULN2003 Driver
// library : Stepper.h

#include <Stepper.h>

const int numStep = 2048;
const int rpm1 = 5, rpm2 = 10;
// 모터 연결 핀
const int in1 = 8, in2 = 9, in3 = 10, in4 = 11;
const int delayTime = 1000;
Stepper mySM(numStep, in1, in3, in2, in4);

void setup() {
}

void loop() {
    // 5 rpm 속도로 모터를 정 회전 시킨다.
    mySM.setSpeed(rpm1);
    mySM.step(numStep);
    delay(delayTime);
    // 10 rpm 속도로 모터를 정 회전 시킨다.
    mySM.setSpeed(rpm2);
    mySM.step(-numStep);
    delay(delayTime);
}
  
```

---

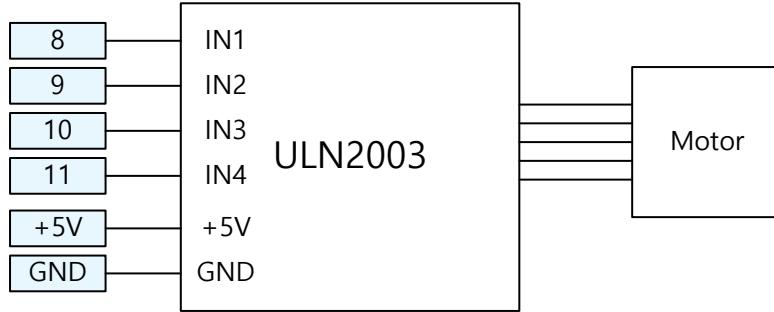
예제 5-4-2

목표

- ✓ 아두이노 시리얼 모니터에서 회전 각(도)을 입력하면 스텝 모터가 주어진 각도 만큼 회전(음수를 입력하면 역 방향 회전)
- ✓ 회전 속도 : 10 rpm

#### ▣ 회로도

- ✓ 디지털 핀 8, 9, 10, 11 번에 드라이버 모듈 IN1, IN3, IN2, IN4를 연결한다.



#### ▣ 스케치 프로그램

- ✓ Stepper.h 라이브러리 사용

```
// Stepper Motor Control using ULN2003 Driver

#include <Stepper.h>

const int numStep = 2048;
// 모터 연결 핀
const int in1 = 8, in2 = 9, in3 = 10, in4 = 11;
String myString;
Stepper mySM(numStep, in1, in3, in2, in4);

void setup() {
    Serial.begin(9600);
    // 모터 회전 속도 설정
    mySM.setSpeed(10);
}

void loop() {
    if (Serial.available()) {
        // 시리얼 모니터에서 문자열을 읽는다.
        myString = Serial.readString();
        // 읽은 문자열을 정수 값으로 변환한다.
        int angle = myString.toInt();
        // 읽은 정수값(각도)를 회전 스텝 수로 변환한다.
        int nStep = numStep / 360.0 * angle;
        Serial.println(angle);
    }
}
```

```
// 모터를 회전 시킨다.  
mySM.step(nStep);  
}  
}
```

---

## 연습 문제

### 연습 문제 5-4-1

- ✓ 스마트 폰에 시계 반대 방향으로 180 도, 90 도, 시계 방향으로 90 도, 180 도 회전 버튼을 만들고 이를 누를 경우 블루투스 통신을 통해 스텝퍼 모터가 해당 행동을 하는 회로를 구성하고 프로그램을 작성하라.

## 6. 스피커 제어

### 6.1 스피커 제어

#### ✚ 아두이노 용 스피커 종류

- ✓ 피에조 부저
  - ✗ 피에조 효과를 이용한 소자로 인가된 전압에 의해 진동시켜 음을 재생한다.
  - ✗ 자기 손실이 적고 효율적으로 전기 에너지를 소리로 변환한다.



#### ✚ 아두이노에서의 음 출력

- ✓ 주어진 주파수를 갖고 duty cycle 이 50 %인 구형파를 출력한다.

#### ✚ 관련 함수

**tone(pin, freq);**

- ✗ 아두이노 핀에 duty cycle이 50 %이고 주파수가 freq [Hz]인 구형파를 noTone() 함수를 부를 때까지 계속 내 보낸다.
- ✗ 한 번에 한 tone만 발생시킨다. (tone이 연주되고 있으면 tone() 함수를 불러도 아무 일이 일어나지 않는다.)

**tone(pin, freq, duration);**

- ✗ 아두이노 핀에 duty cycle이 50 %이고 주파수가 freq [Hz]인 구형파를 duration [ms] 동안 내 보낸다.

**noTone(pin);**

- ✗ 아두이노 핀에 tone()에 의해 발생된 구형파를 내 보내지 않는다.
- ✗ tone() 함수로 원하는 시간만큼 소리를 내려면 delay() 함수를 불러 원하는 시간만큼 아무 일도 하지 않고 시간이 흐른 후 notone() 함수를 불러야 한다.

- ✓ `analogWrite()` 함수는 주파수가 정해져 있고 duty cycle 을 조절할 수 있는데 반해 `tone()`함수는 duty cycle 은 50%로 정해져 있고 주파수를 조절할 수 있다.

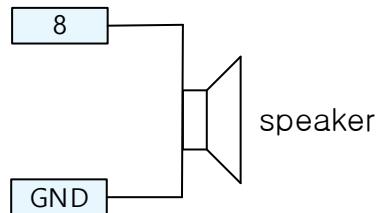
### 예제 6-1-1

#### 목표

- ✓ 500 Hz 의 소리를 계속해서 1 초 간격으로 소리 발생과 정지를 반복하도록 한다.

#### 회로도

- ✓ 아두이노 디지털 8 번 핀에 스피커를 연결한다.



#### 스케치 프로그램

---

```

// Speaker control

const int tonePin = 8;
int freq = 500;
int dur1 = 1000, dur2 = 2000;

void setup() {
}

void loop() {
    // 500 Hz의 소리를 1초 동안 내 보낸다.
    tone(tonePin, freq, dur1);
    // 소리를 내 보내고 2초 후 noTone() 함수를 부른다.
    // 1초 동안 소리가 나오고 1초 동안 소리를 멈춘다.
    delay(dur2);
    // 다음에 tone() 함수를 부르기 위해 noTone() 함수를 부른다.
    noTone(tonePin);
}
  
```

---

- ✓ 주의 사항

- \* `tone()` 함수에서 duration 시간을 주어도 `tone()` 함수를 실행한 후 지연

시간을 주지 않고 바로 `notone()` 함수를 부르면 소리가 바로 끝난다.  
(`duration` 시간이 의미가 없어진다.)

- ✖ `duration` 시간보다 자연 시간을 크게 주고 `notone()` 함수를 불러야 소리가 `duration` 시간 동안 유지된다.

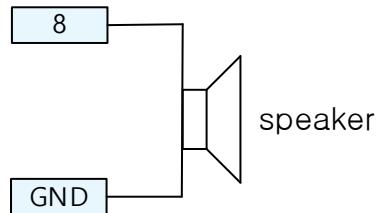
### 예제 6-1-2

#### ✚ 목표

- ✓ 주파수가 300 Hz, 500 Hz, 700 Hz 인 소리를 1초 간격으로 계속 발생시킨다.

#### ✚ 회로도

- ✓ 아두이노 디지털 8 번 핀에 스피커를 연결한다.



#### ✚ 스케치 프로그램

```
// Speaker control

const int tonePin = 8;
int freq[] = {300, 500, 700};
int dura = 1000;

void setup() {
}

void loop() {
    for (int n = 0; n < 3; n++) {
        // 주어진 주파수로 소리를 발생시킨다.
        tone(tonePin, freq[n]);
        delay(dura);
        noTone(tonePin);
    }
}
```

#### ✚ 새로운 라이브러리 1

**NewTone.h**

- ✓ 속도가 빠르고 크기가 작은 라이브러리

- ✓ 고 품질의 소리 출력
- ✓ 스피커와 아두이노 핀 사이에  $100\ \Omega$  저항을 연결해야 한다. (헤더 파일 안에  $100\ \Omega$  저항을 삽입하라고 적혀 있지만 직접 연결해도 동작한다.)
- ✓ <https://bitbucket.org/teckel12/arduino-new-tone/wiki/Home>

#### ✚ 관련 함수

`NewTone(pin, freq, duration);`

- \* 아두이노 핀에 duty cycle이 50 %이고 주파수가 freq [Hz]인 구형파를 duration [ms] 동안 내 보낸다.
- \* duration : 구형파가 나가는 시간 (ms), 0이면 계속 나간다. (생략 가능, default = 0)

`noNewTone(pin);`

- \* 아두이노 핀에 newTone()에 의해 발생된 구형파를 내 보내지 않는다.

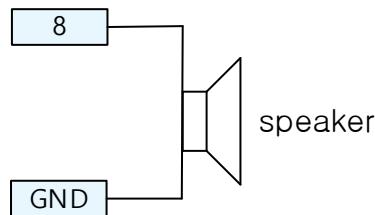
#### 예제 6-1-3

#### ✚ 목표

- ✓ 주파수가 300 Hz, 500 Hz, 700 Hz 인 소리를 1 초 간격으로 계속 발생시킨다.
- ✓ NewTone.h 라이브러리를 이용한다.

#### ✚ 회로도

- ✓ 아두이노 디지털 8 번 핀에 스피커를 연결한다.



#### ✚ 스케치 프로그램

- ✓ NewTone.h 라이브러리를 사용

---

```

// Speaker control
// Library : NewTone.h

#include <NewTone.h>

const int tonePin = 8;
int freq[] = { 300, 500, 700 };
  
```

```

int dur = 1000;

void setup() {
}

void loop() {
    for (int n = 0; n < 3; n++) {
        // 주어진 주파수로 소리를 발생시킨다.
        NewTone(tonePin, freq[n]);
        delay(dur);
        noNewTone(tonePin);
    }
}

```

### ▣ 새로운 라이브러리 2

#### TimerFreeTone.h

- ✓ Timer 를 사용하지 않는 tone 라이브러리
- ✓ Timer 를 사용하는 라이브러리 (예: NewPing.h) 와 tone 라이브러리를 같이 사용하는 경우 충돌이 일어나기 때문에 timer 를 사용하지 않는 tone 라이브러리를 사용해야 한다.
- ✓ 스피커와 아두이노 핀 사이에 100 Ω 저항을 연결해야 한다. (NewTone.h 와 마찬가지로 헤더 파일 안에 100 Ω 저항을 삽입하라고 적혀 있지만 직접 연결해도 동작한다.)
- ✓ <https://bitbucket.org/teckel12/arduino-timer-free-tone/wiki/Home>

### ▣ 관련 함수

#### TimerFreeTone(pin, freq, duration, vol = 10);

- ✗ 아두이노 핀에 duty cycle이 50 %이고 주파수가 freq [Hz]인 구형파를 duration [ms] 동안 내 보낸다.
- ✗ Freq : 주파수, 100 ~ 15kHz일 때가 잘 동작한다.
- ✗ duration : 구형파가 나가는 시간 (ms), 0 ~ 65535 (65.5 sec)
- ✗ vol : 음량 (1 ~ 10), 10이면 최대 음량 (생략 가능, default=10)

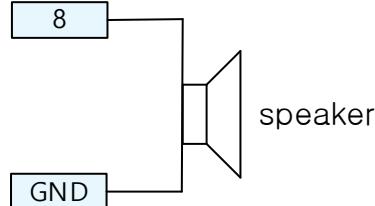
### 예제 6-1-4

### ▣ 목표

- ✓ 주파수가 300 Hz, 500 Hz, 700 Hz 인 소리를 1 초 간격으로 계속 왕복 발생시킨다.
- ✓ TimerFreeTone.h 라이브러리를 이용한다.

 회로도

- ✓ 아두이노 디지털 8 번 핀에 스피커를 연결한다.



 스케치 프로그램

- ✓ TimerFreeTone.h 라이브러리를 사용

---

```

// Speaker control
// Library : TimerFreeTone.h

#include <TimerFreeTone.h>

const int tonePin = 8;
int freq[] = { 300, 500, 700, 500 };
int dur = 1000;

void setup() {
}

void loop() {
    for (int n = 0; n < 4; n++) {
        // 주어진 주파수로 소리를 발생시킨다.
        TimerFreeTone(tonePin, freq[n], dur);
    }
}
  
```

---

## 연습 문제

 연습 문제 6-1-1

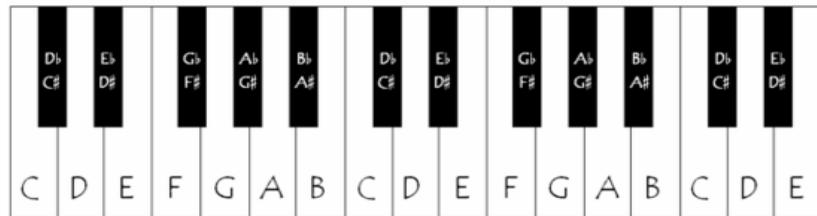
- ✓ 아두이노 시리얼 모니터에서 숫자를 입력하고 입력한 숫자가 0~10 사이이면 400 Hz 의 소리를 계속해서 발생시키고, 11~30 사이이면 0.2 초 간격으로, 31~60 사이이면 0.4 초 간격으로, 61~100 사이이면 0.8 초

간격으로 발생과 정지를 반복하고, 그 외의 숫자이면 아무 소리도 나지 않도록 회로를 구성하고 프로그램을 작성하라.

## 6.2 음악 연주

### ✚ 음계

- ✓ 한 옥타브 – 주파수 2 배
- ✓ 가운데 '라' (A) 음 – 440 Hz
- ✓ 12 음계 – 주파수가  $2^{(1/12)}$  배씩 증가



- ✓ 음계 주파수

Tone	Do1	Re1	Mi1	Fa1	Sol1	La1	Si1	Do2	Re2	Mi2	Fa2	Sol2	La2	Si2
Tone	C	D	E	F	G	A	B	C	D	E	F	G	A	B
f [Hz]	262	294	330	349	392	440	494	523	587	659	698	784	880	988

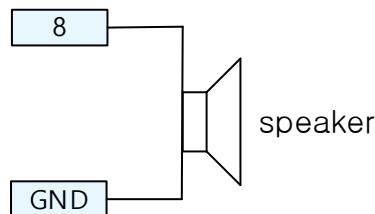
### 예제 6-2-1 : 음악 연주

### ✚ 목표

- ✓ '학교 종이 땡땡땡' 을 연주한다.
- ✓ 음계와 박자 벡터가 주어질 때 주파수와 음의 길이를 계산하여 연주한다.

### ✚ 회로도

- ✓ 아두이노 디지털 8 번 핀에 스피커를 연결한다.



### ✚ 스케치 프로그램

```
// Play music (Melody)
// 학교 종이 땡땡땡
```

```
// 음계 주파수 설정
#define D01 262
#define RE1 294
#define MI1 330
```

```
#define FA1 349
#define SOL1 392
#define LA1 440
#define SI1 494
#define DO2 523
#define RE2 587
#define MI2 659
#define FA2 698
#define SOL2 784
#define LA2 880
#define SI2 988

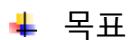
const int notePin = 8;
// 노래 음계
int noteFreq[] = {
    SOL1,SOL1,LA1,LA1,SOL1,SOL1,MI1,SOL1,SOL1,MI1,MI1,RE1,
    SOL1,SOL1,LA1,LA1,SOL1,SOL1,MI1,SOL1,MI1,RE1,MI1,DO1
};
// 노래 박자
int noteLeng[] = {
    4,4,4,4,4,4,2,4,4,4,4,1,4,4,4,4,4,4,2,4,4,4,4,1
};

void setup() {
    // 음계의 개수를 계산한다.
    int numNote = sizeof(noteLeng) / sizeof(noteLeng[0]);
    for (int n = 0; n < numNote; n++) {
        // 노래 박자에 따라 지연 시간을 계산한다.
        int noteDur = 1000/noteLeng[n];
        // 소리를 발생시킨다.
        tone(notePin, noteFreq[n], noteDur);
        // 소리 발생이 멈춘 후 소리 지연 시간의 30% 정도 지연 시간을 준다.
        int pauseNote = noteDur*1.3;
        delay(pauseNote);
        noTone(notePin);
    }
}

void loop() {
```

---

### 예제 6-2-2 : 음악 연주

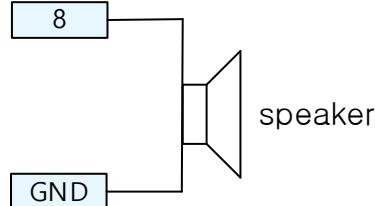


#### 목표

- ✓ '학교 종이 땡땡땡' 을 연주한다.
- ✓ 음계와 박자 벡터가 주어질 때 주파수와 음의 길이를 계산하여 연주한다.
- ✓ 음계 주파수를 별도의 헤더 파일에 저장한다. (note\_freq.h)

### 회로도

- ✓ 아두이노 디지털 8 번 핀에 스피커를 연결한다.



### 스케치 프로그램

```

// Play music (Melody)
// 학교 종이 땡땡땡

#include "note_freq.h"

const int notePin = 8;
// 노래 음계
int noteFreq[] = {
    SOL1,SOL1,LA1,LA1,SOL1,SOL1,MI1,SOL1,SOL1,MI1,MI1,RE1,
    SOL1,SOL1,LA1,LA1,SOL1,SOL1,MI1,SOL1,MI1,RE1,MI1,DO1
};
// 노래 박자
int noteLeng[] = {
    4,4,4,4,4,4,2,4,4,4,4,1,4,4,4,4,4,4,2,4,4,4,4,1
};

void setup() {
    // 음계의 개수를 계산한다.
    int numNote = sizeof(noteLeng) / sizeof(noteLeng[0]);
    for (int n = 0; n < numNote; n++) {
        // 노래 박자에 따라 지연 시간을 계산한다.
        int noteDur = 1000/noteLeng[n];
        // 소리를 발생시킨다.
        tone(notePin, noteFreq[n], noteDur);
        // 소리 발생이 멈춘 후 소리 지연 시간의 30% 정도 지연 시간을 준다.
        int pauseNote = noteDur*1.3;
        delay(pauseNote);
        noTone(notePin);
    }
}
  
```

```
    }  
}  
  
void loop() {  
}
```

### ✓ 헤더 파일

```
// Note Frequency  
  
#define D01 262  
#define RE1 294  
#define MI1 330  
#define FA1 349  
#define SOL1 392  
#define LA1 440  
#define SI1 494  
#define D02 523  
#define RE2 587  
#define MI2 659  
#define FA2 698  
#define SOL2 784  
#define LA2 880  
#define SI2 988
```

## 연습 문제

### ▶ 연습 문제 6-2-1

- ✓ '반짝 반짝 작은 별' 동요를 연주하는 회로를 구성하고 프로그램을 작성하라.

## 7. 초음파 센서

### ➊ 거리 측정용 초음파 센서

- ✓ 대상에 초음파를 발사하여 반사되어 오는 시간을 측정하여 대상과의 거리를 측정하는 센서

### ➋ 아두이노 용 초음파 센서 (HC-SR04)

- ✓ 규격
  - ✗ 정격 전압 : DC 5 V
  - ✗ 동작 전류 : 15 mA
  - ✗ 동작 주기 : 40 Hz (한 번 측정하고 25 ms 정도 쉬어야 한다.)
  - ✗ 측정 거리 : 2 cm ~ 400 cm
  - ✗ 유효 각도 : 15 도
- ✓ 자료: <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>



### ✓ 핀 배치

- ✗ VCC (5 V), GND (접지)
- ✗ Trig (초음파 송신 단자)
- ✗ Echo (초음파 수신 단자)

### ➌ 라이브러리

#### NewPing.h

- ✓ 초음파를 쏘고 반향을 수신하여 시간 및 거리 측정
- ✓ 시간과 거리 변환 가능
- ✓ 주어진 거리 내에서 반향이 있는지 체크 가능
- ✓ 여러 개의 초음파 센서 동작 가능
- ✓ <http://playground.arduino.cc/Code/NewPing>

### ✚ 관련 함수

`NewPing(trigPin, echoPin, max_cm);`

- ✖ trigPin : 초음파 센서 모듈 trig에 연결할 아두이노 핀 번호
- ✖ echoPin : 초음파 센서 모듈 echo에 연결할 아두이노 핀 번호
- ✖ max\_cm : 최대 측정 거리, 옵션 (default = 500 cm)

`.ping();`

- ✖ 초음파를 쏘고 반향을 수신하는데 걸리는 시간 ( $\mu\text{s}$ )을 돌려 준다. 주어진 거리 내에 반향 없으면 0을 돌려 준다.

`.ping_cm();`

- ✖ 초음파를 쏘고 반향을 수신하여 측정한 거리 (cm)를 돌려 준다. 주어진 거리 내에 반향 없으면 0을 돌려 준다.

`.ping_median(iteration);`

- ✖ 초음파를 iteration 수만큼 쏘고 (default=5) 수신하는데 걸리는 시간 중 중앙값을 돌려 준다.

`.convert_cm(echoTime);`

- ✖ 반향 시간 ( $\mu\text{s}$ )을 거리 (cm)로 변환한다.

`.check_timer();`

- ✖ 주어진 거리 내에서 반향이 수신되는지 체크한다.

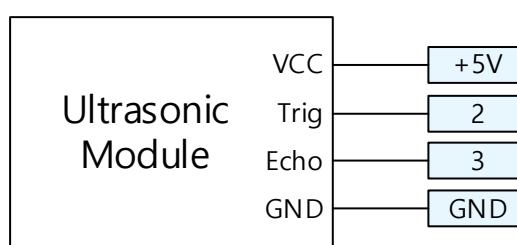
### 예제 7-1-1

### ✚ 목표

- ✓ 초음파 센서를 통해 여러가지 상황에서 거리를 측정하고 아두이노 시리얼 모니터에 표시한다.
- ✓ NewPing.h 라이브러리 이용한다.

### ✚ 회로도

- ✓ 초음파 센서 모듈의 trig 핀을 아두이노 디지털 2 번 핀에 연결한다.
- ✓ 초음파 센서 모듈의 echo 핀을 아두이노 디지털 3 번 핀에 연결한다.



 스케치 프로그램

- ✓ NewPing.h 라이브러리 사용

```
// Measure the distance using the supersonic sensor
// Display the distance (cm) in the serial monitor
// Library : NewPing.h

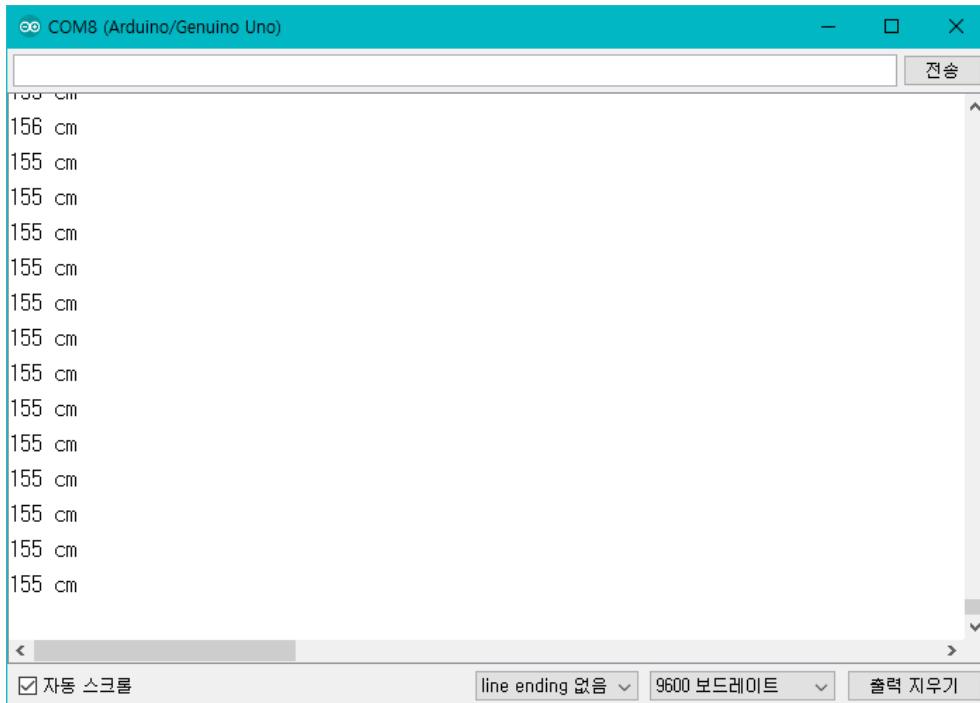
#include <NewPing.h>

// 아두이노 연결 핀 번호
const int trigPin = 2;
const int echoPin = 3;
const int maxDistance = 300;
const int delayTime = 200;
// NewPing 객체 선언
NewPing myPing(trigPin, echoPin, maxDistance);

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

void loop() {
    // 초음파 센서로 거리(cm)를 측정하고 시리얼 모니터에 나타낸다.
    int distance = myPing.ping_cm();
    Serial.print(distance);
    Serial.println(" cm");
    delay(delayTime);
}
```

- ✓ 시리얼 모니터 창



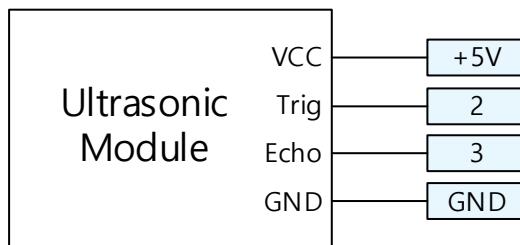
### 예제 7-1-2

목표

- ✓ 초음파 센서를 통해 여러가지 상황에서 거리를 측정하고 아두이노 시리얼 모니터에 표시한다.
  - ✓ 여러 번 측정하여 중간 값을 표시한다.
  - ✓ NewPing.h 라이브러리를 사용한다.

+ 회로도

- ✓ 초음파 센서 모듈의 trig 핀을 아두이노 디지털 2 번 핀에 연결한다.
  - ✓ 초음파 센서 모듈의 echo 핀을 아두이노 디지털 3 번 핀에 연결한다.



+ 스케치 프로그램

- ✓ NewPing.h 라이브러리 사용

```
// Measure the distance using the supersonic sensor  
// Display the distance (cm) in the serial monitor  
// Display median value
```

```
// Library : NewPing.h

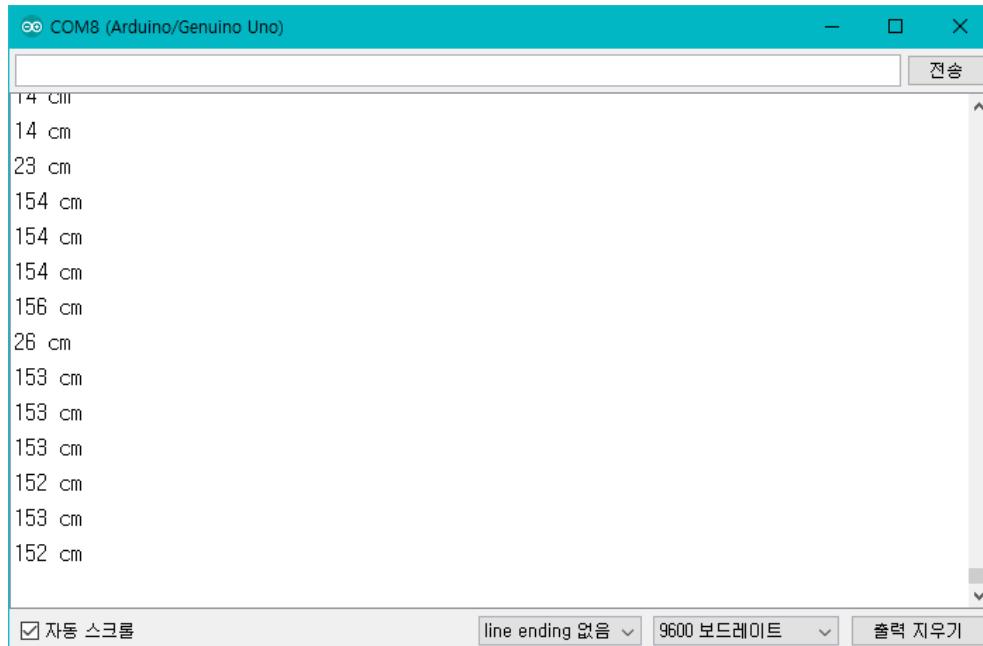
#include <NewPing.h>

const int pingIter = 5;
const int delayTime = 100;
// NewPing 객체 선언
NewPing myPing(2, 3, 300);

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

void loop() {
    // 초음파 센서로 5회 거리를 측정한 후 중앙 값(걸리는 시간)을 읽는다.
    int pingTime = myPing.ping_median(pingIter);
    // 걸리는 시간을 거리로 변환한다.
    int distance = myPing.convert_cm(pingTime);
    Serial.print(distance);
    Serial.println(" cm");
    delay(delayTime);
}
```

### ✓ 시리얼 모니터 창



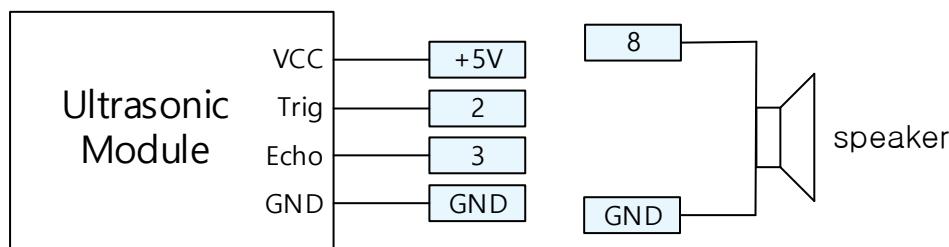
### 예제 7-1-3

#### ▣ 목표

- ✓ 초음파 센서를 통해 여러가지 상황에서 거리를 측정하고 아두이노 시리얼 모니터에 표시한다.
- ✓ 거리가 20 cm 이내이면 300 Hz, 20 ~ 40 cm 사이이면 500 Hz, 40 ~ 60 cm 사이이면 700 Hz, 60 cm 이상이면 900 Hz의 음을 발생시킨다.
- ✓ 초음파 센서는 NewPing.h 라이브러리를 이용한다.
- ✓ 스피커는 TimerFreeTone.h 라이브러리를 사용한다. (NewPing.h 와 tone() 함수는 timer를 사용하기 때문에 충돌이 발생하여 컴파일 에러가 발생한다. 충돌을 피하려면 timer를 사용하지 않는 TimerFreeTone.h 라이브러리를 사용해야 한다.)

#### ▣ 회로도

- ✓ 초음파 센서 모듈의 trig 와 echo 핀을 아두이노 디지털 2 번과 3 번 핀에 연결한다.
- ✓ 스피커는 아두이노 디지털 8 번 핀에 연결한다.



#### ▣ 스케치 프로그램

- ✓ NewPing.h, TimerFreeTone.h 라이브러리 사용

```
// Measure the distance using the supersonic sensor
// Sound control by distance
// Library : NewPing.h, TimerFreeTone.h

#include <NewPing.h>
#include <TimerFreeTone.h>

// 초음파 센서와 스피커 연결 핀
const int trigPin = 2;
const int echoPin = 3;
const int speakerPin = 8;
// 소리 주파수
const int toneFreq[] = { 300, 500, 700, 900 };
// 거리 구분
```

```
const int distArray[] = { 20, 40, 60 };
const int delayTime = 50;

// NewPing 객체 선언
NewPing myPing(trigPin, echoPin, 300);

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

void loop() {
    // 초음파 센서로 거리를 측정한다.
    int distance = myPing.ping_cm();

    Serial.print(distance);
    Serial.println(" cm");

    // 거리에 따른 소리를 출력한다.
    int freq = soundFreq(distance);
    TimerFreeTone(speakerPin, freq, delayTime);
}

// 거리에 따른 소리 주파수를 돌려주는 함수
int soundFreq(int dist) {
    // 거리가 20 cm 이하이면 300 Hz를 돌려준다.
    if (dist <= distArray[0]) {
        return toneFreq[0];
    }
    // 거리가 20 ~ 40 cm 이내면 500 Hz를 돌려준다.
    else if (dist <= distArray[1]) {
        return toneFreq[1];
    }
    // 거리가 40 ~ 60cm 이내면 700 Hz를 돌려준다.
    else if (dist <= distArray[2]) {
        return toneFreq[2];
    }
    // 거리가 60 cm 이상이면 900 Hz를 돌려준다.
    else {
        return toneFreq[3];
    }
}
```

---

## 연습 문제

 연습 문제 7-1-1

- ✓ 초음파 센서 모듈을 이용하여 거리를 측정하고 측정한 거리가 20 cm 이하이면 600 Hz 의 소리를 0.2 초 간격으로, 20 ~ 40 cm 사이이면 0.5 초 간격으로, 40 ~ 60 cm 사이이면 0.8 초 간격으로 계속해서 발생과 정지를 반복하고, 60 cm 이상이면 아무 소리도 나지 않도록 회로를 구성하고 프로그램을 작성하라.

## 8. 블루투스 통신

### 8.1 아두이노 블루투스 통신

#### 8.1.1 블루투스 시리얼 통신

##### 블루투스 통신

- ✓ 아두이노와 스마트폰 사이의 블루투스 통신
- ✓ 아두이노와 아두이노 사이의 블루투스 통신

##### 하드웨어 시리얼 통신

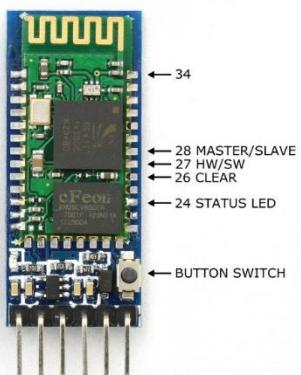
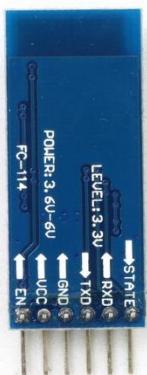
- ✓ RX(1 번 핀), TX(2 번 핀) : 아두이노와 컴퓨터 사이의 통신에 사용하기 위해 예약되어 있다.
- ✓ RX, TX 핀을 다른 용도로 사용하면 아두이노와 컴퓨터 사이의 시리얼 통신에 에러가 발생한다.
- ✓ 스케치 프로그램을 업로드할 때 사용한다.
- ✓ Serial.xxx 함수를 사용한다.

##### 소프트웨어 시리얼 통신

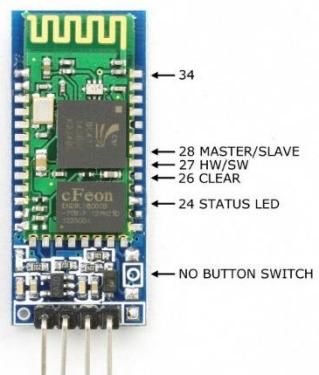
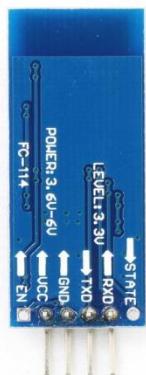
- ✓ 다른 디지털 핀을 RX, TX 핀으로 사용 가능하다.
- ✓ 우노는 모든 디지털 핀이 사용 가능하며, 다른 보드는 일부분만 사용이 가능하다. (소프트웨어 시리얼이 인터럽트를 사용하는데 일부 핀만이 인터럽트를 지원하기 때문이다.)
- ✓ SoftwareSerial 객체를 사용한다.

#### 8.1.2 블루투스 통신 모듈

HC-05 FC-114



HC-06 FC-114



 HC-06 모듈

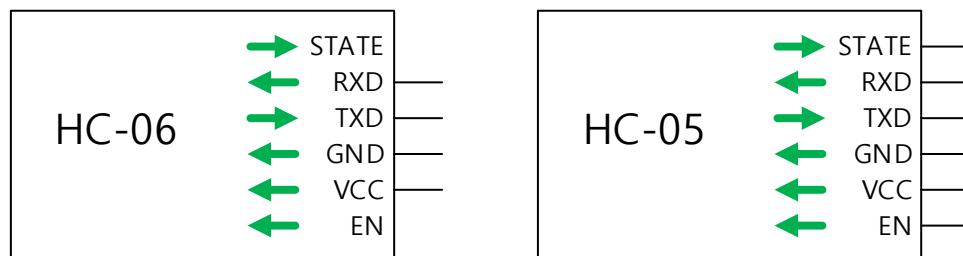
- ✓ 마스터 모듈 (마스터 모듈과 슬레이브 모듈 모두 사용 가능)
- ✓ 핀 배치도 (4 개)
  - ✗ VCC : 전원 핀 (3.6~5 V)
  - ✗ GND : 접지
  - ✗ TXD : 송신 핀, 아두이노 수신 핀 (RX)과 연결
  - ✗ RXD : 수신 핀, 아두이노 송신 핀 (TX)과 연결
- ✓ 시리얼 모니터를 'Both NL & CR' 상태로 설정한다.

 HC-05 모듈

- ✓ 슬레이브 모듈 (펌웨어 1.7 이상에서는 마스터 모드로 설정 가능)
- ✓ 핀 배치도 (6 개, HC-06 과 동일)
  - ✗ VCC : 전원 핀 (3.6~5 V)
  - ✗ GND : 접지
  - ✗ TXD : 송신 핀, 아두이노 수신 핀 (RX)과 연결
  - ✗ RXD : 수신 핀, 아두이노 송신 핀 (TX)과 연결
  - ✗ STATE : 연결 상태를 알려주는 핀 (연결 불필요)
  - ✗ EN : AT 명령 모드로 들어가기 위한 핀, VCC 전원을 끊고 EN 핀 옆 버튼 스위치를 누른 상태로 전원을 인가하면 AT 명령 모드로 들어간다.
- ✓ Baud rate 가 내부적으로 38400 으로 되어 있다. – 펌웨어를 업데이트하면 9600 으로 사용 가능하다.
- ✓ 시리얼 모니터를 'Ending line 없음' 상태로 설정한다.

 HC-06, HC-05 핀 배치도 (HC-06 Z40, HC-05 Z40)

- ✓ HC-06 은 핀이 4 개, HC-05 는 핀이 6 개



### 8.1.3 블루투스 통신 모듈 AT 명령어

 HC-06 모듈은 대문자만 유효하고 HC-05 모듈은 소문자도 가능하다.

- ✚ HC-05 모듈은 AT 명령 모드로 들어가기 위해 별도의 조치가 필요하다.
  - ✓ HC-05 AT 모드
    - ✗ Mini AT 모드
    - ✗ Full AT 모드
  - ✓ Mini AT 모드 진입 방법 (버튼 스위치를 이용하는 방법)
    - ✗ 버튼 스위치를 누른 상태에서 VCC 핀을 해제하고 (전원을 빼고) 다시 연결한다. (버튼을 누르면 34 번 핀이 HIGH로 연결된다. 따라서 버튼을 누르는 대신 전원 인가할 때 34 번 핀을 HIGH로 놓아도 된다.)
    - ✗ 빨간 LED 불빛이 2초마다 깜빡이면 AT 명령 모드에 진입 성공을 나타낸다.
  - ✓ Full AT 모드 진입 방법
    - ✗ 특정 명령어는 34 번 핀이 HIGH일 때만 동작한다.
    - ✗ 버튼을 계속 누르면서 명령을 전송하는 것이 불가능하므로 34 번 핀을 5 V나 3.3 V에 연결해 놓고 전원을 인가한 후 계속 34 번 핀에 HIGH를 유지하면 full AT 모드가 된다.



- ✗ Full AT 모드는 통신 속도가 38400에서 동작한다.
- ✓ HC-05 관련 문서 참조
  - ✗ <https://www.gme.cz/data/attachments/dsh.772-148.2.pdf>
  - ✗ <http://www.martyncurrey.com/arduino-with-hc-05-bluetooth-module-at-mode/>
- ✚ HC-06 모듈 AT 명령어
  - ✓ 참고 자료
    - ✗ <http://www.micro4you.com/files/ElecFreaks/Bluetooth%20HC-06.pdf>
  - ✓ 연결 확인
    - ✗ AT

- 출력 : OK
- ✓ 이름 변경
  - ✗ AT+NAME\*\*\*\* (\*\*는 이름, 크기 무관)
    - 출력 : OKsetname, (default = HC-06)
- ✓ PIN 번호 변경
  - ✗ AT+PIN#### (###는 번호, 4자리 숫자)
    - 출력 : OKsetPIN, (default = 1234)
- ✓ 통신 속도 변경
  - ✗ AT+BAUD\* (\*는 숫자)
    - 출력 : \*가 4 이면 OK9600

No	Baud rate	No	Baud rate
1	1200	5	19200
2	2400	6	38400
3	4800	7	57600
4	9600	8	115200

- ✓ 버전 확인
  - ✗ AT+VERSION
    - 출력 : OKlinvorV1.8
- ✓ 마스터 모드 설정
  - ✗ AT+ROLE=M
    - 출력 : OK+ROLE:M
- ✓ 슬레이브 모드 설정
  - ✗ AT+ROLE=S
    - 출력 : OK+ROLE:S
- HC-05 모듈 AT 명령어
  - ✓ 참고 자료
    - ✗ [http://www.linotux.ch/arduino/HC-0305\\_serial\\_module\\_AT\\_command\\_set\\_201104\\_revised.pdf](http://www.linotux.ch/arduino/HC-0305_serial_module_AT_command_set_201104_revised.pdf)
  - ✓ 연결 확인

- ✖ AT
  - 출력 : OK
- ✓ 버전 확인
  - ✖ AT+VERSION?
    - 출력 : +VERSION:2.0-20100601 OK
- ✓ 이름 변경, 출력 (이름 출력은 full AT 모드에서만 가능)
  - ✖ AT+NAME\*\*\*\* (\*\*는 이름, 크기 무관)
    - 출력 : OK
  - ✖ AT+NAME?
    - 출력 : +NAME:\*\*\*\* OK, (default = HC-05)
- ✓ 비밀 번호 변경, 출력
  - ✖ AT+PSWD=##### (####는 숫자)
    - 출력 : OK
  - ✖ AT+PSWD?
    - 출력 : +PSWD:##### OK, (default = 1234)
- ✓ 모드 변경, 출력
  - ✖ AT+ROLE=1 (마스터 모드로 변경)
    - 출력 : OK
  - ✖ AT+ROLE=0 (슬레이브 모드로 변경)
    - 출력 : OK
  - ✖ AT+ROLE?
    - 출력 : +ROLE:0 OK (슬레이브 모드), +ROLE:1 OK (마스터 모드)
- ✓ Baud rate 변경, 확인
  - ✖ AT+UART=9600,0,0
    - 출력 : OK
  - ✖ AT+UART?
    - 출력 : +UART:9600,0,0 OK
- ✓ 다른 블루투스 모듈과의 연결 모드 변경, 확인
  - ✖ AT+CMODE=1

- 출력 : OK, 아무 모듈과 연결이 가능해 진다. (바인딩이 필요 없다.)
- ✗ AT+CMODE=0
  - 출력 : OK, 특정 주소의 모듈과 연결이 가능해 진다. (바인딩이 필요 하다.)
- ✗ AT+CMODE?
  - 출력 : +CMOD:1 OK, +CMOD:0 OK
- ✓ 다른 블루투스와 바인딩 확인
  - ✗ AT+BIND?
    - 출력 : +BIND: 98d3:61:fd4fcf OK

#### 8.1.4 블루투스 통신용 라이브러리

##### 블루투스 통신용 라이브러리

###### **SoftwareSerial**

- ✓ 다른 디지털 핀을 RX, TX 핀으로 사용할 수 있도록 하는 라이브러리
- ✓ 우노 보드는 모든 디지털 핀 사용 가능, 다른 보드는 일부 핀만 사용 가능하다. (소프트웨어 시리얼이 인터럽트 기능을 사용하는데 일부 핀만이 인터럽트 지원)
- ✓ <https://www.arduino.cc/en/Reference/SoftwareSerial>

##### 관련 함수

**SoftwareSerial(rxPin, txPin);**

- ✗ SoftwareSerial 객체를 생성하는데 사용한다.
- ✗ rxPin : 시리얼 데이터 수신 핀, 블루투스 모듈 TXD와 연결한다.
- ✗ txPin : 시리얼 데이터 송신 핀, 블루투스 RXD와 연결한다.

**.begin(speed);**

- ✗ speed : baud rate (시리얼 통신 속도)

**.available();**

- ✗ 소프트웨어 시리얼 포트로부터 읽을 수 있는 바이트 수를 얻는다.
- ✗ 0이면 데이터가 없고 0이 아니면 데이터가 있다.

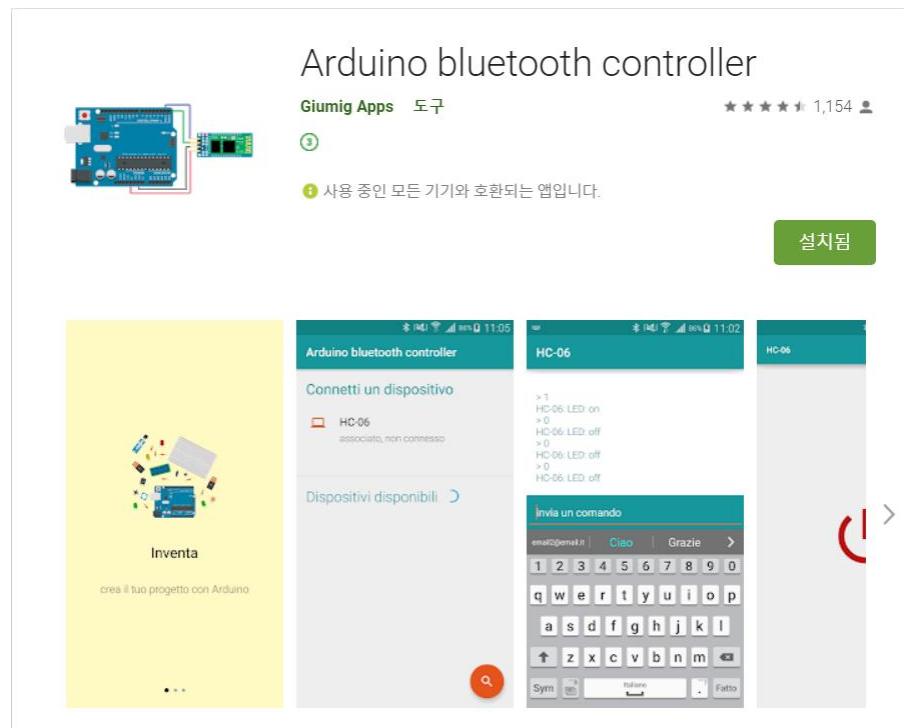
**.print(data);**

- ✗ data를 software serial 포트에 전송 한다.
- ✗ Serial.print()와 동일하다.

- ```
.println(data);  
    ✕ data와 캐리지 리턴을 software serial 포트에 전송 한다.  
    ✕ Serial.println()와 동일하다.  
.read();  
    ✕ software serial 포트에 수신된 데이터를 읽고 버퍼에서 제거한다.  
.write(data);  
    ✕ data를 software serial 포트에 raw byte로 전송한다.  
    ✕ Serial.write()와 동일하다.
```

### 8.1.5 아두이노 블루투스 제어를 위한 스마트 폰 앱

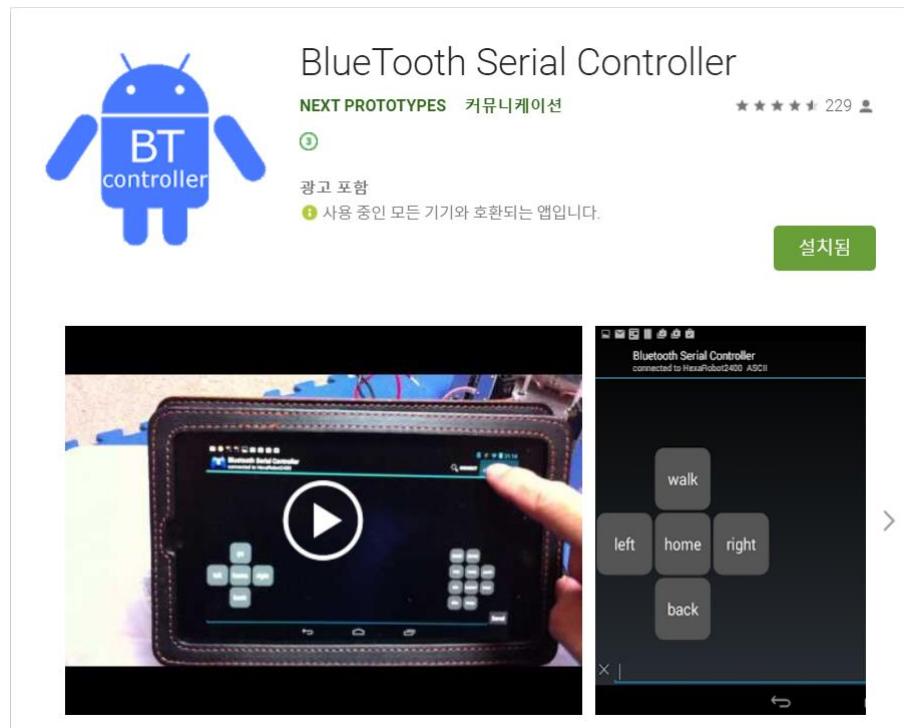
- ✚ 블루투스 통신을 통해 아두이노를 제어할 수 있는 스마트 폰 앱
  - ✓ 메시지 전송
  - ✓ LED ON/OFF 제어
  - ✓ 아두이노 모형 자동차 제어 (조이스틱 모드)
- ✚ 안드로이드 폰 앱
  - ✓ Arduino bluetooth controller
  - ✓ Bluetooth Serial Controller
  - ✓ 구글 플레이에서 Bluetooth controller 로 검색
- ✚ 아이폰 앱
  - ✓ LightBlue Explorer
- ✚ Arduino bluetooth controller



✓ 용도 별 메뉴 선택 가능

- ✗ Controller mode
- ✗ Switch mode
- ✗ Dimmer mode
- ✗ Terminal mode

#### ✚ Bluetooth Serial Controller



- ✓ PREFERENCE에서 용도 별 화면 선택 가능
- ✓ 버튼 배열 및 위치 수정 가능
- ✓ 5 개의 인터페이스 화면 설정 가능

## 8.2 아두이노와 스마트 폰 사이의 블루투스 통신

### 8.2.1 아두이노와 스마트 폰의 블루투스 연결

- ✚ HC-06 모듈을 이용한 아두이노와 스마트 폰의 블루투스 연결

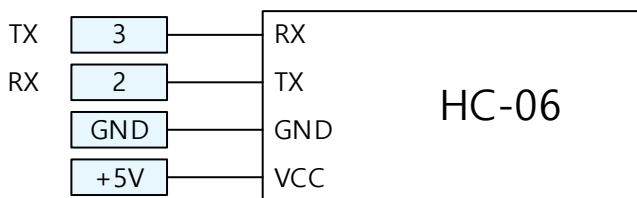
#### 예제 8-2-1

- ✚ 목표

- 아두이노에 HC-06 모듈을 연결하고 블루투스 모듈의 이름과 PIN 번호를 바꾼 후 스마트 폰과 연결한다.

- ✚ 회로도

- HC-06의 RX 단자를 아두이노 3번 핀에, HC-06의 TX 단자를 아두이노 2번 핀에 연결한다. (VCC, GND 도 연결한다.)



- ✚ 스케치 프로그램

- SoftwareSerial.h 사용

```

// Bluetooth Communication
// BT_RX <- A_TX <- D3
// BT_TX -> A_RX -> D2
// Library : SoftrwareSerial.h

#include <SoftwareSerial.h>

// 아두이노 시리얼 데이터 수신 핀, 블루투스 TXD와 연결한다.
const int rxPin = 2;
// 아두이노 시리얼 데이터 송신 핀, 블루투스 RXD와 연결한다.
const int txPin = 3;
// SoftwareSerial 객체 선언
SoftwareSerial BT_Serial(rxPin, txPin);

void setup() {
    // 시리얼과 블루투스 통신 속도를 설정한다.
    Serial.begin(9600);
    BT_Serial.begin(9600);
    Serial.println("Bluetooth HC-06 Module");
}

```

```

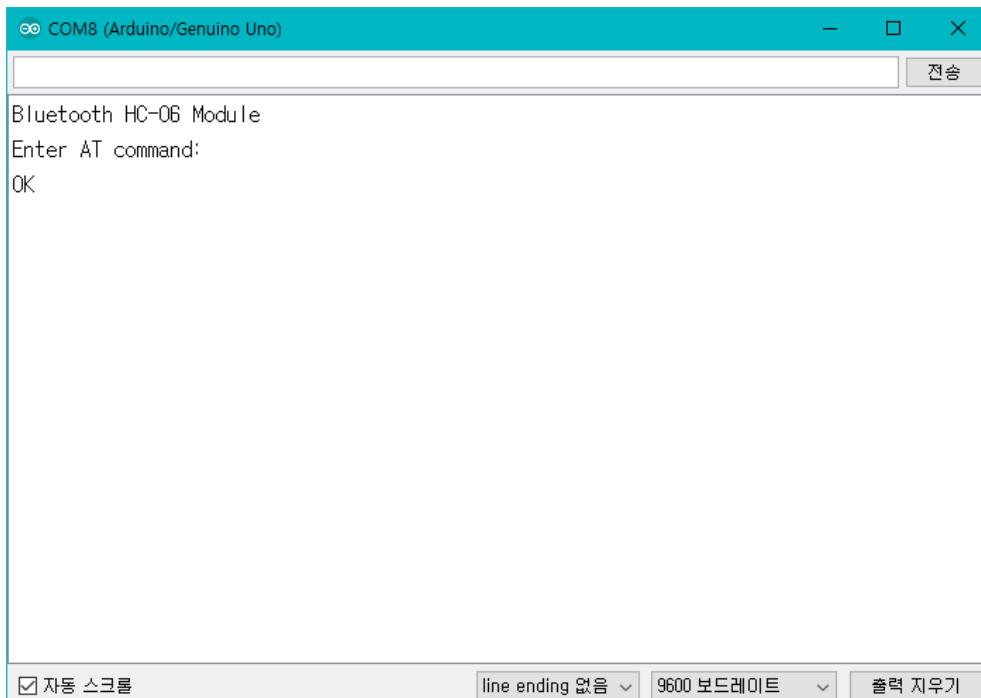
    Serial.println("Enter AT command:");
}

void loop() {
    // 블루투스 통신 버퍼에 데이터가 있으면 데이터를 읽어
    // 시리얼 통신으로 송신한다. (결과가 시리얼 모니터에 나타난다.)
    if (BT_Serial.available()) {
        Serial.write(BT_Serial.read());
    }
    // 시리얼 통신 버퍼에 데이터가 있으면 데이터를 읽어
    // 블루투스 통신으로 송신한다.
    if (Serial.available()) {
        BT_Serial.write(Serial.read());
    }
}

```

#### ➊ 블루투스 연결 확인

- ✓ 아두이노 시리얼 모니터의 옵션을 'No line ending'으로 설정한다.
- ✓ 아두이노 시리얼 모니터 입력 창에 'AT'를 입력하고 엔터를 친다. (또는 전송 버튼을 클릭한다.)
- ✓ 블루투스로부터 'OK'라는 답변이 오고 아두이노 시리얼 모니터에 답변이 표시된다. (블루투스 모듈과 통신 성공)



- ✓ 블루투스 모듈의 이름과 PIN 번호 변경, 버전 확인

✖ AT+NAMEHC-06, AT+PIN1234, AT+VERSION 실행 결과

```

COM8 (Arduino/Genuino Uno)

Bluetooth HC-06 Module
Enter AT command:
OKOKsetnameOKsetPINOKIinvorV1.8

자동 스크롤 line ending 없음 9600 보드레이트 출력 지우기

```

➊ 스마트 폰과 연결 확인

- ✓ 스마트 폰의 블루투스 설정에서 위에서 변경한 블루투스 모듈의 이름을 찾고 핀 번호를 입력하여 연결한다. (모듈 이름 : HC-06)
- ➋ HC-05 모듈을 이용한 아두이노와 스마트 폰의 블루투스 연결
  - ✓ HC-05 모듈을 펌웨어 업데이트하여 baud rate 를 9600 으로 설정한다.
  - ✓ 모듈을 바꾸면 COM 포트 번호도 적절하게 수정해야 한다.

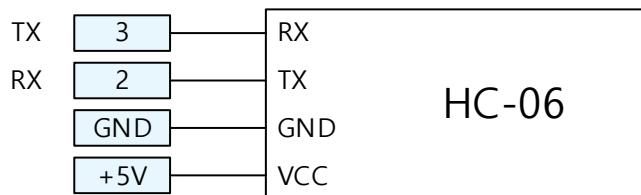
**예제 8-2-2**

➊ 목표

- ✓ 아두이노에 HC-05 모듈을 연결하고 블루투스 모듈의 이름과 PIN 번호를 바꾼 후 스마트 폰과 연결한다.

➋ 회로도

- ✓ HC-05의 RX 단자를 아두이노 3번 핀에, HC-05의 TX 단자를 아두이노 2번 핀에 연결한다. (VCC, GND 도 연결한다. STATE, EN 핀은 연결하지 않는다.)



 스케치 프로그램

- ✓ SoftwareSerial.h 사용

```
// Bluetooth Communication
// BT_RX <- A_TX <- D3
// BT_TX -> A_RX -> D2
// Library : SoftrwareSerial.h

#include <SoftwareSerial.h>

// 아두이노 시리얼 데이터 수신 핀, 블루투스 TXD와 연결한다.
const int rxPin = 2;
// 아두이노 시리얼 데이터 송신 핀, 블루투스 RXD와 연결한다.
const int txPin = 3;
// SoftwareSerial 객체 선언
SoftwareSerial BT_Serial(rxPin, txPin);

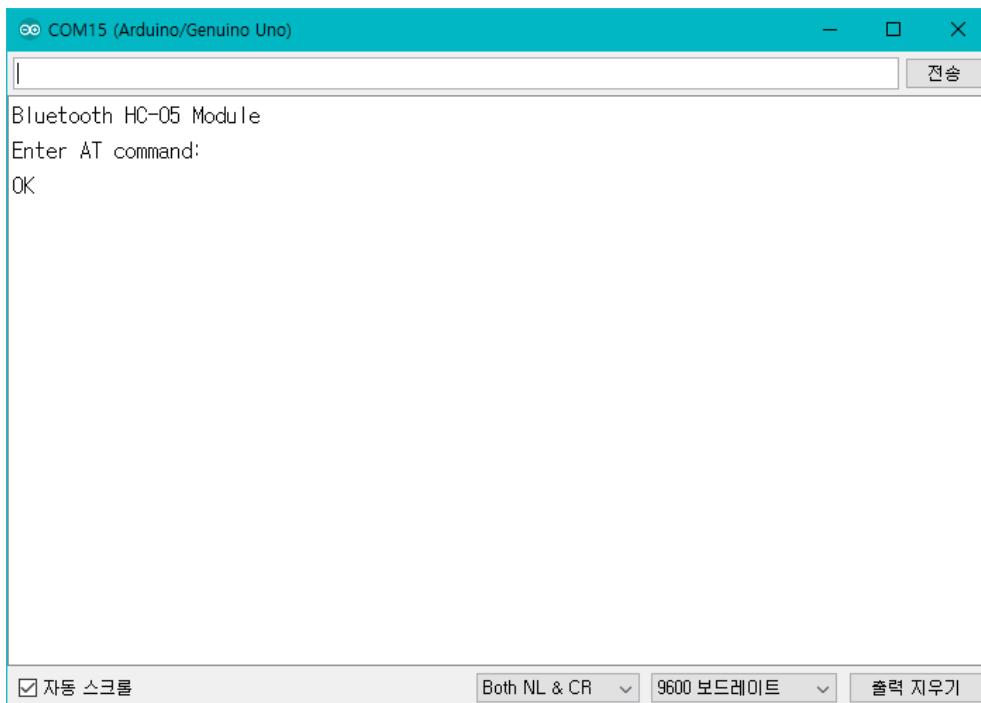
void setup() {
    // 시리얼과 블루투스 통신 속도를 설정한다.
    Serial.begin(9600);
    BT_Serial.begin(38400);
    Serial.println("Bluetooth HC-05 Module");
    Serial.println("Enter AT command:");
}

void loop() {
    // 블루투스 통신 버퍼에 데이터가 있으면 데이터를 읽어
    // 시리얼 통신으로 송신한다. (결과가 시리얼 모니터에 나타난다.)
    if (BT_Serial.available()) {
        Serial.write(BT_Serial.read());
    }
    // 시리얼 통신 버퍼에 데이터가 있으면 데이터를 읽어
    // 블루투스 통신으로 송신한다.
    if (Serial.available()) {
        BT_Serial.write(Serial.read());
    }
}
```

 블루투스 연결 확인

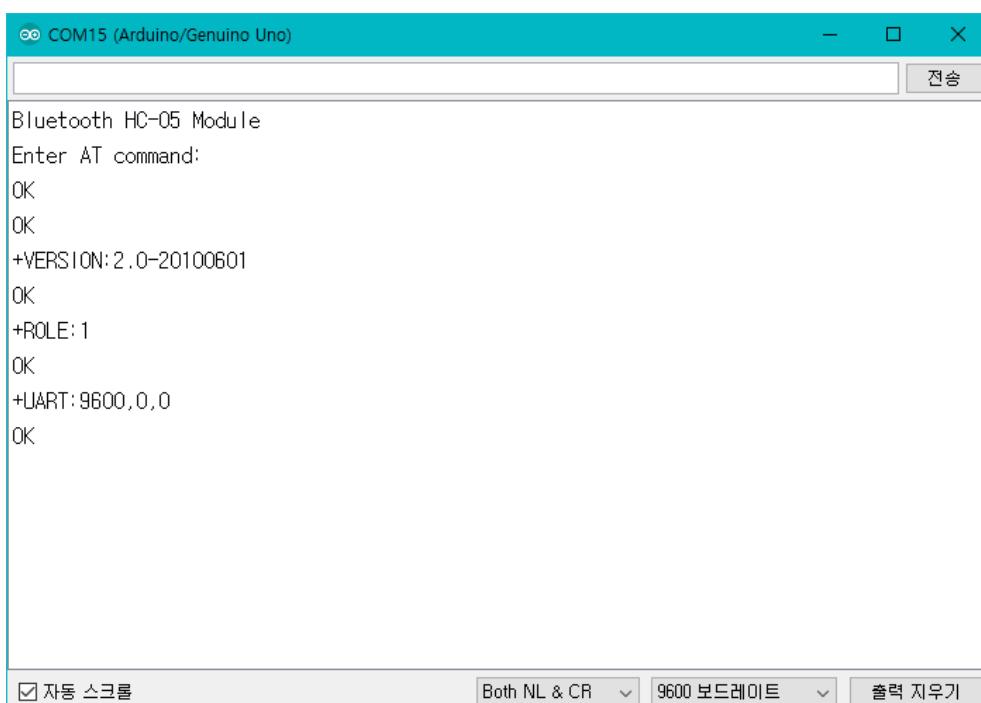
- ✓ AT 명령 모드로 들어간다. (mini AT 모드)
- ✓ 아두이노 시리얼 모니터의 옵션을 'Both NL & CR'로 설정한다.

- ✓ 아두이노 시리얼 모니터 입력 창에 'AT'를 입력하고 엔터를 친다. (또는 전송 버튼을 클릭한다.)
- ✓ 블루투스로부터 'OK'라는 답변이 오고 아두이노 시리얼 모니터에 답변이 표시된다. (블루투스 모듈과 통신 성공)



#### ▶ 블루투스 AT 명령어

- ✓ 블루투스 모듈의 이름 변경, 버전, 마스터/슬레이브 상태, baud rate 확인
- ✗ at+name=HC-05, at+version?, at+role?, at+uart? 실행 결과



스마트 폰과 연결 확인

- ✓ 스마트 폰의 블루투스 설정에서 위에서 변경한 블루투스 모듈의 이름을 찾고 핀 번호를 입력하여 연결한다. (모듈 이름 : HC-05)

### 8.2.2 스마트 폰에서 아두이노로 메시지 전송

HC-06 모듈을 이용하여 스마트 폰에서 아두이노로 메시지 전송

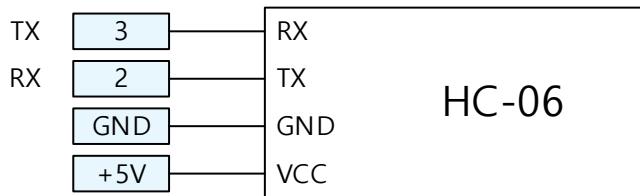
**예제 8-2-3**

목표

- ✓ HC-06 모듈을 이용하여 스마트 폰에서 아두이노로 문자열을 보내고 문자열을 받아 아두이노 시리얼 모니터 창에 표시한다.

회로도

- ✓ HC-06의 RX 단자를 아두이노 2번 핀에, HC-06의 TX 단자를 아두이노 3번 핀에 연결한다. (VCC, GND 도 연결한다.)



스케치 프로그램

- ✓ SoftwareSerial.h 사용

---

```

// Bluetooth Communication
// BT_RX <- A_TX <- D3
// BT_TX -> A_RX -> D2
// Library : SoftrwareSerial.h

#include <SoftwareSerial.h>

// 아두이노 시리얼 데이터 수신 핀, 블루투스 TXD와 연결한다.
const int rxPin = 2;
// 아두이노 시리얼 데이터 송신 핀, 블루투스 RXD와 연결한다.
const int txPin = 3;
// SoftwareSerial 객체 선언
SoftwareSerial BT_Serial(rxPin,txPin);
String myStr = "";

void setup() {

```

```

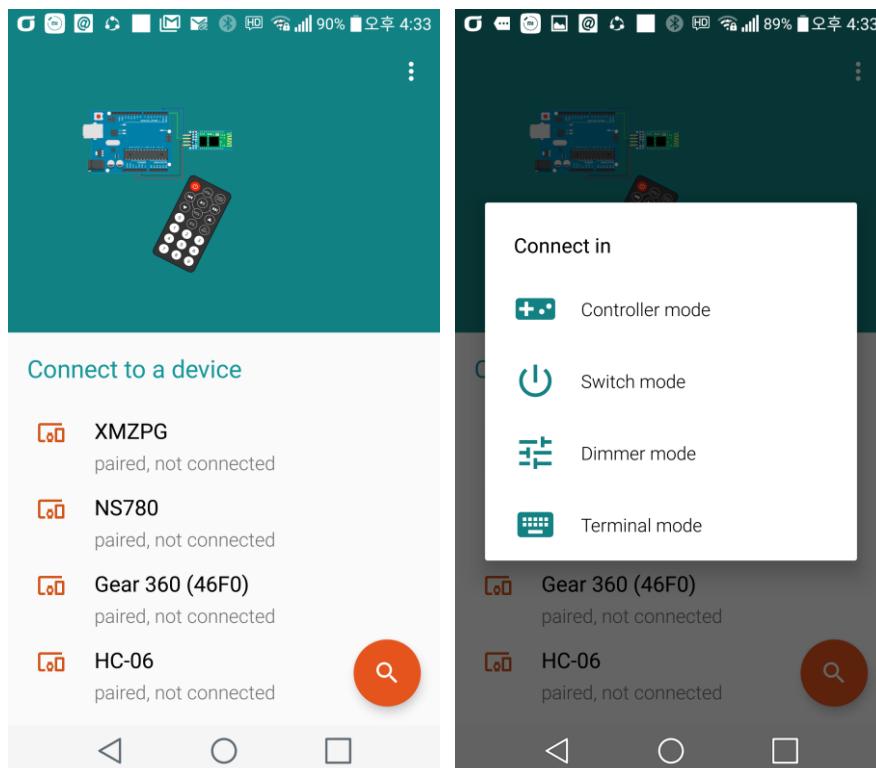
// 시리얼과 블루투스 통신 속도를 설정한다.
Serial.begin(9600);
Serial.println("Enter message in the smartphone:");
BT_Serial.begin(9600);
}

void loop() {
    // 블루투스 통신 버퍼에 데이터가 있으면 데이터(문자)를 읽어
    // 문자열을 만들어 myStr 변수에 저장한다.
    while (BT_Serial.available()) {
        char myChar = (char)BT_Serial.read();
        myStr += myChar;
        delay(5);
    }
    // myStr 변수가 빈 문자열이 아니면 시리얼 모니터에 나타내고
    // 다음을 위해 빈 문자열로 만든다.
    if (!myStr.equals("")) {
        Serial.println("My smartphone sent message : ");
        Serial.println(myStr);
        myStr = "";
    }
}

```

#### Arduino bluetooth controller

- ✓ 블루투스 연결 – 기기를 선택한 후 Terminal mode를 선택한다.

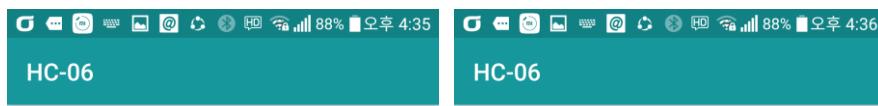


- ✖ 처음 연결 시 핀 번호를 입력한다.

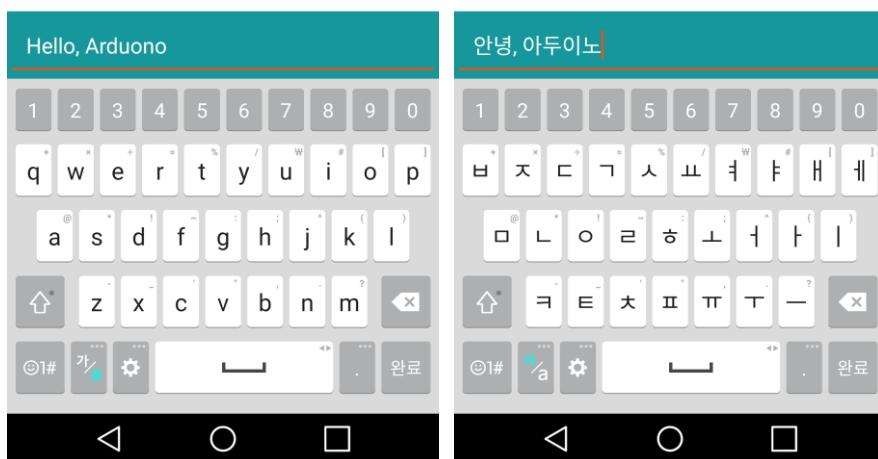


✓ 메시지 입력 창

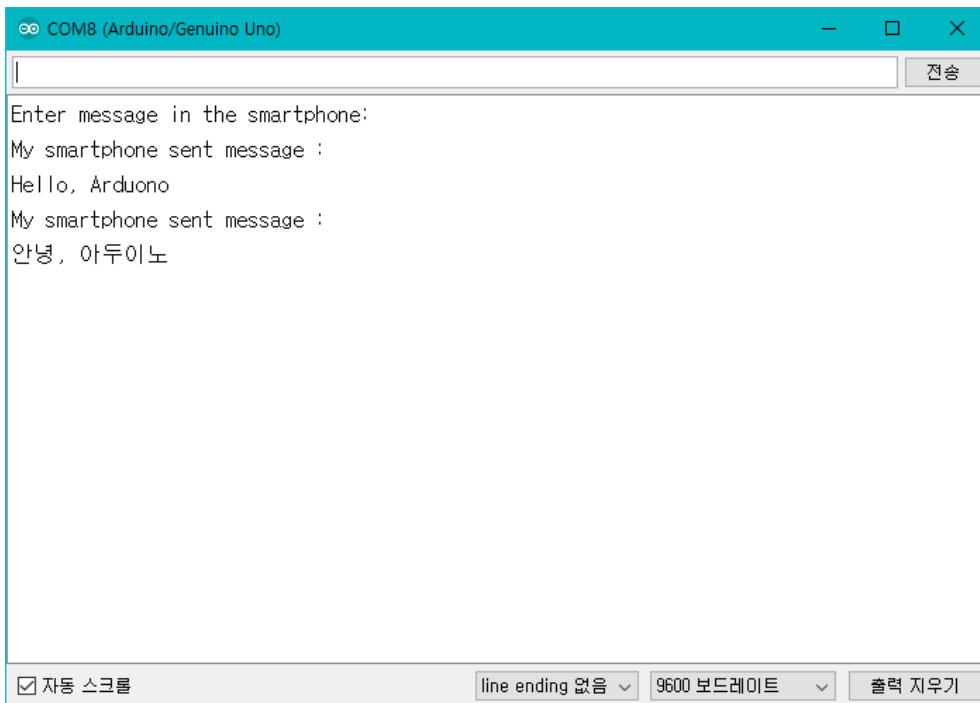
- ✖ Type in command를 클릭한 후 메시지를 입력하고 완료를 클릭한다.



> Hello, Arduono

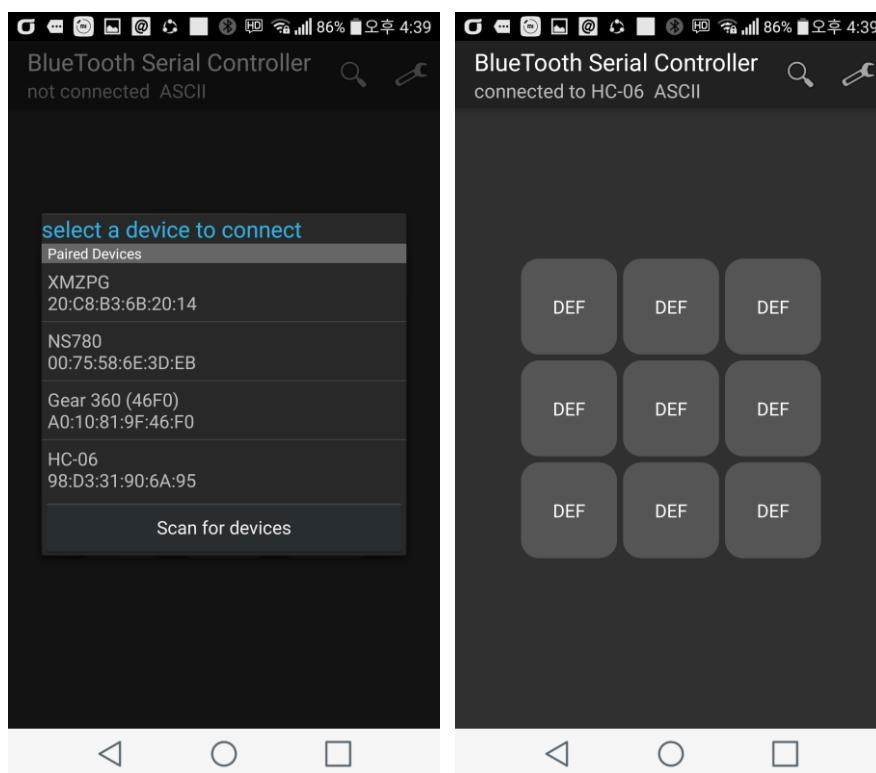


✓ 아두이노 시리얼 모니터

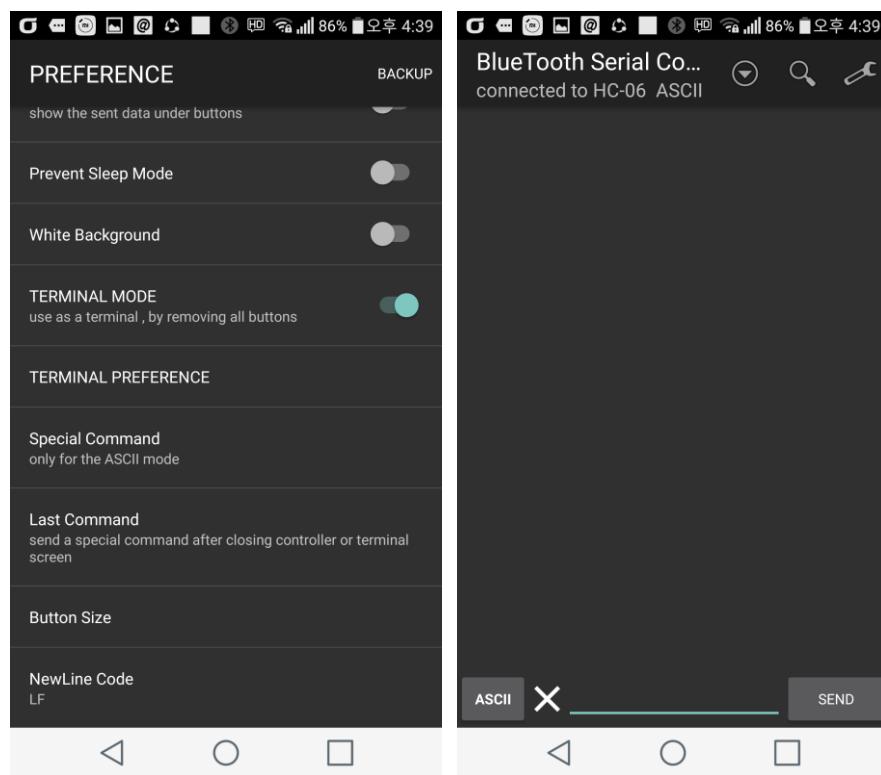


### ➊ BlueTooth Serial Controller

- ✓ 블루투스 연결 – CONNECT(확대경)를 클릭하여 기기(HC-06)를 선택한다.

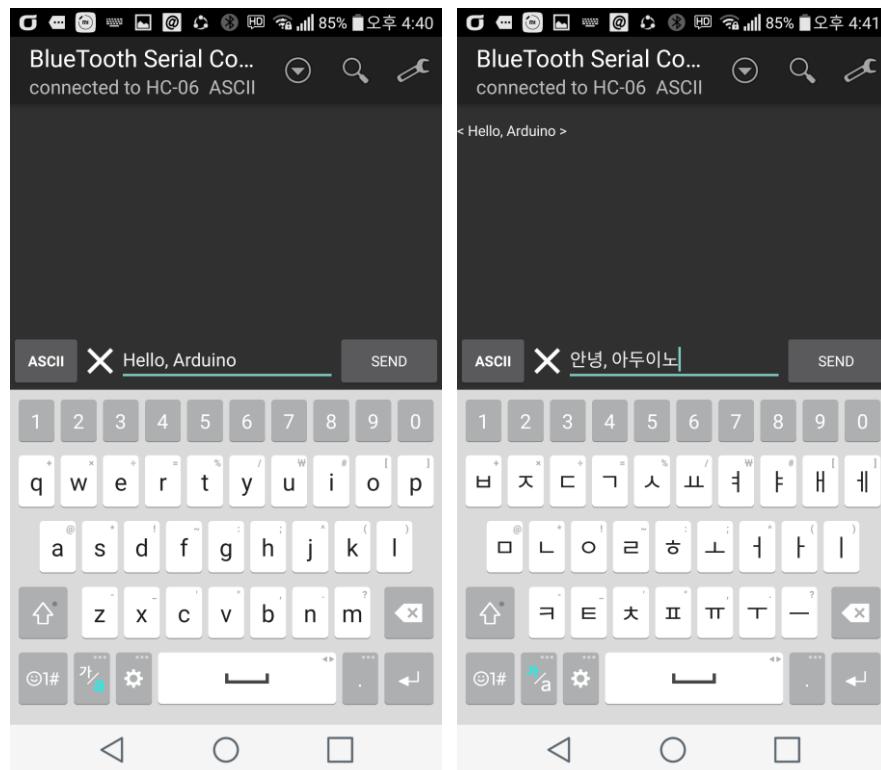


- ✓ PREFERENCE 를 클릭한 후 TERMINAL MODE 를 켠다.

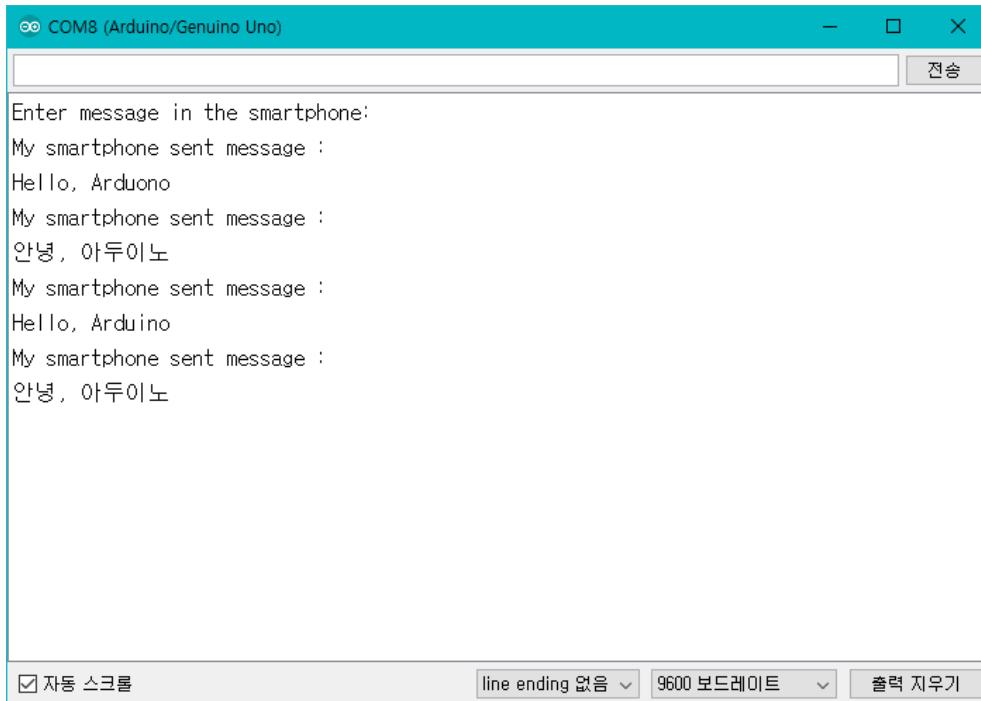


✓ 메시지 입력 창

✗ 메시지를 입력하고 SEND 버튼을 클릭한다.



✓ 아두이노 시리얼 모니터



#### HC-05 모듈을 이용하여 스마트 폰에서 아두이노로 메시지 전송

- ✓ HC-05 모듈을 슬레이브 모드로 바꾼다.
- ✓ AT 명령 모드에서 나온다. (전원을 껐다 켠다.)
- ✓ 스마트 폰에서 블루투스 기기를 검색한다.

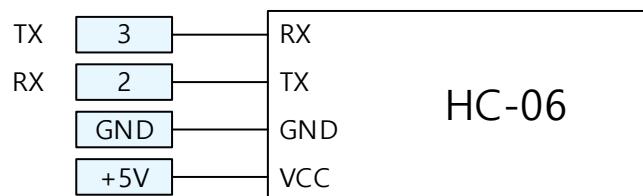
#### 예제 8-2-4

##### 목표

- ✓ HC-05 모듈을 이용하여 스마트 폰에서 아두이노로 문자열을 보내고 문자열을 받아 아두이노 시리얼 모니터 창에 표시한다.

##### 회로도

- ✓ HC-05의 RX 단자를 아두이노 2 번 핀에, HC-05의 TX 단자를 아두이노 3 번 핀에 연결한다. (VCC, GND 도 연결한다. STATE, EN 핀은 연결하지 않는다.)



##### 스케치 프로그램

- ✓ SoftwareSerial.h 사용

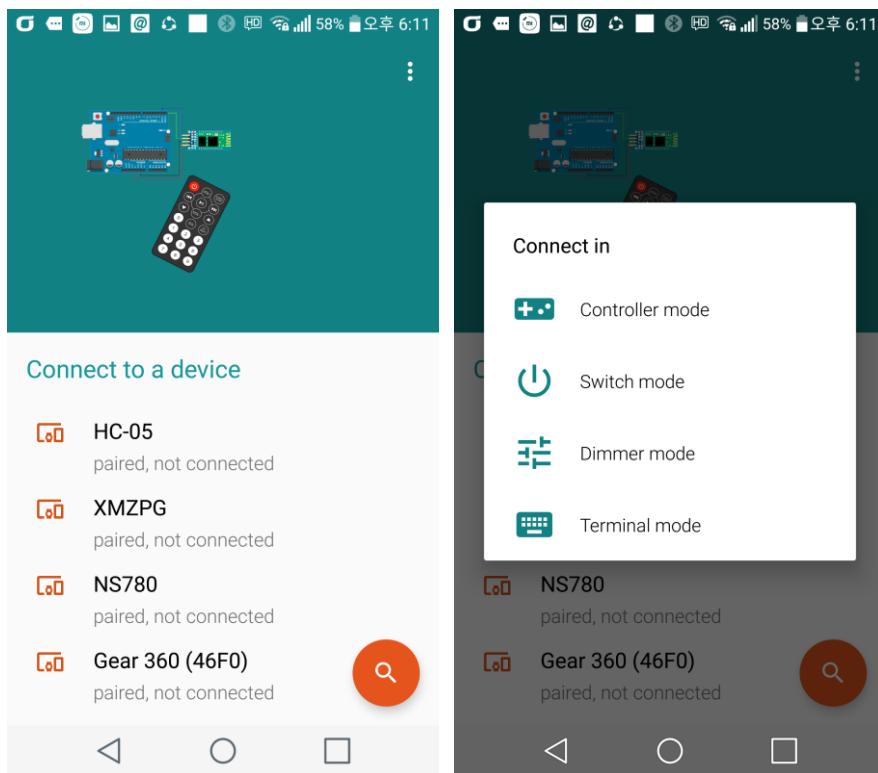
---

```
// Bluetooth Communication
```

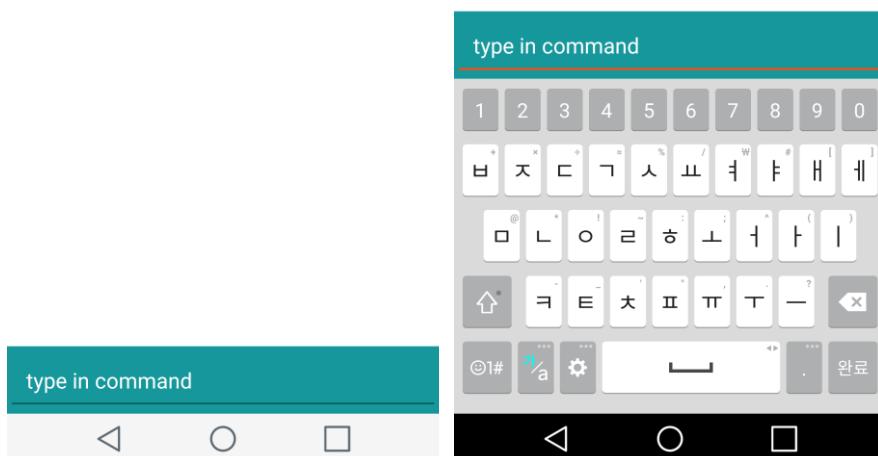
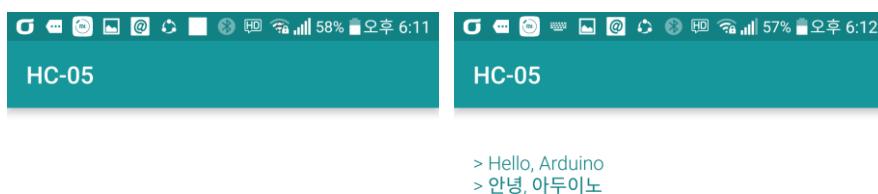
```
// BT_RX <- A_TX <- D3  
// BT_TX -> A_RX -> D2  
// Library : SoftrwareSerial.h  
  
#include <SoftwareSerial.h>  
  
// 아두이노 시리얼 데이터 수신 핀, 블루투스 TXD와 연결한다.  
const int rxPin = 2;  
// 아두이노 시리얼 데이터 송신 핀, 블루투스 RXD와 연결한다.  
const int txPin = 3;  
// SoftwareSerial 객체 선언  
SoftwareSerial BT_Serial(rxPin, txPin);  
  
void setup() {  
    // 시리얼과 블루투스 통신 속도를 설정한다.  
    Serial.begin(9600);  
    Serial.println("Enter message in the smartphone:");  
    BT_Serial.begin(9600);  
}  
  
void loop() {  
    // 블루투스 통신 버퍼에 데이터가 있으면 데이터(문자)를 읽어  
    // 문자열을 만들어 myStr 변수에 저장한다.  
    while (BT_Serial.available()) {  
        char myChar = (char)BT_Serial.read();  
        myStr += myChar;  
        delay(5);  
    }  
    // myStr 변수가 빈 문자열이 아니면 시리얼 모니터에 나타내고  
    // 다음을 위해 빈 문자열로 만든다.  
    if (!myStr.equals("")) {  
        Serial.println("My smartphone sent message : ");  
        Serial.println(myStr);  
        myStr = "";  
    }  
}
```

#### ✚ Arduino bluetooth controller

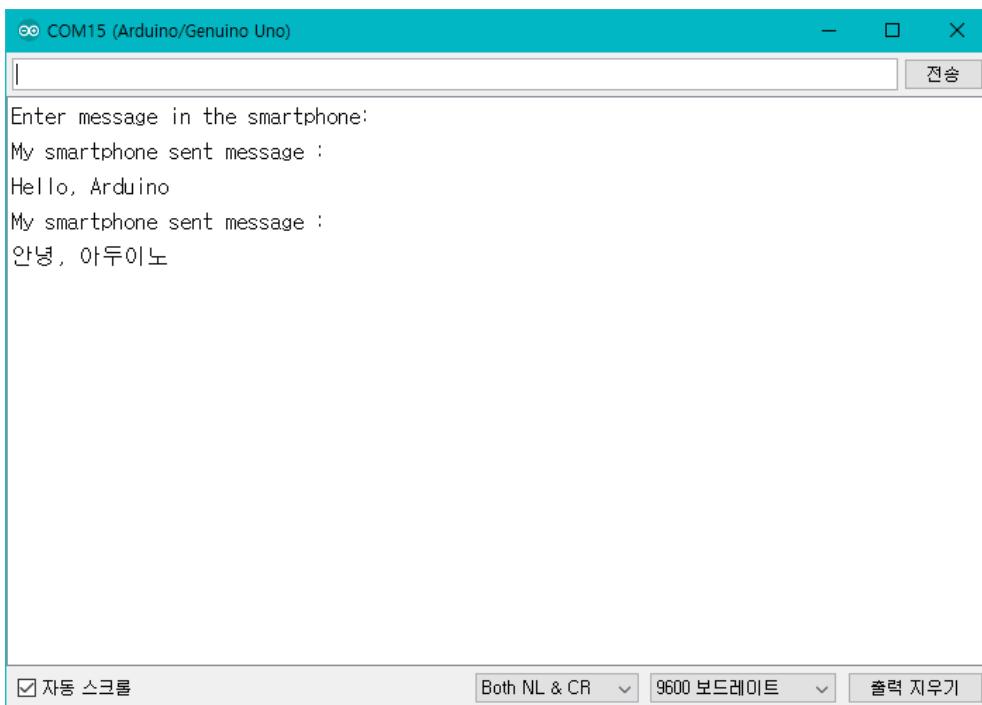
- ✓ 예제 8-2-3 과 마찬가지 과정을 반복한다.
- ✓ 블루투스 연결 – 기기 (HC-05)를 선택한 후 Terminal mode 를 선택한다.



- ✖ 처음 연결 시 핀 번호를 입력한다.
- ✓ 메시지 입력 창
  - ✖ Type in command를 클릭한 후 메시지를 입력하고 완료를 클릭한다.

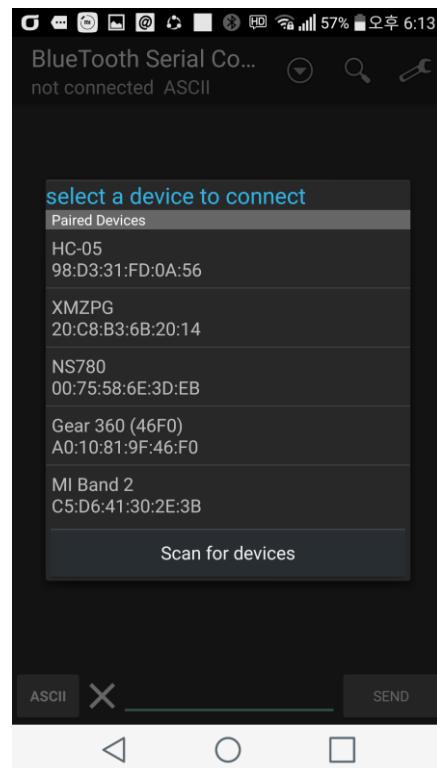


- ✓ 아두이노 시리얼 모니터



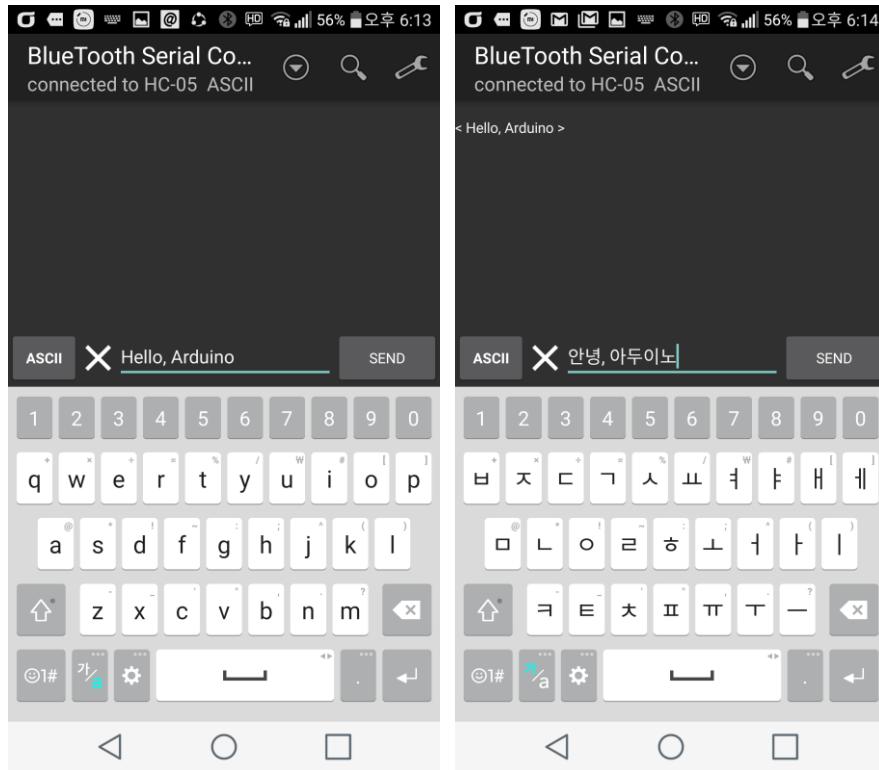
### ➊ BlueTooth Serial Controller

- ✓ 예제 6-2-3 과 마찬가지 과정을 반복한다.
- ✓ 블루투스 연결 – CONNECT(확대경)를 클릭하여 기기(HC-05)를 선택한다.

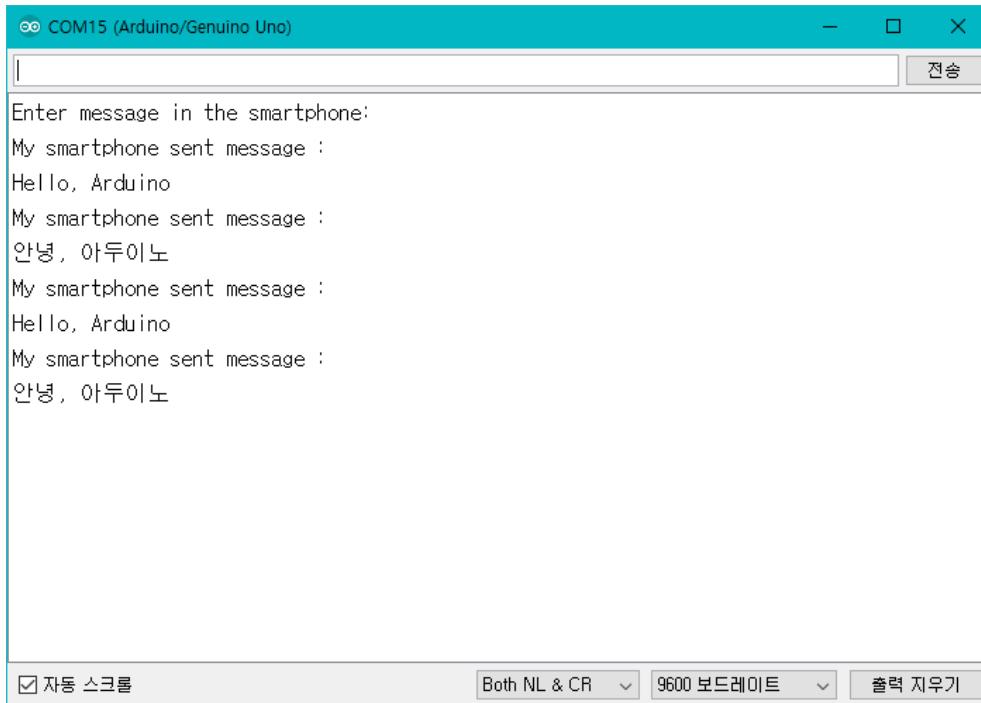


- ✓ TERMINAL MODE 가 아닌 경우 PREFERENCE 를 클릭한 후 TERMINAL MODE 를 켠다.
- ✓ 메시지 입력 창

- ✖ 메시지를 입력하고 SEND 버튼을 클릭한다.



- ✓ 아두이노 시리얼 모니터



### 8.2.3 스마트폰으로 아두이노 제어

- ▶ 스마트 폰의 블루투스 제어 앱에서 버튼을 만들어 아두이노를 제어한다.

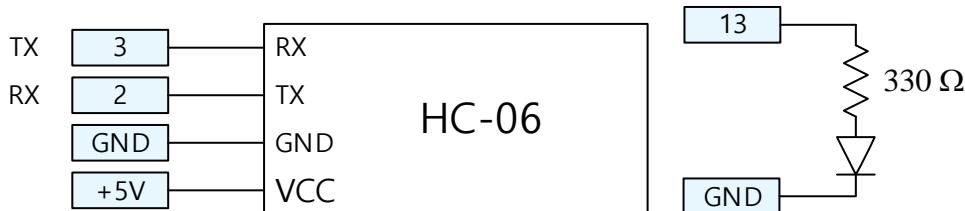
**예제 8-2-5**

### ▣ 목표

- ✓ Arduino bluetooth controller 앱과 BlueTooth Serial Controller 앱에 ON/OFF 버튼을 만들고 HC-06 모듈을 이용하여 블루투스 통신을 통해 LED 를 ON/OFF 제어한다.

### ▣ 회로도

- ✓ HC-06의 RX 단자를 아두이노 2번 핀에, HC-06의 TX 단자를 아두이노 3번 핀에 연결한다. (VCC, GND 도 연결)
- ✓ 디지털 13 번 핀에 LED 연결



### ▣ 스케치 프로그램

- ✓ SoftwareSerial.h 사용

```
// Bluetooth Communication
// BT_RX <- A_TX <- D3
// BT_TX -> A_RX -> D2
// Library : SoftrwareSerial.h

#include <SoftwareSerial.h>

// 아두이노 시리얼 데이터 수신 핀, 블루투스 TXD와 연결한다.
const int rxPin = 2;
// 아두이노 시리얼 데이터 송신 핀, 블루투스 RXD와 연결한다.
const int txPin = 3;
const int ledPin = 13;
// SoftwareSerial 객체 선언
SoftwareSerial BT_Serial(rxPin,txPin);

void setup() {
    // 시리얼과 블루투스 통신 속도를 설정한다.
    Serial.begin(9600);
    Serial.println("Press ON/OFF button in the smartphone");
    BT_Serial.begin(9600);
    pinMode(ledPin, OUTPUT);
}
```

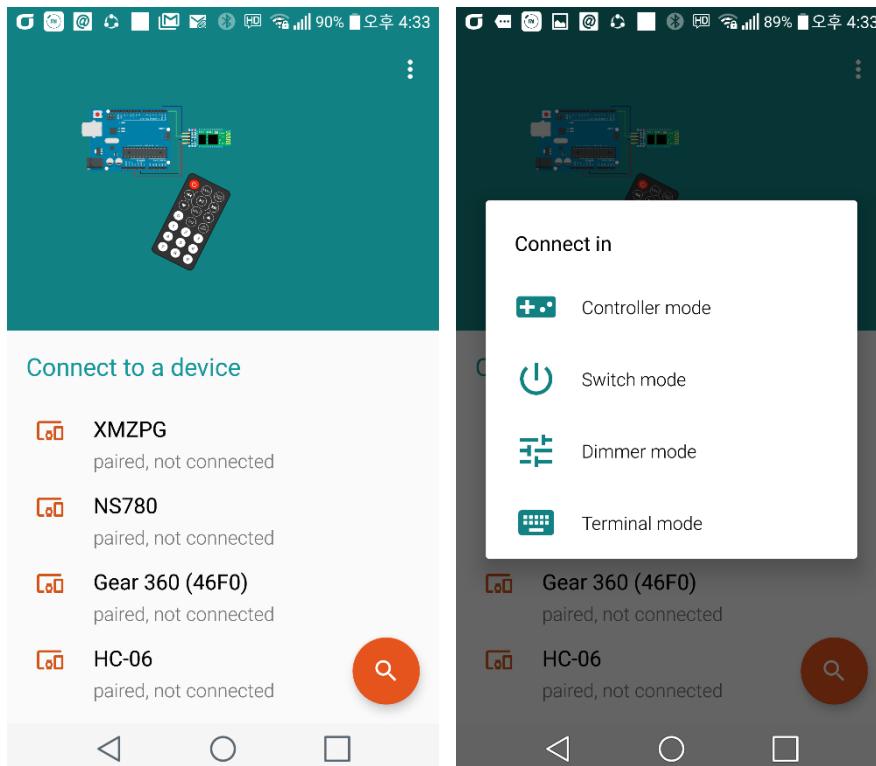
```

void loop() {
    // 블루투스 통신 버퍼에 데이터가 있으면 데이터(문자)를 읽는다.
    if (BT_Serial.available()) {
        char myChar = (char)BT_Serial.read();
        // 읽은 문자가 1이면 LED를 켠다.
        if (myChar == '1') {
            digitalWrite(ledPin, HIGH);
            Serial.println("LED is ON");
        }
        // 읽은 문자가 0이면 LED를 끈다.
        else if (myChar == '0') {
            digitalWrite(ledPin, LOW);
            Serial.println("LED is OFF");
        }
    }
}

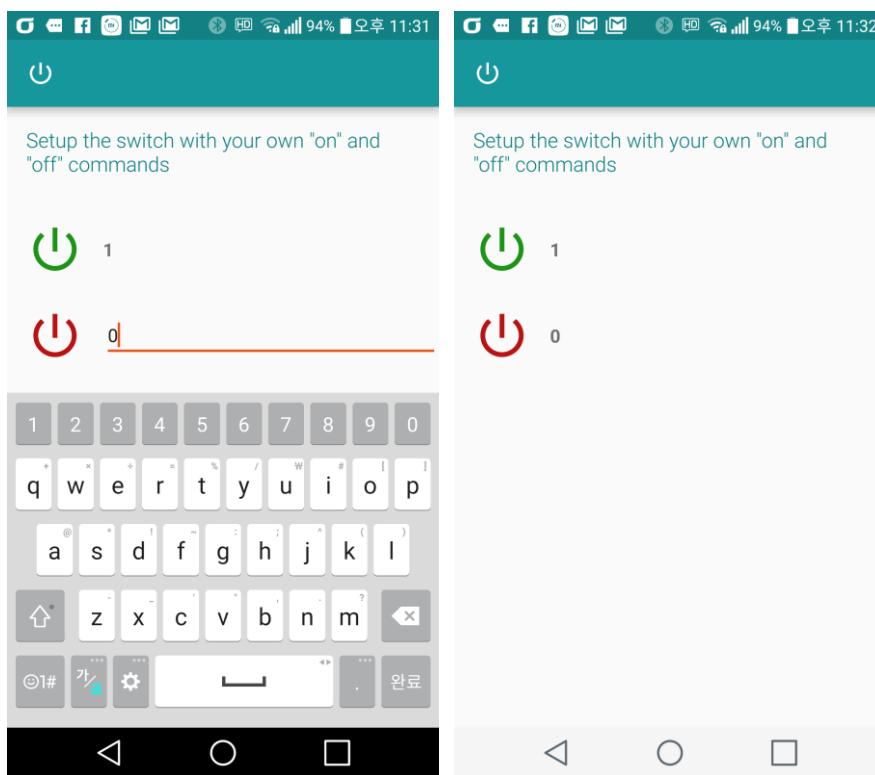
```

#### Arduino bluetooth controller

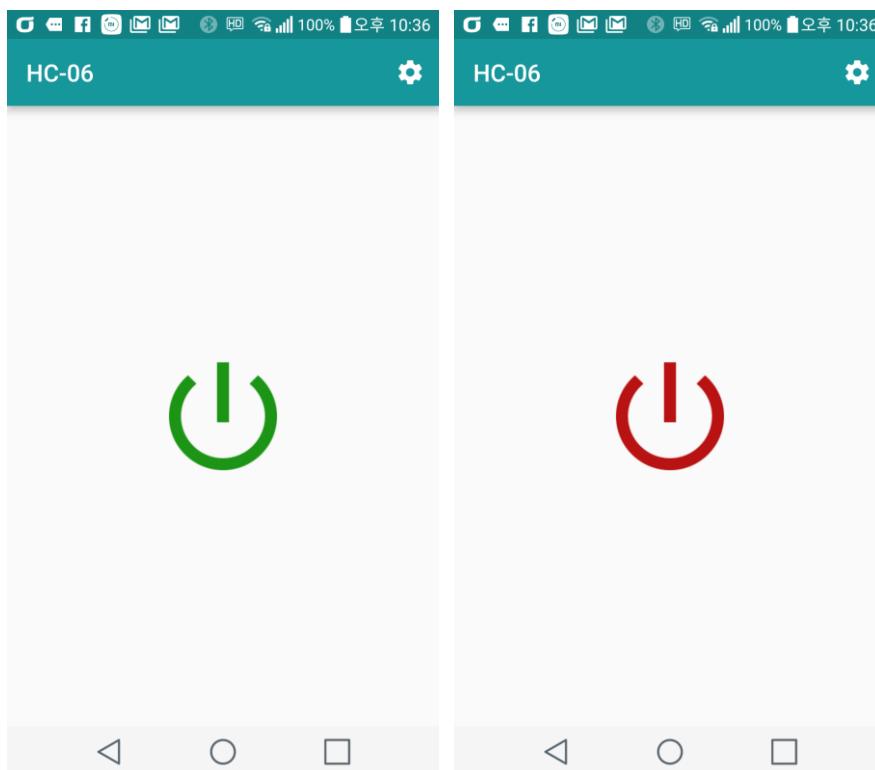
- ✓ 블루투스 연결 – 기기(HC-06)를 선택한 후 Switch mode 를 선택한다.



- ✓ 설정을 클릭하여 ON/OFF 버튼에 명령 값을 할당한다.
- ✗ 스케치 프로그램에서 '1'이 전송되면 LED가 켜지고 '0'이 전송되면 LED 가 꺼지므로 아래와 같이 0과 1을 할당한다.



- ✓ 스케치 프로그램을 실행하고 스마트 폰의 ON/OFF 버튼을 실행하면 LED 가 ON/OFF 된다.

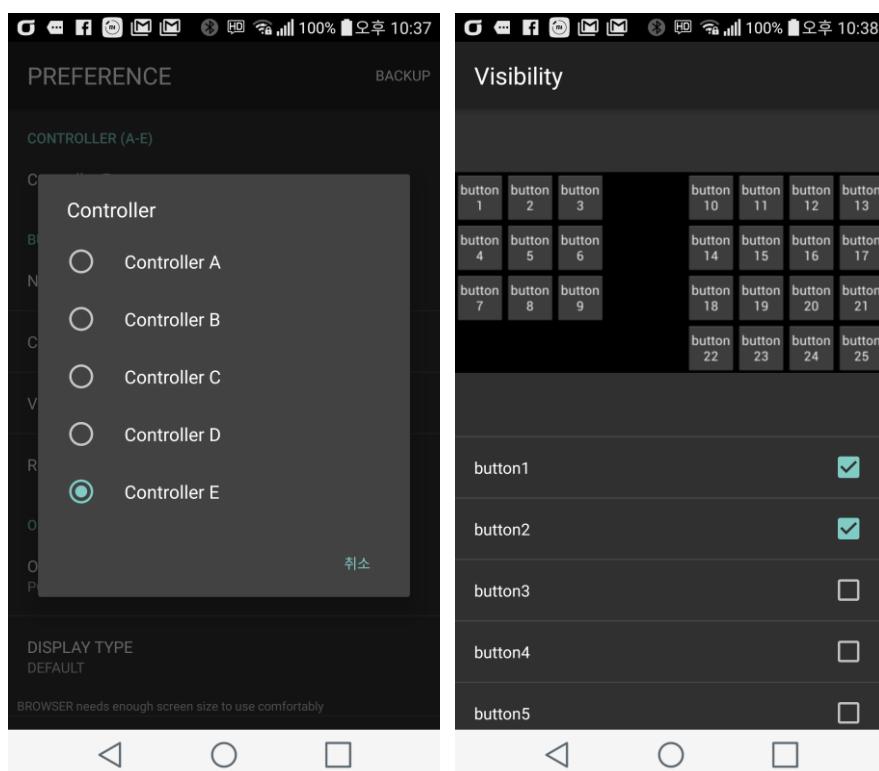


- ✓ ON/OFF 버튼을 누른 후 아두이노 시리얼 모니터



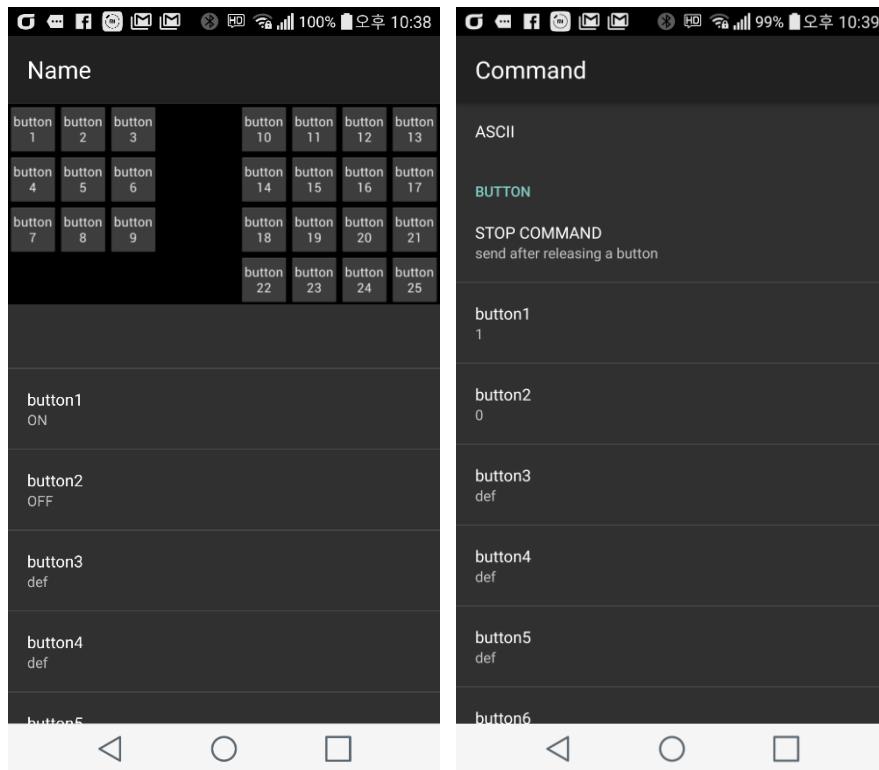
### ➊ BlueTooth Serial Controller

- ✓ 앱의 인터페이스 화면 설정
- ✗ 설정 – PREFERENCE
- ✗ 컨트롤러 선택 – CONTROLLER E
- ✗ 버튼 표시 유무 설정 – Visibility (Button1~Button3 외 모두 off)

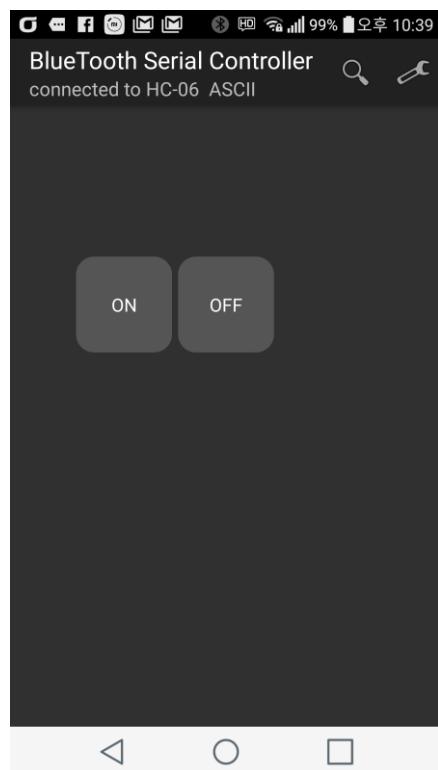


- ✗ 버튼 이름 설정 – Name (ON, OFF)

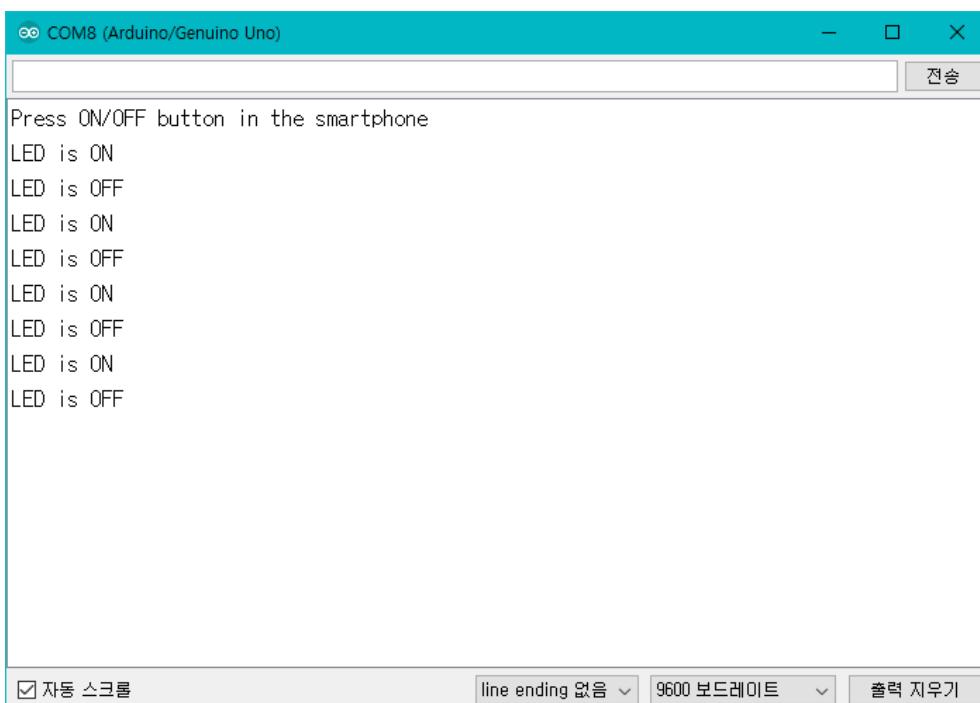
✗ 버튼 명령 설정 – Command (1, 0)



✗ 최종 인터페이스 화면



- ✓ ON/OFF 버튼을 누른 후 아두이노 시리얼 모니터



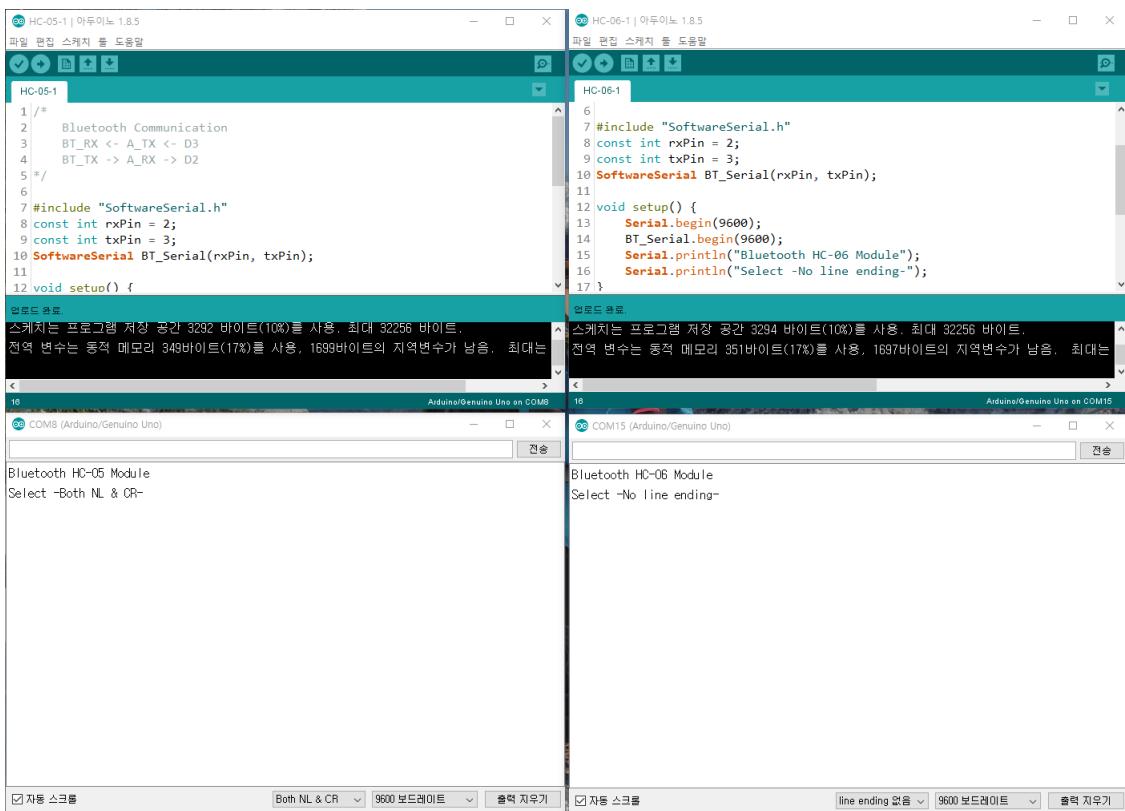
## 연습 문제

### ➊ 연습 문제 8-2-1

- ✓ BlueTooth Serial Controller 앱에 0~9 의 버튼을 만들고 이를 이용하여 블루투스 통신을 통해 7-세그먼트에 원하는 숫자를 나타내는 회로를 구성하고 프로그램을 작성하라. (HC-06 모듈을 사용하고 숫자도 아두이노 시리얼 모니터에 나타내라.)

### 8.3 아두이노 사이의 블루투스 통신

- ✚ 아두이노 사이를 블루투스 통신으로 연결하려면 HC-05 와 HC-06 을 사용한다.
  - ✓ HC-05 는 마스터 모드로 HC-06 은 슬레이브 모드로 사용한다.
- ✚ HC-05 와 HC-06 를 연결하는 방법
  - ✓ HC-05 가 어떤 HC-06 와도 연결 가능한 방법
    - ✗ HC-05 모듈에서 CMODE=1로 설정한다.
    - ✗ Pairing과 binding이 필요 없다. (간단하다.)
    - ✗ <http://www.martyncurrey.com/connecting-2-arduinobysbluetooth-using-a-hc-05-and-a-hc-06-easy-method-using-cmode/>
  - ✓ HC-05 가 특정한 HC-06 와 연결하는 방법
    - ✗ HC-05 모듈에서 CMODE=0로 설정한다.
    - ✗ Pairing과 binding이 필요하다. (복잡하다.)
    - ✗ <http://www.martyncurrey.com/connecting-2-arduinobysbluetooth-using-a-hc-05-and-a-hc-06-pair-bind-and-link/>
- ✚ 컴퓨터 한 대로 2 개의 블루투스 모듈을 제어하는 방법
  - ✓ 아두이노 IDE 를 2 개 연다.
  - ✓ 각각 IDE 에서 블루투스 모듈에 맞는 COM 포트를 설정한다.
  - ✓ 각각의 IDE 에서 각각 다른 시리얼 모니터를 열 수 있다.
  - ✓ 시리얼 모니터 2 개를 연 상태 (COM8, COM15)



### 8.3.1 CMODE를 이용한 블루투스 연결

#### HC-05 설정

- ✓ HC-05 를 AT 모드로 만든다. (전원 제거 후 스위치 누른 상태로 전원을 인가한다.)
- ✓ 시리얼 모니터에서 AT 명령어를 이용하여 다음과 같이 설정한다.
  - ✗ HC-05를 마스터로 설정한다. (AT+ROLE=1)
  - ✗ CMODE를 1로 설정한다. (AT+CMODE=1)
- ✓ HC-05 의 전원을 제거한 후 다시 인가한다. (AT 모드를 해제한다.)
- ✓ HC-05 과 자동으로 연결된다.

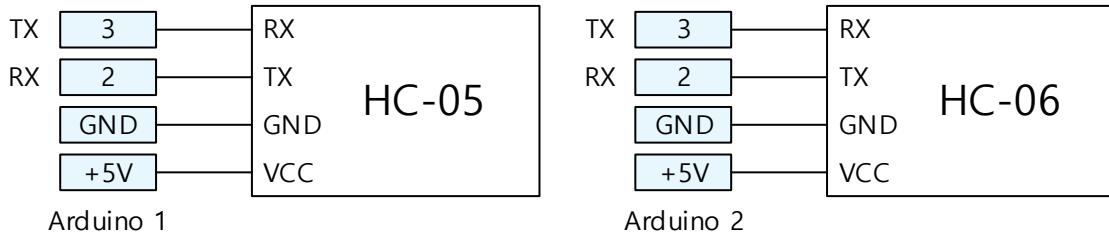
#### 예제 8-3-1

#### 목표

- ✓ HC-05 모듈과 HC-06 모듈을 이용하여 두 아두이노 사이에 블루투스 통신을 이용하여 메시지를 서로 교환한다.
- ✓ 각각의 시리얼 모니터에 문자열을 입력하면 시리얼 통신이 읽어 블루투스에 쓰고 자동으로 다른 블루투스에 전달되어 다른 시리얼 모니터에 쓴다.

### ✚ 회로도

- ✓ HC-05 모듈과 HC-06 모듈의 RX 단자를 아두이노 2 번 핀에, HC-06 의 TX 단자를 아두이노 3 번 핀에 각각 연결한다. (VCC, GND 도 연결)



### ✚ 스케치 프로그램 1 (HC-05 모듈)

- ✓ SoftwareSerial.h 사용

```
// Bluetooth Communication
// BT_RX <- A_TX <- D3
// BT_TX -> A_RX -> D2
// Library : SoftrwareSerial.h

#include <SoftwareSerial.h>

// 아두이노 시리얼 데이터 수신 핀, 블루투스 TXD와 연결한다.
const int rxPin = 2;
// 아두이노 시리얼 데이터 송신 핀, 블루투스 RXD와 연결한다.
const int txPin = 3;
// SoftwareSerial 객체 선언
SoftwareSerial BT_Serial(rxPin, txPin);

void setup() {
    // 시리얼과 블루투스 통신 속도를 설정한다.
    Serial.begin(9600);
    Serial.println("Bluetooth HC-05 Module");
    Serial.println("Select -Both NL & CR-");
    BT_Serial.begin(9600);
}

void loop() {
    // 블루투스 통신 버퍼에 데이터가 있으면 데이터를 읽어
    // 시리얼 통신으로 송신한다. (결과가 시리얼 모니터에 나타난다.)
    if (BT_Serial.available()) {
        Serial.write(BT_Serial.read());
    }
    // 시리얼 통신 버퍼에 데이터가 있으면 데이터를 읽어
}
```

---

```
// 블루투스 통신으로 송신한다.
if (Serial.available()) {
    BT_Serial.write(Serial.read());
}
}
```

---

- ✓ HC-05 모듈에 맞는 COM 포트를 선택하고 시리얼 모니터에서 'Both NL & CR'을 선택한다.

 스케치 프로그램 2 (HC-06 모듈)

- ✓ SoftwareSerial.h 사용
- 

```
// Bluetooth Communication
// BT_RX <- A_TX <- D3
// BT_TX -> A_RX -> D2
// Library : SoftwareSerial.h

#include <SoftwareSerial.h>

// 아두이노 시리얼 데이터 수신 핀, 블루투스 TXD와 연결한다.
const int rxPin = 2;
// 아두이노 시리얼 데이터 송신 핀, 블루투스 RXD와 연결한다.
const int txPin = 3;
// SoftwareSerial 객체 선언
SoftwareSerial BT_Serial(rxPin, txPin);

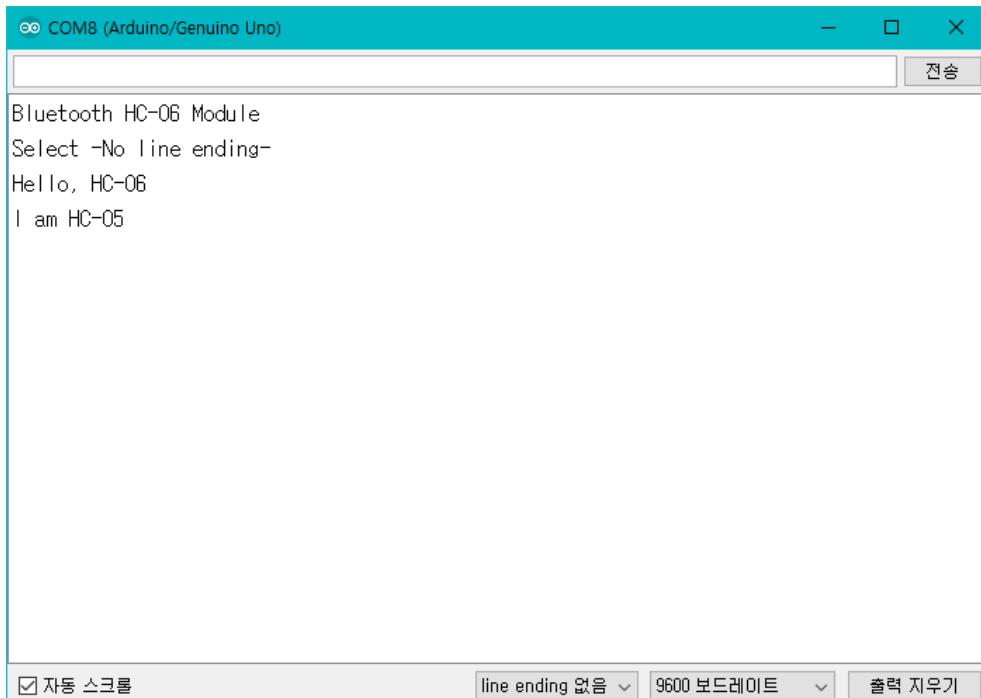
void setup() {
    // 시리얼과 블루투스 통신 속도를 설정한다.
    Serial.begin(9600);
    Serial.println("Bluetooth HC-06 Module");
    Serial.println("Select -No line ending-");
    BT_Serial.begin(9600);
}

void loop() {
    // 블루투스 통신 버퍼에 데이터가 있으면 데이터를 읽어
    // 시리얼 통신으로 송신한다. (결과가 시리얼 모니터에 나타난다.)
    if (BT_Serial.available()) {
        Serial.write(BT_Serial.read());
    }
    // 시리얼 통신 버퍼에 데이터가 있으면 데이터를 읽어
    // 블루투스 통신으로 송신한다.
    if (Serial.available()) {
```

```
    BT_Serial.write(Serial.read());  
}  
}
```

- ✓ HC-06 모듈에 맞는 COM 포트를 선택하고 시리얼 모니터에서 'Line ending 없음'을 선택한다.

- ▶ 시리얼 모니터 1에서 'Hello, HC-06', 'I am HC-05'를 전송했을 때 시리얼 모니터 2



- ▶ 시리얼 모니터 2에서 'Hi, HC-05', 'I am HC-06'을 전송했을 때 시리얼 모니터 1



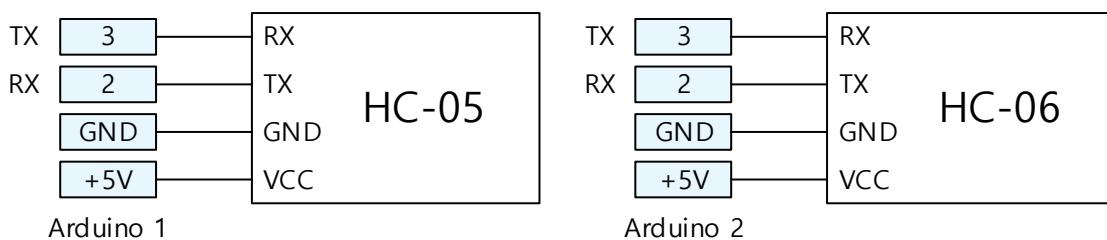
### 예제 8-3-2

#### ▣ 목표

- ✓ HC-05 모듈과 HC-06 모듈을 이용하여 두 아두이노 사이에 블루투스 통신을 이용하여 LED를 ON/OFF 제어한다.
- ✓ 아두이노 1의 시리얼 모니터에 0/1을 입력하고 이에 따라 아두이노 2의 LED를 ON/OFF 제어한다.

#### ▣ 회로도

- ✓ HC-05 모듈과 HC-06 모듈의 RX 단자를 아두이노 2 번 핀에, HC-06의 TX 단자를 아두이노 3 번 핀에 각각 연결한다. (VCC, GND도 연결)
- ✓ 13 번에 내장된 LED를 사용하기 때문에 별도의 회로가 필요 없다.
- ✓ 별도의 LED를 제어하려면 LED와 저항을 추가로 연결해야 한다.



#### ▣ 스케치 프로그램 1 (HC-05 모듈)

- ✓ SoftwareSerial.h 사용

```
// Bluetooth Communication
// BT_RX <- A_TX <- D3
// BT_TX -> A_RX -> D2
// Library : SoftwareSerial.h

#include <SoftwareSerial.h>

// 아두이노 시리얼 데이터 수신 핀, 블루투스 TXD와 연결한다.
const int rxPin = 2;
// 아두이노 시리얼 데이터 송신 핀, 블루투스 RXD와 연결한다.
const int txPin = 3;
// SoftwareSerial 객체 선언
SoftwareSerial BT_Serial(rxPin, txPin);

void setup() {
    // 시리얼과 블루투스 통신 속도를 설정한다.
    Serial.begin(9600);
    Serial.println("Bluetooth HC-05 Module");
    Serial.println("Select -Both NL & CR-");
    BT_Serial.begin(9600);
}

void loop() {
    // 블루투스 통신 버퍼에 데이터가 있으면 데이터를 읽어
    // 시리얼 통신으로 송신한다. (결과가 시리얼 모니터에 나타난다.)
    if (BT_Serial.available()) {
        Serial.write(BT_Serial.read());
    }
    // 시리얼 통신 버퍼에 데이터가 있으면 데이터를 읽어
    // 블루투스 통신으로 송신한다.
    if (Serial.available()) {
        BT_Serial.write(Serial.read());
    }
}
```

- 
- ✓ HC-05 모듈에 맞는 COM 포트를 선택하고 시리얼 모니터에서 'Both NL & CR'을 선택한다.
  - ✚ 스케치 프로그램 2 (HC-06 모듈)
  - ✓ SoftwareSerial.h 사용
- 

```
// Bluetooth Communication
// BT_RX <- A_TX <- D3
```

```

// BT_TX -> A_RX -> D2
// Library : SoftrwareSerial.h

#include <SoftwareSerial.h>

// 아두이노 시리얼 데이터 수신 핀, 블루투스 TXD와 연결한다.
const int rxPin = 2;
// 아두이노 시리얼 데이터 송신 핀, 블루투스 RXD와 연결한다.
const int txPin = 3;
const ledPin = 13;
// SoftwareSerial 객체 선언
SoftwareSerial BT_Serial(rxPin, txPin);

void setup() {
    // 시리얼과 블루투스 통신 속도를 설정한다.
    Serial.begin(9600);
    Serial.println("Bluetooth HC-06 Module");
    Serial.println("Select -No line ending-");
    BT_Serial.begin(9600);
    pinMode(ledPin, OUTPUT);
}

void loop() {
    // 블루투스 통신 버퍼에 데이터가 있으면 데이터(문자)를 읽는다.
    if (BT_Serial.available()) {
        char myChar = BT_Serial.read();
        // 읽은 문자가 1이면 13 번 핀에 내장된 LED를 켠다.
        if (myChar == '1') {
            digitalWrite(ledPin, HIGH);
            Serial.println("LED is ON");
        }
        // 읽은 문자가 0이면 13 번 핀에 내장된 LED를 끈다.
        else if (myChar == '0') {
            digitalWrite(ledPin, LOW);
            Serial.println("LED is OFF");
        }
    }
}

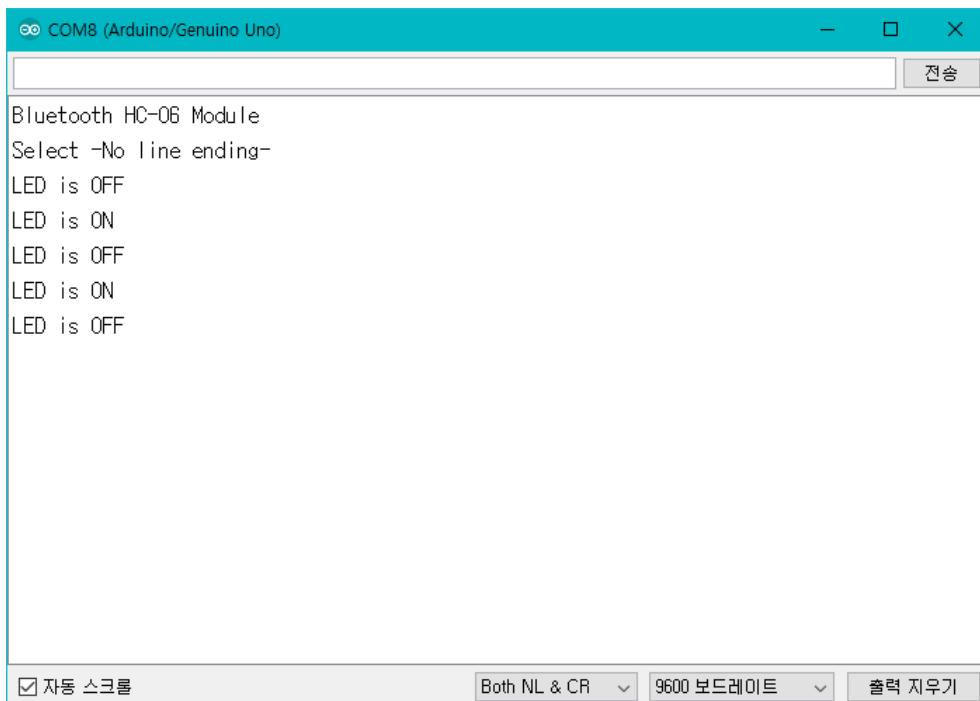
```

- ✓ HC-06 모듈에 맞는 COM 포트를 선택하고 시리얼 모니터에서 'Line ending 없음'을 선택한다.

#### 아두이노 시리얼 모니터 1



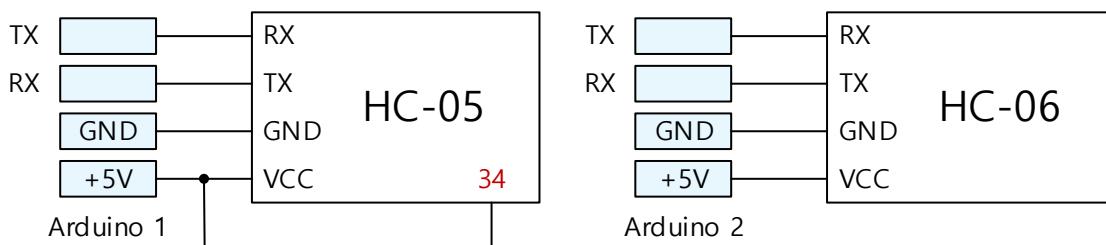
### ▣ 아두이노 시리얼 모니터 2



### 8.3.2 특정한 주소의 블루투스 모듈을 연결하는 방법

#### ▣ 회로도

- ✓ HC-05 를 full AT 모드로 들어가기 위해 34 번 핀을 5 V에 연결한다.



### ✚ 스케치 프로그램 1 (HC-06 모듈)

- ✓ SoftwareSerial.h 사용

```
// Bluetooth Communication
// BT_RX <- A_TX <- D3
// BT_TX -> A_RX -> D2
// Library : SoftwareSerial.h

#include <SoftwareSerial.h>

// 아두이노 시리얼 데이터 수신 핀, 블루투스 TXD와 연결한다.
const int rxPin = 2;
// 아두이노 시리얼 데이터 송신 핀, 블루투스 RXD와 연결한다.
const int txPin = 3;
// SoftwareSerial 객체 선언
SoftwareSerial BT_Serial(rxPin, txPin);

void setup() {
    // 시리얼과 블루투스 통신 속도를 설정한다.
    Serial.begin(9600);
    BT_Serial.begin(9600);
    Serial.println("Bluetooth HC-06 Module");
    Serial.println("Enter AT command:");
}

void loop() {
    // 블루투스 통신 버퍼에 데이터가 있으면 데이터를 읽어
    // 시리얼 통신으로 송신한다. (결과가 시리얼 모니터에 나타난다.)
    if (BT_Serial.available()) {
        Serial.write(BT_Serial.read());
    }
    // 시리얼 통신 버퍼에 데이터가 있으면 데이터를 읽어
    // 블루투스 통신으로 송신한다.
    if (Serial.available()) {
        BT_Serial.write(Serial.read());
    }
}
```

 스케치 프로그램 2 (HC-05 모듈)

## ✓ SoftwareSerial.h 사용

```
// Bluetooth Communication
// BT_RX <- A_TX <- D3
// BT_TX -> A_RX -> D2
// Library : SoftrwareSerial.h

#include <SoftwareSerial.h>

// 아두이노 시리얼 데이터 수신 핀, 블루투스 TXD와 연결한다.
const int rxPin = 2;
// 아두이노 시리얼 데이터 송신 핀, 블루투스 RXD와 연결한다.
const int txPin = 3;
// SoftwareSerial 객체 선언
SoftwareSerial BT_Serial(rxPin, txPin);
Char myChar;

void setup() {
    // 시리얼과 블루투스 통신 속도를 설정한다.
    Serial.begin(9600);
    // Full AT 모드로 들어가기 위해 38400으로 설정한다.
    BT_Serial.begin(38400);
    Serial.println("Bluetooth HC-05 Module");
    Serial.println("Enter AT command:");
}

void loop() {
    // 블루투스 통신 버퍼에 데이터가 있으면 데이터를 읽어
    // 시리얼 통신으로 송신한다. (결과가 시리얼 모니터에 나타난다.)
    if (BT_Serial.available()) {
        myChar = BT_Serial.read();
        Serial.write();
    }
    // 시리얼 통신 버퍼에 데이터가 있으면 데이터를 읽어
    // 블루투스 통신으로 송신한다.
    // 입력한 명령어를 시리얼 모니터에 나타내어 결과와 함께 볼 수 있도록 한다.
    if (Serial.available()) {
        myChar = Serial.read();
        Serial.write(myChar);
        BT_Serial.write(myChar);
    }
}
```

- ✓ Full AT 모드로 들어가기 위해 통신 속도를 38400으로 설정한다.
- ✚ HC-05 와 HC-06 의 설정
  - ✓ HC-05 설정
    - ✗ HC-05를 AT 모드로 만든다. (전원 제거 후 스위치 누른 상태로 전원을 인가한다.)
    - ✗ HC-05를 마스터로 설정한다. (AT+ROLE=1)
    - ✗ CMODE를 0으로 설정한다. (AT+CMODE=0)
    - ✗ 속도를 9600으로 설정한다. (AT+UART=9600)
    - ✗ 비밀번호를 HC-06과 같게 설정한다. (AT+PSWD=1234)
    - ✗ HC-05의 전원을 재 인가한다. (AT 모드 해제한다.)
  - ✓ HC-06 설정
    - ✗ 속도를 9600으로 설정한다. (AT+UART4)
- ✚ 페어링(pairing), 바인딩(binding), link
  - ✓ HC-05 는 full AT 모드에 있어야 한다.
  - ✓ HC-05 와 HC-06 의 통신 속도를 같게 한다.
    - ✗ 9600으로 통일
  - ✓ HC-05 와 HC-06 의 비밀 번호는 같아야 한다.
    - ✗ 1234로 통일
  - ✓ HC-06 의 주소를 찾는다. (full AT 모드에 있어야 한다.)
    - ✗ AT+RMAAD
      - 전에 페어링된 모든 기기를 클리어한다.
    - ✗ AT+ROLE=1
      - HC-05를 마스터 모드로 한다.
    - ✗ AT+RESET
      - HC-05를 리셋 시킨다.
    - ✗ AT+CMODE=0
      - HC-05를 모든 기기와 연결하게 한다.
    - ✗ AT+INQM=0,5,9
      - 9 초 동안 5 개까지 기기를 찾도록 설정한다.

✖ AT+INIT

- SPP 프로파일을 초기화한다. SPP가 이미 활성화 되어 있으면 error(17) 문장이 나온다. (무시하라.)

✖ AN+INQ

- 1 개의 블루투스 기기를 찾았다.
- 찾은 블루투스 기기의 주소는 98D3:61:FD4FCF 이다.
- 2개 이상인 경우 HC-06을 찾기 위해 'AT+RNAME?주소' 를 입력하면 기기 종류를 알 수 있다. (주소 입력 시 콜론을 컴마로 바꾸어 입력한다.)

```

COM15 (Arduino/Genuino Uno)
Enter AT command:
AT+RMAAD
OK
AT+ROLE=1
OK
AT+RESET
OK
AT+CMODE=0
OK
AT+INQM=0,5,9
OK
AT+INIT
OK
AT+INQ
+INQ: 98D3:61:FD4FCF,1F00,7FFF
OK
AT+RNAME?98D3,61,FD4FCF
+RNAME: HC-06
OK

자동 스크롤 Both NL & CR 9600 보드레이트 출력 지우기

```

✓ HC-05 와 HC-06 를 짹짓는다. (pairing 한다.)

- AT+PAIR=<addr>,<timeout>
- AT+PAIR=98D3,61,FD4FCF,9
- 9 초 이내에 페어링 되면 OK, 안 되면 에러가 표시된다.

✓ HC-05 와 HC-06 를 묶는다. (binding 한다.)

- AT+BIND=<addr>

✓ HC-05 가 한 개의 장치와 연결되도록 설정한다.

- At+CMODE=1

✓ HC-06 를 연결한다. (link)

- AT+LINK=<addr>

```

OK
AT+INIT
OK
AT+INQ
+INQ: 98D3:61:FD4FCF,1F00,7FFF
OK
AT+RNAME?
+RNAME: HC-06
OK
AT+PAIR=
ERROR: (7)
AT+PAIR=
ERROR: (7)
OK
AT+BIND=
ERROR: (7)
OK
AT+CMODE=1
OK
AT+LINK=
ERROR: (7)
OK

```

Both NL & CR 9600 보드레이트 출력 지우기

- ✓ 참고 : 주소와 컴마 사이에 빈 칸이 들어가니 예러가 났음을 알 수 있다.
- ✓ HC-05는 2초에 한 번 LED가 점멸하고, HC-06은 LED가 항상 켜 있다.
- ✓ 연결이 완성되면 HC-05는 전원이 인가될 때마다 자동으로 HC-06을 연결한다.
- ✓ 연결이 완성되면 HC-05는 통신 모드로 들어간다. AT 모드가 종료된다.

```

OK
AT+INQ
+INQ: 98D3:61:FD4FCF,1F00,7FFF
OK
AT+RNAME?
+RNAME: HC-06
OK
AT+PAIR=
ERROR: (7)
AT+PAIR=
ERROR: (7)
OK
AT+BIND=
ERROR: (7)
OK
AT+CMODE=1
OK
AT+LINK=
ERROR: (7)
OK
AT
Hello?

```

Both NL & CR 9600 보드레이트 출력 지우기

- ✓ AT를 입력해도 아무 반응이 없고 입력한 문장을 반복해서 나타내 준다.

#### ▶ 연결 테스트

- ✓ HC-05 와 HC-06 의 전원을 제거한다.
- ✓ HC-06 의 전원을 인가한다. – LED 가 초당 5 회로 빠르게 점멸하며 짹 (pair)과 연결을 기다린다.
- ✓ HC-05 의 전원을 인가한다. – HC-05 의 LED 가 처음에 수 초 동안 점멸하다가 정상적인 패턴으로 점멸한다. HC-06 의 LED 는 항상 켜져 있다. (둘이 연결되었다.)
- ✓ 예제 8-3-1 의 스케치 프로그램을 실행하고 두 개의 시리얼 모니터를 열고 서로 메시지를 입력하면 상대방 시리얼 모니터에 메시지가 표시된다.

#### ▶ 연결에 문제가 생길 때

- ✓ 두 개의 모듈이 전원이 켜질 때마다 연결해야 한다.
- ✓ 만일 연결되지 않으면 전원을 껐다 켰다를 반복한다.
- ✓ 가끔 HC-05 가 연결을 시도하는데 막힐 때가 있다. – HC-05 를 리셋하지 않고 HC-06 의 전원을 껐다 켜면 발생한다.

## 연습 문제

#### ▶ 연습 문제 8-3-1

- ✓ 아두이노 1 의 시리얼 모니터에서 0~9 의 숫자를 입력하면 블루투스 통신을 통해 아두이노 2 에 연결된 7-세그먼트에 입력된 숫자를 나타내는 회로를 구성하고 프로그램을 작성하라. (CMODE=0 모드를 이용하라.)

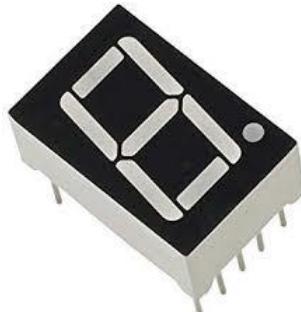
#### ▶ 연습 문제 8-3-2

- ✓ 아두이노 1 의 시리얼 모니터에서 0~9 의 숫자를 입력하면 블루투스 통신을 통해 아두이노 2 에 연결된 7-세그먼트에 입력된 숫자를 나타내는 회로를 구성하고 프로그램을 작성하라. (CMODE=1 모드를 이용하라.)

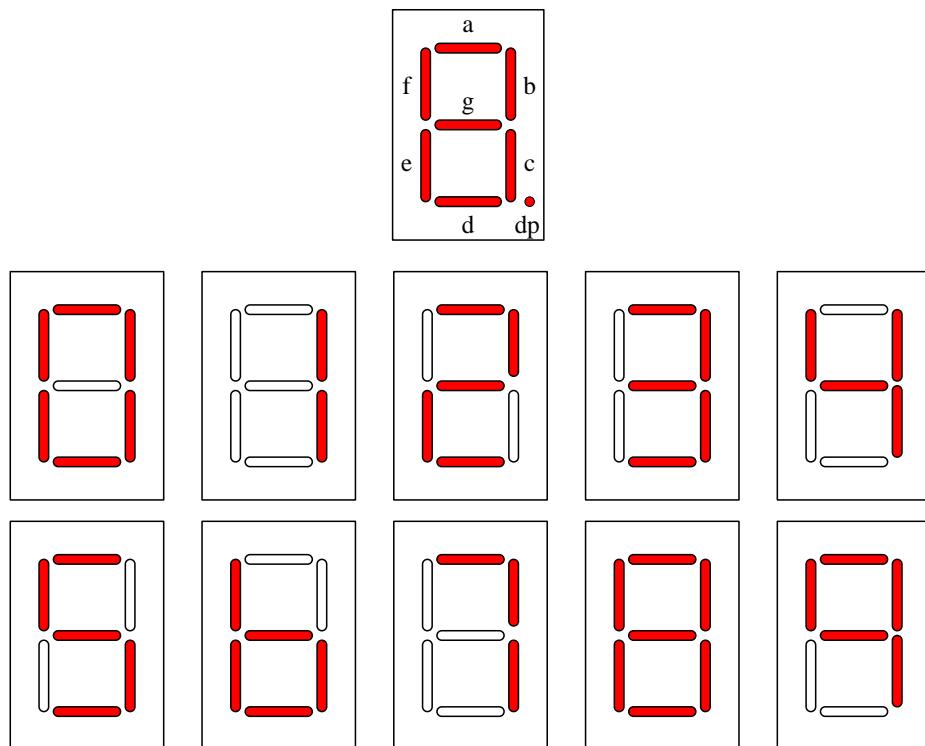
## 9. 디스플레이 제어

### 9.1 7-세그먼트 제어

#### ■ 7-세그먼트 표시기

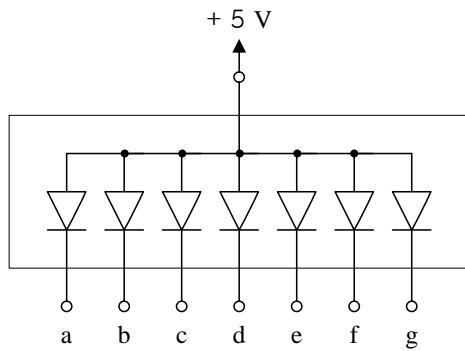


- ✓ 0~9 의 숫자 표시가 가능한 디스플레이
- ✓ 16 진수 숫자 표시도 가능함 (0~9, A~F)
- ✓ 소수점을 포함하여 8 개의 LED 로 구성

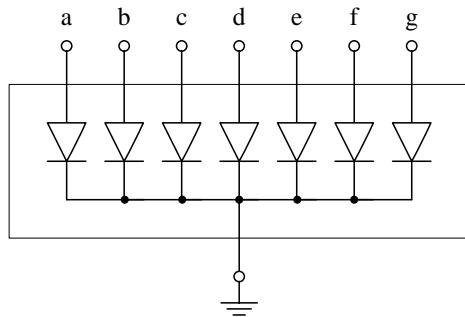


#### ■ 7-세그먼트 표시기의 종류

- ✓ 공통 애노드(anode) 7-세그먼트

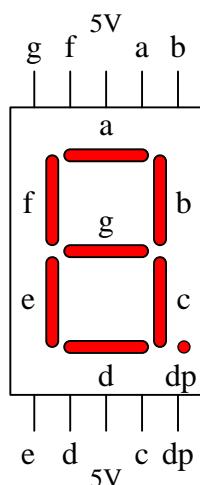


- ✖ LED의 애노드를 공통으로 연결
- ✖ 공통 애노드에 5 V를 인가하고 각 세그먼트에 0 V (LOW)를 인가하면 켜진다.
- ✓ 공통 캐소드(cathode) 7-세그먼트

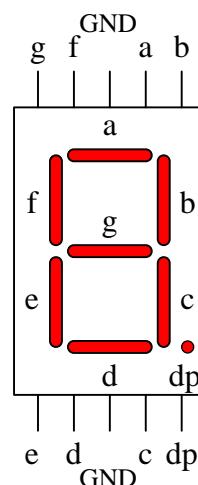


- ✖ LED의 캐소드를 공통으로 연결
- ✖ 공통 캐소드에 0 V를 인가하고 각 세그먼트에 5 V (HIGH)를 인가하면 켜진다.

#### 7-세그먼트의 핀 배치도



공통 애노드



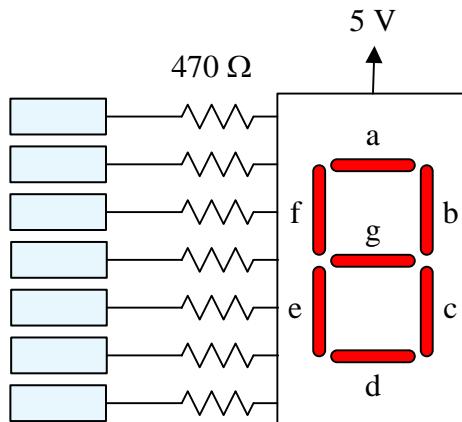
공통 캐소드

✚ 아두이노와 공통 애노드 7-세그먼트의 연결

- ✓ LOW 가 출력될 때 세그먼트가 켜진다.

|   | <i>a</i> | <i>b</i> | <i>c</i> | <i>d</i> | <i>e</i> | <i>f</i> | <i>g</i> |
|---|----------|----------|----------|----------|----------|----------|----------|
| 0 | 0        | 0        | 0        | 0        | 0        | 0        | 1        |
| 1 | 1        | 0        | 0        | 1        | 1        | 1        | 1        |
| 2 | 0        | 0        | 1        | 0        | 0        | 1        | 0        |
| 3 | 0        | 0        | 0        | 0        | 1        | 1        | 0        |
| 4 | 1        | 0        | 0        | 1        | 1        | 0        | 0        |
| 5 | 0        | 1        | 0        | 0        | 1        | 0        | 0        |
| 6 | 1        | 1        | 0        | 0        | 0        | 0        | 1        |
| 7 | 0        | 0        | 0        | 1        | 1        | 1        | 1        |
| 8 | 0        | 0        | 0        | 0        | 0        | 0        | 0        |
| 9 | 0        | 0        | 0        | 1        | 1        | 0        | 0        |

- ✓ 저항 값으로 7-세그먼트의 밝기를 조정

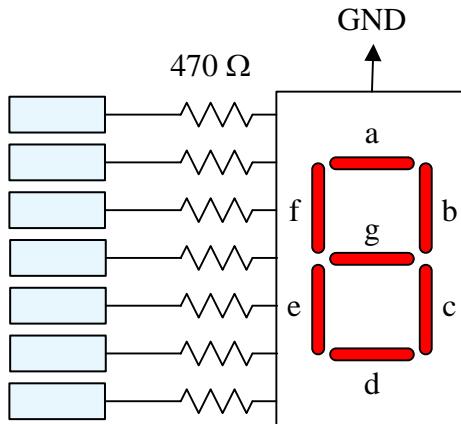


✚ 아두이노와 공통 캐소드 7-세그먼트의 연결

- ✓ HIGH 가 출력될 때 세그먼트가 켜진다.

|   | <i>a</i> | <i>b</i> | <i>c</i> | <i>d</i> | <i>e</i> | <i>f</i> | <i>g</i> |
|---|----------|----------|----------|----------|----------|----------|----------|
| 0 | 1        | 1        | 1        | 1        | 1        | 1        | 0        |
| 1 | 0        | 1        | 1        | 0        | 0        | 0        | 0        |
| 2 | 1        | 1        | 0        | 1        | 1        | 0        | 1        |
| 3 | 1        | 1        | 1        | 1        | 0        | 0        | 1        |
| 4 | 0        | 1        | 1        | 0        | 0        | 1        | 1        |
| 5 | 1        | 0        | 1        | 1        | 0        | 1        | 1        |
| 6 | 0        | 0        | 1        | 1        | 1        | 1        | 0        |
| 7 | 1        | 1        | 1        | 0        | 0        | 0        | 0        |
| 8 | 1        | 1        | 1        | 1        | 1        | 1        | 1        |
| 9 | 1        | 1        | 1        | 0        | 0        | 1        | 1        |

- ✓ 저항 값으로 7-세그먼트의 밝기를 조정



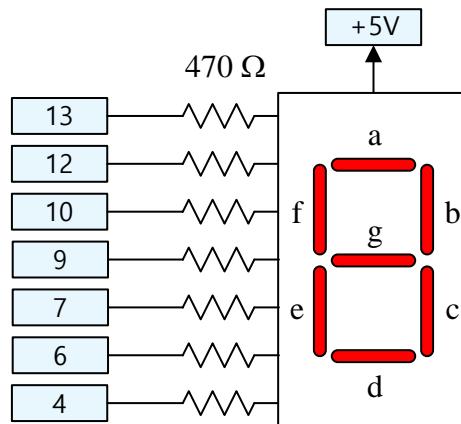
### 예제 9-1-1

#### ▣ 목표

- ✓ 0 ~ 9 의 숫자를 1 Hz 의 속도로 7-세그먼트에 표시 (계속 순환)
- ✓ 7-세그먼트는 공통 애노드를 사용한다.

#### ▣ 회로도

- ✓ 디지털 13, 12, 10, 9, 7, 6, 4 번 핀에 7-세그먼트 a, b, c, d, e, f, g 를 연결한다.
- ✓ 저항  $470\Omega$  은  $330\Omega \sim 680\Omega$  사이의 저항으로 대체할 수 있다. 저항이 커지면 세그먼트의 밝기가 약해진다.



#### ▣ 스케치 프로그램

```
// Display 10 digit in the 7-segment display
// Use common anode 7-segment
// Automatic change of digit display

const int delayTime = 1000;
// 7-세그먼트에 연결되는 아두이노 핀 번호
```

```

int fndPin[] = {13, 12, 10, 9, 7, 6, 4};
// 0 ~ 9 숫자를 나타내는 세그먼트 값 (0이면 세그먼트가 on이다.)
int digits[10][7] = {
    {0,0,0,0,0,0,1}, {1,0,0,1,1,1,1}, {0,0,1,0,0,1,0},
    {0,0,0,0,1,1,0}, {1,0,0,1,1,0,0}, {0,1,0,0,1,0,0},
    {0,1,0,0,0,0,0}, {0,0,0,1,1,1,1}, {0,0,0,0,0,0,0},
    {0,0,0,1,1,0,0}
};

void setup() {
    // 7-세그먼트에 연결되는 아두이노 핀을 디지털 출력으로 설정한다.
    for (int pn = 0; pn < 7; pn++) {
        pinMode(fndPin[pn], OUTPUT);
    }
}

void loop() {
    // 0 ~ 9의 숫자를 7-세그먼트에 나타낸다.
    for (int dn = 0; dn < 10; dn++) {
        displayDigit(dn);
        delay(delayTime);
    }
}

// 숫자 num을 7-세그먼트에 나타내는 함수
void displayDigit(int num) {
    for (int pn = 0; pn < 7; pn++) {
        digitalWrite(fndPin[pn], digits[num][pn]);
    }
}

```

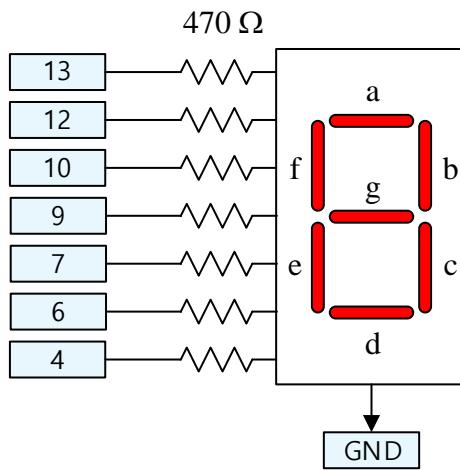
### 예제 9-1-2

#### ▣ 목표

- ✓ 0 ~ 9 의 숫자를 1 Hz 의 속도로 7-세그먼트에 표시 (계속 순환)
- ✓ 7-세그먼트는 공통 캐소드를 사용한다.

#### ▣ 회로도

- ✓ 디지털 13, 12, 10, 9, 7, 6, 4 번 핀에 7-세그먼트 a, b, c, d, e, f, g 를 연결한다.



## 스케치 프로그램

```
// Display 10 digit in the 7-segment display
// Use common cathode 7-segment
// Automatic change of digit display

const int delayTime = 1000;
// 7-세그먼트에 연결되는 아두이노 핀 번호
int fndPin[] = {13, 12, 10, 9, 7, 6, 4};
// 0 ~ 9 숫자를 나타내는 세그먼트 값 (1이면 세그먼트가 on이다.)
int digits[10][7] = {
    {1,1,1,1,1,1,0}, {0,1,1,0,0,0,0}, {1,1,0,1,1,0,1},
    {1,1,1,1,0,0,1}, {0,1,1,0,0,1,1}, {1,0,1,1,0,1,1},
    {1,0,1,1,1,1,1}, {1,1,1,0,0,0,0}, {1,1,1,1,1,1,1},
    {1,1,1,0,0,1,1}
};

void setup() {
    // 7-세그먼트에 연결되는 아두이노 핀을 디지털 출력으로 설정한다.
    for (int pn = 0; pn < 7; pn++) {
        pinMode(fndPin[pn], OUTPUT);
    }
}

void loop() {
    // 0 ~ 9의 숫자를 7-세그먼트에 나타낸다.
    for (int dn = 0; dn < 10; dn++) {
        displayDigit(dn);
        delay(delayTime);
    }
}
```

---

```
// 숫자 num을 7-세그먼트에 나타내는 함수
void displayDigit(int num) {
    for (int pn = 0; pn < 7; pn++) {
        digitalWrite(fndPin[pn], digits[num][pn]);
    }
}
```

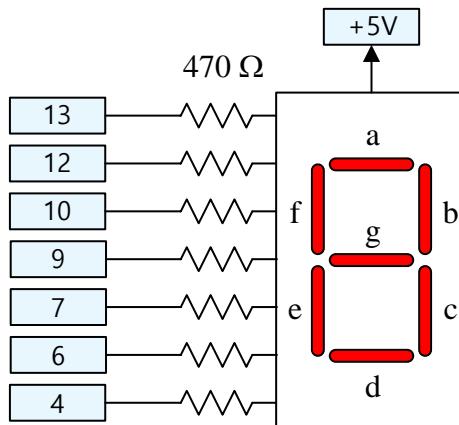
---

**예제 9-1-3****▣ 목표**

- ✓ 아두이노의 시리얼 모니터에서 입력된 숫자를 7-세그먼트에 나타낸다.
- ✓ 7-세그먼트는 공통 애노드를 사용한다.

**▣ 회로도**

- ✓ 디지털 13, 12, 10, 9, 7, 6, 4 번 핀에 7-세그먼트 a, b, c, d, e, f, g 를 연결한다.

**▣ 스케치 프로그램**


---

```
// Display 10 digit in the 7-segment display
// Use common anode 7-segment
// Display number given through serial communication

// 7-세그먼트에 연결되는 아두이노 핀 번호
int fndPin[] = {13, 12, 10, 9, 7, 6, 4};
// 0 ~ 9 숫자를 나타내는 세그먼트 값 (0이면 세그먼트가 off이다.)
int digits[10][7] = {
    {0,0,0,0,0,0,1}, {1,0,0,1,1,1,1}, {0,0,1,0,0,1,0},
    {0,0,0,0,1,1,0}, {1,0,0,1,1,0,0}, {0,1,0,0,1,0,0},
    {0,1,0,0,0,0,0}, {0,0,0,1,1,1,1}, {0,0,0,0,0,0,0},
    {0,0,0,1,1,0,0}
};

void setup() {
```

---

```

Serial.begin(9600);
for (int pn = 0; pn < 7; pn++) {
    // 7-세그먼트에 연결되는 아두이노 핀을 디지털 출력으로 설정한다.
    pinMode(fndPin[pn], OUTPUT);
    // 7-세그먼트를 다 끈다.
    digitalWrite(fndPin[pn], HIGH);
}
}

void loop() {
    // 시리얼 버퍼에 데이터가 있으면 데이터 (문자)를 읽는다.
    if (Serial.available()) {
        char myChar = Serial.read();
        Serial.println(myChar);
        // 시리얼 모니터에 입력한 문자가 0 ~ 9 사이의 숫자인지 검사한다.
        // 읽은 문자의 ASCII 코드 값에서 48을 빼면 읽은 숫자 값이 얻어진다.
        int myNum = (int)myChar - 48;
        // 얻어진 숫자가 0 ~ 9 사이에 있으면 숫자를 7-세그먼트에 나타낸다.
        if (myNum >= 0 && myNum < 10) {
            displayDigit(myNum);
        }
    }
}

// 숫자 num을 7-세그먼트에 나타내는 함수
void displayDigit(int num) {
    for (int pn = 0; pn < 7; pn++) {
        digitalWrite(fndPin[pn], digits[num][pn]);
    }
}

```

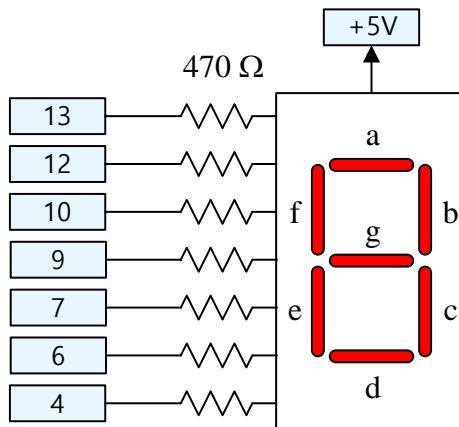
### 예제 9-1-4

#### ▣ 목표

- ✓ 스위치 버튼을 누를 때마다 7-세그먼트의 숫자를 증가시킨다.
- ✓ 7-세그먼트는 공통 애노드를 사용한다.

#### ▣ 회로도

- ✓ 디지털 13, 12, 10, 9, 7, 6, 4 번 핀에 7-세그먼트 a, b, c, d, e, f, g를 연결한다.



### 스케치 프로그램

```
// Display 10 digit in the 7-segment display //
// Manual change of digit display by pressing button//
// Button pin : 2 //
// FND pin : 13, 12, 10, 9, 7, 6, 4 //

// 스위치 상태를 읽는 아두이노 핀 번호
const int btnPin = 2;
const int delayTime = 1000;
const int btnDelayTime = 50;
// 7-세그먼트에 연결되는 아두이노 핀 번호
int fndPin[] = {13, 12, 10, 9, 7, 6, 4};
// 0 ~ 9 숫자를 나타내는 세그먼트 값 (0이면 세그먼트가 on이다.)
int digits[10][7] = {
    {0,0,0,0,0,0,1}, {1,0,0,1,1,1,1}, {0,0,1,0,0,1,0},
    {0,0,0,0,1,1,0}, {1,0,0,1,1,0,0}, {0,1,0,0,1,0,0},
    {0,1,0,0,0,0,0}, {0,0,0,1,1,1,1}, {0,0,0,0,0,0,0},
    {0,0,0,1,1,0,0}
};
int pn;
int dn = 0;
int lastBtnState = 0;

void setup() {
    // 스위치 상태를 읽는 아두이노 핀을 디지털 입력으로 설정한다.
    pinMode(btnPin, INPUT);
    // 7-세그먼트에 연결되는 아두이노 핀을 디지털 출력으로 설정한다.
    for (pn = 0; pn < 7; pn++) {
        pinMode(fndPin[pn], OUTPUT);
    }
    displayDigit(0);
}
```

```
void loop() {
    // 스위치가 눌려지면 숫자 값을 증가시키고 7-세그먼트에 나타낸다.
    if (pressBtn(btnPin) == 1) {
        // 숫자 값을 증가시키고 9를 초과하면 다시 0이 되게 한다.
        dn++;
        dn %= 10;
        displayDigit(dn);
    }
    delay(btnDelayTime);
}

// 숫자 num을 7-세그먼트에 나타내는 함수
void displayDigit(int num) {
    for (pn = 0; pn < 7; pn++) {
        digitalWrite(fndPin[pn], digits[num][pn]);
    }
}

// 스위치가 눌려지면 1을 돌려주는 함수
int pressBtn(int bPin) {
    int btnState = digitalRead(bPin);
    if (btnState == lastBtnState) { return 0; }
    else {
        lastBtnState = btnState;
        if (btnState == 1) { return 1; }
        else { return 0; }
    }
}
```

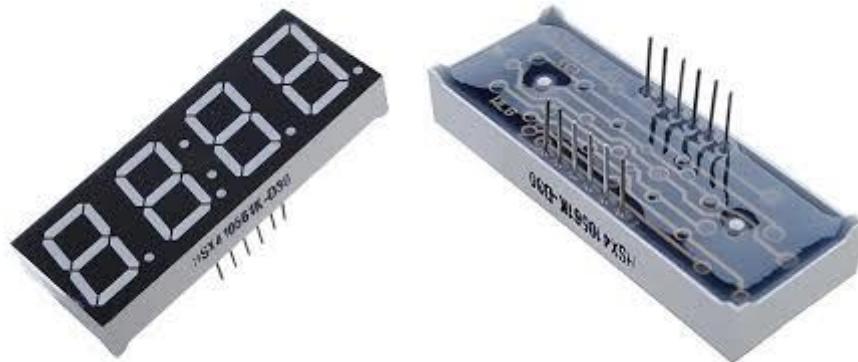
---

## 연습 문제

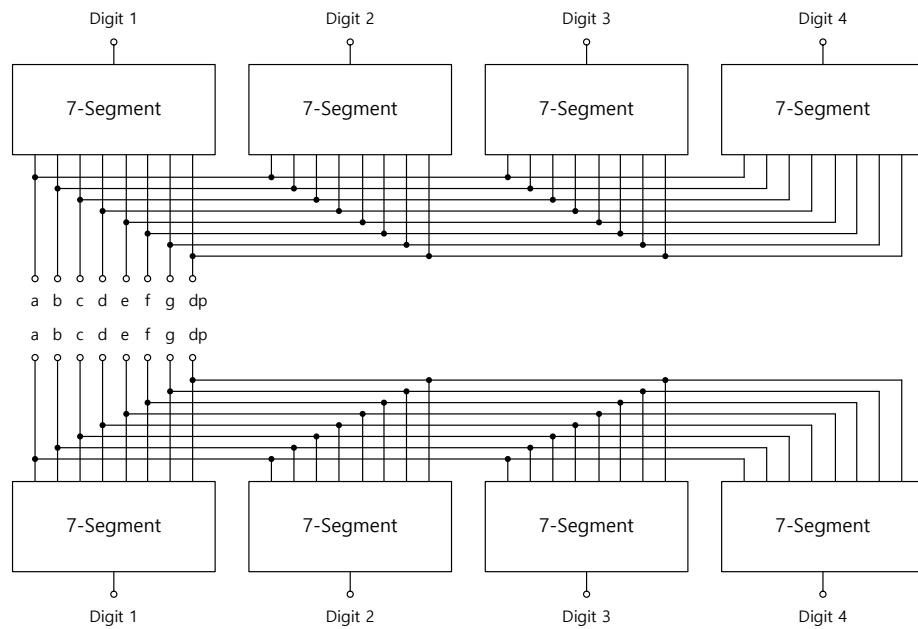
- ➊ 연습 문제 9-1-1
- ➋ 2 개의 스위치 버튼을 누를 때마다 7-세그먼트의 숫자를 증가/감소시키는 회로를 구성하고 프로그램을 작성하라.

## 9.2 4 자리 7-세그먼트 제어

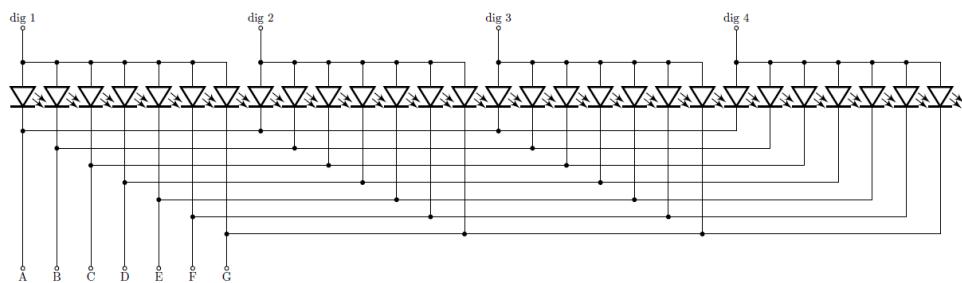
### 9.2.1 4 자리 7-세그먼트 표시기 1



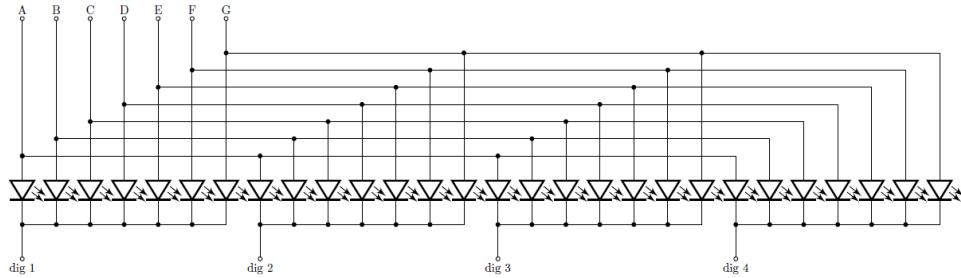
#### ▶ 공통 애노드 형, 공통 캐소드 형 (12 핀)



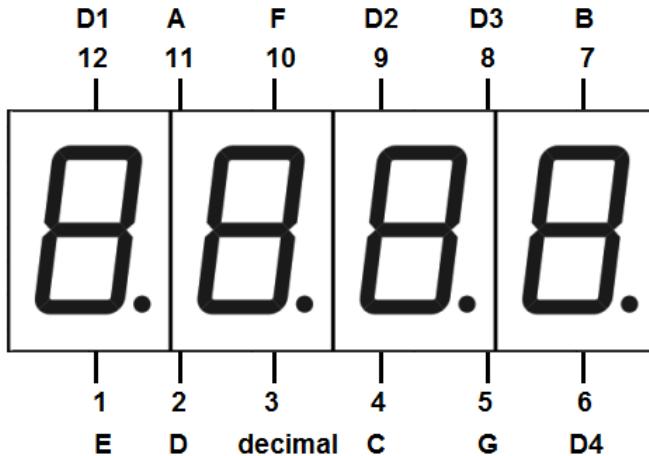
#### ▶ 공통 애노드 형



#### ▶ 공통 캐소드 형



### 핀 배치도



### 라이브러리

#### SevSeg.h

- ✓ 소수점, 콜론, 따옴표 지원
- ✓ 공통 애노드, 공통 캐소드 지원
- ✓ 밝기 조절 가능
- ✓ 8 개 digit 지원 (확장 가능)
- ✓ <https://playground.arduino.cc/Main/SevenSeg>
- ✓ <https://github.com/DeanIsMe/SevSeg>

### 관련 함수

```
.begin(byte hwCfg, byte numDigit, byte dgtPin[], byte segPin[],
bool rstOnSeg=0, bool udtDelay=0, bool leadZero=0);

    ✗ hwCfg : byte, COMMON_ANODE (공통 애노드), COMMON_CATHODE (공
        통 캐소드)
    ✗ numDigit : Digit 수 (자리 수)
    ✗ dgtPin : Digit가 연결되는 아두이노 핀 벡터 (왼쪽부터 오른쪽 순)
    ✗ segPin : 세그먼트가 연결되는 아두이노 8 개 핀 벡터
```

- ✖ rstOnSeg : 전류 제어용 저항을 digit 핀에 연결하면 false, 세그먼트에 연결하면 true, (default=false)
  - ✖ udtDelay : false 권장, (default=false)
  - ✖ leadZero : 앞에 0을 나타내려면 true, (default=false)
- ```
.setNumber(int num, char decPlace=-1, bool hex=0);
.setNumber(float num, char decPlace=-1, bool hex=0);
```
- ✖ num : 숫자, 범위를 벗어나면 ---- 표시
  - ✖ decPlace : 소수점 위치, -1이면 소수점 없음, (default=-1)
  - ✖ hex : true이면 16진수 표시, (default=false)
- ```
.setChars(char str[]);
```
- ✖ str : 문자 열
- ```
.setSegments(byte segs[]);
```
- ✖ segs[] : 세그먼트 데이터 (비트 0 – 세그먼트 a, 비트 1 – 세그먼트 b, ..., 비트 6 – 세그먼트 g)
  - ✖ - 표시 : 1000000, 0x40
  - ✖ 모두 off : 0000000, 0x00
- ```
.brightness(int num)
```
- ✖ num : 0 ~ 100 사이의 값, 일반적으로 90 사용
- ```
.refreshDisplay();
```
- ✖ 데이터를 나타내기 위해 지속적으로 실행시켜야 한다.
- ```
.blank();
```
- ✖ 데이터 표시

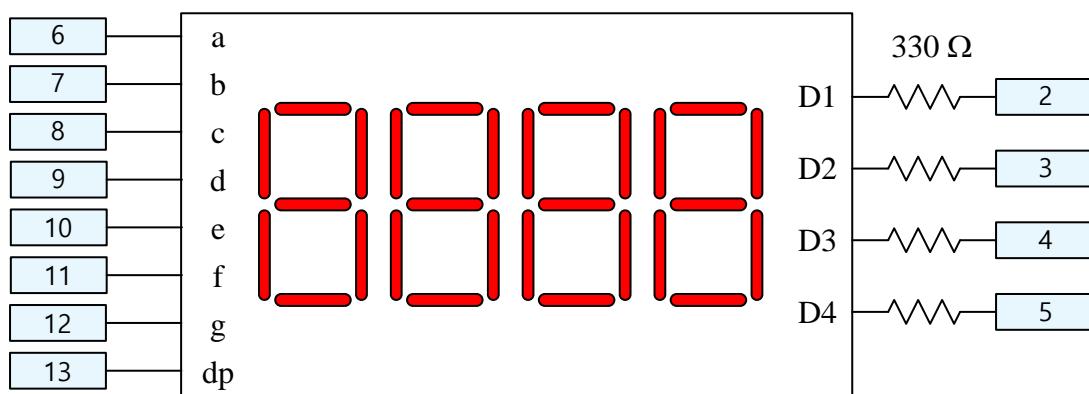
### 예제 9-2-1

#### ▣ 목표

- ✓ 1234 의 숫자 (정수)를 7-세그먼트에 표시한다.

#### ▣ 회로도

- ✓ WCN4-0036GU-A11 (공통 애노드)를 사용한다.
- ✓ 7-세그먼트 핀 a ~ dp 를 디지털 6 ~ 13 에, digit 핀 (D1, D2, D3, D4)을 디지털 2, 3, 4, 5 에 연결한다.
- ✓ 전류 제어용 저항은 digit 핀에 연결한다.



▶ 스케치 프로그램

- ✓ SevSeg.h 라이브러리 사용

```
// 4 digits 7-segment display
// Library : SevSeg.h

#include <SevSeg.h>

SevSeg mySS;
byte hwCfg = COMMON_ANODE;
byte numDgt = 4;
// Digit 선택 핀 번호
byte dgtPin[] = {2, 3, 4, 5};
// 7-세그먼트 연결 핀 번호
byte segPin[] = {6, 7, 8, 9, 10, 11, 12, 13};

void setup() {
    // SevSeg 객체 선언
    mySS.begin(hwCfg, numDgt, dgtPin, segPin);
    // 밝기 설정
    mySS.setBrightness(90);
}

void loop() {
    // 숫자 1234를 표시한다.
    int num = 1234;
    mySS.setNumber(num);
    mySS.refreshDisplay();
}
```

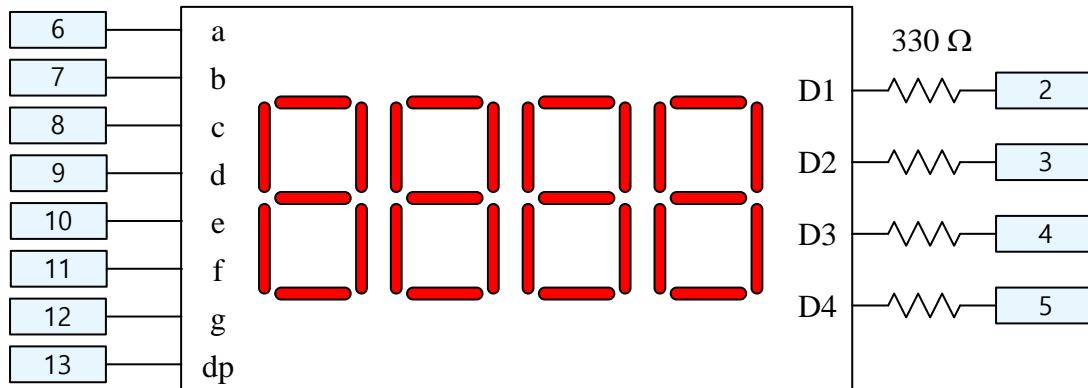
예제 9-2-2

▶ 목표

- ✓ 3.14 의 숫자 (소수점 있는 숫자)를 7-세그먼트에 표시한다.

#### 회로도

- ✓ WCN4-0036GU-A11 (공통 애노드)를 사용한다.
- ✓ 7-세그먼트 핀 a ~ dp 를 디지털 6 ~ 13 에, digit 핀 (D1, D2, D3, D4)을 디지털 2, 3, 4, 5 에 연결한다.



#### 스케치 프로그램

- ✓ SevSeg.h 라이브러리 사용

```
// 4 digits 7-segment display
// Library : SevSeg.h

#include <SevSeg.h>

SevSeg mySS;
byte hwCfg = COMMON_ANODE;
byte numDgt = 4;
// Digit 선택 핀 번호
byte dgtPin[] = {2, 3, 4, 5};
// 7-세그먼트 연결 핀 번호
byte segPin[] = {6, 7, 8, 9, 10, 11, 12, 13};

void setup() {
    // SevSeg 객체 선언
    mySS.begin(hwCfg, numDgt, dgtPin, segPin);
    // 밝기 설정
    mySS.setBrightness(90);
}

void loop() {
    // 숫자 3.14를 소수점 2자리로 표시한다.
    float num = 3.14;
```

```

    mySS.setNumber(num, 2);
    mySS.refreshDisplay();
}

```

## 연습 문제

✚ 연습 문제 9-2-1

- ✓ 아두이노 시리얼 모니터에서 입력된 숫자를 7-세그먼트에 나타내는 회로를 구성하고 프로그램을 작성하라. 0~9999 을 벗어나면 '----'로 표시하라.

✚ 연습 문제 9-2-2

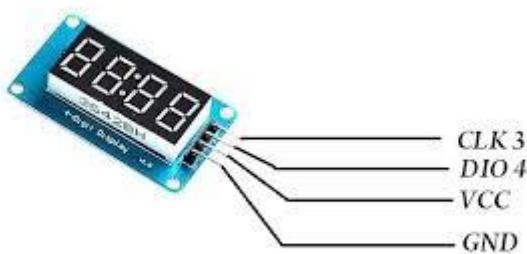
- ✓ 0~9999 의 숫자를 4 자리 7-세그먼트에 100 Hz 의 속도로 표시하는 회로를 구성하고 프로그램을 작성하라. 9999 를 초과하면 다시 0 이 되도록 하라.

### 9.2.2 4 자리 7-세그먼트 표시기 2



✚ 4 자리 7-세그먼트 표시기 2

- ✓ TM1637 IC 사용
- ✓ 4 자리 7-세그먼트 표시기
- ✓ 4 개의 연결 핀 (VCC, GND, CLK, DIO)



✚ 라이브러리

**TM1637Display.h**

- ✓ 소수점, 콜론 지원
- ✓ 밝기 조절 가능
- ✓ <https://playground.arduino.cc/Main/TM1637>
- ✓ <https://github.com/avishorp/TM1637>
- ✓ 매뉴얼 다운로드
  - ✗ User guide for TM1637 4 digits display
- ✓ 센터 콜론 on/off 지원 함수 추가 라이브러리
  - ✗ TM1637-1.1.0.zip
  - ✗ <http://forum.arduino.cc/index.php?topic=271238.0>

 관련 함수

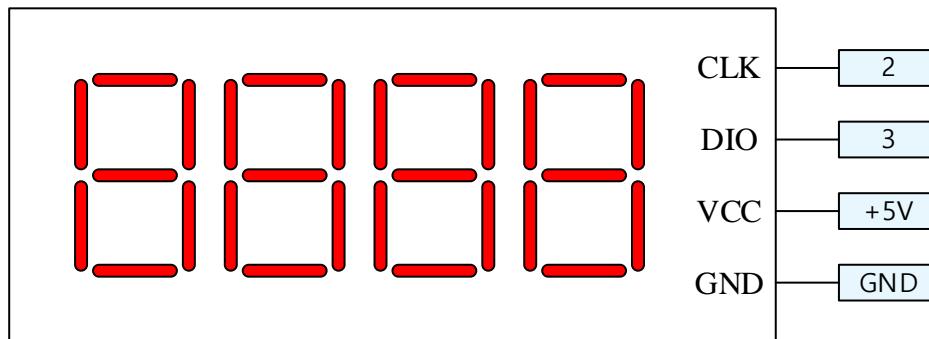
```
TM1637Display(clkPin, dioPin);
  ✗ clkPin : 모듈의 clock 핀에 연결되는 아두이노 디지털 핀 번호
  ✗ dioPin : 모듈의 DIO 핀에 연결되는 아두이노 디지털 핀 번호
.setBrightness(uint8_t num, bool on=true);
  ✗ num : 0~7 사이의 정수 값
.showNumberDec(int num, bool leadZero=0, uint8_t numDigit=4,
uint8_t pos=0);
  ✗ num : 표시할 숫자
  ✗ leadZero : 앞에 0을 나타내려면 true, (default=false)
  ✗ numDigit : Digit 수 (자리 수), (default=4)
  ✗ pos : LSD의 위치 (0~3, 0: 왼쪽, 3: 오른쪽)
.setSegments(const uint8_t seg[], uint8_t len=4, uint8_t pos=0);
  ✗ seg[] : 세그먼트 데이터 (비트 0 – 세그먼트 a, 비트 1 – 세그먼트 b,
..., 비트 6 – 세그먼트 g)
  ✗ - 표시 : 1000000, 0x40
  ✗ 모두 off : 0000000, 0x00
.showNumberDecEx(int num, uint8_t dots=0, bool leadZero=0,
uint8_t numDigit=4, uint8_t pos=0);
  ✗ version 1.1.0에 추가된 함수
```

**예제 9-2-3****▣ 목표**

- ✓ 0~9999 의 숫자를 100 Hz 의 속도로 7-세그먼트에 표시한다.

**▣ 회로도**

- ✓ 7-세그먼트 모듈 CLK, DIO 단자를 디지털 2, 3 번 핀에 연결한다.

**▣ 스케치 프로그램**

- ✓ TM1637Display.h 라이브러리 사용

```
// 4 Digits 7-Segment Display
// Library : TM1637Display.h

#include <TM1637Display.h>
#define CLK 2
#define DIO 3

// TM1637display 객체 선언
TM1637Display myTMD(CLK, DIO);
int maxNum = 10000;
int delayTime = 10;

void setup() {
    myTMD.setBrightness(7);
}

void loop() {
    for (int n = 0; n < maxNum; n++) {
        // 숫자를 표시한다.
        myTMD.showNumberDec(n);
        delay(delayTime);
    }
}
```

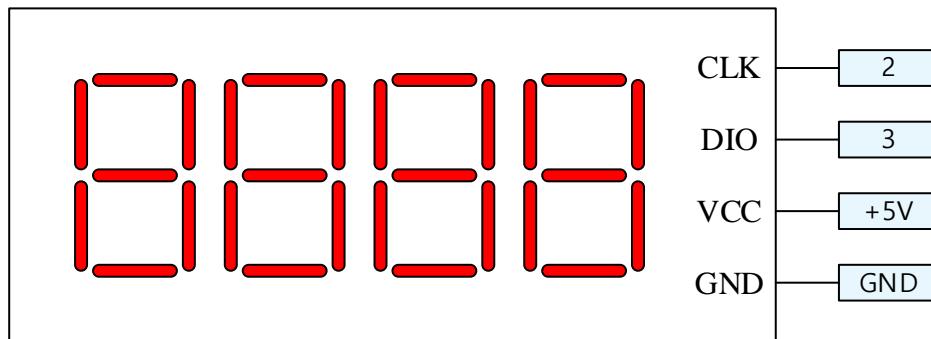
### 예제 9-2-4

#### ▣ 목표

- ✓ 0~9999 의 숫자를 100 Hz 의 속도로 7-세그먼트에 표시한다.
- ✓ 예제 9-2-3 의 프로그램은 100 Hz 보다 느리게 바뀐다. (왜?)
- ✓ TM1637Display.h 라이브러리 사용

#### ▣ 회로도

- ✓ 7-세그먼트 모듈 CLK, DIO 단자를 디지털 2, 3 번 핀에 연결한다.



#### ▣ 스케치 프로그램

- ✓ TM1637Display.h 라이브러리 사용

```
// 4 Digits 7-Segment Display
// Library : TM1637Display.h

#include <TM1637Display.h>
#define CLK 2
#define DIO 3

// TM1637display 객체 선언
TM1637Display myTMD(CLK, DIO);
int maxNum = 10000;
int delayTime = 10;
int num = 0;

void setup() {
    myTMD.setBrightness(7);
}

void loop() {
    // 전에 숫자를 표시한 시간보다 10 ms 지난 후에 숫자를 표시한다.
    static long timer = millis();
    if (millis() >= timer) {

```

```
    num++;
    num %= 10000;
    timer += delayTime;
    myTMD.showNumberDec(num);
}
}
```

---

## 연습 문제

### ▣ 연습 문제 9-3-1

- ✓ 아두이노 시리얼 모니터에서 입력된 숫자를 7-세그먼트에 나타내는 회로를 구성하고 프로그램을 작성하라. 0~9999 을 벗어나면 '----'로 표시하라.

### ▣ 연습 문제 9-3-2

- ✓ 4 자리 7-세그먼트를 이용하여 분과 초가 표시되고 매 초마다 가운데 콜론이 점멸하는 시계 회로를 구성하고 프로그램을 작성하라.

### 9.3 LCD 디스플레이 제어

#### LCD 디스플레이

- ✓ 1602 LCD 디스플레이
  - ✗ 16x2 (16 자, 2 줄) 디스플레이
- ✓ 2004 LCD 디스플레이
  - ✗ 20x4 (20 자, 4 줄) 디스플레이
- ✓ I2C(inter-integrated circuit) 인터페이스



#### 단자와 핀

- ✓ GND
- ✓ VCC : 5 V
- ✓ SDA : 데이터 전송, 아두이노 아날로그 핀 A4에 연결한다.
- ✓ SCL : 클럭 신호, 아두이노 아날로그 핀 A5에 연결한다.

#### 라이브러리

##### LiquidCrystal\_I2C.h

- ✗ LCD 라이브러리
- ✗ 1602 LCD 디스플레이, 2004 LCD 디스플레이에 적용
- ✗ <https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library>
- ✗ LiquidCrystal\_I2C.h 라이브러리는 종류가 많아 헤더 파일을 확인하고 사용해야 한다.

### ✚ 관련 함수

```
LiquidCrystal_I2C(uint8_t lcd_addr, uint8_t lcd_cols, uint8_t
lcd_rows, uint8_t charsize=LCD_5x8DOTS);

    ✕ lcd_addr : LCD 주소
    ✕ lcd_cols : LCD 글자 열 수 (16)
    ✕ lcd_rows : LCD 글자 행 수 (2)
    ✕ charsize : LCD 글자의 도트 크기 (LCD_5x10DOTS 또는 LCD_5x8DOTS)

.begin();
    ✕ LCD 초기화 (모든 글자 지움), 최초 반드시 실행해야 한다.

.clear();
    ✕ 모든 글자 제거 후, 커서를 (0,0) 위치에 놓는다.

.write(char);
    ✕ 문자 1개를 LCD에 쓴다.

.print(char[]);
.printstr(char[]);
    ✕ 문자 열을 LCD에 쓴다. (String 클래스는 불가, char[] 만 가능)

.setCursor(col, row);
    ✕ 커서 위치를 제어한다., 첫 줄 처음 (0,0), 둘째 줄 처음 (0,1)

.blink(), .noBlink();
    ✕ 글씨가 쓰이는 곳 점멸 제어한다.

.cursor(), .noCursor();
    ✕ 커서 on/off 제어한다.

.display(), .noDisplay();
    ✕ 글씨 보이기/안 보이기 제어한다.

.backlight(), .noBacklight();
    ✕ 백 라이트 on/off 제어한다.
```

### ✚ 1602 LCD 디스플레이 주소

- ✓ 일반적으로 0x27 로 나와 있지만 일부 제품은 0x3F 이다.
- ✓ 주소 스캔 프로그램으로 주소를 확인 필요하다.
- ✓ 주소 스캔 프로그램
  - ✖ <http://henrysbench.capnfatz.com/henrys-bench/arduino-projects-tips->

[and-more/arduino-quick-tip-find-your-i2c-address/](http://www.arduino.cc/en/Tutorial/and-more/arduino-quick-tip-find-your-i2c-address/)

- ✖ 프로그램과 시리얼 모니터의 baud rate 속도를 맞춘다. (프로그램의 속도가 115200이므로 시리얼 모니터도 115200으로 맞춘다. 둘 다 9600으로 맞추어도 동작.)
- ✓ 1602 LCD 주소 스캔 결과



The screenshot shows the Arduino Serial Monitor window titled "COM8 (Arduino/Genuino Uno)". The text output is as follows:

```
I2C scanner. Scanning ...
Found address: 63 (0x3F)
Done.
Found 1 device(s).
```

At the bottom of the window, there are three buttons: "자동 스크롤" (Auto Scroll), "line ending 없음" (No Line Ending), and "115200 보드레이트" (Board Rate 115200). There is also a "전송" (Send) button.

#### ■ 2004 LCD 디스플레이 주소

- ✓ 일반적 0x27
- ✖ 위 스캔 프로그램으로 확인 필요하다.

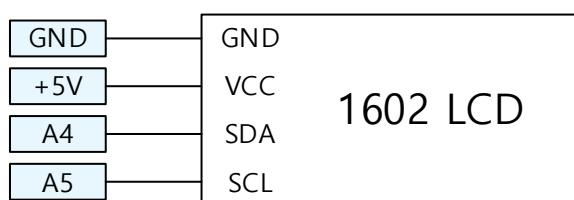
#### 예제 9-3-1

##### ■ 목표

- ✓ 1602 LCD 디스플레이에 적당한 두 줄의 문장을 나타낸다. (모든 줄의 문장은 16자 이내로 할 것)

##### ■ 회로도

- ✓ SDA : 아두이노 아날로그 핀 A4에 연결한다.
- ✓ SCL : 아두이노 아날로그 핀 A5에 연결한다.



✚ 스케치 프로그램

- ✓ LiquidDisplay\_I2C.h 라이브러리 사용

---

```
// Display Text in the 1602 LCD Display
// SDA -> A4, SCL -> A5
// Library : LiquidCrystal_I2C.h

#include <LiquidCrystal_I2C.h>

// LiquidCrystal_I2C 객체 선언
LiquidCrystal_I2C myLCD(0x3F, 16, 2);
const char myStr1[] = "Hello, Arduino";
const char myStr2[] = "My name is Baik";

void setup() {
    // LCD를 초기화 한다.
    myLCD.begin();
    // LCD의 백 라이트를 켠다.
    myLCD.backlight();
    // LCD를 다 지우고 커서를 1 행 1 열에 놓는다.
    myLCD.clear();
    // LCD에 문자열을 쓴다.
    myLCD.printstr(myStr1);
    // 커서를 2 행 1 열에 놓는다.
    myLCD.setCursor(0, 1);
    // LCD에 문자열을 쓴다.
    myLCD.printstr(myStr2);
}

void loop() {
```

---

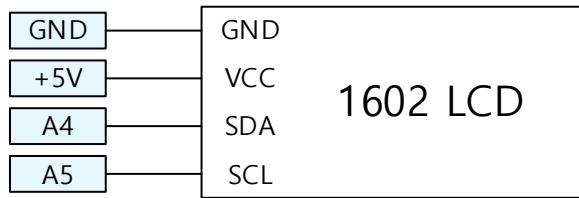
### 예제 9-3-2

✚ 목표

- ✓ 시리얼 모니터에서 입력한 2 줄의 문장을 1602 LCD 디스플레이에 표시한다. (각 줄은 16자 이내)

✚ 회로도

- ✓ SDA : 아두이노 아날로그 핀 A4에 연결한다.
- ✓ SCL : 아두이노 아날로그 핀 A5에 연결한다.



▶ 주의 사항

- ✓ String 을 문자 열로 변환해야 한다. (String.toCharArray 사용)

▶ 스케치 프로그램

- ✓ LiquidDisplay\_I2C.h 라이브러리 사용

```

// Display Text in the 1602 LCD Display
// Text is given in serial monitor
// SDA -> A4, SCL -> A5
// Library : LiquidCrystal_I2C.h

#include <LiquidCrystal_I2C.h>

// LiquidCrystal_I2C 객체 선언
LiquidCrystal_I2C myLCD(0x3F, 16, 2);
// 문자열을 쓰는 LCD의 행 번호
int charLine = 0;

void setup() {
  Serial.begin(9600);
  // LCD를 초기화 한다.
  myLCD.begin();
  // LCD의 백 라이트를 켠다.
  myLCD.backlight();
  // LCD를 다 지우고 커서를 1 행 1 열에 놓는다.
  myLCD.clear();
}

void loop() {
  // 시리얼 버퍼에 데이터가 있으면 문자열을 읽고 시리얼 모니터에 나타낸다.
  if (Serial.available()) {
    String myStr = Serial.readString();
    Serial.println(myStr);
    char charArray[17];
    // 문자열을 변환한다. (String 문자열을 char[] 문자열로 바꾼다.)
    myStr.toCharArray(charArray, 17);
    // 커서를 1 열에 놓는다. (charLine=0이면 1 행, charLine=1이면 2 행)
    myLCD.setCursor(0, charLine);
}
  
```

```
    myLCD.printstr(charArray);
    charLine = 1;
}
}
```

---

## 연습 문제

### ■ 연습 문제 9-3-1

- ✓ 초음파 센서를 이용하여 거리를 측정하고 거리를 1602 LCD 디스플레이에 표시하도록 회로를 구성하고 프로그램을 작성하라. (첫 줄 : Distance is, 둘째 줄 : \*\*\* cm)

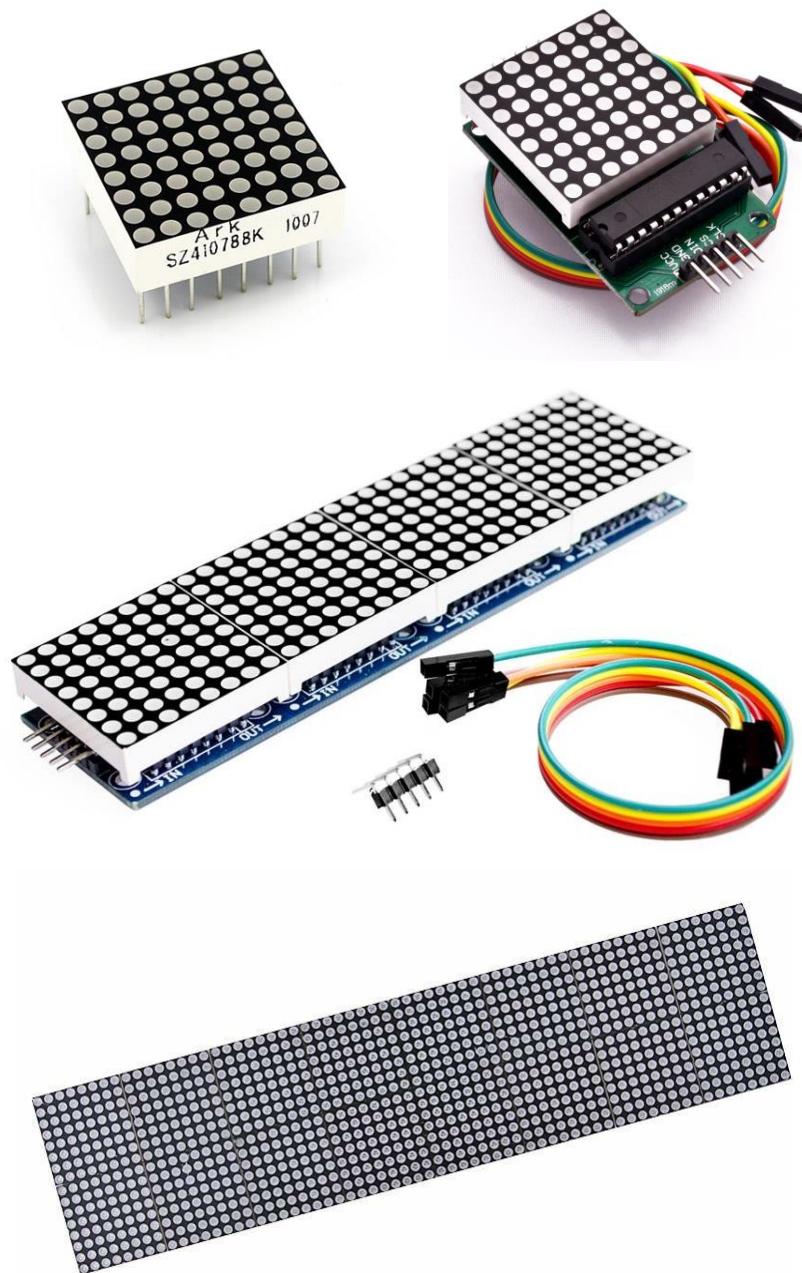
### ■ 연습 문제 9-3-2

- ✓ 아두이노 시리얼 모니터에서 입력한 문장을 1602 LCD 디스플레이에 표시하는 회로를 구성하고 프로그램을 작성하라. 단 2 줄을 초과하면 위로 스크롤하도록 하라.

## 9.4 Matrix LED 제어

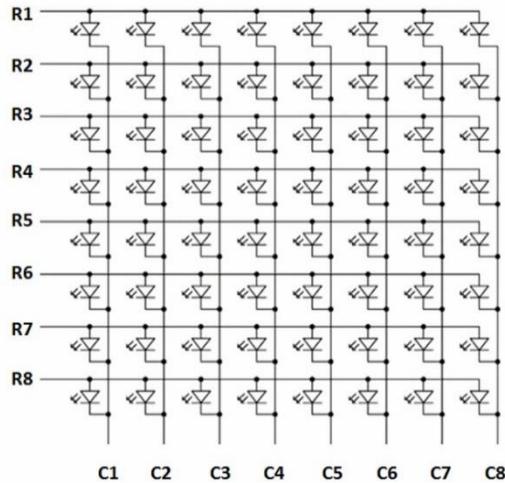
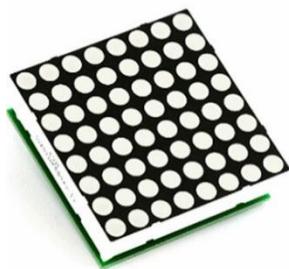
### 9.4.1 Matrix LED 제어

#### ✚ Matrix LED 종류



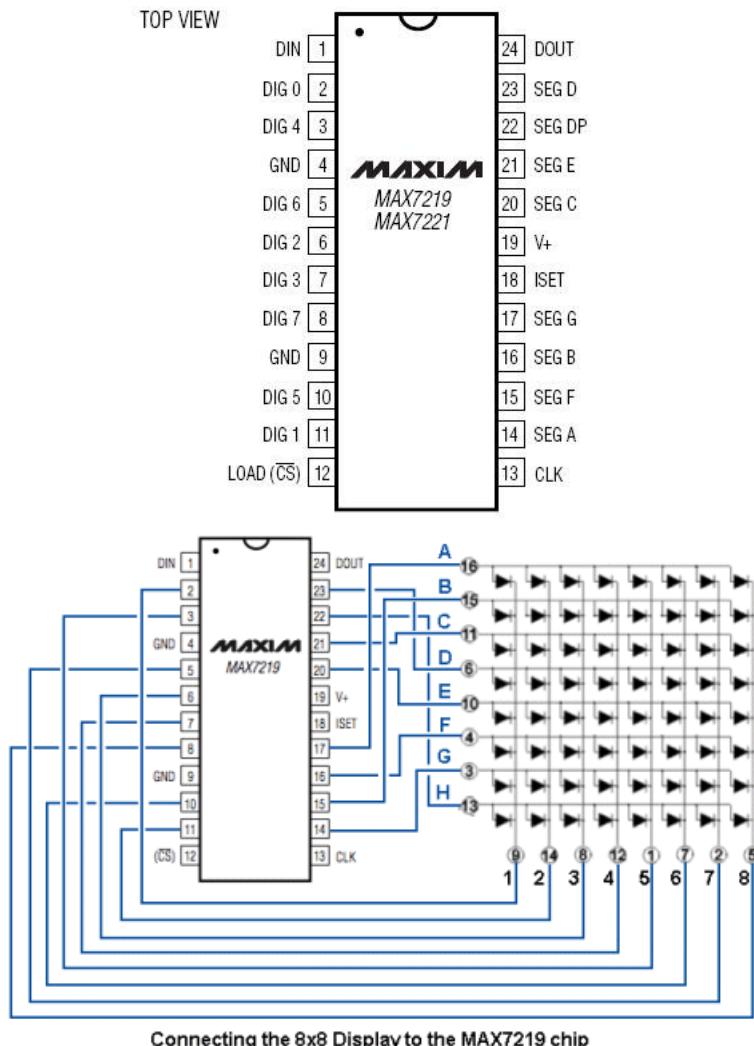
#### ✚ 8x8 matrix LED

- ✓ Row 핀 (8 개), Column 핀 (8 개)
- ✓ R1~R8, C1~C8 의 값에 따라 64 개의 LED 상태 결정



✚ MAX7219 를 이용한 8x8 matrix LED 제어

- ✓ MAX7219 시프트 레지스터
- ✓ MAX7219 는 8 자리 7-세그먼트에도 적용 가능



✚ 8x8 matrix LED 핀 배치



#### ■ Matrix LED 라이브러리

##### `LedControl.h`

- ✖ 8 개의 matrix LED가 연결 가능하다.
- ✖ 각각의 LED on/off 가능
- ✖ 행 및 열의 LED on/off 가능
- ✖ 밝기 조절 가능
- ✖ <https://playground.arduino.cc/Main/LedControl>
- ✖ <https://github.com/wayoda/LedControl>

#### ■ 관련 함수

```

LedControl(int dataPin, int clkPin, int csPin, int numDevice);
    ✖ LedControl 객체 선언
    ✖ dataPin : matrix LED의 DIN에 연결되는 아두이노 핀
    ✖ clkPin : matrix LED의 CLK에 연결되는 아두이노 핀
    ✖ csPin : matrix LED의 CS에 연결되는 아두이노 핀
    ✖ numDevice : 연결된 matrix LED의 개수
.setIntensity(int noMLed, int intensity);
    ✖ noMLed : 연결한 matrix LED의 주소(0~numDevice-1)
    ✖ intensity : LED의 밝기 (0~15)
.clearDisplay(int noMLed);
    ✖ 선택한 matrix LED 모두 off
.setshutdown(int noMLed, bool state);
    ✖ state : true (shut-down 모드), false (정상 모드)
.setLed(int noMLed, int row, int col, bool state);
    ✖ int row, col : 선택한 행 및 열 (0~7)
    ✖ state : 선택한 LED 상태 (false : off, true : on)
.setRow(int noMLed, int row, byte value);

```

- ✖ row : 선택한 행 번호 (0~7)
- ✖ value : 선택한 행의 LED 상태 (1이면 on)

```
.setColumn(int noMLed, int col, byte value);
```

- ✖ col : 선택한 열 번호 (0~7)
- ✖ value : 선택한 열의 LED 상태 (1이면 on)

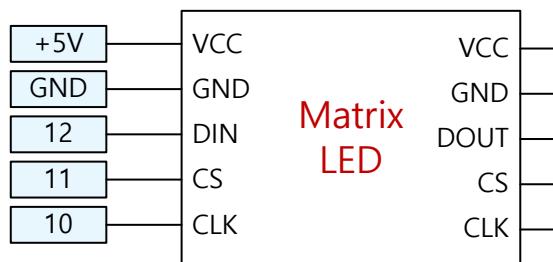
### 예제 9-4-1

#### ▣ 목표

- ✓ 행과 열을 랜덤하게 선택하여 해당 LED 를 20  $\mu$ s 만큼 켠다.

#### ▣ 회로도

- ✓ 디지털 12 번, 10 번, 11 번 핀에 matrix LED 의 DIN, CS, CLK 을 연결한다.



#### ▣ 스케치 프로그램

- ✓ LedControl.h 라이브러리 사용

```
// 8 x 8 Matrix LED Display
// DIN = 12, CLK = 11, CS = 10
// Library : LedControl.h

#include <LedControl.h>

const int delayTime = 20;
// LedControl 객체 선언
LedControl myLC = LedControl(12, 11, 10, 1);

void setup() {
    // LED의 셋 다운을 해제하고 밝기를 설정하며 모두 지운다.
    myLC.shutdown(0, false);
    myLC.setIntensity(0, 8);
    myLC.clearDisplay(0);
}

void loop() {
```

```
// 행과 열 번호를 랜덤하게 선택한다.
int row = random(0, 8);
int col = random(0, 8);
// 선택한 행과 열의 LED를 켰다가 일정 시간 후 끈다.
myLC.setLed(0, row, col, true);
delay(delayTime);
myLC.setLed(0, row, col, false);
}
```

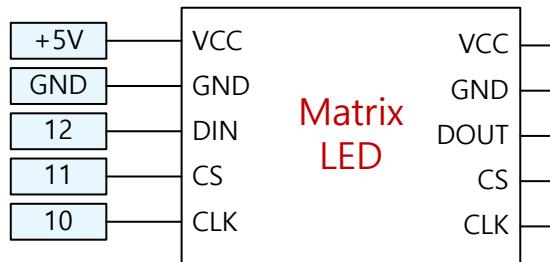
### 예제 9-4-2 :

#### ▣ 목표

- ✓ 대각선(/) 아래의 LED 를 위에서 아래로, 왼쪽에서 오른쪽으로 on, off 시킨다.

#### ▣ 회로도

- ✓ 디지털 12 번, 10 번, 11 번 핀에 matrix LED 의 DIN, CS, CLK 을 연결한다.



#### ▣ 스케치 프로그램

- ✓ LedControl.h 라이브러리 사용

```
// 8 x 8 Matrix LED Display
// DIN = 12, CLK = 11, CS = 10
// Library : LedControl.h

#include <LedControl.h>

const int delayTime = 100;
// 이진수로 {00000001, 00000011, 00000111, 00001111, 00011111, 00111111,
// 01111111, 11111111}이다. LED를 켜는데 사용한다.
const byte value[] = {0x01, 0x03, 0x07, 0x0F, 0x1F, 0x3F, 0x7F, 0xFF};
// LedControl 객체 선언
LedControl myLC = LedControl(12,11,10,1);

void setup() {
    // LED의 셋 다운을 해제하고 밝기를 설정하여 모두 지운다.
```

```

    myLC.shutdown(0, false);
    myLC.setIntensity(0, 8);
    myLC.clearDisplay(0);
}

void loop() {
    turn_on_off_down();
    turn_on_off_right();
}

// LED를 행 순서에 따라 대각선으로 켜고 끄는 함수
void turn_on_off_down() {
    // LED를 1 행 1 개, 2 행 2 개 식으로 켠다.
    for (int row = 0; row < 8; row++) {
        myLC.setRow(0, row, value[row]);
        delay(delayTime);
    }
    // LED를 1 행, 2 행 순으로 모두 끈다.
    for (int row = 0; row < 8; row++) {
        myLC.setRow(0, row, 0x00);
        delay(delayTime);
    }
}

// LED를 열 순서에 따라 대각선으로 켜고 끄는 함수
void turn_on_off_right() {
    // LED를 1 열 1 개, 2 열 2 개 식으로 켠다.
    for (int col = 0; col < 8; col++) {
        myLC.setColumn(0, col, value[col]);
        delay(delayTime);
    }
    // LED를 1 열, 2 열 순으로 모두 끈다.
    for (int col = 0; col < 8; col++) {
        myLC.setColumn(0, col, 0x00);
        delay(delayTime);
    }
}

```

## 연습 문제

 연습 문제 9-4-1

- ✓ 8x8 matrix LED 에서 대각선으로 LED 를 켜고 끄는 회로를 구성하고 프로그램을 작성하라.

▣ 연습 문제 9-4-2

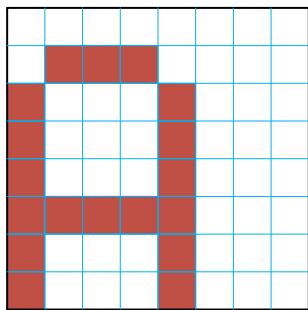
- ✓ 8x8 matrix LED에서 시계 방향으로 LED를 켜고 끄는 회로를 구성하고 프로그램을 작성하라.

▣ 연습 문제 9-4-3

- ✓ 8x8 matrix LED에서 8개의 열이 랜덤한 크기로 LED를 켜는 회로를 구성하고 프로그램을 작성하라.

#### 9.4.2 Matrix LED에 문자 표시

▣ 5x7 문자 폰트



|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 |

- ✓ 5개 열의 비트 값

✗ 1 열 : 00111111 → 0x3F

✗ 2 열 : 01000100 → 0x44

✗ 3 열 : 01000100 → 0x44

✗ 4 열 : 01000100 → 0x44

✗ 5 열 : 00111111 → 0x3F

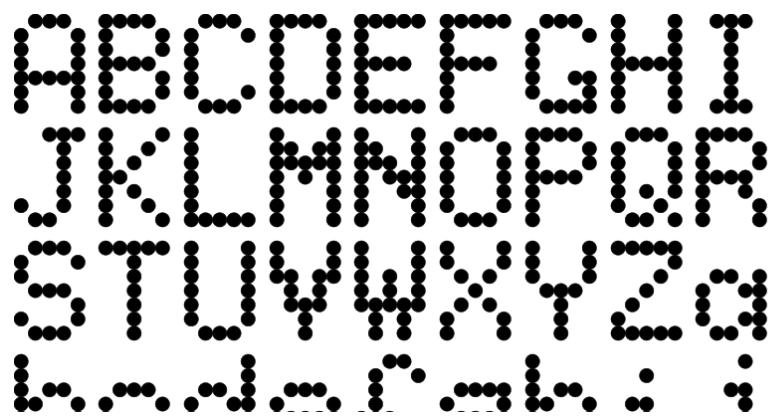
✓ A → {0x3F, 0x44, 0x44, 0x44, 0x3F}

✓ B → {0x7F, 0x49, 0x49, 0x49, 0x36}

✓ C → {0x3E, 0x41, 0x41, 0x41, 0x22}

✓ D → {0x7F, 0x41, 0x41, 0x22, 0x1C}

▣ 5x7 문자 폰트



#### ■ 5x7 문자 폰트 (ASCII 코드)

- ✓ 0x00, 0x00, 0x00, 0x00, 0x00, // space
- ✓ 0x00, 0x00, 0x7D, 0x00, 0x00, // !
- ✓ 0x00, 0x70, 0x00, 0x70, 0x00, // "
- ✓ 0x14, 0x7F, 0x14, 0x7F, 0x14, // #
- ✓ 0x12, 0x2A, 0x7F, 0x2A, 0x24, // \$
- ✓ 0x62, 0x64, 0x08, 0x13, 0x23, // %
- ✓ 0x36, 0x49, 0x55, 0x22, 0x05, // &
- ✓ 0x00, 0x50, 0x60, 0x00, 0x00, // '
- ✓ 0x00, 0x1C, 0x22, 0x41, 0x00, // (
- ✓ 0x00, 0x41, 0x22, 0x1C, 0x00, // )
- ✓ 0x08, 0x2A, 0x1C, 0x2A, 0x08, // \*
- ✓ 0x08, 0x08, 0x3E, 0x08, 0x08, // +
- ✓ 0x00, 0x05, 0x06, 0x00, 0x00, // ,
- ✓ 0x08, 0x08, 0x08, 0x08, 0x08, // -
- ✓ 0x00, 0x03, 0x03, 0x00, 0x00, // .
- ✓ 0x02, 0x04, 0x08, 0x10, 0x20, // /
- ✓ 0x3E, 0x45, 0x49, 0x51, 0x3E, // 0
- ✓ 0x00, 0x21, 0x7F, 0x01, 0x00, // 1
- ✓ 0x21, 0x43, 0x45, 0x49, 0x31, // 2
- ✓ 0x42, 0x41, 0x51, 0x69, 0x46, // 3
- ✓ 0x0C, 0x14, 0x24, 0x7F, 0x04, // 4
- ✓ 0x72, 0x51, 0x51, 0x51, 0x4E, // 5

- ✓ 0x1E, 0x29, 0x49, 0x49, 0x06, // 6
- ✓ 0x40, 0x47, 0x48, 0x50, 0x60, // 7
- ✓ 0x36, 0x49, 0x49, 0x49, 0x36, // 8
- ✓ 0x30, 0x49, 0x49, 0x4A, 0x3C, // 9
- ✓ 0x00, 0x36, 0x36, 0x00, 0x00, // :
- ✓ 0x00, 0x35, 0x36, 0x00, 0x00, // ;
- ✓ 0x00, 0x08, 0x14, 0x22, 0x41, // <
- ✓ 0x14, 0x14, 0x14, 0x14, 0x14, // =
- ✓ 0x41, 0x22, 0x14, 0x08, 0x00, // >
- ✓ 0x20, 0x40, 0x45, 0x48, 0x30, // ?
- ✓ 0x26, 0x49, 0x4F, 0x41, 0x3E, // @
- ✓ 0x3F, 0x44, 0x44, 0x44, 0x3F, // A
- ✓ 0x7F, 0x49, 0x49, 0x49, 0x36, // B
- ✓ 0x3E, 0x41, 0x41, 0x41, 0x22, // C
- ✓ 0x7F, 0x41, 0x41, 0x22, 0x1C, // D
- ✓ 0x7F, 0x49, 0x49, 0x49, 0x41, // E
- ✓ 0x7F, 0x48, 0x48, 0x40, 0x40, // F
- ✓ 0x3E, 0x41, 0x41, 0x45, 0x26, // G
- ✓ 0x7F, 0x08, 0x08, 0x08, 0x7F, // H
- ✓ 0x00, 0x41, 0x7F, 0x41, 0x00, // I
- ✓ 0x02, 0x01, 0x41, 0x7E, 0x40, // J
- ✓ 0x7F, 0x08, 0x14, 0x22, 0x41, // K
- ✓ 0x7F, 0x01, 0x01, 0x01, 0x01, // L
- ✓ 0x7F, 0x20, 0x10, 0x20, 0x7F, // M
- ✓ 0x7F, 0x10, 0x08, 0x04, 0x7F, // N
- ✓ 0x3E, 0x41, 0x41, 0x41, 0x3E, // O
- ✓ 0x7F, 0x48, 0x48, 0x48, 0x30, // P
- ✓ 0x3E, 0x41, 0x45, 0x42, 0x3D, // Q
- ✓ 0x7F, 0x48, 0x4C, 0x4A, 0x31, // R

- ✓ 0x31, 0x49, 0x49, 0x49, 0x46, // S
- ✓ 0x40, 0x40, 0x7F, 0x40, 0x40, // T
- ✓ 0x7E, 0x01, 0x01, 0x01, 0x7E, // U
- ✓ 0x7C, 0x02, 0x01, 0x02, 0x7C, // V
- ✓ 0x7F, 0x02, 0x0C, 0x02, 0x7F, // W
- ✓ 0x63, 0x14, 0x08, 0x14, 0x63, // X
- ✓ 0x60, 0x10, 0x0F, 0x10, 0x60, // Y
- ✓ 0x43, 0x45, 0x49, 0x51, 0x61, // Z
- ✓ 0x00, 0x00, 0x7F, 0x41, 0x41, // [
- ✓ 0x20, 0x10, 0x08, 0x04, 0x02, // inverse slash
- ✓ 0x41, 0x41, 0x7F, 0x00, 0x00, // ]
- ✓ 0x10, 0x20, 0x40, 0x20, 0x10, // ^
- ✓ 0x01, 0x01, 0x01, 0x01, 0x01, // \_
- ✓ 0x00, 0x40, 0x20, 0x10, 0x00, // `
- ✓ 0x02, 0x15, 0x15, 0x15, 0x0F, // a
- ✓ 0x7F, 0x09, 0x11, 0x11, 0x0E, // b
- ✓ 0x0E, 0x11, 0x11, 0x11, 0x02, // c
- ✓ 0x0E, 0x11, 0x11, 0x09, 0x7F, // d
- ✓ 0x0E, 0x15, 0x15, 0x15, 0x0C, // e
- ✓ 0x08, 0x3F, 0x48, 0x40, 0x20, // f
- ✓ 0x08, 0x14, 0x15, 0x15, 0x1E, // g
- ✓ 0x7F, 0x08, 0x10, 0x10, 0x0F, // h
- ✓ 0x00, 0x11, 0x5F, 0x01, 0x00, // i
- ✓ 0x02, 0x01, 0x11, 0x5E, 0x00, // j
- ✓ 0x00, 0x7F, 0x04, 0x0A, 0x11, // k
- ✓ 0x00, 0x41, 0x7F, 0x01, 0x00, // l
- ✓ 0x1F, 0x10, 0x0C, 0x10, 0x0F, // m
- ✓ 0x1F, 0x08, 0x10, 0x10, 0x0F, // n
- ✓ 0x0E, 0x11, 0x11, 0x11, 0x0E, // o

- ✓ 0x1F, 0x14, 0x14, 0x14, 0x08, // p
- ✓ 0x08, 0x14, 0x14, 0x0C, 0x1F, // q
- ✓ 0x1F, 0x08, 0x10, 0x10, 0x08, // r
- ✓ 0x09, 0x15, 0x15, 0x15, 0x02, // s
- ✓ 0x10, 0x7E, 0x11, 0x01, 0x02, // t
- ✓ 0x1E, 0x01, 0x01, 0x02, 0x1F, // u
- ✓ 0x1C, 0x02, 0x01, 0x02, 0x1C, // v
- ✓ 0x1E, 0x01, 0x06, 0x01, 0x1E, // w
- ✓ 0x11, 0x0A, 0x04, 0x0A, 0x11, // x
- ✓ 0x18, 0x05, 0x05, 0x05, 0x1E, // y
- ✓ 0x11, 0x13, 0x15, 0x19, 0x11, // z
- ✓ 0x00, 0x08, 0x36, 0x41, 0x00, // {
- ✓ 0x00, 0x00, 0x7F, 0x00, 0x00, // |
- ✓ 0x00, 0x41, 0x36, 0x08, 0x00, // }
- ✓ 0x08, 0x08, 0x2A, 0x1C, 0x08, // ~
- ✓ 0x08, 0x1C, 0x2A, 0x08, 0x08, // DEL

#### ▣ 문자 표시 프로그래밍 방법

- ✓ 문자 수가 적을 경우

✗ 문자에 해당하는 폰트 데이터를 2차원 배열로 선언한다.

```
const byte charName[][] = {
    {0x7F, 0x49, 0x49, 0x49, 0x36}, // B
    {0x3F, 0x44, 0x44, 0x44, 0x3F}, // A
    {0x00, 0x41, 0x7F, 0x41, 0x00}, // I
    {0x7F, 0x08, 0x14, 0x22, 0x41}, // K
};
```

- ✓ 문자 수가 많고 문자가 정해지지 않았을 경우

✗ ASCII 코드 데이터를 헤더 파일로 저장한다.

✗ 헤더 파일 : asciiCode.h

```
// 5x7 Font Image of ASCII Code
const byte asciiCode[][] = {
    {0x00, 0x00, 0x00, 0x00, 0x00}, // space
```

```

    ...
    {0x08, 0x1C, 0x2A, 0x08, 0x08} // DEL
};

```

✖ 문자의 인덱스 찾는 방법

- 문자의 인덱스 = 문자의 ASCII 코드 번호 – 오프셋 값(32)
- 예 : A ( $65 - 32 = 33$ ) → `asciicode[32] = A`
- 예 : a ( $97 - 32 = 65$ ) → `asciicode[65] = a`

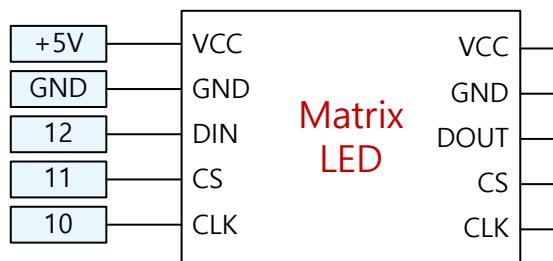
**예제** 9-4-3

✚ 목표

- ✓ 영문 이름을  $500\ \mu\text{s}$  간격으로 표시한다. (마지막은 모두 off 한다.)

✚ 회로도

- ✓ 디지털 12 번, 10 번, 11 번 핀에 matrix LED 의 DIN, CS, CLK 을 연결한다.



✚ 스케치 프로그램

- ✓ `LedControl.h` 라이브러리 사용

```

// 8 x 8 Matrix LED Display
// DIN = 12, CLK = 11, CS = 10
// Library : LedControl.h

#include <LedControl.h>

const int delayTime = 2000;
// 나타낼 문자 열 코드
const byte nameCode[][] = {
    {0x7F, 0x49, 0x49, 0x49, 0x36}, // B
    {0x3F, 0x44, 0x44, 0x44, 0x3F}, // A
    {0x00, 0x41, 0x7F, 0x41, 0x00}, // I
    {0x7F, 0x08, 0x14, 0x22, 0x41}, // K
    {0x7F, 0x08, 0x08, 0x08, 0x7F}, // H
    {0x7F, 0x49, 0x49, 0x49, 0x41}, // E
    {0x7E, 0x01, 0x01, 0x01, 0x7E}, // U
}

```

```

{0x7F, 0x10, 0x08, 0x04, 0x7F}, // N
{0x3E, 0x41, 0x41, 0x45, 0x26}, // G
{0x7F, 0x08, 0x14, 0x22, 0x41}, // K
{0x00, 0x41, 0x7F, 0x41, 0x00} // I
};

// LedControl 객체 선언
LedControl myLC = LedControl(12,11,10,1);
int numChar;

void setup() {
    // LED의 셋 다운을 해제하고 밝기를 설정하여 모두 지운다.
    myLC.shutdown(0, false);
    myLC.setIntensity(0, 8);
    myLC.clearDisplay(0);
    // 문자열의 문자 개수를 계산한다.
    numChar = sizeof(nameCode) / 5;
}

void loop() {
    // 문자열의 문자를 하나씩 8x8 matrix LED에 나타내고 지운다.
    for (int nc = 0; nc < numChar; nc++) {
        displayChar(nameCode[nc]);
        delay(delayTime);
    }
    myLC.clearDisplay(0);
    delay(delayTime);
}

// 주어진 문자를 8x8 matrix LED에 나타내는 함수
void displayChar(byte charArray[]) {
    for (int col = 0; col < 5; col++) {
        myLC.setColumn(0, col, charArray[col]);
    }
}

```

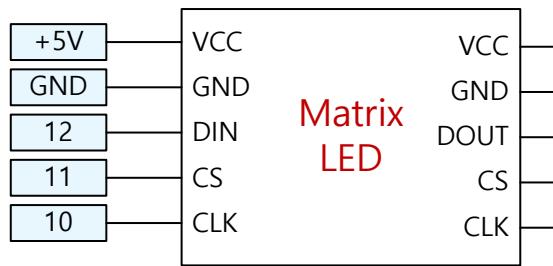
### 예제 9-4-4

#### ▣ 목표

- ✓ 임의의 문자열을 500 μs 간격으로 표시한다. (마지막은 모두 off 한다.)

#### ▣ 회로도

- ✓ 디지털 12 번, 10 번, 11 번 핀에 matrix LED의 DIN, CS, CLK을 연결한다.



## ▶ 스케치 프로그램

- ✓ LedControl.h 라이브러리 사용

```

// 8 x 8 Matrix LED Display
// DIN = 12, CLK = 11, CS = 10
// Library : LedControl.h
// Header file : asciiicode.h

#include <LedControl.h>
// 헤더 파일 포함
#include "asciiicode.h"

const int numCode = 95;
const int offsetCode = 32;
const int delayTime = 500;
// LedControl 객체 선언
LedControl myLC = LedControl(12,11,10,1);
char myChar[] = "Hello, Arduino";
int numChar, codeIndex;

void setup() {
    // LED의 셋 다운을 해제하고 밝기를 설정하며 모두 지운다.
    myLC.shutdown(0, false);
    myLC.setIntensity(0, 8);
    myLC.clearDisplay(0);
    numChar = sizeof(myChar);
}

void loop() {
    // 문자열의 문자를 하나씩 8x8 matrix LED에 나타내고 지운다.
    for (int nc = 0; nc < numChar; nc++) {
        displayChar(myChar[nc]);
        delay(delayTime);
    }
    delay(delayTime);
    myLC.clearDisplay(0);
    delay(delayTime);
}
  
```

```

}

// 주어진 문자를 8x8 matrix LED에 나타내는 함수
void displayChar(char charCode) {
    // 헤더 파일의 문자 코드를 이용하여 주어진 문자의 코드를 나타낸다.
    codeIndex = charCode - offsetCode;
    for (int col = 0; col < 5; col++) {
        myLC.setColumn(0, col+1, asciiCode[codeIndex][col]);
    }
}

```

---

✓ 헤더 파일 (asciiCode.h)

---

```

// 5x7 Font Image of ASCII Code

const byte asciiCode[][][5] = {
    {0x00, 0x00, 0x00, 0x00, 0x00}, // space
    {0x00, 0x00, 0x7D, 0x00, 0x00}, // !
    {0x00, 0x70, 0x00, 0x70, 0x00}, // "
    {0x14, 0x7F, 0x14, 0x7F, 0x14}, // #
    {0x12, 0x2A, 0x7F, 0x2A, 0x24}, // $
    {0x62, 0x64, 0x08, 0x13, 0x23}, // %
    {0x36, 0x49, 0x55, 0x22, 0x05}, // &
    {0x00, 0x50, 0x60, 0x00, 0x00}, // '
    {0x00, 0x1C, 0x22, 0x41, 0x00}, // (
    {0x00, 0x41, 0x22, 0x1C, 0x00}, // )
    {0x08, 0x2A, 0x1C, 0x2A, 0x08}, // *
    {0x08, 0x08, 0x3E, 0x08, 0x08}, // +
    {0x00, 0x05, 0x06, 0x00, 0x00}, // ,
    {0x08, 0x08, 0x08, 0x08, 0x08}, // -
    {0x00, 0x03, 0x03, 0x00, 0x00}, // .
    {0x02, 0x04, 0x08, 0x10, 0x20}, // /
    {0x3E, 0x45, 0x49, 0x51, 0x3E}, // 0
    {0x00, 0x21, 0x7F, 0x01, 0x00}, // 1
    {0x21, 0x43, 0x45, 0x49, 0x31}, // 2
    {0x42, 0x41, 0x51, 0x69, 0x46}, // 3
    {0x0C, 0x14, 0x24, 0x7F, 0x04}, // 4
    {0x72, 0x51, 0x51, 0x51, 0x4E}, // 5
    {0x1E, 0x29, 0x49, 0x49, 0x06}, // 6
    {0x40, 0x47, 0x48, 0x50, 0x60}, // 7
    {0x36, 0x49, 0x49, 0x49, 0x36}, // 8
    {0x30, 0x49, 0x49, 0x4A, 0x3C}, // 9
    {0x00, 0x36, 0x36, 0x00, 0x00}, // :
    {0x00, 0x35, 0x36, 0x00, 0x00}, // ;
}
```

```
{0x00, 0x08, 0x14, 0x22, 0x41}, // <
{0x14, 0x14, 0x14, 0x14, 0x14}, // =
{0x41, 0x22, 0x14, 0x08, 0x00}, // >
{0x20, 0x40, 0x45, 0x48, 0x30}, // ?
{0x26, 0x49, 0x4F, 0x41, 0x3E}, // @@
{0x3F, 0x44, 0x44, 0x44, 0x3F}, // A
{0x7F, 0x49, 0x49, 0x49, 0x36}, // B
{0x3E, 0x41, 0x41, 0x41, 0x22}, // C
{0x7F, 0x41, 0x41, 0x22, 0x1C}, // D
{0x7F, 0x49, 0x49, 0x49, 0x41}, // E
{0x7F, 0x48, 0x48, 0x40, 0x40}, // F
{0x3E, 0x41, 0x41, 0x45, 0x26}, // G
{0x7F, 0x08, 0x08, 0x08, 0x7F}, // H
{0x00, 0x41, 0x7F, 0x41, 0x00}, // I
{0x02, 0x01, 0x41, 0x7E, 0x40}, // J
{0x7F, 0x08, 0x14, 0x22, 0x41}, // K
{0x7F, 0x01, 0x01, 0x01, 0x01}, // L
{0x7F, 0x20, 0x10, 0x20, 0x7F}, // M
{0x7F, 0x10, 0x08, 0x04, 0x7F}, // N
{0x3E, 0x41, 0x41, 0x41, 0x3E}, // O
{0x7F, 0x48, 0x48, 0x48, 0x30}, // P
{0x3E, 0x41, 0x45, 0x42, 0x3D}, // Q
{0x7F, 0x48, 0x4C, 0x4A, 0x31}, // R
{0x31, 0x49, 0x49, 0x49, 0x46}, // S
{0x40, 0x40, 0x7F, 0x40, 0x40}, // T
{0x7E, 0x01, 0x01, 0x01, 0x7E}, // U
{0x7C, 0x02, 0x01, 0x02, 0x7C}, // V
{0x7F, 0x02, 0x0C, 0x02, 0x7F}, // W
{0x63, 0x14, 0x08, 0x14, 0x63}, // X
{0x60, 0x10, 0x0F, 0x10, 0x60}, // Y
{0x43, 0x45, 0x49, 0x51, 0x61}, // Z
{0x00, 0x00, 0x7F, 0x41, 0x41}, // [
{0x20, 0x10, 0x08, 0x04, 0x02}, // inverse slash
{0x41, 0x41, 0x7F, 0x00, 0x00}, // ]
{0x10, 0x20, 0x40, 0x20, 0x10}, // ^
{0x01, 0x01, 0x01, 0x01, 0x01}, // -
{0x00, 0x40, 0x20, 0x10, 0x00}, // `
{0x02, 0x15, 0x15, 0x15, 0x0F}, // a
{0x7F, 0x09, 0x11, 0x11, 0x0E}, // b
{0x0E, 0x11, 0x11, 0x11, 0x02}, // c
{0x0E, 0x11, 0x11, 0x09, 0x7F}, // d
{0x0E, 0x15, 0x15, 0x15, 0x0C}, // e
{0x08, 0x3F, 0x48, 0x40, 0x20}, // f
{0x08, 0x14, 0x15, 0x15, 0x1E}, // g
```

```
{0x7F, 0x08, 0x10, 0x10, 0x0F}, // h  
{0x00, 0x11, 0x5F, 0x01, 0x00}, // i  
{0x02, 0x01, 0x11, 0x5E, 0x00}, // j  
{0x00, 0x7F, 0x04, 0x0A, 0x11}, // k  
{0x00, 0x41, 0x7F, 0x01, 0x00}, // l  
{0x1F, 0x10, 0x0C, 0x10, 0x0F}, // m  
{0x1F, 0x08, 0x10, 0x10, 0x0F}, // n  
{0x0E, 0x11, 0x11, 0x11, 0x0E}, // o  
{0x1F, 0x14, 0x14, 0x14, 0x08}, // p  
{0x08, 0x14, 0x14, 0x0C, 0x1F}, // q  
{0x1F, 0x08, 0x10, 0x10, 0x08}, // r  
{0x09, 0x15, 0x15, 0x15, 0x02}, // s  
{0x10, 0x7E, 0x11, 0x01, 0x02}, // t  
{0x1E, 0x01, 0x01, 0x02, 0x1F}, // u  
{0x1C, 0x02, 0x01, 0x02, 0x1C}, // v  
{0x1E, 0x01, 0x06, 0x01, 0x1E}, // w  
{0x11, 0x0A, 0x04, 0x0A, 0x11}, // x  
{0x18, 0x05, 0x05, 0x05, 0x1E}, // y  
{0x11, 0x13, 0x15, 0x19, 0x11}, // z  
{0x00, 0x08, 0x36, 0x41, 0x00}, // {  
{0x00, 0x00, 0x7F, 0x00, 0x00}, // |  
{0x00, 0x41, 0x36, 0x08, 0x00}, // }  
{0x08, 0x08, 0x2A, 0x1C, 0x08}, // ~  
{0x08, 0x1C, 0x2A, 0x08, 0x08}, // DEL  
};
```

## 연습 문제

### ■ 연습 문제 9-4-1

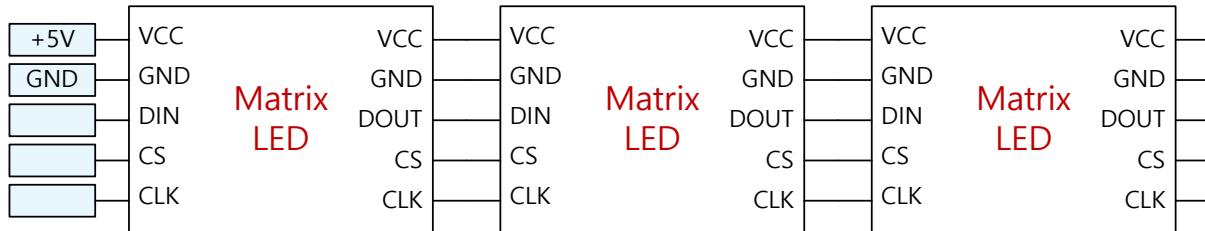
- ✓ 시리얼 모니터에서 문장을 입력하고 그 문장을 matrix LED 에 쓰는 회로를 구성하고 프로그램을 작성하라. 문자의 변환 속도는 적당히 하라.

### ■ 연습 문제 9-4-2

- ✓ 스마트 폰의 Bluetooth Serial Controller 에서 문장을 입력하고 그 문장을 matrix LED 에 쓰는 회로를 구성하고 프로그램을 작성하라. 문자의 변환 속도는 적당히 하라.

### 9-4-3 여러 개의 matrix LED 연결

- 8x8 matrix LED 를 여려 개 연결하여 사용할 수 있다.



- ✓ 5 V 와 접지는 공유하고 DOUT 단자를 다음 단의 DIN 단자에 연결한다.
- ✓ LedControl.h 헤더 파일을 사용하면 8 개의 matrix LED 연결이 가능하다.

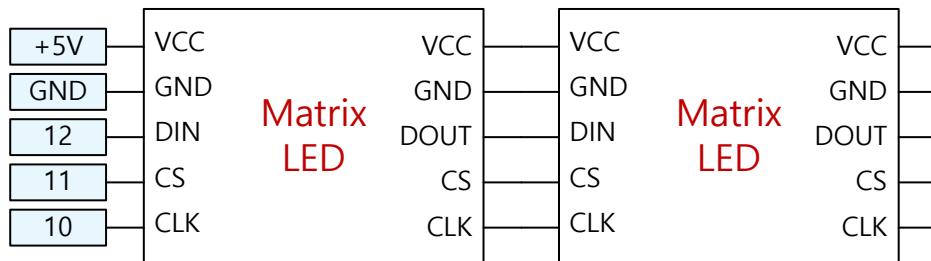
#### 예제 9-4-5

- 목표

- ✓ 2 개의 matrix LED 를 연결하고 8 개의 행과 16 개의 열을 랜덤하게 선택하여 해당 LED 를 20  $\mu$ s 만큼 on 시킨다.

- 회로도

- ✓ 디지털 12 번, 10 번, 11 번 핀에 matrix LED 의 DIN, CS, CLK 을 연결한다.
- ✓ 5 V 와 접지를 공유하고 DOUT 단자를 다음 단의 DIN 단자에 연결한다.



- 스케치 프로그램

- ✓ LedControl.h 라이브러리 사용

```
// 8 x 8 Matrix LED Display
// DIN = 12, CLK = 11, CS = 10
// Library : LedControl.h

#include <LedControl.h>

const int delayTime = 20;
// LedControl 객체 선언
LedControl myLC = LedControl(12, 11, 10, 2);
int nMLed;
```

```

void setup() {
    // 2 개의 LED의 셋 다운을 해제하고 밝기를 설정하여 모두 지운다.
    myLC.shutdown(0, false);
    myLC.setIntensity(0, 8);
    myLC.clearDisplay(0);
    myLC.shutdown(1, false);
    myLC.setIntensity(1, 8);
    myLC.clearDisplay(1);
}

void loop() {
    // 행과 열 번호를 랜덤하게 선택한다.
    int row = random(0, 8);
    int col = random(0, 16);
    // 열 번호가 8~15이면 두 번째 LED를 선택하고 열 번호를 0~7로 바꾼다.
    if (col >= 8) {
        nMLed = 1;
        col -= 8;
    }
    else {
        nMLed = 0;
    }
    // 선택한 행과 열의 LED를 켰다가 일정 시간 후 끈다.
    myLC.setLed(nMLed, row, col, true);
    delay(delayTime);
    myLC.setLed(nMLed, row, col, false);
}

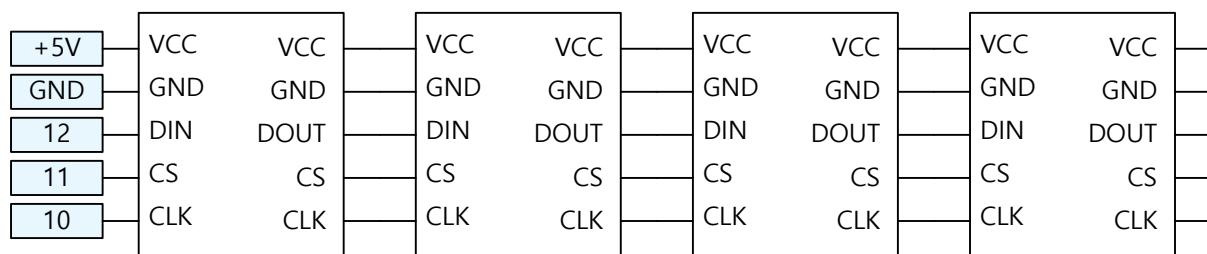
```

**예제 9-4-6****▣ 목표**

- ✓ 4 개의 matrix LED 를 연결하고 4 개의 문자로 구성된 주어진 임의의 문자열을 나타낸다.

**▣ 회로도**

- ✓ 디지털 12 번, 10 번, 11 번 핀에 matrix LED 의 DIN, CS, CLK 을 연결한다.
- ✓ 5 V 와 접지를 공유하고 DOUT 단자를 다음 단의 DIN 단자에 연결한다.



### ▶ 스케치 프로그램

- ✓ LedControl.h 라이브러리 사용

```
// 8 x 8 Matrix LED Display
// DIN = 12, CLK = 11, CS = 10
// Library : LedControl.h
// Header file : asciiicode.h

#include <LedControl.h>
// 헤더 파일 포함
#include "asciiicode.h"

const int numCode = 95;
const int offsetCode = 32;
const int numMLed = 4;
// LedControl 객체 선언
LedControl myLC = LedControl(12, 11, 10, numMLed);
char myChar[] = "BAIK";
int numChar;

void setup() {
    // LED의 셋 다운을 해제하고 밝기를 설정하여 모두 지운다.
    for (int nc = 0; nc < numMLed; nc++) {
        myLC.shutdown(nc, false);
        myLC.setIntensity(nc, 8);
        myLC.clearDisplay(nc);
    }
    // 문자 열의 문자 수를 계산한다.
    numChar = sizeof(myChar);
}

void loop() {
    // 문자 열의 문자를 8x8 matrix LED에 나타낸다.
    for (int nc = 0; nc < numChar; nc++) {
        displayChar(nc, myChar[numChar-nc-1]);
    }
}
```

```
// 주어진 문자를 8x8 matrix LED에 나타내는 함수
void displayChar(int noMLed, char charCode) {
    // 헤더 파일의 문자 코드를 이용하여 주어진 문자의 코드를 나타낸다.
    int codeIndex = charCode - offsetCode;
    for (int col = 0; col < 5; col++) {
        myLC.setColumn(noMLed, col+1, asciiCode[codeIndex][col]);
    }
}
```

---

## 10. 스펙트럼 분석

### 10.1 MSGEQ7 IC 를 이용한 스펙트럼 분석

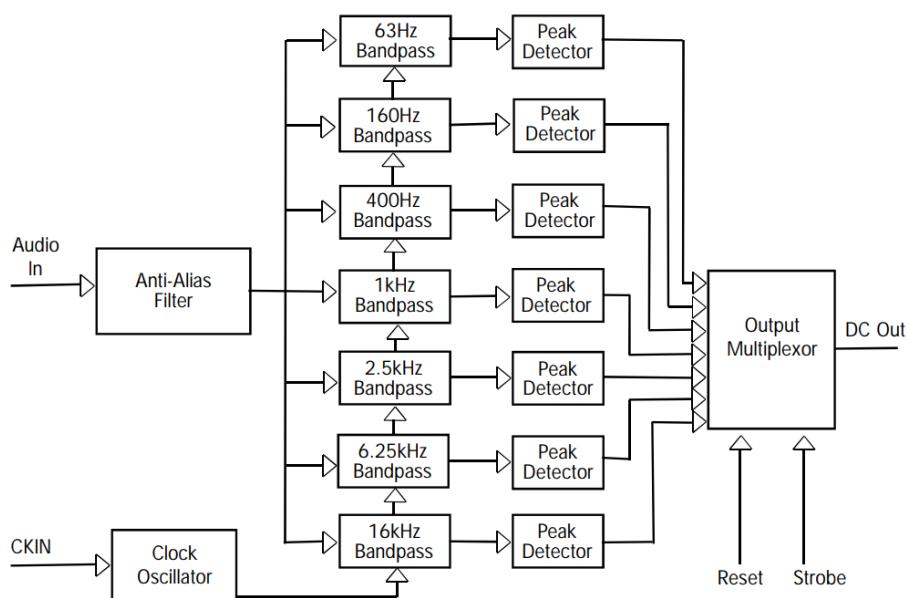
- ✚ MSGEQ7 IC 를 사용하여 오디오 신호의 스펙트럼을 분석한다.

#### 10.1.1 MSGEQ7 IC

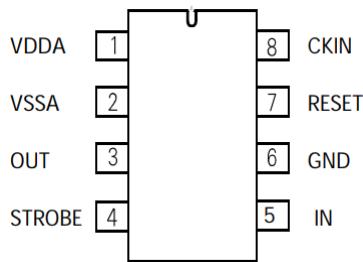
- ✚ MSGEQ7 IC



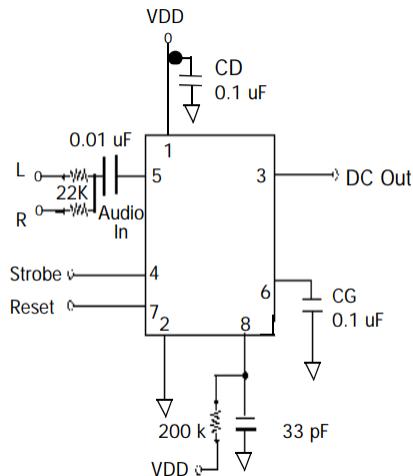
- ✓ 7 band graphic equalizer
  - ✖ 밴드 중간 주파수: 63 Hz, 160 Hz, 400 Hz, 1000 Hz, 2500 Hz, 6250 Hz, 16000 Hz
  - ✖ 7 개 대역의 피크값을 순차적으로 출력한다.
  - ✖ 동작 전압 : 3.3 V, 5 V (아두이노 전압 사용 가능)
- ✓ 블록도



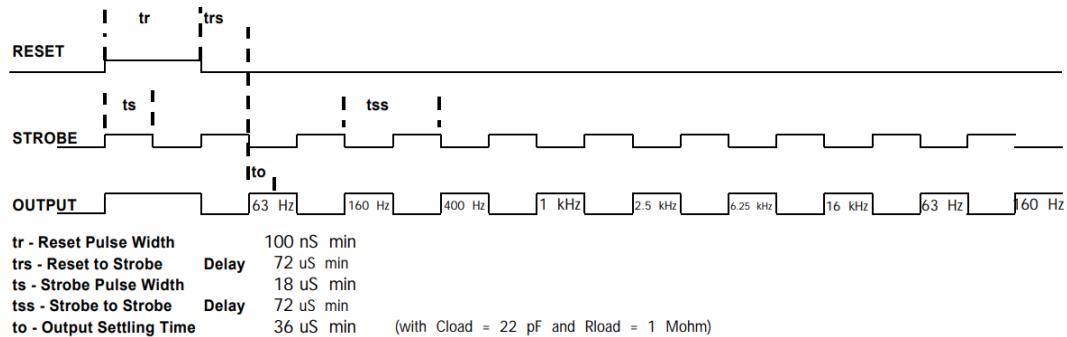
✓ 핀 배치도



✓ MSGEQ7 회로도



✓ Strobe timing diagram



### 10.1.2 MSGEQ7 을 이용한 스펙트럼 분석

✚ 아두이노를 이용하여 RESET, STROBE 신호를 발생시키고 스펙트럼 읽는 방법

- ✓ 초기화 : RESET = 0, STROBE = 0
- ✓ RESET : tr(100 ns 이상) 동안 1을 유지한 후 0으로 한다.
- ✓ STROBE : RESET 이 0 이 되고 trs(72  $\mu$ s 이상) 후 STROBE 를 0 으로 한다.
- ✓ STROBE 가 0 이 되고 to(36  $\mu$ s 이상) 후 값을 읽는다.
- ✓ 값을 읽은 후 STROBE 를 1 로 한다. 출력은 0 이 된다.

- ✓ STROBE가 0이 되고 tss(72 μs 이상) 후 다시 STROBE를 다시 0으로 한다.  
이 과정을 7 번 반복한다.
- ✓ 다시 초기화 하고 위의 과정을 반복한다.

 Data sheet

- ✓ <https://www.sparkfun.com/datasheets/Components/General/MSGEQ7.pdf>

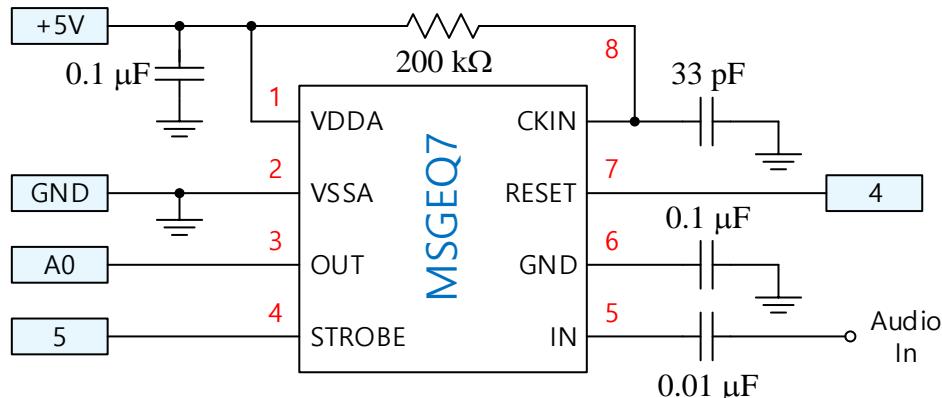
**예제 10-1-1**

 목표

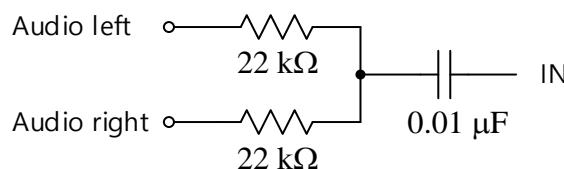
- ✓ MSGEQ7 을 사용하여 오디오 신호의 스펙트럼을 구하고 이 값을 시리얼 모니터에 나타낸다.
- ✓ 아두이노를 이용하여 RESET, STROBE 신호를 발생시켜 신호의 스펙트럼을 얻는다.

 회로도

- ✓ MSGEQ7 의 전원과 접지 단자를 아두이노 +5 V, GND 판에 연결한다.
- ✓ MSGEQ7 의 STROBE, RESET 단자를 아두이노 디지털 5, 4 번 핀에 연결한다.
- ✓ MSGEQ7 의 OUT 단자를 아두이노 아날로그 입력 A0 에 연결한다.



- ✓ 오디오 신호가 스테레오일 경우 다음과 같이 모노 신호로 변환하여 인가한다.



 스케치 프로그램

```
// Spectrum Analyzer using MSGEQ7 (1 CH)
// Reset(p7) of MSGEQ7 : D4
// Strobe(p4) of MSGEQ7 : D5
// Audio_In(p5) of MSGEQ7
// Data_Out(p3) : A0

// 아두이노 연결 핀
const int resetPin = 4;
const int strobePin = 5;
const int sigPin = A0;
const int trsTime = 72;
const int tssTime = 72;
const int toTime = 36;
int sigBand[7];
int nb;

void setup()
{
    Serial.begin(9600);
    // STROBE, RESET 핀을 디지털 출력으로 설정한다.
    pinMode(resetPin, OUTPUT);
    pinMode(strobePin, OUTPUT);
    // STROBE, RESET 신호를 초기화 한다.
    digitalWrite(resetPin, LOW);
    digitalWrite(strobePin, HIGH);
}

void loop()
{
    // 7 개 대역의 스펙트럼을 읽고 값을 시리얼 모니터에 나타낸다.
    readMSGEQ7();
    for (nb = 0; nb < 7; nb++) {
        Serial.print(sigBand[nb]);
        Serial.print(" ");
    }
    Serial.println();
}

// 7 개 대역의 스펙트럼을 읽는 함수
void readMSGEQ7() {
    // RESET 신호를 발생시킨다. (tr=100 ns로 짧아 바로 LOW를 내 보낸다.)
    digitalWrite(resetPin, HIGH);
    digitalWrite(resetPin, LOW);
    // RESET 신호가 0이되고 72 μs(trs) 후 STROBE 신호를 0으로 한다.
```

```

delayMicroseconds(trsTime);
// 7 개 대역의 스펙트럼 값을 읽는다.
for (nb = 0; nb < 7; nb++) {
    // STROBE 신호를 0으로 하고 36 µs(to) 후 스펙트럼 값을 읽는다.
    digitalWrite(strobePin, LOW);
    delayMicroseconds(toTime);
    sigBand[nb] = analogRead(sigPin);
    // STROBE 신호를 1로 하고 36 µs(tss-to) 지연시킨다.
    digitalWrite(strobePin, HIGH);
    delayMicroseconds(tssTime-toTime);
}
}

```

- ✓ 스펙트럼이 적절하게 출력되지 않으면 delayStrobe 값을 증가시킨다.

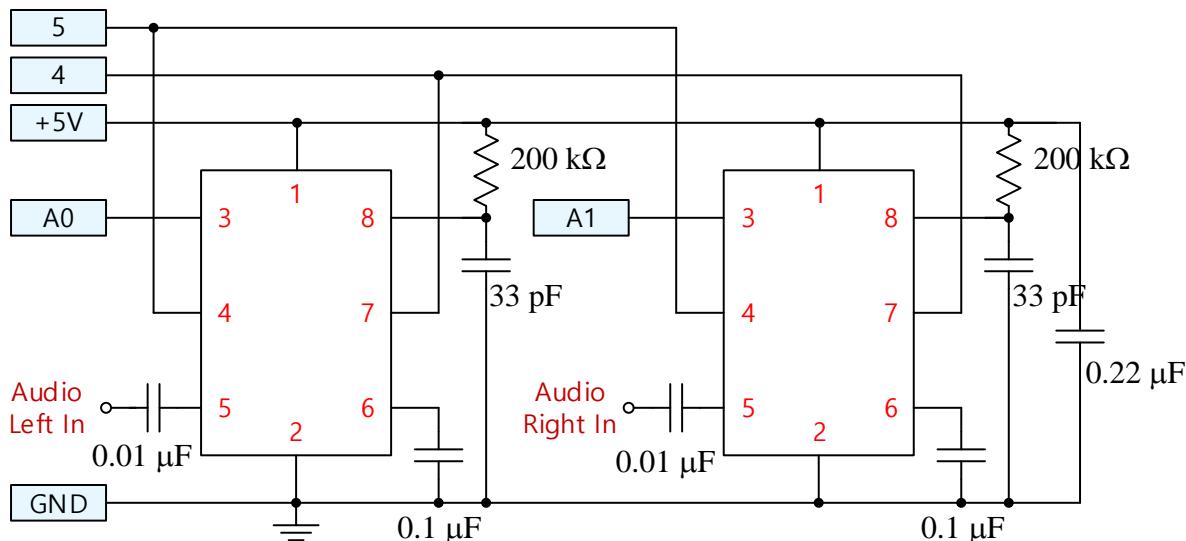
### 예제 10-1-2

#### ▣ 목표

- ✓ 2 개의 MSGEQ7 을 사용하여 스테레오 오디오 신호의 스펙트럼을 구하고 이 값을 시리얼 모니터에 나타낸다.
- ✓ 아두이노를 이용하여 RESET, STROBE 신호를 발생시켜 신호의 스펙트럼을 얻는다.

#### ▣ 회로도

- ✓ MSGEQ7 의 전원과 접지 단자를 아두이노 +5V, GND 핀에 연결한다.
- ✓ MSGEQ7 의 STROBE, RESET 단자를 아두이노 디지털 5, 4 번 핀에 연결한다.
- ✓ MSGEQ7 의 OUT 단자를 아두이노 아날로그 입력 A0 와 A1 에 연결한다.



 스케치 프로그램

```
// Spectrum Analyzer using MSGEQ7 (1 CH)
// Reset(p7) of MSGEQ7 : D4
// Strobe(p4) of MSGEQ7 : D5
// Audio_In(p5) of MSGEQ7
// Data_Out(p3) : A0

// 아두이노 연결 핀
const int resetPin = 4;
const int strobePin = 5;
const int leftSigPin = A0;
const int rightSigPin = A1;
const int trsTime = 72;
const int tssTime = 72;
const int toTime = 36;
int leftBand[7];
int rightBand[7];
int nb;

void setup()
{
    Serial.begin(9600);
    // STROBE, RESET 핀을 디지털 출력으로 설정한다.
    pinMode(resetPin, OUTPUT);
    pinMode(strobePin, OUTPUT);
    // STROBE, RESET 신호를 초기화 한다.
    digitalWrite(resetPin, LOW);
    digitalWrite(strobePin, HIGH);
}

void loop()
{
    // 7 개 대역의 스펙트럼을 읽고 값을 시리얼 모니터에 나타낸다.
    readMSGEQ7();
    for (nb = 0; nb < 7; nb++) {
        Serial.print(leftBand[nb]);
        Serial.print(" ");
    }
    for (nb = 0; nb < 7; nb++) {
        Serial.print(rightBand[nb]);
        Serial.print(" ");
    }
    Serial.println();
}
```

```

}

// 7 개 대역의 스펙트럼을 읽는 함수
void readMSGEQ7() {
    // RESET 신호를 발생시킨다. (tr=100 ns로 짧아 바로 LOW를 내 보낸다.)
    digitalWrite(resetPin, HIGH);
    digitalWrite(resetPin, LOW);
    // RESET 신호가 0이되고 72 µs(trs) 후 STROBE 신호를 0으로 한다.
    delayMicroseconds(trsTime);
    // 7 개 대역의 스펙트럼 값을 읽는다.
    for (nb = 0; nb < 7; nb++) {
        // STROBE 신호를 0으로 하고 36 µs(to) 후 스펙트럼 값을 읽는다.
        digitalWrite(strobePin, LOW);
        delayMicroseconds(toTime);
        leftBand[nb] = analogRead(leftSigPin);
        rightBand[nb] = analogRead(rightSigPin);
        // STROBE 신호를 1로 하고 36 µs(tss-to) 지연시킨다.
        digitalWrite(strobePin, HIGH);
        delayMicroseconds(tssTime-toTime);
    }
}

```

### 10.1.3 MSGEQ7 라이브러리를 이용한 스펙트럼 분석

#### ■ 라이브러리 1

**MD\_MSGEQ7.h**

- ✓ MSGEQ7 을 제어하기 위한 라이브러리
- ✓ [https://github.com/MajicDesigns/MD\\_MSGEQ7](https://github.com/MajicDesigns/MD_MSGEQ7)

#### ■ 관련 함수

**MD\_MSGEQ7(resetPin, strobePin, dataPin)**

- \* MD\_MSGEQ7 객체를 생성하는데 사용한다.
- \* resetPin : MSGEQ7의 RESET 단자에 연결하는 디지털 출력 핀
- \* strobePin : MSGEQ7의 STROBE 단자에 연결하는 디지털 출력 핀
- \* dataPin : MSGEQ7의 OUT 단자에 연결하는 아날로그 입력 핀

**.begin();**

- \* 스펙트럼을 읽기 위해 MSGEQ7을 준비한다.

**.reset();**

- \* MSGEQ7에서 스펙트럼 값을 읽기 위해 초기화 시킨다.

- ✖ 중간에 다시 밴드 0 (63 Hz)부터 데이터를 읽을 때 사용한다.
- ✖ 일반적으로 .read() 함수를 부를 때 실행하기 때문에 별도로 실행할 필요 없다.

**.read(bReset);**

- ✖ MSGEQ7에서 7 개 대역의 스펙트럼 값을 읽어 버퍼에 저장한다.
- ✖ bReset : 데이터를 읽기 전에 .reset() 함수를 실행할지 여부 선택, 옵션 (default=true)

**.get(band);**

- ✖ 버퍼에 있는 데이터를 읽는다.
- ✖ band : (0~7), 주파수 대역 선택 변수 (0: 63 Hz, 1: 160 Hz, . . . , 7: 2.5 kHz)

## ▣ 라이브러리 2

**MSGEQ7.h**

- ✓ MSGEQ7 을 제어하기 위한 라이브러리
- ✓ 7 개의 대역 주파수의 스펙트럼 값을 8 비트(0~255 사이의 값)로 읽는다.
- ✓ 모노와 스테레오 신호 적용 가능 (여러 개의 MSGEQ7 사용 가능)
- ✓ 읽은 값의 평활화(smoothing) 가능
- ✓ <https://github.com/NicoHood/MSGEQ7>

## ▣ 관련 함수

**CMSGEQ7<smooth, resetPin, strobePin, dataPin1, . . . , dataPinn>;**

- ✖ MSGEQ7 클래스를 생성하는데 사용
- ✖ smooth : 읽은 값을 평활화(smoothing)하는 변수 (0: 평활화 없음, 255: 100% 평활화, 평활화가 필요하면 191~233 추천)
- ✖ resetPin : MSGEQ7의 RESET 단자에 연결하는 디지털 출력 핀
- ✖ strobePin : MSGEQ7의 STROBE 단자에 연결하는 디지털 출력 핀
- ✖ dataPin1 : 1 번 MSGEQ7의 OUT 단자에 연결하는 아날로그 입력 핀
- ✖ dataPinn : n 번 MSGEQ7의 OUT 단자에 연결하는 아날로그 입력 핀

**.begin();**

- ✖ 스펙트럼을 읽기 위해 MSGEQ7을 준비한다.

**.reset();**

- ✖ MSGEQ7에서 스펙트럼 값을 읽기 위해 초기화 시킨다.
- ✖ 중간에 다시 밴드 0 (63 Hz)부터 데이터를 읽을 때 사용한다.
- ✖ 일반적으로 .read() 함수를 부를 때 실행하기 때문에 별도로 실행할 필요 없다.

`.read();`

`.read(interval);`

- ✖ MSGEQ7에서 7 개 대역의 스펙트럼 값을 읽어 버퍼에 저장한다.

- ✖ 2 채널의 경우 아두이노 우노 (16 MHz)에서 400FPS 가능

- ✖ interval : 주어진 시간 간격으로 데이터 값을 읽는다. (옵션)

`.get(band);`

`.get(band, channel);`

- ✖ 버퍼에 있는 데이터를 읽는다.

- ✖ 8 비트 (0~255 레벨) 값으로 읽는다.

- ✖ band : (0~7), 주파수 대역 선택 변수 (0: 63 Hz, 1: 160 Hz, . . . , 7: 2.5 kHz)

- ✖ channel : MSGEQ7이 여러 개 있을 때 선택하는 값 (1개면 옵션)

`.end();`

- ✖ MSGEQ7에 할당한 핀을 입력으로 설정하여 연결을 끊는다.

- ✖ 일반적으로 불필요하다.

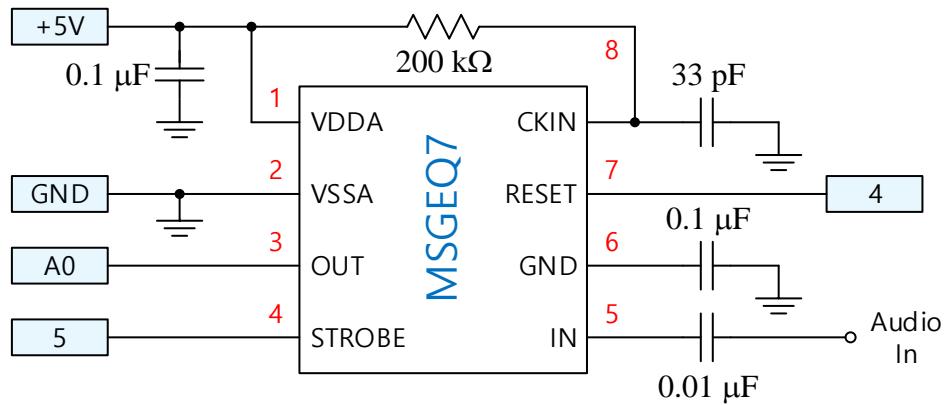
### 예제 10-1-3

#### ▣ 목표

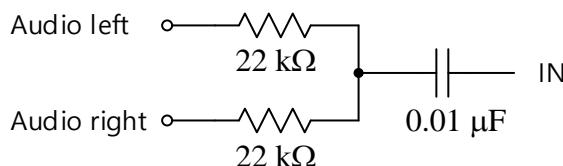
- ✓ MSGEQ7 을 사용하여 오디오 신호의 스펙트럼을 구하고 이 값을 시리얼 모니터에 나타낸다.
- ✓ 50 μs 마다 스펙트럼 값을 읽어라.
- ✓ 라이브러리 MD\_MSGEQ7.h 를 사용하라.

#### ▣ 회로도

- ✓ MSGEQ7 의 전원과 접지 단자를 아두이노 +5 V, GND 핀에 연결한다.
- ✓ MSGEQ7 의 STROBE, RESET 단자를 아두이노 디지털 5, 4 번 핀에 연결한다.
- ✓ MSGEQ7 의 OUT 단자를 아두이노 아날로그 입력 A0 에 연결한다.



- ✓ 오디오 신호가 스테레오일 경우 다음과 같이 모노 신호로 변환하여 인가한다.



### ▶ 스케치 프로그램

- ✓ MD\_MSGEQ7.h 사용

```
// Mono Spectrum Analyzer
// Library : MD_MSGEQ7

#include <MD_MSGEQ7.h>

// 아두이노 연결 핀
#define DATA_PIN A0
#define RESET_PIN 4
#define STROBE_PIN 5

// MD_MSGEQ7 객체 선언
MD_MSGEQ7 myMSG(RESET_PIN, STROBE_PIN, DATA_PIN);
// 데이터를 읽는 시간 간격 선언 (50 μs)
const int sampTime = 50;

void setup() {
    // MSGEQ7 준비
    myMSG.begin();
    Serial.begin(57600);
    Serial.println("[MD_MSGEQ7 Serial]");
}
```

```
void loop() {
    static long prevTime = 0;
    // 전에 데이터를 읽었던 시간부터 지금까지의 시간 차가 sampTime보다
    // 클 때만 데이터를 읽는다. (작으면 아무 일도 하지 않는다.)
    if (millis() - prevTime >= sampTime) {
        // 데이터를 읽을 때 현재 시간을 저장한다.
        prevTime = millis();
        // 데이터를 읽는다.
        myMSG.read();
        // 7 개 대역의 값을 꺼내 시리얼 모니터에 나타낸다.
        for (int nb = 0; nb < 7; nb++) {
            Serial.print(myMSG.get(nb));
            Serial.print('\t');
        }
        Serial.println();
    }
}
```

- ✓ 위의 스케치 프로그램은 sampTime 시간마다 데이터를 읽는다.

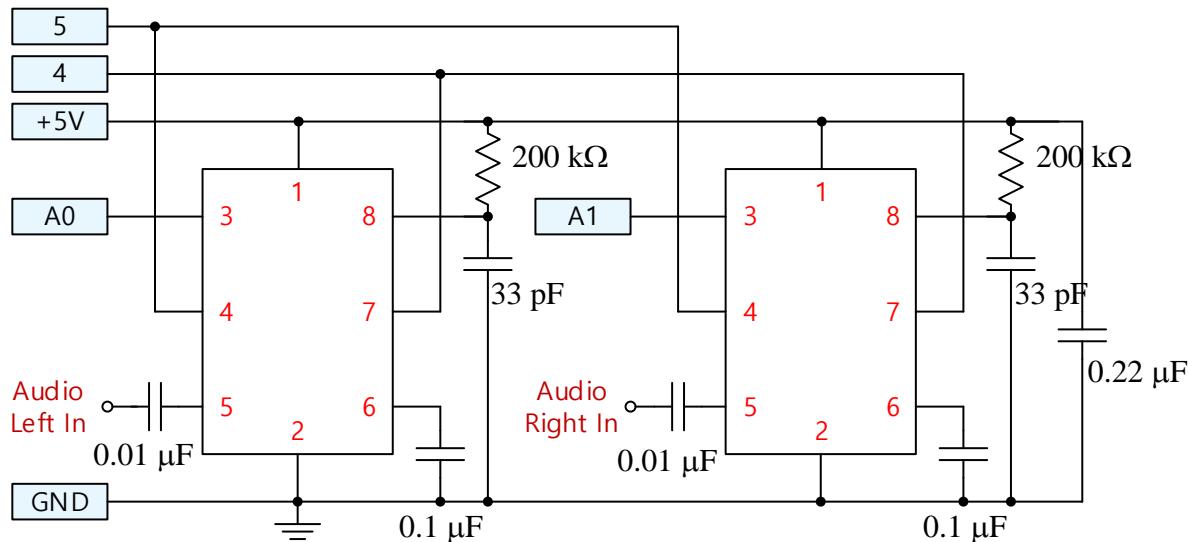
#### 예제 10-1-4

##### ▣ 목표

- ✓ 2 개의 MSGEQ7 을 사용하여 스테레오 오디오 신호의 스펙트럼을 구하고 이 값을 시리얼 모니터에 나타낸다.
- ✓ 50 µs 마다 스펙트럼 값을 읽어라.
- ✓ 라이브러리 MSGEQ7.h 를 사용하라.

##### ▣ 회로도

- ✓ MSGEQ7 의 전원과 접지 단자를 아두이노 +5 V, GND 핀에 연결한다.
- ✓ MSGEQ7 의 STROBE, RESET 단자를 아두이노 디지털 5, 4 번 핀에 연결한다.
- ✓ MSGEQ7 의 OUT 단자를 아두이노 아날로그 입력 A0 와 A1 에 연결한다.



## 스케치 프로그램

- ### ✓ MSGEQ7.h 사용

```
// Stereo Spectrum Analyzer
// Library : MSGEQ7

#include <MSGEQ7.h>

// 아두이노 연결 핀
#define leftDataPin A0
#define rightDataPin A1
#define resetPin 4
#define strobePin 5
// 데이터를 평활화 하기 위한 변수 값
#define SMOOTH 191

// MSGEQ7 클래스 선언 (MSGEQ7 2 개 사용)
CMSGEQ7<SMOOTH, resetPin, strobePin, leftDataPin, rightDataPin> myMSG;
// 데이터를 읽는 시간 간격 선언 (50 μs)
const int sampTime = 50;

void setup() {
    // MSGEQ7 준비
    myMSG.begin();
    Serial.begin(115200);
    Serial.println("[MSGEQ7 Serial]");
}

void loop() {
    // readDelay 시간마다 데이터를 읽는다.
```

```
bool newReading = myMSG.read(sampTime);
// 0 번 채널의 7 개 대역의 값을 꺼내 시리얼 모니터에 나타낸다.
for (int nb = 0; nb < 7; nb++) {
    Serial.print(myMSG.get(nb, 0));
    Serial.print('\t');
}
Serial.println();
// 1 번 채널의 7 개 대역의 값을 꺼내 시리얼 모니터에 나타낸다.
for (int nb = 0; nb < 7; nb++) {
    Serial.print(myMSG.get(nb, 1));
    Serial.print('\t');
}
Serial.println();
Serial.println();
}
```

## 연습 문제

### 색연습 문제 10-1-1

- ✓ MSGEQ7을 사용하여 오디오 신호의 스펙트럼을 구하고 스펙트럼의 크기에 따라 matrix-LED 에 7 개의 스펙트럼을 세로 크기로 나타내는 회로를 구성하고 프로그램을 작성하라. 라이브러리 MD\_MSGEQ7.h 를 사용하라.

### 색연습 문제 10-1-2

- ✓ 2 개의 MSGEQ7 을 사용하여 스테레오 오디오 신호의 스펙트럼을 구하고 스펙트럼의 크기에 따라 2 개의 matrix-LED 에 7 개의 스펙트럼을 세로 크기로 나타내는 회로를 구성하고 프로그램을 작성하라. 라이브러리 MSGEQ7.h 를 사용하라.

## 10.2 FFT 를 이용한 스펙트럼 분석

### ▣ 아두이노 FFT 라이브러리

- ✓ ArduinoFFT 라이브러리
  - ✗ Fast Fourier transform
  - ✗ C 언어로 구성되어 있다.
  - ✗ <https://github.com/kosme/arduinoFFT>
- ✓ FFT 라이브러리
  - ✗ Fast Fourier transform
  - ✗ AVR로 구성되어 있다.
  - ✗ 데이터가 복소수일 때 사용한다.
  - ✗ <http://wiki.openmusiclabs.com/wiki/ArduinoFFT>
- ✓ FHT 라이브러리
  - ✗ Fast Hartley transform
  - ✗ AVR로 구성되어 있다.
  - ✗ FFT 라이브러리와 사용 방법이 같다.
  - ✗ 데이터가 실수일 때 사용한다.
  - ✗ FFT 라이브러리에 비해 계산량과 메모리 사용이 절반이다.
  - ✗ <http://wiki.openmusiclabs.com/wiki/ArduinoFHT>

### ▣ 라이브러리 속도 비교

- ✓ FFT 크기가 128 일 때 함수 별 속도 비교

|           | ArduinoFFT          | FHT                | FFT                |
|-----------|---------------------|--------------------|--------------------|
| Windowing | 11864 $\mu\text{s}$ | 288 $\mu\text{s}$  | 304 $\mu\text{s}$  |
| Run       | 46060 $\mu\text{s}$ | 1423 $\mu\text{s}$ | 2783 $\mu\text{s}$ |
| Abs       | 7388 $\mu\text{s}$  | 287 $\mu\text{s}$  | 486 $\mu\text{s}$  |

### ▣ 라이브러리 장단점

- ✓ ArduinoFFT : 프로그램 작성이 쉽지만 속도가 느리다.
- ✓ FFT, FHT : 속도가 빠르지만 프로그램 작성이 어렵다.

### 10.2.1 ArduinoFFT 라이브러리

✚ ArduinoFFT 라이브러리

**arduinoFFT.h**

- ✓ FFT 를 실행하는 아두이노 라이브러리
- ✓ FFT 크기 : 16~128 (최대 주파수 대역 : 128)
- ✓ <https://github.com/kosme/arduinoFFT>

✚ 관련 함수

**arduinoFFT();**

- ✖ arduinoFFT 객체

**.Windowing(vData, samples, windowType, dir);**

- ✖ FFT를 하기 전에 창 함수를 곱해주는 함수
- ✖ vData[] : 창 함수를 곱해주는 신호 배열 (double), 크기는 samples
- ✖ samples : FFT 크기
- ✖ WindowType : 창 함수 종류 (아래는 예약어)
  - FFT\_WIN\_TYP\_RECTANGLE : 구형 창
  - FFT\_WIN\_TYP\_HAMMING : 해밍(hamming) 창
  - FFT\_WIN\_TYP\_HANN : 해닝(hann) 창
  - FFT\_WIN\_TYP\_TRIANGLE : 삼각 창, 바틀릿(bartlett) 창
  - FFT\_WIN\_TYP\_BLACKMAN : 블랙만(blackman) 창
  - FFT\_WIN\_TYP\_FLT\_TOP : flat top 창
  - FFT\_WIN\_TYP\_WELCH : welch 창
- ✖ dir : FFT, IFFT를 결정하는 값
  - FFT\_FORWARD : FFT를 선택하는 예약어
  - FFT\_BACKWARD : IFFT를 선택하는 예약어

**.Compute(vReal, vImag, samples, dir);**

- ✖ FFT를 수행하는 주 함수
- ✖ vReal[] : 실수 입력 데이터를 저장하는 배열 (double), 크기는 samples
- ✖ vImag[] : 허수 입력 데이터를 저장하는 배열 (double), 크기는 samples
- ✖ samples : FFT 크기 (uint16\_t), 16 ~ 128 사이의 2의 지수 승 정수 (16, 32, 64, 128) (uint16\_t)
- ✖ dir : FFT, IFFT를 결정하는 값

- FFT\_FORWARD : FFT를 선택하는 예약어
  - FFT\_BACKWARD : IFFT를 선택하는 예약어
- .ComplexToMagnitude(vReal, vImag, samples);**
- ✖ 복소수 값을 크기 (magnitude)로 변환하는 함수
  - ✖ vReal[], vImag[] : 복소수 데이터 배열, 크기를 vReal[]에 저장한다.
  - ✖ samples : FFT 크기
- .DCRemoval(vData, samples);**
- ✖ 직류 성분을 제거하는 함수
  - ✖ vData[] : 입력과 출력이 저장되는 배열 변수, 크기는 samples
- .MajorPeak(vData, samples, sampFreq);**
- ✖ FFT를 실행한 후 결과로부터 최대값을 나타내는 주파수를 알려 주는 함수
  - ✖ vData[] : FFT를 실행한 값의 크기(magnitude)를 저장한 배열 변수
  - ✖ sampFreq : 샘플링 주파수 (double)

### 예제 10-2-1

#### ▣ 목표

- ✓ 주파수가 200 Hz 인 사인파를 1 kHz 로 샘플링한 신호와 백색 잡음을 섞은 신호의 스펙트럼을 구한다.
- ✓ 스펙트럼을 시리얼 통신으로 송신하고 시리얼 플로터와 SerialPlot 프로그램에서 받아 스펙트럼을 그린다.
- ✓ FFT 크기는 128 로 한다.

#### ▣ 회로도

- ✓ 없음

#### ▣ 스케치 프로그램

- ✓ arduinoFFT.h 사용

---

```
// Spectrum Estimation
// Library : ArduinoFFT
// FFT size : 128

#include <arduinoFFT.h>
// FFT 크기
```

```
#define FFT_N 128

// 샘플링 주파수
const float FS = 1000;
float F0 = 200;
float w0;
// arduinoFFT 객체 선언
arduinoFFT FFT = arduinoFFT();
// 입력 데이터가 저장되는 변수
double xReal[FFT_N], xImag[FFT_N];
// 랜덤값을 자세하게 구해주기 위해 곱해지는 값
float Ax = 10000;
// 신호와 잡음의 비 값
float SNR = 0.2;

void setup() {
    Serial.begin(115200);
    w0 = 2 * PI * F0 / FS;
}

void loop() {
    // 입력 데이터를 얻는다.
    getData();
    // 해밍 창을 써운다.
    FFT.Windowing(xReal,FFT_N, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
    // FFT를 실행한다.
    FFT.Compute(xReal, xImag, FFT_N, FFT_FORWARD);
    // FFT 결과가 복소수이므로 크기를 계산한다.
    FFT.ComplexToMagnitude(xReal, xImag, FFT_N);
    // 결과를 시리얼 통신으로 보낸다.
    printData();
    // 위의 과정을 실행한 후 계속 헛돌아 1번만 실행하게 한다.
    while(1);
}

// 입력 신호를 만드는 함수
void getData() {
    for (int n = 0; n < FFT_N; n++) {
        // 사인파 신호
        float x = sin(w0 * n);
        // 잡음 신호
        float xn = random(-Ax, Ax) / Ax;
        xReal[n] = x + SNR * xn;;
        xImag[n] = 0;
```

```

    }
}

// FFT 결과를 시리얼 통신으로 보내는 함수
void printData() {
    for (int k = 0; k < FFT_N/2; k++) {
        Serial.println(xReal[k], 2);
    }
}

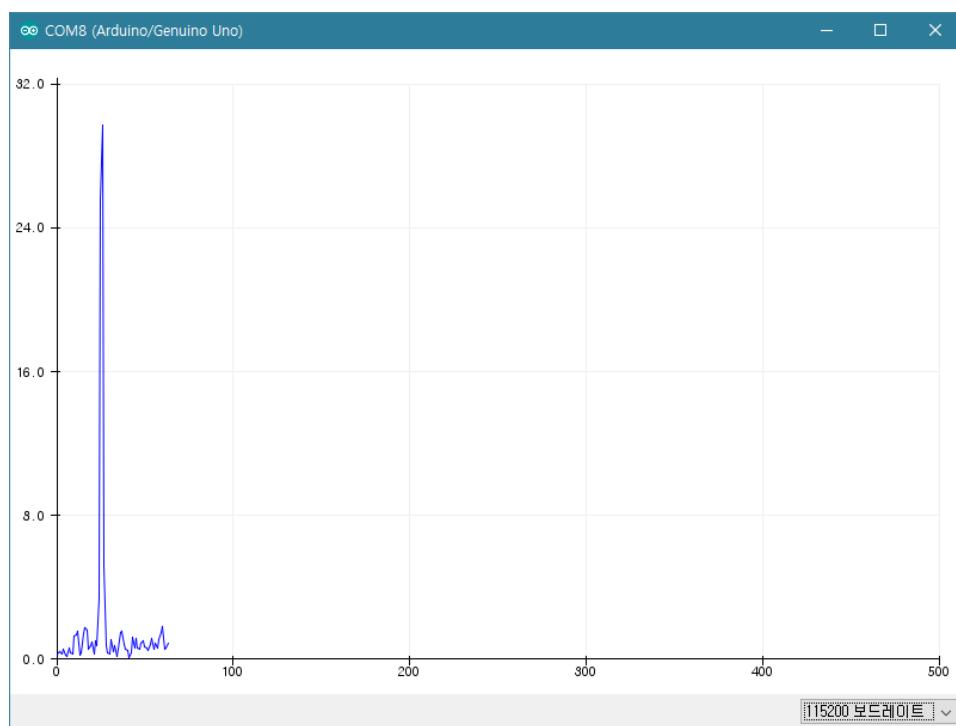
```

### 스펙트럼 파형

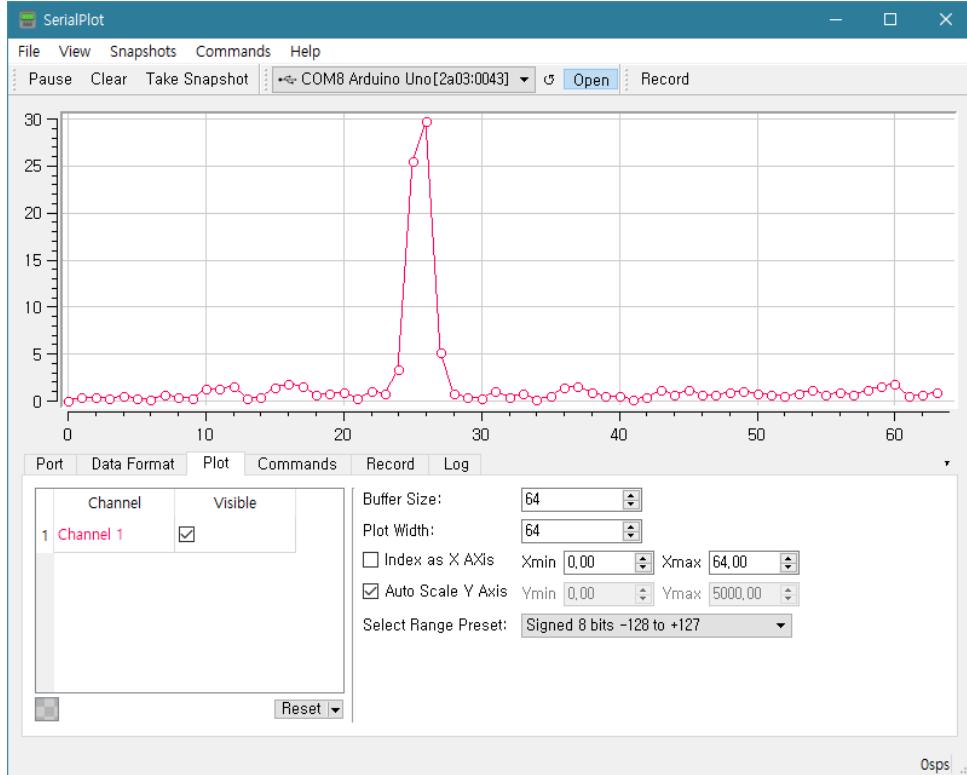
- ✓ 실수 신호의 스펙트럼은 우대칭이므로 절반인 64 개만 그린다. (0~500 Hz)
- ✓ 일반적으로 가로 축이 시간인 경우는 흘러가는 파형을 그리지만 가로 축이 주파수인 경우 64 개의 데이터를 받아 계속 같은 자리에 그려야 한다.
- ✓ loop() 가 반복되면 데이터가 계속 출력되어 시리얼 플로터에 출력하면 그림이 계속 흘러간다. 따라서 while(1)을 사용하여 한 번만 데이터를 출력하고 프로그램이 계속 헛돌아 가게 한다.
- ✓ while(1) 대신 delay(1000)으로 바꾸면 1 초에 한 번씩 그림을 갱신할 수 있어 계속 스펙트럼을 관찰할 수 있다.

### 출력 파형

- ✓ 시리얼 플로터 출력



- ✖ 가로 축의 값을 변경할 수 없어 파형이 좁게 나왔다.
- ✓ SerialPlot 출력
- ✖ SerialPlot은 가로 축의 값을 조정할 수 있다.



- ✖ Port 탭에서 Port와 Baud rate를 조정한다.
- ✖ Data Format 탭에서 ASCII를 선택한다.
- ✖ Plot 탭에서 Buffer Size와 Plot Width를 64로 선택하면 64 개의 데이터를 볼 수 있다. Xmax도 64로 설정한다.
- ✖ FFT 크기가 128인 경우 인덱스 128이 샘플링 주파수 1 kHz에 해당된다. 200 Hz에 해당하는 인덱스는 다음과 같다.

$$k = 128 \times \frac{200}{1000} = 25.6$$

- ✖ 따라서  $k = 25, 26$  일 때가 크기가 최대임을 알 수 있다.

### 예제 10-2-2

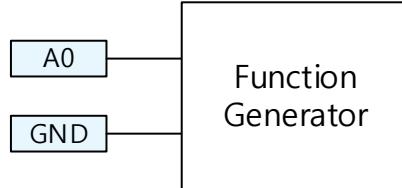
#### ▣ 목표

- ✓ 함수 발생기에서 주파수가 200 Hz, 진폭이 5 V, 직류 오프셋이 2.5 V 인 사인파를 발생시키고 1 kHz로 샘플링한 신호의 스펙트럼을 구한다.

- ✓ 스펙트럼을 시리얼 통신으로 송신하고 시리얼 플로터와 SerialPlot 프로그램에서 받아 1초 간격으로 스펙트럼을 갱신하여 그린다.
- ✓ FFT 크기는 128로 한다.

#### ✚ 회로도

- ✓ 함수 발생기의 출력을 아날로그 입력 핀 A0에 연결하고 접지끼리 연결한다.



#### ✚ 스케치 프로그램

- ✓ arduinoFFT.h 사용

---

```

// Spectrum Estimation
// Library : ArduinoFFT
// FFT size : 128

#include <arduinoFFT.h>
// FFT 크기
#define FFT_N 128
#define readPin A0

// 샘플링 주파수
const float FS = 1000;
// 샘플링 주기 1/FS
unsigned int sampTime;
// 현재 시간과 전에 샘플링한 시간
unsigned long currTime, lastTime;
// arduinoFFT 객체 선언
arduinoFFT FFT = arduinoFFT();
// 입력 데이터가 저장되는 변수
double xReal[FFT_N], xImag[FFT_N];

void setup() {
    Serial.begin(115200);
    sampTime = 1000000 / FS;
}

void loop() {

```

```

// 데이터를 얻는다.
getData();
// 직류 성분을 제거한다.
FFT.DCRemoval(xReal, FFT_N);
// 해밍 창을 씌운다.
FFT.Windowing(xReal,FFT_N, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
// FFT를 실행한다.
FFT.Compute(xReal, xImag, FFT_N, FFT_FORWARD);
// FFT 결과가 복소수이므로 크기를 계산한다.
FFT.ComplexToMagnitude(xReal, xImag, FFT_N);
// 결과를 시리얼 통신으로 보낸다.
printData();
// 1 초가 지난 후에 위의 과정을 다시 실행한다.
delay(1000);
}

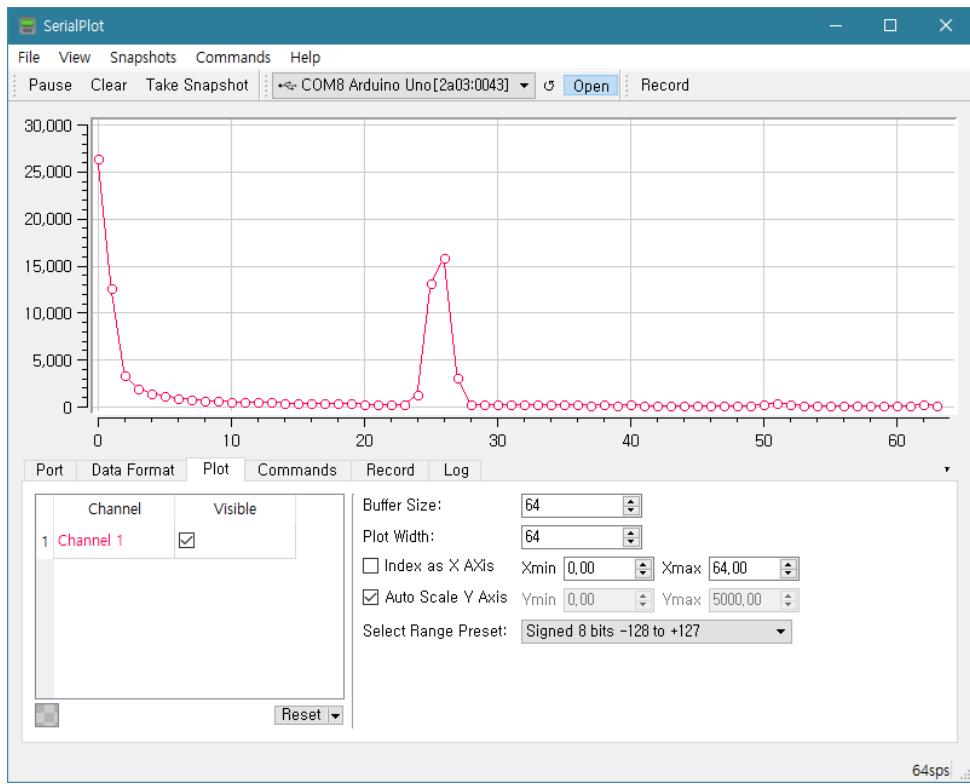
// 함수 발생기로부터 1 ms 마다 입력을 읽는 함수
void getData() {
    for (int n = 0; n < FFT_N; n++) {
        // 전에 샘플링한 시간에서 샘플링 주기가 지난 후에 작업을 진행한다.
        do {
            currTime = micros();
        } while ((currTime - lastTime) < sampTime);
        lastTime = currTime;
        // 함수 발생기의 출력을 읽는다.
        xReal[n] = ANALOGREAD(readPin);;
        xImag[n] = 0;
    }
}

// FFT 결과를 시리얼 통신으로 보내는 함수
void printData() {
    for (int k = 0; k < FFT_N/2; k++) {
        Serial.println(xReal[k], 2);
    }
}

```

#### ✚ 출력 파형

- ✓ 예제 12-2-1 과 같이 SerialPlot 을 설정한다.



- ✓ 예제 12-2-1 과 마찬가지로  $k = 25, 26$  일 때가 크기가 최대임을 알 수 있다.
- ✓ 직류 성분이 완전히 제거되지 않았음을 알 수 있다.

### 10.2.2 아두이노 FFT 라이브러리

#### ✚ 아두이노 FFT 라이브러리

##### FFT.h

- ✓ FFT 크기 : 16~256 (최대 주파수 대역 : 256)
- ✓ FFT 크기가 256 일 때 대략 7 ms 시간 소요
- ✓ 출력 : 16 비트 정수 (선형), 8 비트 정수 (선형), 8 비트 정수 (로그), 8 비트 정수 (옥타브)
- ✓ <http://wiki.openmusiclabs.com/wiki/ArduinoFFT>
- ✓ <http://wiki.openmusiclabs.com/wiki/ArduinoFFT?action=AttachFile&do=view&target=ArduinoFFT3.zip>

#### ✚ 관련 변수

- ✓ 모든 변수는 ASCII 가 아닌 이진 변수이다.
- ✓ 모든 변수는 미리 정의되어 있다.

`fft_input[];`

✗ FFT 입력 및 출력 변수

- 메모리 절약을 위해 `fft_input[]` 배열의 데이터를 FFT 후 도로 그 배열에 출력 값을 저장한다.

✗ 입력 데이터 저장

- 복소수 데이터를 실수, 허수, 실수, 허수 순으로 저장한다.
- $\text{fft\_input}[2n] = \text{Re}\{x[n]\}$ ,  $\text{fft\_input}[2n+1] = \text{Im}\{x[n]\}$

✗ 출력 데이터 저장

- 복소수 데이터를 실수, 허수, 실수, 허수 순으로 저장한다.
- $\text{fft\_input}[2k] = \text{Re}\{X[k]\}$ ,  $\text{fft\_input}[2k+1] = \text{Im}\{X[k]\}$

✗ 부호 16 비트 정수형 (`int16_t`) 배열 (크기  $2 \times \text{FFT\_N}$ )

`fft_lin_out8[];`

- ✗ `fft_mag_lin8()` 함수의 출력 변수
- ✗ 무 부호 8 비트 정수형 (`uint8_t`) 배열 (크기  $\text{FFT\_N}/2$ )

`fft_lin_out[];`

- ✗ `fft_mag_lin()` 함수의 출력 변수
- ✗ 무 부호 16 비트 정수형 (`uint8_t`) 배열 (크기  $\text{FFT\_N}/2$ )

`fft_log_out[];`

- ✗ `fft_log_lin()` 함수의 출력 변수
- ✗ 무 부호 8 비트 정수형 (`uint8_t`) 배열 (크기  $\text{FFT\_N}/2$ )

`fft_oct_out[];`

- ✗ `fft_oct_lin()` 함수의 출력 변수
- ✗ 무 부호 8 비트 정수형 (`uint8_t`) 배열 (크기  $\log(\text{FFT\_N})$ )

✚ 관련 함수

- ✓ 모든 함수는 입력 인수와 출력 인수가 없고 미리 정해진 변수에 대해 처리하고 미리 정해진 변수에 출력 값을 저장한다.

`fft_run();`

- ✗ FFT 주 함수
- ✗ `fft_input[]`에 저장된 데이터를 FFT한 후 `fft_input[]`에 저장한다.

`fft_reorder();`

- ✗ 입력 데이터를 재 배열(bit reversal)하는 함수

- ✖ fft\_input[]에 저장된 데이터를 처리 후 fft\_input[]에 저장한다.
- ✖ fft\_run() 함수 실행 전에 실행해야 한다.

**fft\_window();**

- ✖ 입력 데이터에 창 함수를 곱하는 함수
- ✖ 창 함수는 Hanning 창을 사용한다.
- ✖ fft\_input[]에 저장된 데이터를 처리 후 fft\_input[]에 저장한다.
- ✖ fft\_reorder(), fft\_run() 함수 실행 전에 실행해야 한다.

**fft\_mag\_lin8();**

- ✖ 입력 데이터의 크기(실수 제곱과 허수 제곱의 합의 제곱근)를 8 비트 값으로 변환한다.

$$\sqrt{(\operatorname{Re}\{X[k]\})^2 + (\operatorname{Im}\{X[k]\})^2}$$

- ✖ 출력 값은 SCALE 옵션에 따라 해상도를 최대화하기 위해 스케일링 될 수 있다.
- ✖ fft\_input[]에 저장된 데이터를 처리 후 fft\_lin\_out8[]에 저장한다.

**fft\_mag\_lin();**

- ✖ 입력 데이터의 크기(실수 제곱과 허수 제곱의 합의 제곱근)를 16 비트 값으로 변환한다.

$$\sqrt{(\operatorname{Re}\{X[k]\})^2 + (\operatorname{Im}\{X[k]\})^2}$$

- ✖ fft\_input[]에 저장된 데이터를 처리 후 fft\_lin\_out[]에 저장한다.

**fft\_mag\_log();**

- ✖ 입력 데이터의 크기(실수 제곱과 허수 제곱의 합의 제곱근)를 계산하고 밑이 2인 8 비트 로그 값으로 변환한다.

$$16 \times \log_2 \sqrt{(\operatorname{Re}\{X[k]\})^2 + (\operatorname{Im}\{X[k]\})^2}$$

- ✖ fft\_input[]에 저장된 데이터를 처리 후 fft\_log\_out[]에 저장한다.

**fft\_mag\_octave();**

- ✖ 입력 데이터의 크기(실수 제곱과 허수 제곱의 합의 제곱근)를 계산하고 8 비트 옥타브 포맷으로 배열의 RMS 값으로 변환한다.
  - FFT\_N = 256 일 때 [0, 1, 2:4, 5:8, 9:16, 17:32, 65:128]

- FFT\_N = 128 일 때 [0, 1, 2:4, 5:8, 9:16, 17:32]

$$(9:16) \Rightarrow 16 \times \log_2 \sqrt{\frac{1}{16-9+1} \sum_{k=9}^{16} (\operatorname{Re}\{X[k]\})^2 + (\operatorname{Im}\{X[k]\})^2}$$

✖ fft\_input[]에 저장된 데이터를 처리 후 fft\_oct\_out[]에 저장한다.

#### ▣ 옵션 정의

- ✓ FFT\_N : FFT 크기
  - ✖ 16, 32, 64, 128, 256, (default 256)
- ✓ SCALE : fft\_mag\_lin8() 을 사용할 때 필요한 스케일 값
  - ✖ 1 ~ 255 사이의 정수, (default 1)
- ✓ WINDOW : 데이터에 창을 씌울지 여부 결정
  - ✖ fft\_window() 함수를 사용하지 않으면 0으로 선언해야 함, (default 1)
- ✓ REORDER : 데이터의 재 배열 여부 결정
  - ✖ fft\_reorder() 함수 사용하지 않으면 0으로 선언해야 함, (default 1)

#### ▣ 출력 옵션

- ✓ LOG\_OUT : 출력을 8 비트 로그 값으로 할지 여부 결정
  - ✖ fft\_mag\_log() 함수 사용하면 1로 선언해야 함, (default 0)
- ✓ LIN\_OUT : 출력을 16 비트 선형 값으로 할지 여부 결정
  - ✖ fft\_mag\_lin() 함수 사용하면 1로 선언해야 함, (default 0)
- ✓ LIN\_OUT8 : 출력을 8 비트 선형 값으로 할지 여부 결정
  - ✖ fft\_mag\_lin8() 함수 사용하면 1로 선언해야 함, (default 0)
- ✓ OCTAVE : 출력을 8 비트 옥타브 값으로 할지 여부 결정
  - ✖ fft\_mag\_octave() 함수 사용하면 1로 선언해야 함, (default 0)

### 10.2.3 아두이노 FHT 라이브러리

#### ▣ 아두이노 FHT 라이브러리

##### FHT.h

- ✓ FHT 크기 : 16 ~ 256 (최대 주파수 대역 : 256)
- ✓ FHT 크기가 256 일 때 대략 3.5 ms 시간 소요
- ✓ 출력 : 16 비트 선형, 8 비트 선형, 8 비트 로그, 8 비트 옥타브
- ✓ FHT 의 출력은 FFT 의 조금 출력과 다르다.

$$X[-k] = X[N-k]$$

$$\operatorname{Re}\{X[k]\} = \frac{X[k] + X[-k]}{2}, \quad \operatorname{Im}\{X[k]\} = \frac{X[k] - X[-k]}{2}$$

$$|X[k]| = \sqrt{\frac{X^2[k] + X^2[-k]}{2}}, \quad \angle X[k] = \tan^{-1} \left( \frac{X[k] - X[-k]}{X[k] + X[-k]} \right)$$

$$f_k = k \times \frac{f_{Samp}}{N}$$

- ✓ <http://wiki.openmusiclabs.com/wiki/ArduinoFHT>
- ✓ <http://wiki.openmusiclabs.com/wiki/ArduinoFHT?action=AttachFile&do=view&target=ArduinoFHT4.zip>

#### 관련 변수

- ✓ 모든 변수는 ASCII 가 아닌 이진 변수이다.

**fht\_input[];**

✗ FHT 입력 및 출력 변수

- 메모리 절약을 위해 fht\_input[] 배열의 데이터를 FHT 후 도로 그 배열에 출력 값을 저장한다.

✗ 입력 데이터 저장

- $fht\_input[n] = \operatorname{Re}\{x[n]\}$

✗ 출력 데이터 저장

- 변수의 앞 절반은 실수 성분과 허수 성분의 합, 변수의 뒤 절반은 실수 성분과 허수 성분의 차를 저장한다.

- $fht\_input[k] = \operatorname{Re}\{X[k]\} + \operatorname{Im}\{X[k]\}$ ,

- $fht\_input[k+N/2] = \operatorname{Re}\{X[k]\} - \operatorname{Im}\{X[k]\}$

✗ 부호 16 비트 정수형 (int16\_t) 배열 (크기 FFT\_N)

**fht\_lin\_out8[];**

✗ fft\_mag\_lin() 함수의 출력 변수

✗ 무 부호 8 비트 정수형 (uint8\_t) 배열 (크기 FHT\_N/2)

**fht\_lin\_out[];**

✗ fft\_mag\_lin() 함수의 출력 변수

✗ 무 부호 16 비트 정수형 (uint16\_t) 배열 (크기 FHT\_N/2)

**fht\_log\_out[];**

- ✖ fht\_log\_lin() 함수의 출력 변수
- ✖ 무 부호 8 비트 정수형 (uint8\_t) 배열 (크기 FHT\_N/2)

#### **fht\_oct\_out[];**

- ✖ fht\_oct\_lin() 함수의 출력 변수
- ✖ 무 부호 8 비트 정수형 (uint8\_t) 배열 (크기 log(FHT\_N))

#### ✚ 관련 함수

- ✓ 모든 함수는 입력 인수와 출력 인수가 없고 미리 정해진 변수에 대해 처리하고 미리 정해진 변수에 출력 값을 저장한다.

#### **fht\_run();**

- ✖ FHT 주 함수
- ✖ fht\_input[]에 저장된 데이터를 FHT한 후 fht\_input[]에 저장한다.

#### **fht\_reorder();**

- ✖ 입력 데이터를 재 배열(bit reversal)하는 함수
- ✖ fht\_input[]에 저장된 데이터를 처리 후 fht\_input[]에 저장한다.
- ✖ fht\_run() 함수 실행 전에 실행해야 한다.

#### **fht\_window();**

- ✖ 입력 데이터에 창 함수를 곱하는 함수
- ✖ 창 함수는 Hanning 창을 사용한다.
- ✖ fht\_input[]에 저장된 데이터를 처리 후 fht\_input[]에 저장한다.
- ✖ fht\_reorder(), fht\_run() 함수 실행 전에 실행해야 한다.

#### **fht\_mag\_lin8();**

- ✖ 입력 데이터의 크기(실수 제곱과 허수 제곱의 합의 제곱근)를 8 비트 값으로 변환한다.

$$\sqrt{(\operatorname{Re}\{X[k]\})^2 + (\operatorname{Im}\{X[k]\})^2}$$

- ✖ 출력 값은 SCALE 옵션에 따라 해상도를 최대화하기 위해 스케일링 될 수 있다.
- ✖ fht\_input[]에 저장된 데이터를 처리 후 fht\_mag\_lin8[]에 저장한다.

#### **fht\_mag\_lin();**

- ✖ 입력 데이터의 크기(실수 제곱과 허수 제곱의 합의 제곱근)를 16 비트 값으로 변환한다.

$$\sqrt{(\operatorname{Re}\{X[k]\})^2 + (\operatorname{Im}\{X[k]\})^2}$$

- ✖ fht\_input[]에 저장된 데이터를 처리 후 fht\_lin\_out[]에 저장한다.

#### **fht\_mag\_log();**

- ✖ 입력 데이터의 크기(실수 제곱과 허수 제곱의 합의 제곱근)를 계산하고  
밑이 2인 8 비트 로그 값으로 변환한다.

$$16 \times \log_2 \sqrt{(\operatorname{Re}\{X[k]\})^2 + (\operatorname{Im}\{X[k]\})^2}$$

- ✖ fht\_input[]에 저장된 데이터를 처리 후 fht\_log\_out[]에 저장한다.

#### **fht\_mag\_octave();**

- ✖ 입력 데이터의 크기(실수 제곱과 허수 제곱의 합의 제곱근)를 계산하고  
8 비트 옥타브 포맷으로 배열의 RMS 값으로 변환한다.

- FHT\_N = 256일 때 [0, 1, 2:4, 5:8, 9:16, 17:32, 65:128]
- FHT\_N = 128일 때 [0, 1, 2:4, 5:8, 9:16, 17:32]

$$(9:16) \Rightarrow 16 \times \log_2 \sqrt{\frac{1}{16-9+1} \sum_{k=9}^{16} (\operatorname{Re}\{X[k]\})^2 + (\operatorname{Im}\{X[k]\})^2}$$

- ✖ fht\_input[]에 저장된 데이터를 처리 후 fht\_oct\_out[]에 저장한다.

#### 옵션 정의

- ✓ FHT\_N : FHT 크기
  - ✖ 16, 32, 64, 128, 256, (default 256)
- ✓ SCALE : fht\_mag\_lin8() 을 사용할 때 필요한 스케일 값
  - ✖ 1~255 사이의 정수, (default 1)
- ✓ WINDOW : 데이터에 창을 씌울지 여부 결정
  - ✖ fht\_window() 함수를 사용하지 않으면 0으로 선언해야 함, (default 1)
- ✓ REORDER : 데이터의 재 배열 여부 결정
  - ✖ fht\_reorder() 함수 사용하지 않으면 0으로 선언해야 함, (default 1)

#### 출력 옵션

- ✓ LOG\_OUT : 출력을 8 비트 로그 값으로 할지 여부 결정
  - ✖ fht\_mag\_log() 함수 사용하면 1로 선언해야 함, (default 0)
- ✓ LIN\_OUT : 출력을 16 비트 선형 값으로 할지 여부 결정
  - ✖ fht\_mag\_lin() 함수 사용하면 1로 선언해야 함, (default 0)

- ✓ LIN\_OUT8 : 출력을 8 비트 선형 값으로 할지 여부 결정
  - ✗ fht\_mag\_lin8() 함수 사용하면 1로 선언해야 함, (default 0)
- ✓ OCTAVE : 출력을 8 비트 옥타브 값으로 할지 여부 결정
  - ✗ fht\_mag\_octave() 함수 사용하면 1로 선언해야 함, (default 0)

### 예제 10-2-3

#### ▣ 목표

- ✓ 주파수가 200 Hz 인 사인파를 1 kHz 로 샘플링한 신호와 백색 잡음을 섞은 신호의 스펙트럼을 구한다.
- ✓ 스펙트럼을 시리얼 통신으로 송신하고 시리얼 플로터와 SerialPlot 프로그램에서 받아 스펙트럼을 그린다.
- ✓ FFT 크기는 256 으로 한다.
- ✓ 입력 신호가 실수이므로 FHT.h 라이브러리를 사용한다.

#### ▣ 회로도

- ✓ 없음

#### ▣ 스케치 프로그램

- ✓ FHT.h 사용

---

```
// Spectrum Estimation
// Library : FHT
// FFT size : 256

// 8 비트 선형 출력 사용
#define LIN_OUT8 1
// FHT 크기 256
#define FHT_N 256
#include <FHT.h>

// 샘플링 주파수
const float FS = 1000;
// 신호 주파수와 각 주파수
const float F0 = 200;
float w0;
// 랜덤값을 자세하게 구해주기 위해 곱해지는 값
float Ax = 10000;
// 신호와 잡음의 비 값
float SNR = 0.2;
```

```
void setup() {
    Serial.begin(115200);
    w0 = 2 * PI * F0 / FS;
}

void loop() {
    // 입력 데이터를 얻는다.
    getData();
    // 창 함수를 곱한다.
    fht_window();
    // FHT 하기 위해 재 배열한다.
    fht_reorder();
    // FHT를 실행한다.
    fht_run();
    // 출력을 8 비트 선형으로 변환한다.
    fht_mag_lin8();
    // 결과를 시리얼 통신으로 보낸다.
    printData();
    // 위의 과정을 실행한 후 계속 헛돌아 1번만 실행하게 한다.
    while(1);
}

// 입력 신호를 만드는 함수
void getData() {
    for (int n = 0; n < FHT_N; n++) {
        // 사인파 신호
        float xs = Ax * sin(w0 * n);
        // 잡음 신호
        float xn = random(-Ax, Ax);
        float x = xs + SNR * xn;
        // 신호를 10 비트 정수 데이터로 변환한다. (0~1023)
        int xi = map(x, -Ax*(1+SNR), Ax*(1+SNR), 0, 1023);
        // 평균을 0으로 만든다. (512를 뺀다.)
        xi -= 0x0200;
        // 신호를 16 비트 정수로 바꾸기 위해 64를 곱한다.
        xi <<= 6;
        // 입력 변수에 저장한다.
        fht_input[n] = xi;
    }
}

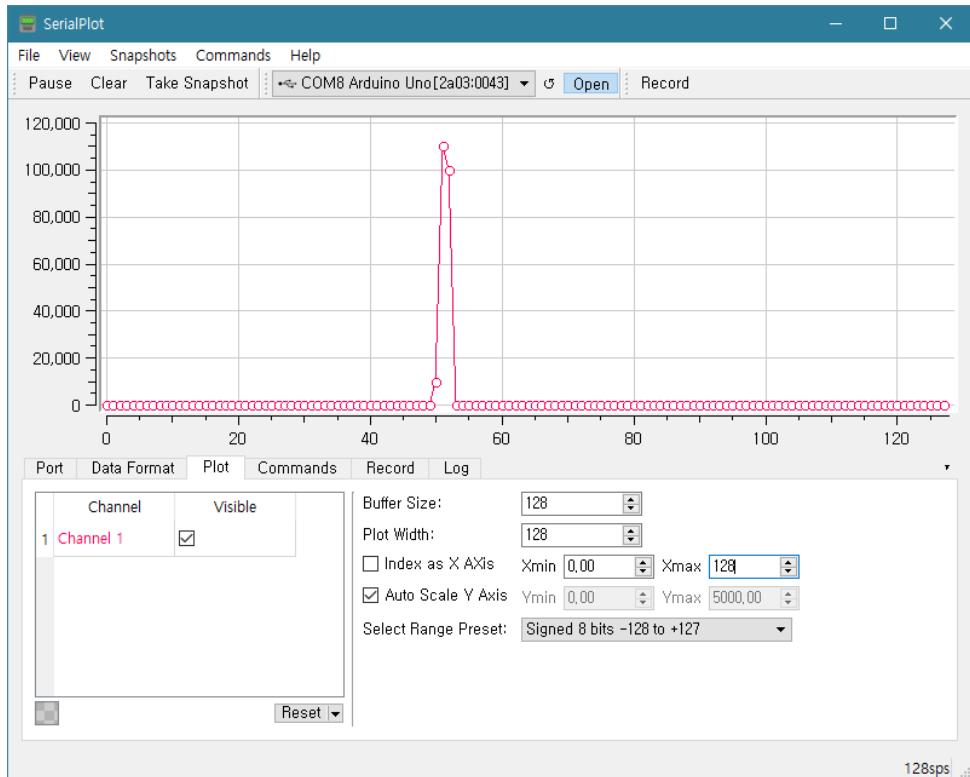
// FFT 결과를 시리얼 통신으로 보내는 함수
void printData() {
```

```

for (int k = 0; k < FHT_N/2; k++) {
    Serial.println(fht_lin_out8[k], 2);
}
}

```

### 스펙트럼 파형 (SerialPlot)



- ✓ Plot 탭에서 Buffer Size와 Plot Width를 128로 선택하면 128 개의 데이터를 볼 수 있다. Xmax도 128로 설정한다.
- ✓ FFT 크기가 256인 경우 인덱스 256이 샘플링 주파수 1 kHz에 해당된다. 200 Hz에 해당하는 인덱스는 다음과 같다.

$$k = 256 \times \frac{200}{1000} = 51.2$$

- ✓ 그림은  $k=51$ 에서 최대 출력을 보여 준다.
- ✓ FFT 크기가 256으로 128에 비해 해상도가 좋아졌다.

### 예제 10-2-4

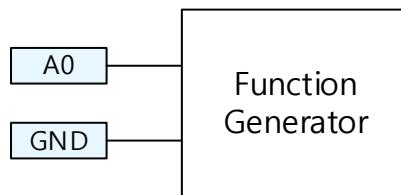
#### 목표

- ✓ 함수 발생기에서 주파수가 200 Hz, 진폭이 5 V, 직류 오프셋이 2.5 V인 사인파를 발생시키고 1 kHz로 샘플링한 신호의 스펙트럼을 구한다.

- ✓ 스펙트럼을 시리얼 통신으로 송신하고 시리얼 플로터와 SerialPlot 프로그램에서 받아 1초 간격으로 스펙트럼을 갱신하여 그린다.
- ✓ FFT 크기는 256으로 한다.
- ✓ 입력 신호가 실수이므로 FHT.h 라이브러리를 사용한다.

#### 회로도

- ✓ 함수 발생기의 출력을 아날로그 입력 핀 A0에 연결하고 접지는 접지끼리 연결한다.



#### 스케치 프로그램

- ✓ FHT.h 사용

---

```

// Spectrum Estimation
// Library : FHT
// FFT size : 256

// 8 비트 선형 출력 사용
#define LIN_OUT8 1
// FHT 크기 256
#define FHT_N 256
#define readPin A0
#include <FHT.h>

// 샘플링 주파수
const float FS = 1000;
// 샘플링 주기 1/FS
unsigned int sampTime;
// 현재 시간과 전에 샘플링한 시간
unsigned long currTime, lastTime;

void setup() {
    Serial.begin(115200);
    sampTime = 1000000 / FS;
}

void loop() {

```

```

// 입력 데이터를 얻는다.
getData();
// 창 함수를 곱한다.
fht_window();
// FHT 하기 위해 재 배열한다.
fht_reorder();
// FHT를 실행한다.
fht_run();
// 출력을 8 비트 선형으로 변환한다.
fht_mag_lin8();
// 결과를 시리얼 통신으로 보낸다.
printData();
// 1 초가 지난 후에 위의 과정을 다시 실행한다.
delay(1000);
}

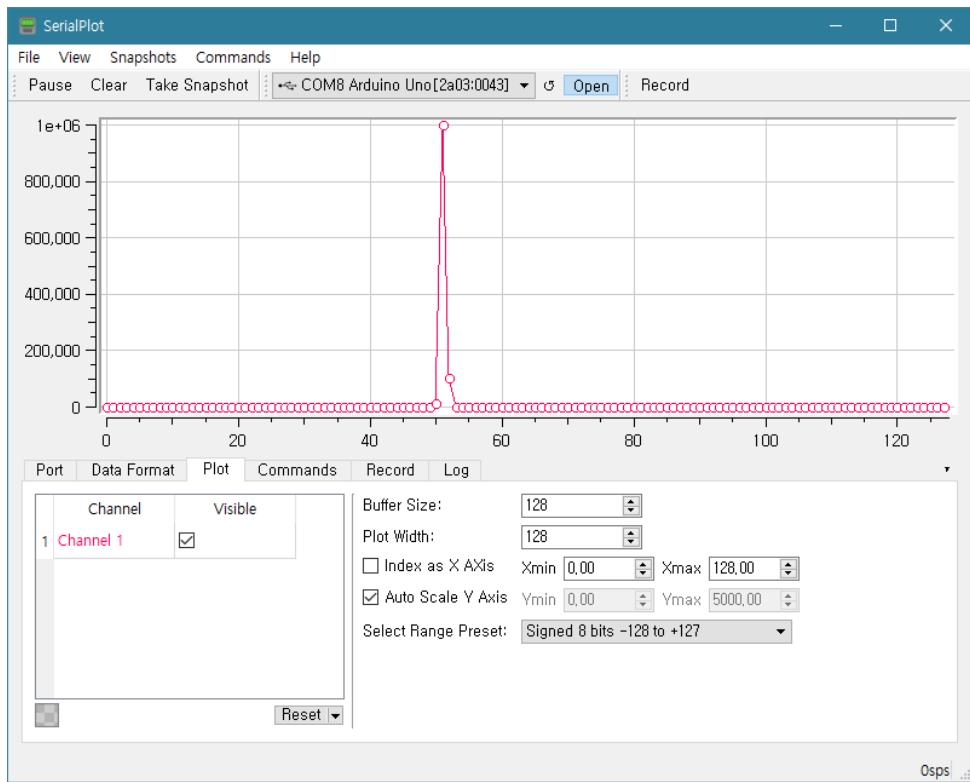
// 함수 발생기로부터 1 ms 마다 입력을 읽는 함수
void getData() {
    for (int n = 0; n < FHT_N; n++) {
        // 전에 샘플링한 시간에서 샘플링 주기가 지난 후에 작업을 진행한다.
        do {
            currTime = micros();
        } while ((currTime - lastTime) < sampTime);
        lastTime = currTime;
        int xi = analogRead(readPin);
        // 평균을 0으로 만든다. (512를 뺀다.)
        xi -= 0x0200;
        // 신호를 16 비트 정수로 바꾸기 위해 64를 곱한다.
        xi <<= 6;
        // 입력 변수에 저장한다.
        fht_input[n] = xi;
    }
}

// FFT 결과를 시리얼 통신으로 보내는 함수
void printData() {
    for (int k = 0; k < FHT_N/2; k++) {
        Serial.println(fht_mag_out8[k], 2);
    }
}

```

### ➊ 출력 파형

- ✓ 예제 12-2-3과 같이 SerialPlot 을 설정한다.



- ✓ 예제 12-2-3 과 마찬가지로  $k = 51$  일 때가 크기가 최대임을 알 수 있다.
- ✓ 예제 12-2-2 와 비교하여 직류 성분이 나타나지 않는다.

### 10.2.3 ADC (analog to digital converter)와 FFT 를 이용한 스펙트럼 분석

#### ✚ 아두이노의 데이터 샘플링 속도를 높이는 방법

- ✓ analogRead() 함수 대신 ADC 의 레지스터를 직접 제어함으로써 샘플링 속도를 높일 수 있다.
  - ✗ <http://maxembedded.com/2011/06/the-adc-of-the-avr/>
- ✓ 시리얼 통신에서 데이터를 이진 데이터로 보냄으로써 속도를 높일 수 있다.
- ✓ 예제 참고
  - ✗ <http://wiki.openmusiclabs.com/wiki/FHTExample>

#### 예제 10-2-5

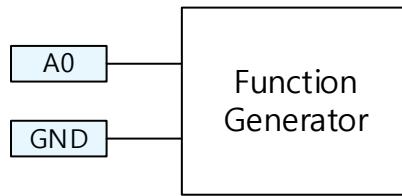
#### ✚ 목표

- ✓ 함수 발생기에서 진폭이 5 V, 직류 오프셋이 2.5 V 인 사인파를 발생시키고 주파수를 변화시켜 가면서 최대 속도로 샘플링한 신호의 스펙트럼을 구한다.

- ✓ 스펙트럼을 시리얼 통신에서 이진 데이터로 송신하고 SerialPlot 프로그램에서 받아 스펙트럼을 갱신하여 그린다.
- ✓ FFT 크기는 256으로 한다.
- ✓ 입력 신호가 실수이므로 FHT.h 라이브러리를 사용한다.

#### 회로도

- ✓ 함수 발생기의 출력을 아날로그 입력 핀 A0에 연결하고 접지는 접지끼리 연결한다.



#### 스케치 프로그램

- ✓ FHT.h 사용

---

```

// Spectrum Estimation
// Library : FHT
// FFT size : 256

// 8 비트 선형 출력 사용
#define LIN_OUT8 1
// FHT 크기 256
#define FHT_N 256
#include <FHT.h>

void setup() {
    Serial.begin(115200);
    // jitter를 낮추기 위해 timer0를 끈다.
    TIMSK0 = 0;
    // ADC를 free running 모드로 설정한다.
    ADCSRA = 0xe5;
    // ADC0와 내부 VCC 전압을 사용한다.
    ADMUX = 0x40;
    // ADC0를 위해 디지털 입력을 끈다.
    DIDR0 = 0x01;
}

void loop() {
    // jitter를 줄이기 위해 사용한다.
  
```

```

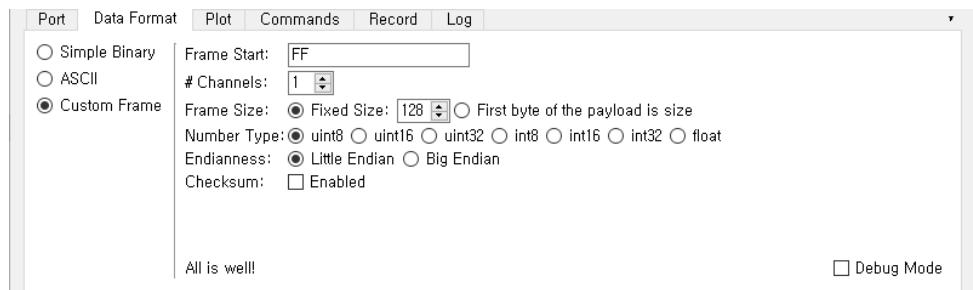
while(1) {
    // 모든 인터럽트를 클리어한다. (불가능하게 한다.)
    cli();
    for (int i = 0 ; i < FHT_N ; i++) {
        // ADC가 준비되기를 기다린다.
        while(!(ADCSRA & 0x10));
        // ADC를 재 시작한다.
        ADCSRA = 0xf5;
        // ADC 데이터를 꺼낸다.
        byte m = ADCL;
        byte j = ADCH;
        // 10 비트 정수 데이터를 만들기 위해 ADCH와 ADCL을 병합한다.
        int k = (j << 8) | m;
        // 10 비트의 반인 512를 빼서 부호가 있는 10 비트 정수로 변환한다.
        k -= 0x0200;
        // 6 비트를 추가하여 부호가 있는 16 비트 정수로 변환한다.
        k <<= 6;
        // 입력 변수 배열에 저장한다.
        fht_input[i] = k;
    }
    // 창 함수를 곱한다.
    fht_window();
    // FHT 하기 위해 재 배열한다.
    fht_reorder();
    // FHT를 실행한다.
    fht_run();
    // 출력을 8 비트 선형으로 변환한다.
    fht_mag_lin8();
    // 인터럽트를 가능하게 한다.
    sei();
    // 헤더 비트 (시작 비트)를 보낸다.
    Serial.write(255);
    // 출력 변수 배열을 이진 데이터로 보낸다.
    Serial.write(fht_mag_lin8, FHT_N/2);
}
}

```

- ✓ cli(), sei() 대신 noInterrupts(), interrupt()를 사용해도 된다.
- ✓ cli(), sei()는 데이터를 읽고 FFT 실행하는 동안 시간에 민감한 모든 인터럽트를 차단하기 위해 사용한다.

#### SerialPlot 출력 파형

- ✓ Data Format 탭에서 Frame Start에 FF를 입력하고 Frame Size에 Fixed Size를 선택하고 128을 입력한 다음 Number Type에 uint8을 선택한다.

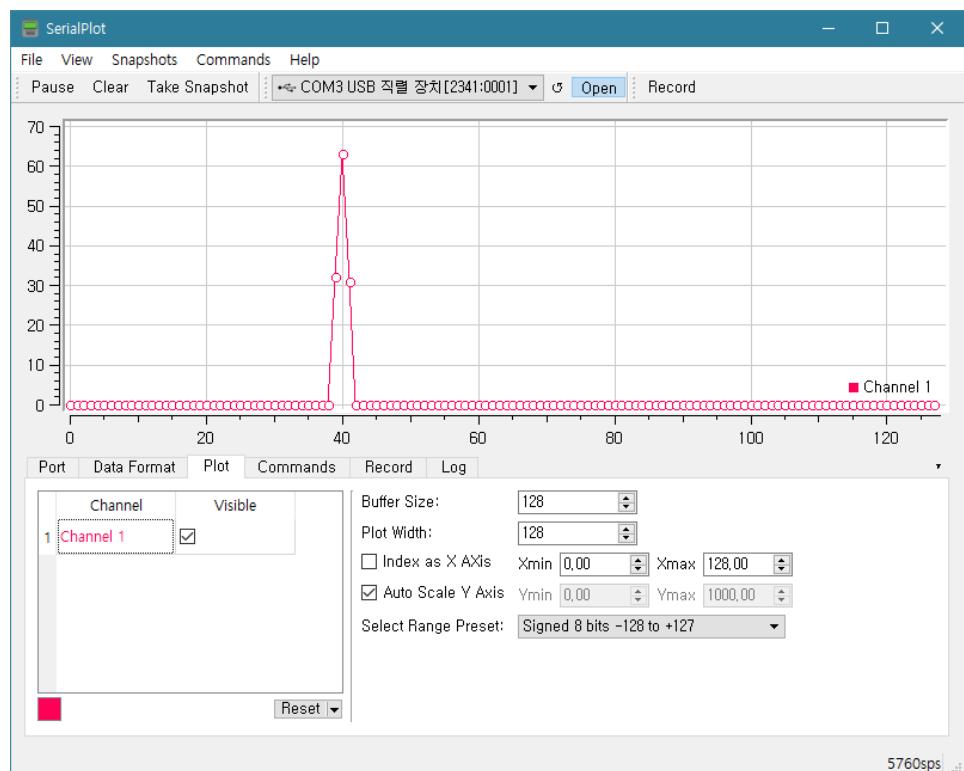


- ✓ Plot 탭에서 Buffer Size, Plot Width, Xmax에 128을 입력한다.

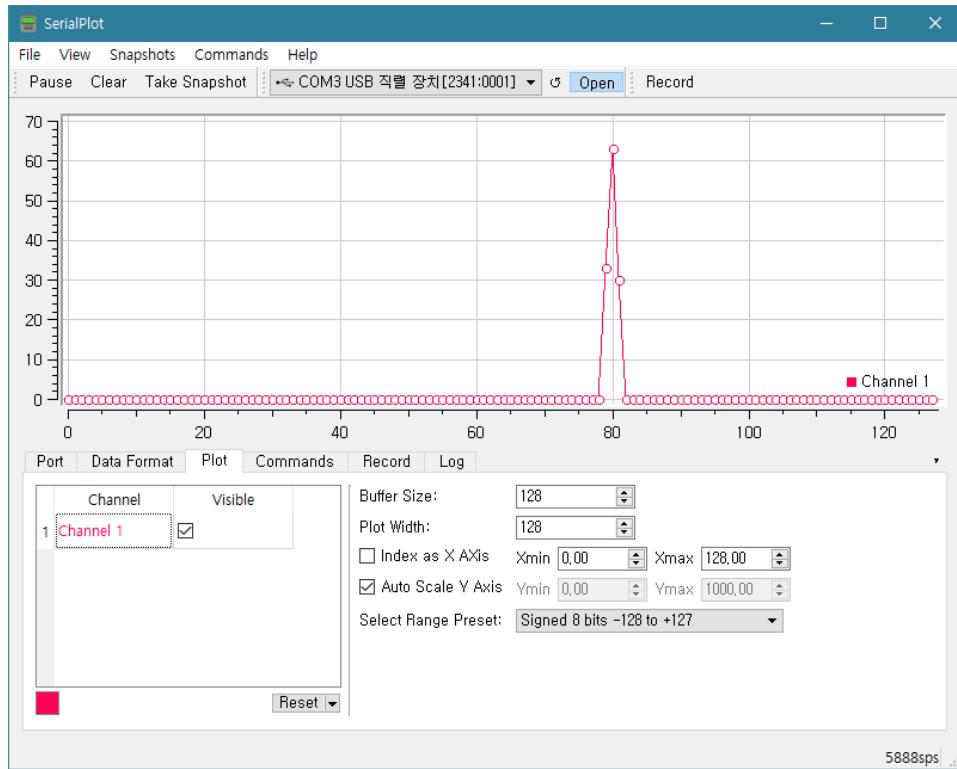


- ✓ 출력 파형

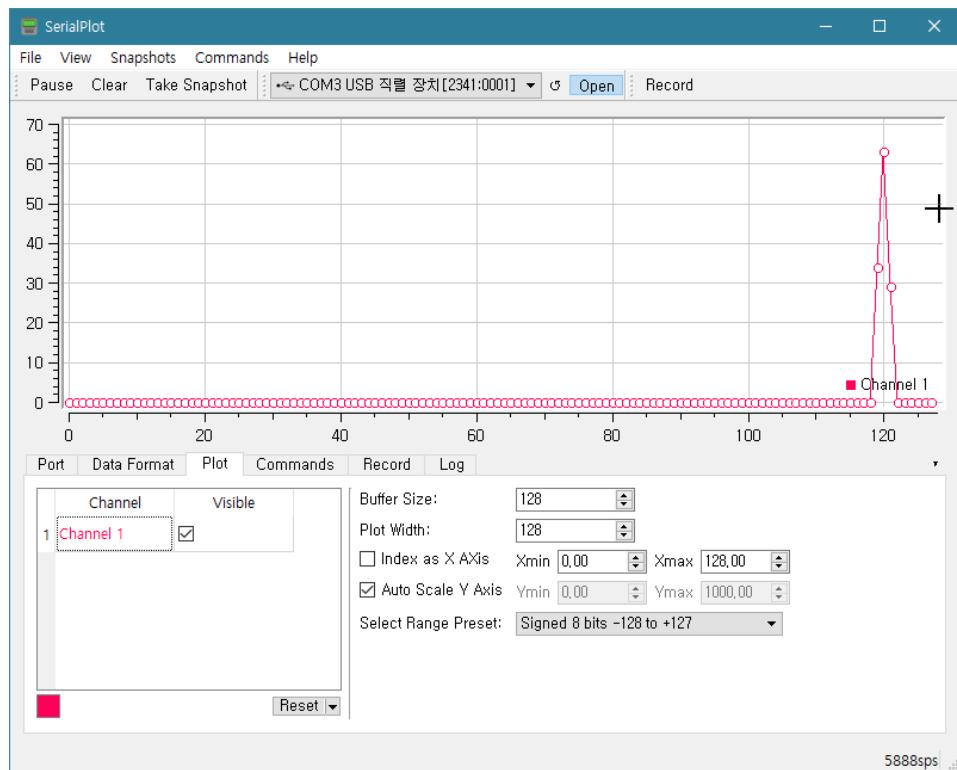
✗  $f = 6\text{ kHz}$  일 때의 출력 파형



✗  $f = 12\text{ kHz}$  일 때의 출력 파형



- ✖  $f = 18 \text{ kHz}$  일 때의 출력 파형



- ✖ 6 kHz 주파수가  $k = 40$ 에 해당하므로 대략 샘플링 주파수를 예측할 수 있다.

$$\frac{6000}{f_s} = \frac{k}{256} \Rightarrow f_s = 6000 \frac{256}{40} = 38400 \text{ Hz}$$

- ✖ 위의 프로그램은 대략 38.4 kHz의 샘플링 주파수로 데이터를 샘플링함을 알 수 있다.

## 연습 문제

### ▶ 연습 문제 10-2-1

- ✓ AudioFFT 라이브러리를 사용하여 오디오 신호의 스펙트럼을 구하고 스펙트럼의 크기에 따라 4 개의 matrix-LED 에 32 개의 스펙트럼을 세로 크기로 나타내는 회로를 구성하고 프로그램을 작성하라.

### ▶ 연습 문제 10-2-2

- ✓ FHT 라이브러리를 사용하여 오디오 신호의 스펙트럼을 구하고 스펙트럼의 크기에 따라 4 개의 matrix-LED 에 32 개의 스펙트럼을 세로 크기로 나타내는 회로를 구성하고 프로그램을 작성하라.

## 11. SD 카드 모듈 제어

### 11.1 SPI (serial peripheral interface) 통신

#### ■ SPI 통신 방식

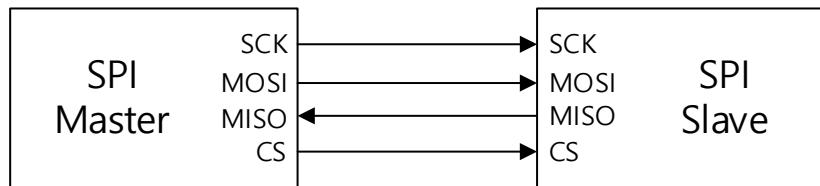
- ✓ 직렬 주변기기 인터페이스
- ✓ 주변 장치와 클럭을 통해 동기화하는 동기식 통신 방식
- ✓ 마스터와 슬레이브 모드로 통신 (한 개의 마스터 기기와 한 개 이상의 슬레이브 기기 간의 통신 방식)

#### ■ SPI에서 사용하는 핀

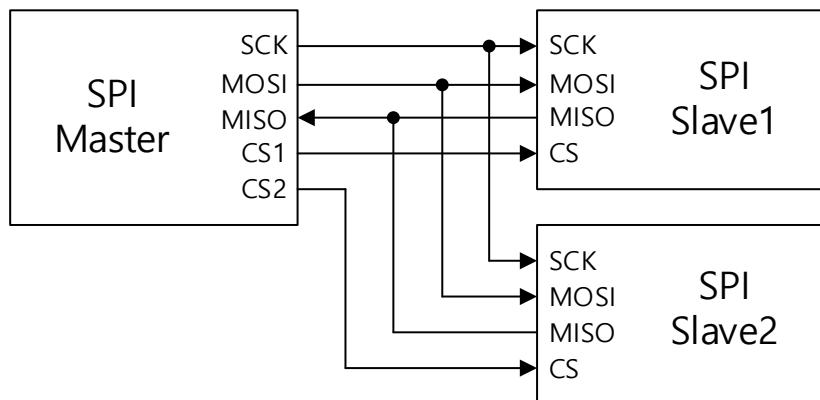
- ✓ SCK (serial clock), CLK : 동기화 클럭
- ✓ CS (chip select), SS (slave select) : 여러 개의 슬레이브 기기가 연결되어 있을 경우 슬레이브 기기를 선택하는 핀, 마스터 기기에서 출력된다.
- ✓ MOSI (master out slave in) : 슬레이브 기기에서 마스터 기기로 가는 데이터 선
- ✓ MISO (master in slave out) : 마스터 기기에서 슬레이브 기기로 가는 데이터 선

#### ■ SPI 연결 방법

- ✓ 슬레이브 기기가 1개인 경우



- ✓ 슬레이브 기기가 2개인 경우



- ✖️ 슬레이브 기기를 선택하기 위해서는 해당 슬레이브 기기에 연결된 마스터 기기의 CS 핀에 LOW를 내 보낸다. (active low)
- ✖️ CS 핀이 HIGH인 기기와는 통신이 이루어지지 않는다.

#### 라이브러리

##### **SPI.h**

- ✓ SPI 통신 방식을 지원하기 위한 라이브러리
- ✓ 아두이노가 마스터 기기, 주변 기기가 슬레이브 기기
- ✓ <https://www.arduino.cc/en/Reference/SPI>

#### 관련 함수

**SPISetting(speedMax, dataOrder, dataMode);**

- ✖️ SPI 통신을 설정하기 위해 필요한 객체
- ✖️ SPI.beginTransaction() 함수의 인수로 사용된다.
- ✖️ speedMax : SPI 통신의 최대 속도
- ✖️ dataOrder : MSBFIRST, LSBFIRST
  - MSBFIRST : MSB를 먼저 보낸다.
  - LSBFIRST : LSB를 먼저 보낸다.
- ✖️ dataMode : SPI\_MODE0, SPI\_MODE1, SPI\_MODE2, SPI\_MODE3

**.begin();**

- ✖️ SPI 버스를 초기화 한다. (SCK, MOSI, SS를 출력으로 설정하고, SCK와 MOSI를 LOW로, SS를 HIGH로 설정한다.)

**.end();**

- ✖️ SPI 버스를 불가능하게 한다.

**.beginTransaction(mySetting);**

- ✖️ SPISetting을 이용하여 SPI 버스를 초기화 한다.

**transfer(val);**

- ✖️ 1 바이트를 송신하고 수신한 값을 돌려 준다.
- ✖️ val : 송신할 데이터 (1 바이트)
- ✖️ return : 수신한 데이터 (1 바이트)

**transfer16(val16);**

- ✖️ 2 바이트를 송신하고 수신한 값을 돌려 준다.

- \* Val16 : 송신할 데이터 (2 바이트)

- \* return : 수신한 데이터 (2 바이트)

`transfer(buffer, size);`

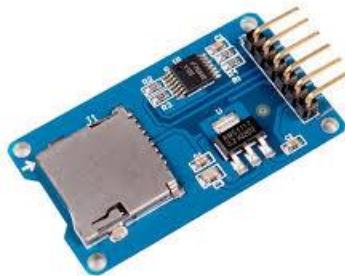
- \* buffer에 있는 데이터를 송신하고 수신한 데이터를 buffer에 저장한다.

## 11.2 SD 카드 모듈

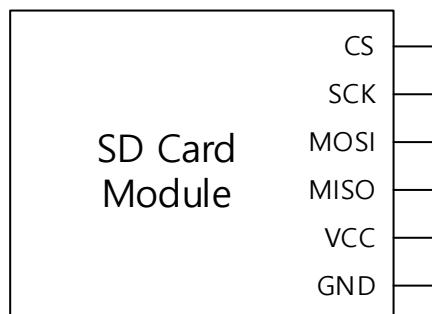
### SD 카드 모듈 제어 필요성

- ✓ GPS 데이터 등 데이터 양이 많은 데이터를 저장하는데 필요

### Micro SD 카드 모듈

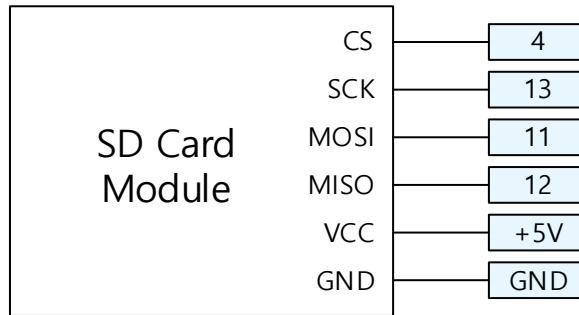


- ✓ 전원 전압: 5 V, 3.3 V
- ✓ 표준 SPI (serial peripheral interface) 인터페이스 사용
- ✓ 저장 용량
  - ✗ Micro SD 카드: 2 G 이하
  - ✗ Micro SDHC 카드: 32 G 이하
- ✓ FAT16, FAT32 포맷 만 이용 가능
- ✓ 핀 배치도



- ✗ CS (chip select)
- ✗ SCK (serial clock)
- ✗ MOSI (master out slave in)
- ✗ MISO (master in slave out)
- ✗ VCC: 전원 핀, 3.3 V 또는 5 V
- ✗ GND: 접지 핀

- ✓ 아두이노 우노와의 연결 (다른 보드는 연결할 핀 번호가 다르다.)
  - ✗ CS (chip select) : 디지털 10번 핀에 연결 (다른 핀도 가능)
  - ✗ SCK (serial clock), CLK (clock) : 디지털 13번 핀에 연결
  - ✗ MOSI (master out slave in) : 디지털 11번 핀에 연결
  - ✗ MISO (master in slave out) : 디지털 12번 핀에 연결



#### ■ 라이브러리

##### SD.h

- ✓ SD 카드에 접근하는데 필요한 클래스, 파일과 디렉토리를 조작한다.
- ✓ FAT16, FAT32 파일 시스템 이용 가능하다.
- ✓ 파일 이름으로 8.3 포맷을 이용, /로 구분된 디렉토리 사용 가능하다. (예 : "directory/filename.txt")
- ✓ <https://www.arduino.cc/en/Reference/SD>

#### ■ 관련 함수

`.begin(csPin);`

- ✗ SD 라이브러리와 SD 카드를 초기화 한다.
- ✗ SPI 버스 (디지털 11, 12, 13번 핀) 사용을 시작한다. (SPI.h 라이브러리를 포함시켜야 한다.)
- ✗ csPin : 칩 선택 핀 (default = 하드웨어 SS핀, 우노의 경우 10번 핀), 다른 핀 사용이 가능하지만 하드웨어 SS핀은 출력으로 유지시켜야 한다.

`.exists(fileName);`

- ✗ fileName 파일이나 디렉토리가 SD 카드에 있는지 검사한다.
- ✗ 있으면 참, 없으면 거짓을 돌려 준다.

`.open(fileName, mode);`

- ✖ fileName의 파일을 연다. 파일이 없으면 만들고 연다.
- ✖ 파일 열기가 성공하면 연 파일과 관련된 File 객체를 돌려 준다.
- ✖ filename : 열고자 하는 파일 이름 (디렉토리 포함 가능)
- ✖ mode (optional) : FILE\_READ (default 값), FILE\_WRITE
  - FILE\_READ : 읽기 위해 파일을 연다. 파일의 처음부터 시작한다.
  - FILE\_WRITE : 읽거나 쓰기 위해 파일을 연다. 파일의 끝에서 시작 한다.

`.close();`

- ✖ 파일을 닫는다.

`.remove(filename)`

- ✖ filename의 파일을 SD 카드에서 삭제한다.

`.mkdir(fileName)`

- ✖ fileName의 디렉토리를 SD 카드에 만든다.

`.rmdir(fileName)`

- ✖ fileName의 디렉토리를 SD 카드에서 제거한다.

### 예제 11-2-1

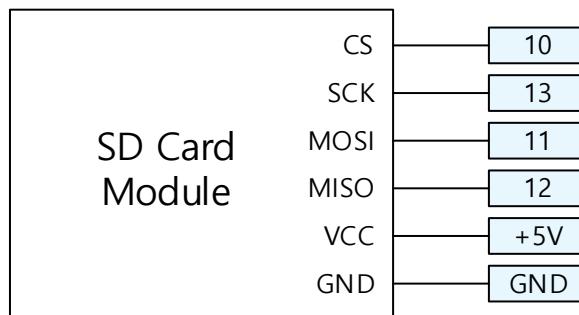
 예제 – SD – Files 참조

 목표

- ✓ SD 카드를 사용하기 위해 초기화한다.
- ✓ SD 카드에 “example.txt” 파일이 있는지 검사하고 존재 여부를 시리얼 모니터에 나타낸다.
- ✓ “example.txt” 파일을 새로 만들어 열고 바로 닫는다.
- ✓ “example.txt” 파일을 SD 카드에서 삭제한다.
- ✓ “example.txt” 파일이 있는지 다시 확인하고 존재 여부를 시리얼 모니터에 나타낸다.

 회로도

- ✓ SD 카드 모듈의 CS, SCK, MOSI, MISO 를 디지털 10, 13, 11, 12 번 핀에 연결한다.



▶ 스케치 프로그램

- ✓ SPI.h, SD.h 사용

```
// SD Card Control
// CS - pin 4, CLK - pin 13, MOSI - pin 11, MISO - pin 12
// Library : SPI.h, SD.h

#include <SPI.h>
#include <SD.h>

const int csPin = 10;
File myFile;

void setup() {
    Serial.begin(9600);
    // 시리얼 포트가 연결되기를 기다린다.
    while (!Serial) {
        ;
    }
    // SD 카드를 초기화한다.
    // 초기화 실패하면 에러 문장을 시리얼 모니터에 표시하고 프로그램을 끝낸다.
    Serial.print("Initializing SD card...");
    if (!SD.begin(csPin)) {
        Serial.println("initialization failed!");
        return;
    }
    // 초기화가 성공하면 성공 문장을 시리얼 모니터에 표시한다.
    Serial.println("initialization done.");

    // 파일이 있는지 검사하고 존재 여부를 시리얼 모니터에 나타낸다.
    if (SD.exists("example.txt")) {
        Serial.println("example.txt exists.");
    }
    else {
        Serial.println("example.txt doesn't exist.");
    }
}
```

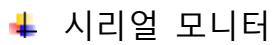
```
}

// 파일이 있으면 파일을 열고, 없으면 만들고 연 후 바로 닫는다.
Serial.println("Creating example.txt...");
myFile = SD.open("example.txt", FILE_WRITE);
myFile.close();
// 파일이 있는지 검사하고 존재 여부를 시리얼 모니터에 나타낸다.
if (SD.exists("example.txt")) {
    Serial.println("example.txt exists.");
}
else {
    Serial.println("example.txt doesn't exist.");
}

// 파일을 삭제한다.
Serial.println("Removing example.txt...");
SD.remove("example.txt");
// 파일이 있는지 검사하고 존재 여부를 시리얼 모니터에 나타낸다.
if (SD.exists("example.txt")) {
    Serial.println("example.txt exists.");
}
else {
    Serial.println("example.txt doesn't exist.");
}

void loop() {
```

- ✓ 한 번만 실행하므로 모든 프로그램은 void setup() 부분에 작성한다.



시리얼 모니터

Initializing SD card...initialization done.  
example.txt doesn't exist.  
Creating example.txt...  
example.txt exists.  
Removing example.txt...  
example.txt doesn't exist.

자동 스크롤  line ending 없음  출력 지우기

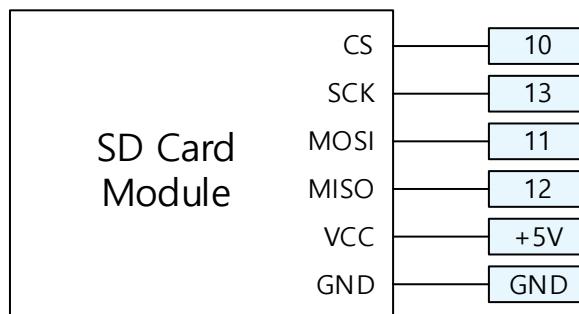
### 예제 11-2-2

#### ▣ 목표

- ✓ SD 카드에 “test1.txt” 파일이 있는지 검사하고 존재 여부를 시리얼 모니터에 나타낸 후 파일이 존재하면 삭제한다..
- ✓ “example.txt” 파일을 새로 만들어 열고 적당한 텍스트 문장을 시리얼 모니터와 파일에 쓴 다음 파일을 닫는다.

#### ▣ 회로도

- ✓ SD 카드 모듈의 CS, SCK, MOSI, MISO 를 디지털 10, 13, 11, 12 번 핀에 연결한다.



#### ▣ 스케치 프로그램

- ✓ SPI.h, SD.h 사용

---

```
// SD Card Write and Read
// CS - pin 4, CLK - pin 13, MOSI - pin 11, MISO - pin 12
```

---

```
// Library : SPI.h, SD.h

#include <SPI.h>
#include <SD.h>

const int csPin = 10;
File myFile;
String fileName = "test1.txt";

void setup() {
    Serial.begin(9600);
    // 시리얼 포트가 연결되기를 기다린다.
    while (!Serial) {
        ;
    }
    Serial.println("Initializing SD card...");
    // SD 카드를 초기화한다.
    if (!SD.begin(csPin)) {
        Serial.println("Initialization of the SD card is failed!");
        return;
    }
    Serial.println("Initialization of the SD card is done.");

    // 파일이 있으면 제거한다.
    // 데이터를 덧붙이려면 이 부분을 제거한다.
    if (SD.exists(fileName)) {
        Serial.println("File exists and will be removed.");
        SD.remove(fileName);
    }
    // 텍스트 문장을 쓰기 위해 파일을 연다.
    // 파일 열기는 한 번에 한 개만 가능하므로 다른 파일을 열기 전에
    // 닫아야 한다.
    myFile = SD.open(fileName, FILE_WRITE);
    if (myFile) {
        Serial.println("File is opened.");
        Serial.println("Writing to SD card...");
        myFile.println("Chonbuk National University");
        myFile.println("Professor Baik Heung-Ki");
        myFile.close();
        Serial.println("File is closed.");
    }
    else {
        Serial.println("File open error!");
    }
}
```

```

// 데이터를 읽기 위해 파일을 연다.
myFile = SD.open("test1.txt");
if (myFile) {
    // 파일 열기가 성공하면 성공 문장을 시리얼 모니터에 나타낸다.
    Serial.println("File is re-opened");
    // 읽을 내용이 없어질 때까지 계속해서 데이터를 읽는다.
    while (myFile.available()) {
        Serial.write(myFile.read());
    }
    // 파일을 닫는다.
    myFile.close();
    Serial.println("File is closed.");
}
else {
    // 파일 열기가 실패하면 실패 문장을 시리얼 모니터에 나타낸다.
    Serial.println("error opening test.txt");
}

void loop() {
}

```

- ✓ 한 번만 실행하므로 모든 프로그램은 void setup() 부분에 작성하고 void loop() 부분은 비워둔다.

### 시리얼 모니터



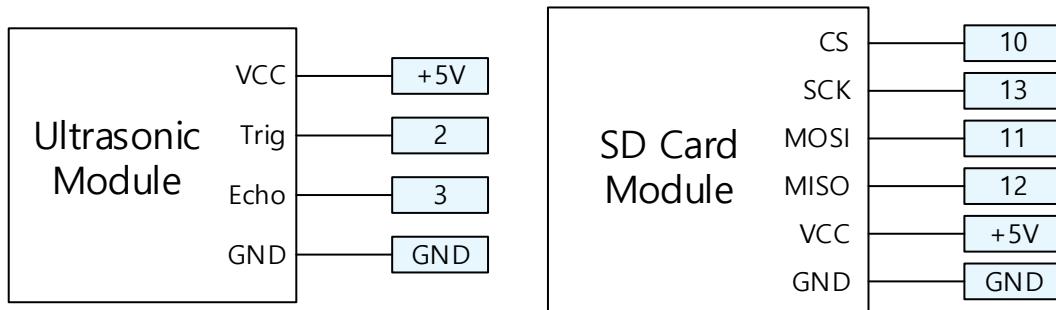
### 예제 11-2-3

#### ▣ 목표

- ✓ SD 카드에 “datalog.txt” 파일이 있는지 검사하고 존재 여부를 시리얼 모니터에 나타낸 후 파일이 존재하면 삭제한다.
- ✓ 초음파 센서로 읽은 거리 값을 시리얼 모니터에 나타내고 Micro SD 카드에 있는 “datalog.txt” 파일에 기록한다.

#### ▣ 회로도

- ✓ SD 카드 모듈의 CS, SCK, MOSI, MISO 를 디지털 10, 13, 11, 12 번 핀에 연결한다.
- ✓ 초음파 센서의 Trig, Echo 를 디지털 2, 3 번 핀에 연결한다.



#### ▣ 스케치 프로그램

- ✓ SPI.h, SD.h, NewPing.h 사용

```

// SD Card Datalogger
// SD card : CS - pin 4, CLK - pin 13, MOSI - pin 11, MISO - pin 12
// Ultrasonic - 2, Echo - 3
// Library : SPI.h, SD.h, NewPing.h

#include <SPI.h>
#include <SD.h>
#include <NewPing.h>

const int trigPin = 2;
const int echoPin = 3;
const int maxDistance = 300;
const int delayTime = 1000;
const int csPin = 10;
File dataFile;
// NewPing 객체 선언
NewPing myPing(trigPin, echoPin, maxDistance);
    
```

```
String fileName = "datalog.txt";

void setup() {
    Serial.begin(9600);
    // 시리얼 포트가 연결되기를 기다린다.
    while (!Serial) {
        ;
    }
    Serial.println("Initializing SD card...");
    // SD 카드를 초기화한다.
    if (!SD.begin(csPin)) {
        Serial.println("Card failed, or not present");
        return;
    }
    Serial.println("SD card is initialized.");

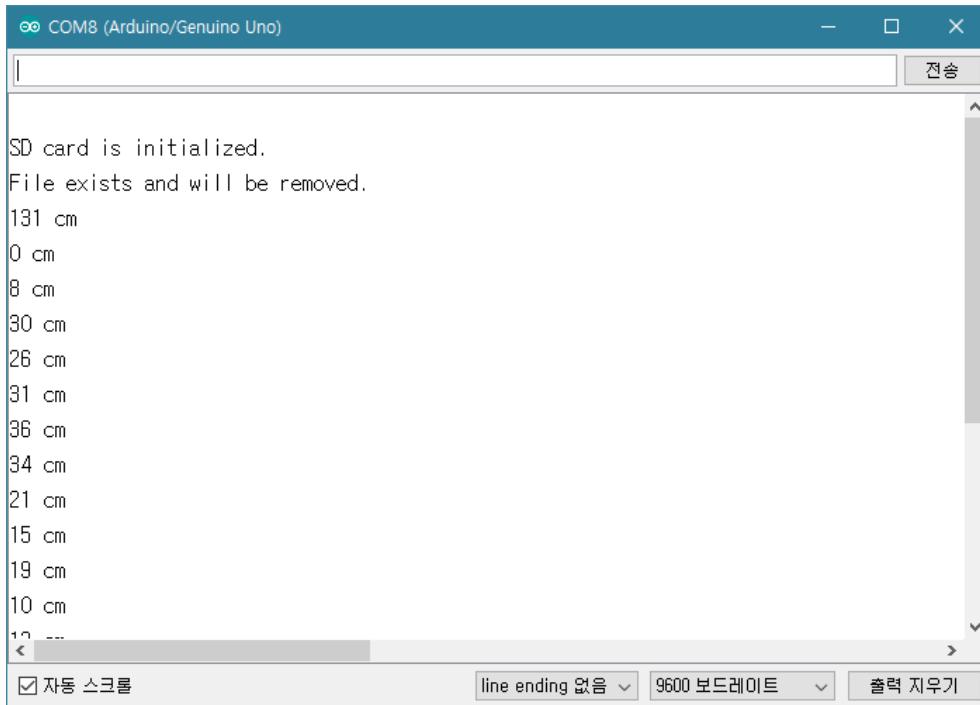
    // 파일이 있으면 제거한다.
    // 만일 기존의 파일에 데이터를 덧붙이려면 이 부분을 제거한다.
    if (SD.exists(fileName)) {
        Serial.println("File exists and will be removed.");
        SD.remove(fileName);
    }
}

void loop() {
    // 초음파 센서를 통해 거리를 측정한 수 시리얼 모니터에 나타낸다.
    int distance = myPing.ping_cm();
    Serial.print(distance);
    Serial.println(" cm");

    // 파일을 열고 에러가 없으면 측정한 데이터를 기록한 후 파일을 닫는다.
    dataFile = SD.open(fileName, FILE_WRITE);
    if (dataFile) {
        dataFile.println(distance);
        dataFile.close();
    }
    // 데이터 파일 열 때 에러가 발생하면 시리얼 모니터에 나타낸다.
    else {
        Serial.println("File open error!");
    }
    delay(delayTime);
}
```

- ✓ SD 카드의 초기화는 setup()에 넣고, 파일을 열고 데이터를 쓰고 파일을 닫는 부분은 loop()에 넣는다.

### ➊ 시리얼 모니터



The screenshot shows the Arduino Serial Monitor window titled "COM8 (Arduino/Genuino Uno)". The text area displays the following data:

```

SD card is initialized.
File exists and will be removed.
131 cm
0 cm
8 cm
30 cm
26 cm
31 cm
36 cm
34 cm
21 cm
15 cm
19 cm
10 cm
12 cm

```

At the bottom of the window, there are three buttons: "자동 스크롤" (Auto Scroll), "line ending 없음" (No Line Ending), and "9600 보드레이트" (9600 Baud Rate). There is also a "Clear" button on the right side of the text area.

### ➋ 데이터 파일 내용



The screenshot shows a Windows Notepad window titled "DATALOG.TXT - 메모장". The text area displays the following data:

```

131
0
8
30
26
31
36
34
21
15
19
10
12
25
160
23
15
15
159
159
159

```

- ✓ 시리얼 모니터와 실제 파일의 내용이 같음을 알 수 있다.

## 연습 문제

 연습 문제 11-2-1

- ✓ 0 ~ 10 사이의 랜덤 값 10 개를 SD 카드에 텍스트 파일로 저장하고 이를 다시 열어 시리얼 모니터에 나타내는 회로를 구성하고 프로그램을 작성하라. 파일 이름은 적당히 정하고 랜덤 값은 소수 3 자리까지 쓰라.