LEVEL UP!

아두이노 in 판타지월드

DEVICEMART 지음

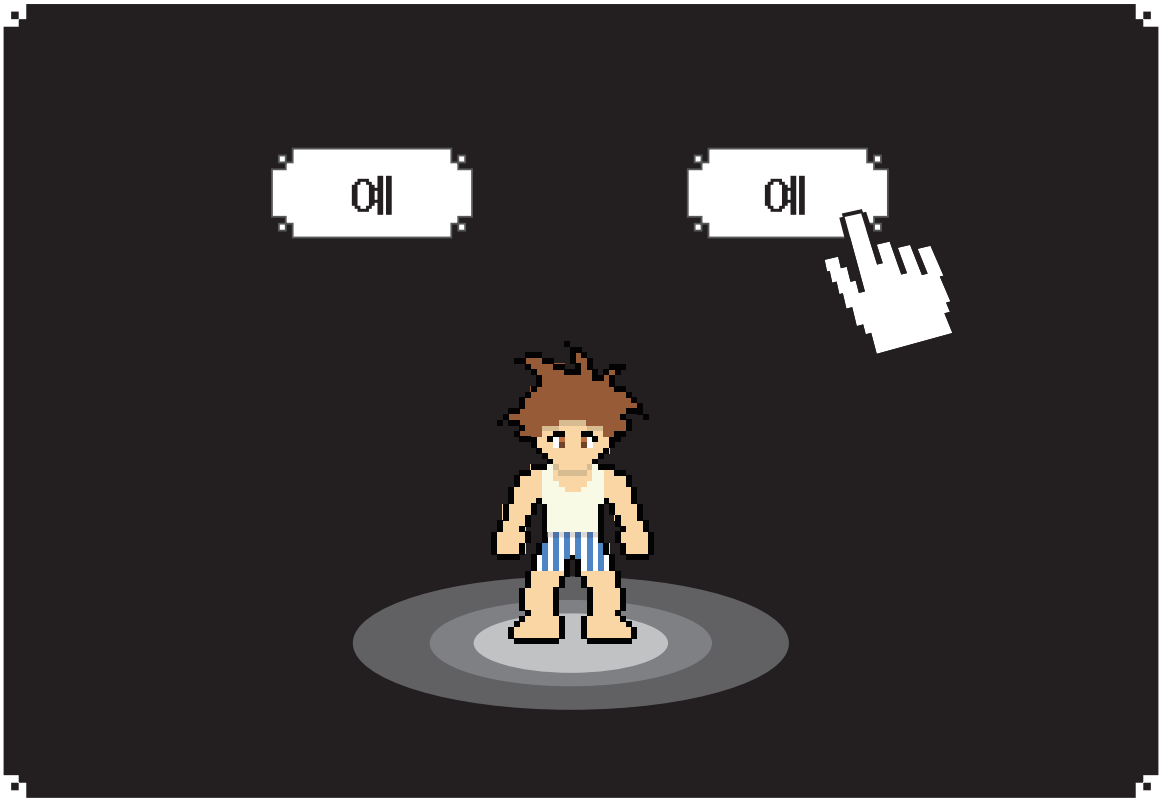
DEVICEMART

레벨업 아두이노
차례

Stage 3. 아두이노 실전 연습

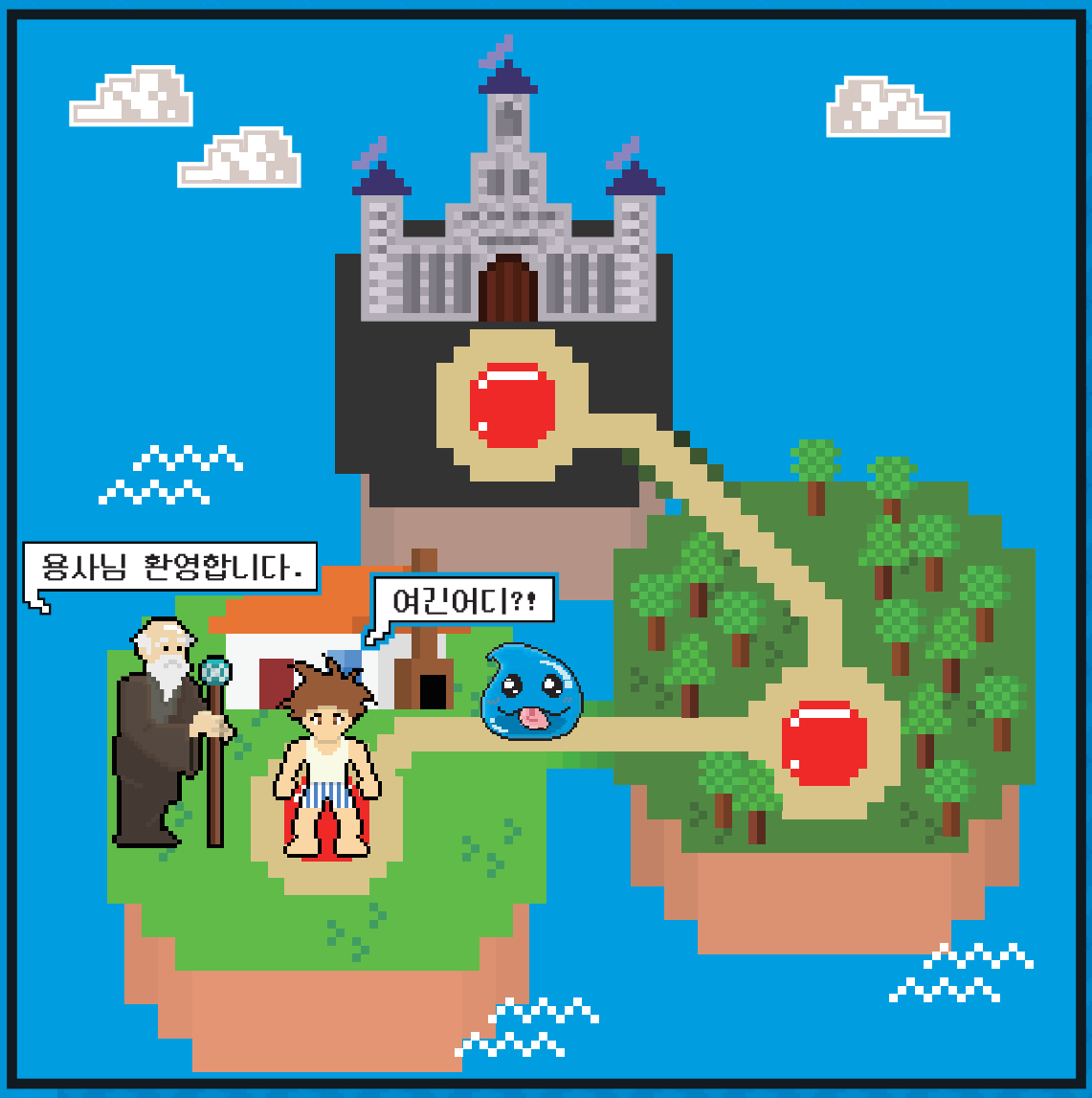
Intro. 아두이노 그리고 용사

당신은 엄마가 밥먹으라고 했는데도 게임에 푹 빠져 제 시간에 가지 않았습니다.
업보를 청산하시겠습니까?



당신은 컴퓨터 속 낯선 세계로 빨려 들어갔습니다.
당신은 아두이노가 선택한 전설의 용사입니다.
마왕을 무찌르면 원래 세계로 돌아가실 수 있습니다.
마왕은 매우 강하기 때문에 장비도 갖춰야 하고 함수검 메소드를 모아야 합니다.
메소드의 힘을 모은 아두이노를 쓰면 마왕을 물리칠 수 있습니다.

스테이지 1.1
전기의 특성



EXP♥

1. 전자 기초 지식

1.1. 전기의 특성

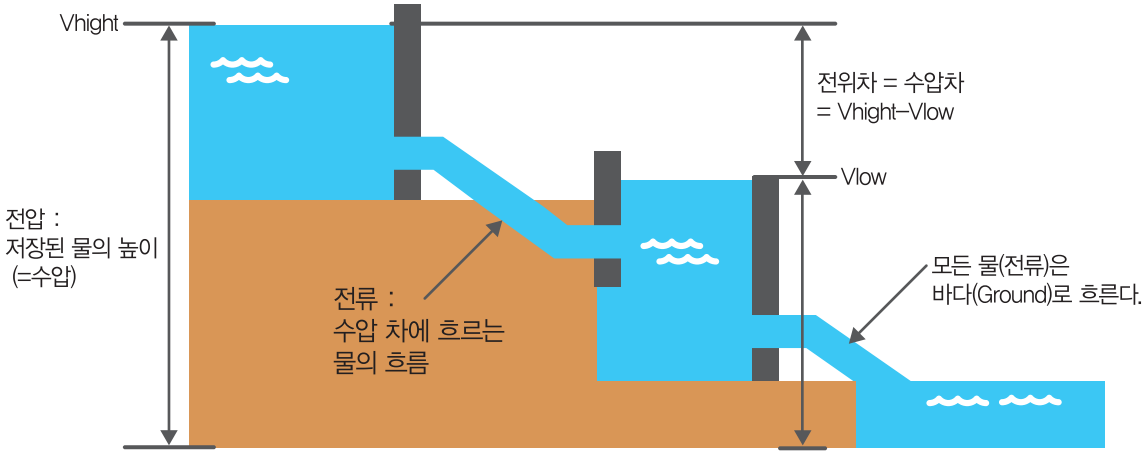
장로 월프레드
용사님 안녕하세요. 저는 이 마을의 장로 월프레드입니다. 홀홀. 용사님이 갖고 계신 아두이노를 능숙하게 사용하시도록 집안 대대로 내려오던 기술을 전수해드리겠습니다.

용사
헉, 꿈이 아니었다니?? 할아버지 여긴 어디죠?
아두..라니 하나도 모르겠어요. 옷은 또 왜 이모양이지? 추워...

장로 월프레드
제가 가르쳐드리는 대로 잘 따라오시면 방어구를 드리겠습니다. 홀홀
아두이노는 '전기'라는 힘으로 동작합니다. 그럼 전기에 대해서 배워볼까요?

용사
네! (나쁜 할아버지 같지는 않으니까 일단 시키는 대로 하자)

1.1.1. 전압, 전류 그리고 저항



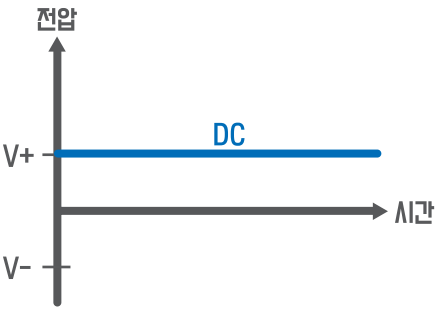
전압은 전기적인 압력을 뜻합니다. 기호는 V, 단위는 볼트 Volt [V] 입니다.
전류는 전기의 흐름을 뜻합니다. 기호는 I, 단위는 암페어 Ampere [A] 입니다.
저항은 전류를 방해하는 것으로 기호는 R, 단위는 옴 Ohm [Ω] 입니다.

장로 월프레드
전압은 수압과 비슷합니다. 물이 높은곳에서 아래로 흐르듯,
전류도 전압이 높은곳에서 낮은 곳으로 흐릅니다.

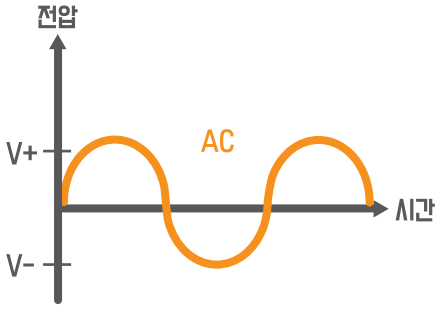
1.1.2. 직류(DC)와 교류(AC)

직류(Direct Current)는 전압의 크기와 방향이 일정하게 유지되는 전류의 형태를 말합니다.

우리가 일반적으로 사용하는 건전지(배터리)가 직류이며, 아두이노에서 사용하는 전류 또한 직류 입니다.



교류(Alternating Current)는 시간에 따라 전압의 크기와 방향이 주기적으로 변하는 전류의 형태를 말합니다. 우리가 가정에서 사용하는 전기가 교류이며, 단상 220V 60Hz입니다. 이때 60Hz는 전류의 방향이 1초에 60번 극성이 바뀌는걸 뜻합니다.

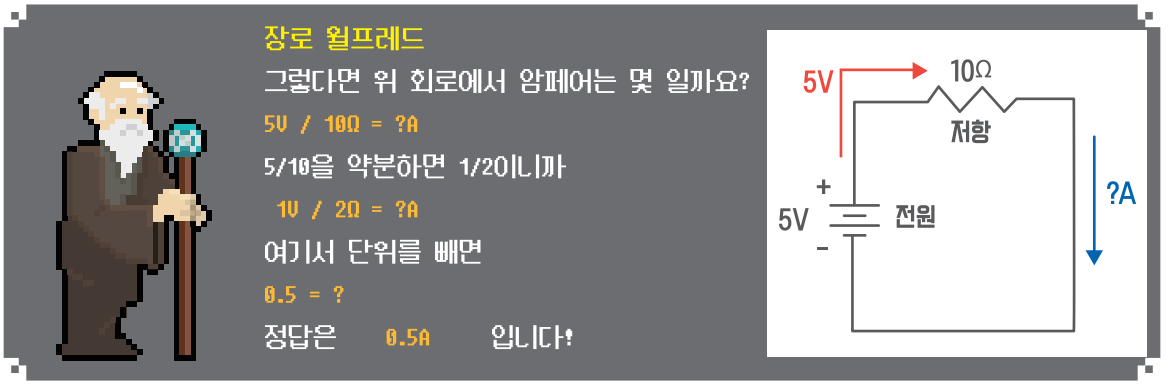


1.1.3. 옴의 법칙

옴의 법칙은 독일의 과학자 게오르그 옴의 이름을 딴 법칙으로, 회로 해석에서 가장 기본이 되는 법칙입니다. 두 지점 사이의 도체(전기가 통하는 물질)에 일정한 전위차가 존재할 때, 도체의 저항의 크기와 전류의 크기는 반비례합니다.

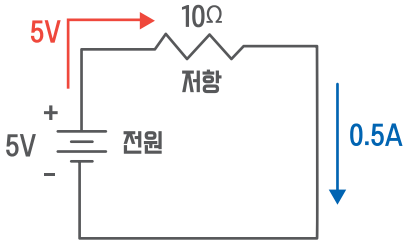
V = IR
I = V/R
R = V/I

즉, **전류(I) = 전압(V) / 저항(R)**입니다.
위 이미지에 나오는 공식 **V=IR, I=V/R, R=V/I** 모두 동일한 공식입니다.
기억이 안 날 땐 위 그림처럼 무당벌레를 그리시고 VIR을 왼쪽 위부터 순서대로 써주세요.

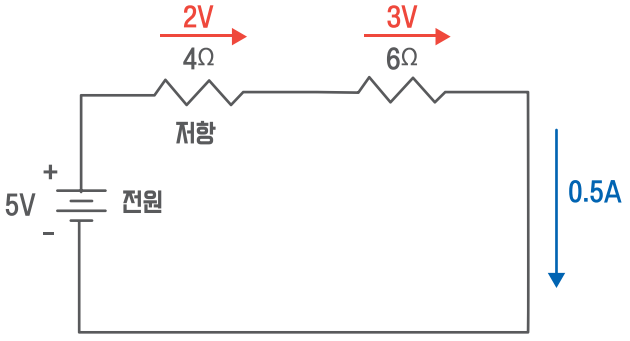


1.1.4. 전압 분배 법칙

위와 같이 저항이 1개인 폐회로(닫혀 있는 회로)에서 저항에 걸리게 되는 전압은 옴의 법칙 (I=V/R)에 의해서 전체 회로에 걸리는 전압과 같습니다.



전류(I)는 0.5A = 전압(5V) / 저항(10Ω)



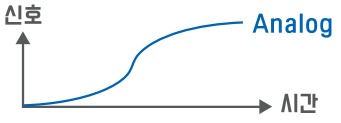
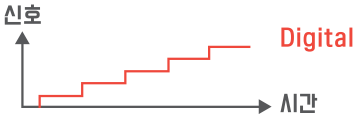
위 회로와 같이 저항이 직렬로 2개 이상이 되면 각 저항에는 저항값에 비례한 전압이 걸리게 되고, 각각의 저항에 걸리는 **전압의 합 = 전체 저항**이 됩니다. 옴의 법칙을 이용해서 각 저항별로 전압을 구할 수 있습니다.

전류(I)는 0.5A = 전압(5V) / 저항(4Ω+6Ω)

4Ω에 걸리는 전압은 V=IR에 의해 4Ω x 0.5A= 2V

6Ω에 걸리는 전압은 V=IR에 의해 6Ω x 0.5A= 3V 입니다.

1.1.5. 아날로그 신호와 디지털 신호

	아날로그 신호	디지털 신호
그래프	 시간이 지날수록 연속적으로 변하는 신호입니다.	 어떠한 특정값이 불연속적으로 변하는 신호입니다.
예시	 자연상태에서 감지할 수 있는 신호	컴퓨터, 휴대폰등 현대 문명에서 사용하는 신호

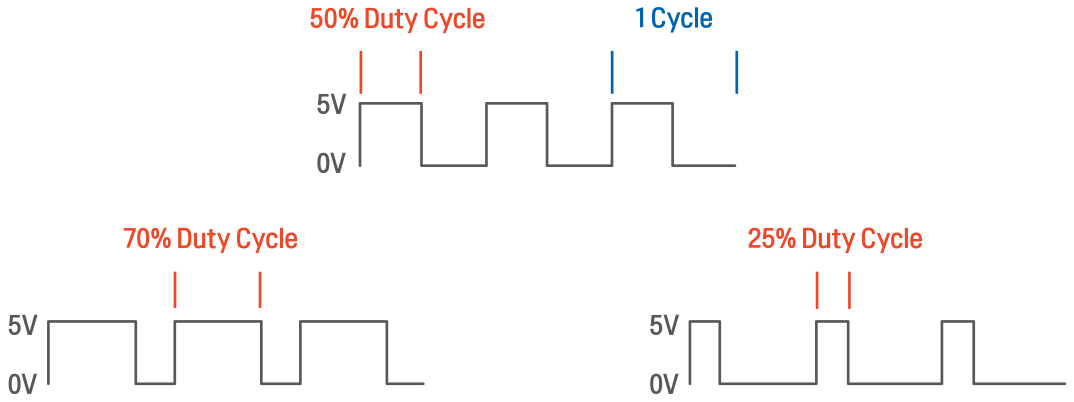
아날로그 신호는 우리가 자연상태에서 감지할 수 있는 신호로, 소리, 빛, 시계바늘, 온도계 등과 같이 시간이 지날수록 연속적으로 변하는 신호입니다.

디지털 신호는 우리가 컴퓨터, 휴대폰등 현대문명에서 사용하는 신호로, 어떠한 특정값이 불연속적으로 변하는 신호입니다.

장로 윌프레드
디지털 신호는 흔히 0과 1로만 이루어져있다고 합니다.
전자에서 0은 0V 신호로는 LOW상태 라고 표현합니다.
전자에서 1은 5V 신호로는 HIGH상태 라고 표현합니다.
아두이노같은 TTL 회로에서는 VCC가 5V이고, GND는 0V입니다.

1.1.6. PWM

펄스 폭 변조(Pulse Width Modulation, PWM)은 펄스의 폭을 조절하는 제어 방법으로, 주로 LED의 밝기를 조절하거나 모터의 속력을 조절할 때 사용됩니다.

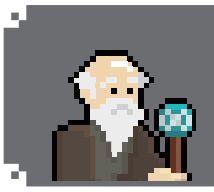


PWM 방식은 일정한 주기(cycle) 동안 On/Off를 반복하며 ON이 되어있는 시간(Duty Cycle)을 조절하여 원하는 출력을 조절하는 방법입니다. Duty Cycle은 '듀티비'라고도 부릅니다.

이를 통해 디지털 값을 아날로그 값처럼 사용할 수 있습니다.

예를 들어 5V 출력 전압에서 Duty Cycle을 50%로 한다면, LED나 모터는 2.5V의 출력으로 동작합니다. 그리고 Duty Cycle을 25%로 한다면 1.75V의 출력으로 동작합니다.

Quest 1. 실력 테스트



장로 윌프레드
그럼 제대로 배우셨는지 한번 점검해 볼까요?
홀홀 마을의 골치거리인 슬라임을 처리하면서 복습해봅시다.
한 문제당 슬라임 한마리랍니다.

용사
헉.. 시험볼줄은 몰랐는데?!
하지만 저런 액괴라면 문제 없지!

Trouble 1 – OX 문제
전압은 낮은곳에서 높은곳으로 흐른다.

Trouble 2 – OX 문제
아두이노는 DC로 작동한다.

Trouble 3 – 객관식 문제
다음 중 옴의 법칙의 공식이 아닌 것은?
① V=I/R ② I=V/R ③ R=V/I ④ V=IR

Trouble 4 – 객관식 문제
100Ω의 저항에 0.2A가 흐르고 있다. 그럼 100Ω의 저항에 걸린 전압은 몇 V인가?
① 20V ② 40V ③ 22V ④ 44V

Trouble 5 – 주관식 문제
10V의 전압으로 구동되는 모터는 정확히 1분에 200회 회전합니다. 이 모터는 1V를 낮출 때마다 20회씩 감속되는 모터입니다. 7.5V일 때 처럼 모터를 제어하고 싶다면 듀티비(Duty Cycle)를 몇 %로 조정해야 할까요?

용사
별로 안어려운데요?

장로 윌프레드
홀홀 슬라임은 양이 많긴 하지만 쉬운 편이지요.
마왕은 매우 강하기 때문에 더 강해지셔야 할 겁니다.

Quest 1. 정답 체크

맞춘 문제 개수만큼 슬라임에 X 표해주세요.

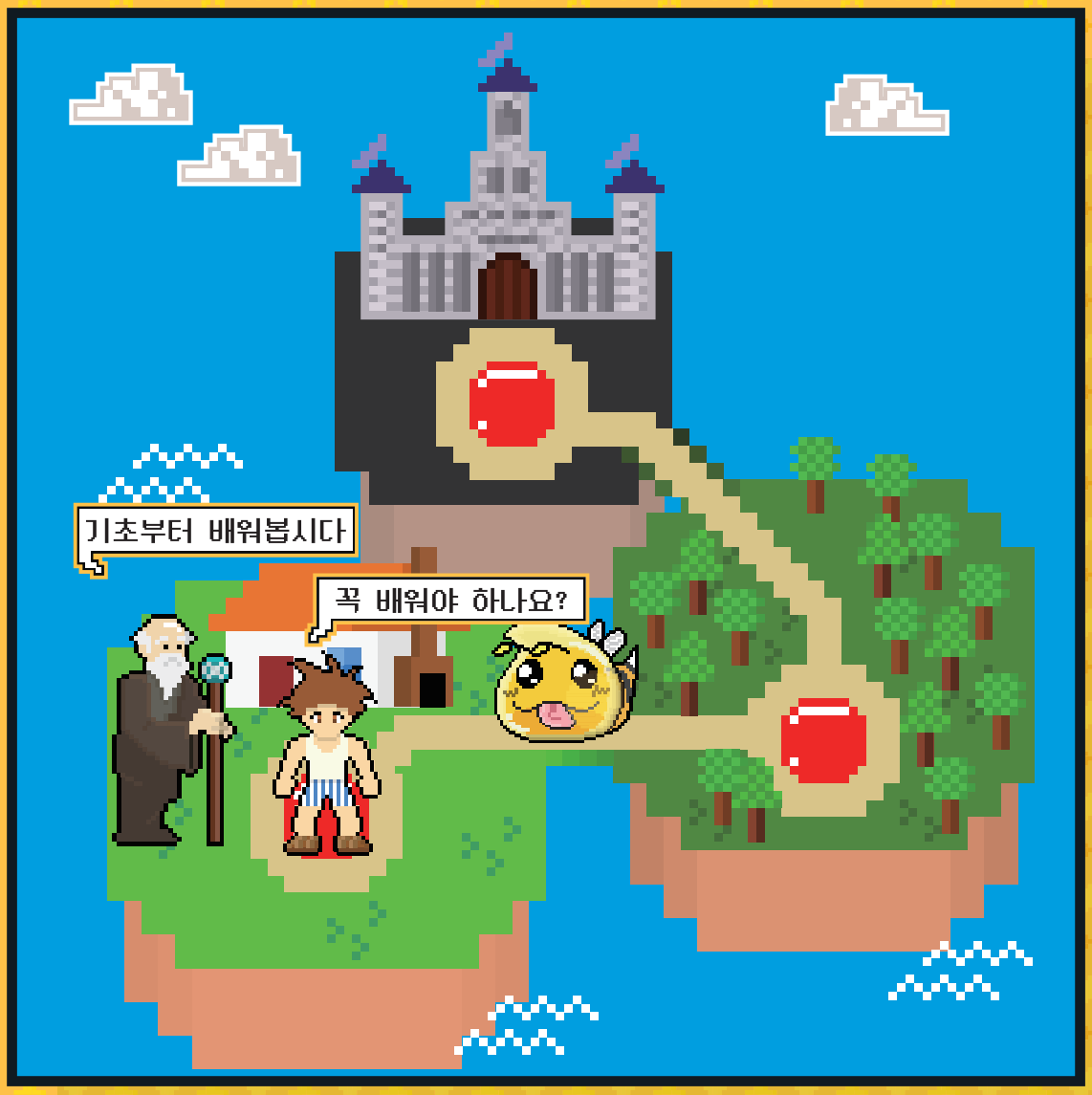
문제	정답
Trouble 1 – OX 문제	X
Trouble 2 – OX 문제	O
Trouble 3 – 객관식 문제	①
Trouble 4 – 객관식 문제	①
Trouble 5 – 주관식 문제	75%

1문제 이상 틀렸다면
이번 스테이지 처음부터 다시!

모두 맞추셨다면 다음페이지로!

용사는 『나무 신발』을 얻었다.

스테이지 1.2
전자 부품 기초



EXP♥

1. 전자 기초 지식

1.2. 전자부품 기초

1.2.1. SI 접두어

미터(m), 킬로그램(kg), 초(s)... 실생활에서도 많이 사용하는 단위들인데요. 이런 단위들은 '국제단위계'라고 불리는 'SI 단위계'입니다. 크고작은 다양한 단위들을 통합했기 때문에 다양한 단위를 보기 편하게 하기 위해 단위 앞에 알파벳 접두어를 붙이기로 한 약속입니다.

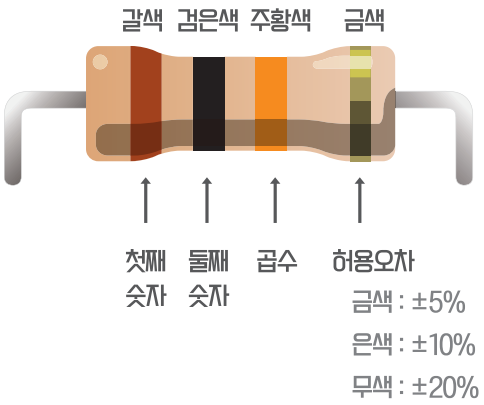
전자·IT분야에서 많이 사용되는 단위 위주의 표로 정리했습니다. 접두어가 많기도 하지만 여러가지 고려해야할 사항이 많습니다.

기호	접두어	발음	배수	10^n	10진수
T	Tera	테라	조	10^{12}	1 000 000 000 000
G	Giga	기가	십억	10^9	1 000 000 000
M	Mega	메가	백만	10^6	1 000 000
k	kilo	킬로	천	10^3	1 000
h	hecto	헥토	백	10^n	100
da	deca	데카	십	10^1	10
			일	10^0	1
d	deci	데시	십분의 일	10^{-1}	0 .1
c	centi	센티	백분의 일	10^{-2}	0 .01
m	milli	밀리	천분의 일	10^{-3}	0 .001
μ, u	micro	마이크로	백만분의 일	10^{-6}	0 .000 0001
n	nano	나노	십억분의 일	10^{-9}	0 .000 000 001
p	pico	피코	일조분의 일	10^{-12}	0 .000 000 000 001

1. 데시(d)는 부피 이외에는 잘 사용하지 않습니다.
2. 마이크로는 그리스문자 'μ'(뮤)를 사용하는데, 입력하기가 까다로워 자주 소문자 'u'로 대체됩니다.
3. 대소문자를 구분합니다. 특히 킬로(k)는 켈빈온도 단위(K)와 구분해야 하기 때문에 꼭 소문자로 표기합니다.

위의 표를 참고해보면 크고 작은 값들을 표기하기가 수월해집니다.
47000Ω의 저항은 47kΩ으로 표현할 수 있고, 0.015A는 15mA로 표현할 수 있습니다.

1.2.2. 저항 읽는 법



색상	쉬운 암기법	1번띠	2번띠	3번띠	승수	오차
검정색	흑	0	0	0	0	
갈색	갈	1	1	1	1	±1(F)
빨간색	적	2	2	2	2	±2(G)
주황색	등	3	3	3	3	
노랑색	황	4	4	4	4	
초록색	녹	5	5	5	5	
파랑색	청	6	6	6	6	
보라색	자	7	7	7	7	
회색	회	8	8	8	8	
흰색	백	9	9	9	9	
금색	금					±5(J)
은색	은					±10(K)

각 색상은 숫자를 의미합니다.
첫번째 띠, 10의 자릿수를 의미합니다.
두번째 띠, 1의 자릿수를 의미합니다.
세번째 띠, 10의 배수를 의미합니다.
네번째 띠, 오차를 의미합니다.

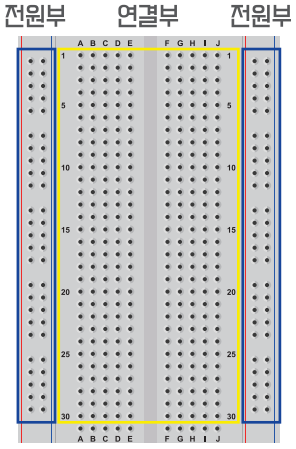
위의 저항을 보면 갈색, 검은색, 주황색, 금색으로 되어있는데요. 금색은 나중에 생각하고 갈색, 검은색, 주황색만 풀이해보면,

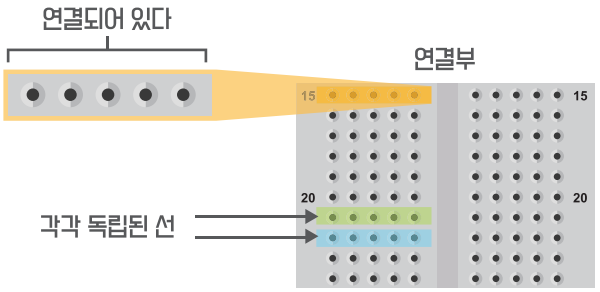
{ 10(갈색) + 0(검은색) } × 10^3(주황색)
= 10 × 10000
= 100000Ω
= 100kΩ

위의 예시의 저항은 100kΩ 저항이라는 걸 알 수 있습니다.
금색은 ±5%의 오차를 뜻합니다.
위 저항의 실제 저항값을 측정해보시면 95kΩ ~ 105kΩ 사이의 값을 측정하실 수 있습니다.

장로 윌프레드
SMD 타입의 매우 작은 저항들은 읽는 방법이 따로 있습니다.
앞으로 아두이노를 테스트하면서 쓰일 저항은 위의 그림과 같은 모양이기 때문에 따로 설명드리지 않았습니다. 홀홀
저항은 극성이 없는 수동소자로 연결방향은 상관없습니다.

1.2.3. 브레드 보드 사용방법





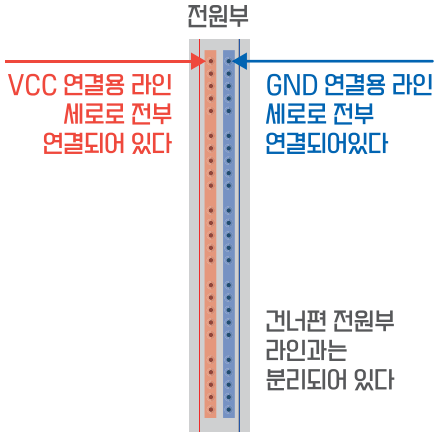
브레드 보드(Bread Board)는 전자 부품을 연결하여 회로를 연결할 수 있는 보드로 납땜을 하지않고 부품의 재사용이 쉬워서 테스트제품을 제작할 때 자주 쓰이는 보드입니다. 속칭 '빵판'이라고 불리며 빵을 썰 때 사용하던 도마에 못이나 철사등으로 전자 부품을 고정해서 임시로 사용했던 데에서 유래한 명칭입니다.

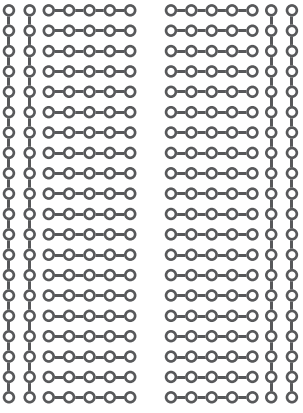
브레드 보드는 크게 두 부분으로 나눌수 있습니다. 빨간선과 파란선 두줄의 선이 있는 전원부와 구멍이 많이 나있는 연결부 입니다.

확대된 부분을 보면 가로 5칸은 서로 연결되어 있는 상태입니다. 위 아래 칸은 연결되어 있지 않습니다.

오른쪽 5칸과 왼쪽 5칸은 가운데 깊은 홈을 기준으로 서로 분리되어 있습니다. 이런 구조를 이용해서 납땜없이 회로를 구성할 수 있습니다.

전원부는 연결부와는 다르게 세로로 전부 연결이 되어 있습니다.





직류 회로 기준으로 빨간색은 전원(VCC)용 검정색 혹은 파란색은 그라운드(GND)용 라인입니다.

연결부와 크게 다른점이라면 구멍이 5칸씩 띄엄띄엄 떨어져 있지만 세로로 전부 연결되어 있다는 점 입니다.

전원부가 따로 없다면, 전원 분배에 많은 라인이 필요하기 때문에 전원 연결을 효율적으로 하기 위해 있는 부분입니다. 브레드 보드의 종류에 따라 전원부가 없는 제품도 있습니다.

브레드 보드의 내부 구조를 투시해보면 위(오른쪽) 같이 연결되어 있는 것을 볼 수 있습니다.

1.2.4. LED 극성 확인하는 방법

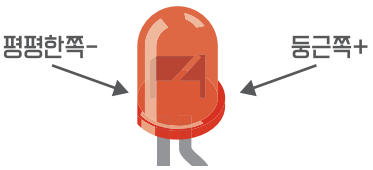
LED는 극성이 있는 반도체 소자입니다. 양극에 +를 음극에는 –를 연결해야 전류가 흐르고 작동합니다.

LED를 보면 **애노드(Anode)라고 부르는 +극과 캐소드(Cathode)라고 부르는 –극**을 구분하는 법이 있습니다.

측정장비를 이용하면 쉽게 확인할 수 있지만 장비없이 부품의 외형으로 확인하는 방법이 세가지 있습니다.

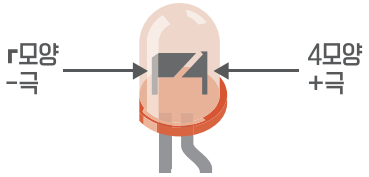
첫번째 방법은 LED의 다리 길이로 구분하는 방법입니다. 보통은 **긴쪽이 애노드(+)**입니다.

이 방법은 다리가 절단되었거나 제조 공정에 따라 차이가 있을 수 있습니다. 외우기는 제일 쉽지만 틀릴 수도 있는 방법입니다.



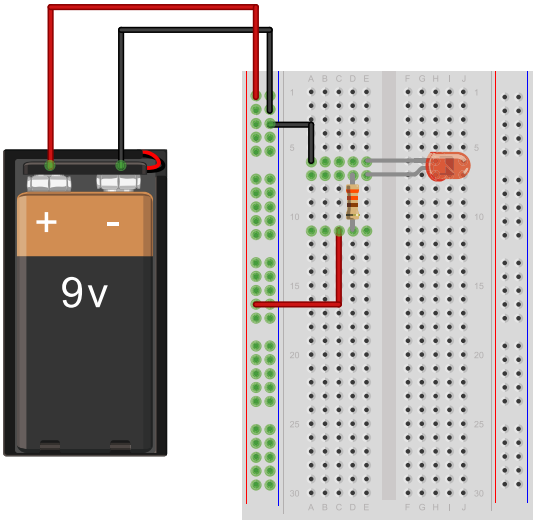
두번째 방법은 LED의 테두리의 평평함으로 구분하는 방법입니다. LED를 보면 알약을 절반 나눈것 처럼 생겼는데 위쪽 둥근부분 말고 다리쪽 면을 보면 부품이 걸릴 수 있게 테두리가 져 있습니다. 이 **테두리가 평평한 쪽이 캐소드(–)**입니다.

테두리가 없는 제품도 있기때문에 이 경우에는 다음 방법을 쓰는게 좋습니다.



세번째 방법은 LED 내부의 구조로 구분하는 방법입니다. 내부를 유심히 들여다 보시면 'r'모양으로 꺾인 부분이 있고 숫자 4 또는 깃발모양의 부분이 있습니다. 보통은 'r'모양이 크기가 더 크고 '4'모양을 덮고있는 형상입니다. 두 모양 중에 **4모양이 애노드(+)**입니다.

1.2.5. LED 점등시 저항 계산 방법



여기 브레드 보드와 9V 배터리, 그리고 일반 원형 LED가 있습니다. 배터리에서 바로 LED로 전원을 주게되면 LED가 감당할 수 있는 전류 이상의 전류가 인가될 수 있습니다. LED에 과전류가 흐르게 되면 LED가 고장날 수 있습니다. 따라서 저항을 달아주어 LED에 인가되는 전류를 조절해야 합니다.

저항에 걸리는 전류만큼 LED에 전류가 인가되기 때문입니다.

소자가 안정적으로 동작할 수 있는 전압/전류를 각각 정격전압, 정격전류라고합니다.

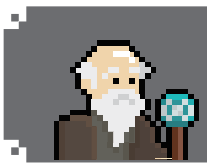
LED의 정격 전압/전류는 LED의 데이터 시트를 찾아보면 알 수 있습니다.

여기서 사용한 LED의 정격 전압이 2V, 정격 전류가

20mA라 가정했을 때, 옴의 법칙을 사용해 저항값을 구해보겠습니다.

저항 값 = (입력전압 9V – 정격 전압 2V) / 정격 전류 0.02A = 350Ω
350Ω의 저항을 달아주시면 됩니다.

정확히 350Ω의 저항을 LED와 직렬로 연결했다면 LED에 2V 20mA의 전류가 흐르기 때문에 안정적으로 동작합니다.

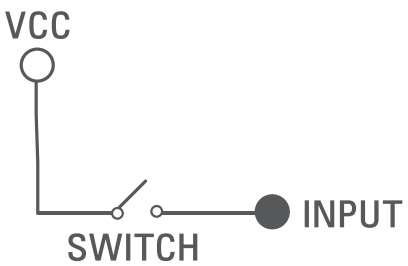


장로 윌프레드
위의 회로에서는 저항을 +극 쪽에 연결했지만 -극에 연결해도
LED에 인가되는 전압은 동일하답니다. 흠흠

1.2.6. 풀업(Pull-up)과 풀다운(Pull-down)

풀업 / 풀다운을 이해하기 위해서는 먼저 플로팅(floating)현상을 알아야 합니다.
아두이노에서 택트 스위치를 연결할 때, 아두이노를 사용하시는 초보자 분들은 저항을 달지 않고 아두이노 디지털 IO포트에 연결하는 일이 있습니다.

이런식으로 연결하게 되면 IO포트의 입력을 인식하지 못하거나 비정상적으로 인식될 수 있습니다. 그 이유는 **플로팅 현상** 때문인데요. 플로팅(floating) 현상이란 영어단어의 뜻 그대로 떠 있는 상태라는 뜻입니다.
디지털신호는 0(LOW, 0V)과 1(HIGH, 5V)로 이루어져 있어야 하는데 양쪽 어디에도 연결되어있지 않기때문에 공중에 붕 떴다는 표현처럼 신호 인식이 흐지부지 됩니다.

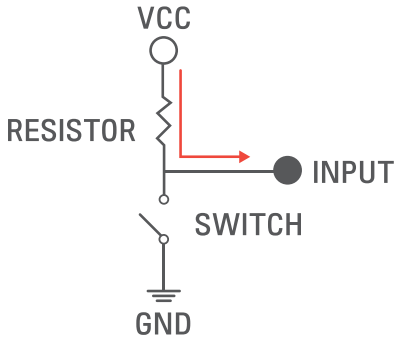


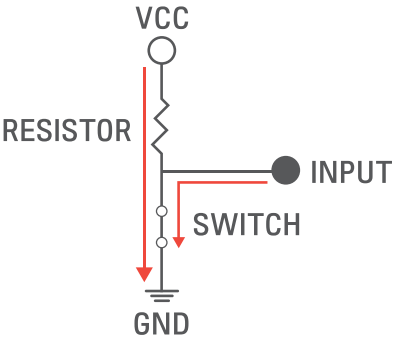
잘못된 회로이긴 하지만 왼쪽 회로처럼 연결하면 INPUT은 원래 HIGH를 인식해야합니다. 하지만 실제로는 값을 일정하게 인식할 수 없는 현상이 발생합니다. 아누이노 등 입력받는 장치 내외부의 주변회로의 영향으로 이런 현상이 발생합니다.

이 현상을 없애기 위해서는 풀업(Pull-up) 저항 혹은 풀다운(Pull-down) 저항을 사용해야 합니다.

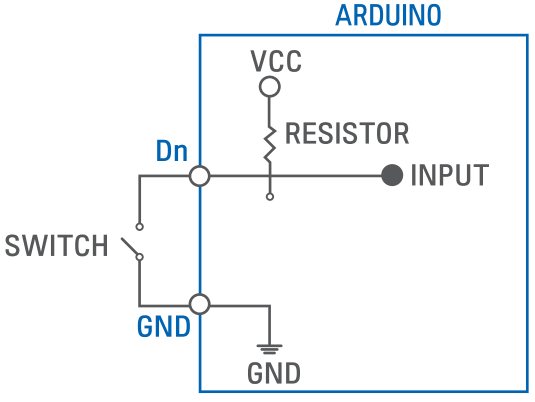
용사
아두이노 바보ㅠㅠ,
연결했는데 왜 못알아 듣는거니?

풀업 저항(Pull-up)이란 전원쪽에 저항을 연결해서 플로팅 현상을 해결하는 방법입니다.
연결하는 방법은 다음과 같습니다.



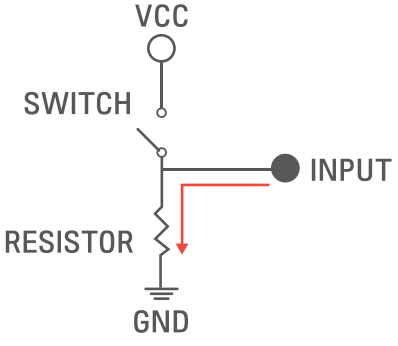


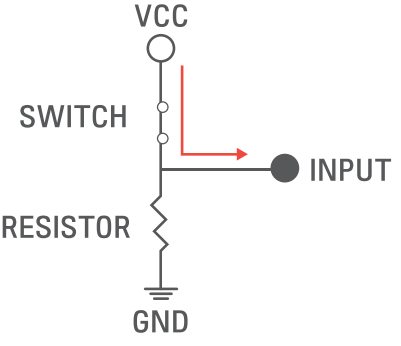
스위치가 눌리지 않았을 때 전류는 전원에서 IO핀으로 흐릅니다. 따라서 IO핀의 값은 HIGH입니다.
스위치가 눌렸을 때, 모든 전류의 방향은 그라운드로 진행합니다. 따라서 IO핀의 값은 LOW가 됩니다.



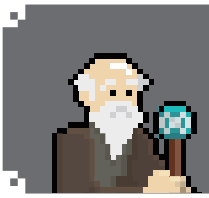
아두이노는 하드웨어적으로 풀업 저항회로가 내장되어있습니다.
그냥 사용할 수 있는건 아니고 프로그래밍을 통해 소프트웨어적으로 설정해주면 사용할 수 있습니다.

풀다운 저항(Pull-down)이란 풀업 저항과는 반대의 개념입니다. 저항을 그라운드(0V)에 연결해서 플로팅 현상을 해결하는 방법입니다.





스위치가 눌리지 않았을 때, 전류의 방향은 IO핀에서 그라운드로 진행합니다. 따라서 INPUT의 값은 LOW입니다.
스위치가 눌렸을 때, 전류의 방향은 전원에서 IO핀으로 진행합니다. 따라서 INPUT의 값은 HIGH입니다.



장로 윌프레드
왜 전류가 GND로 흐르지 않고 INPUT으로 흐르는지 잘 모르시겠죠?
전류는 저항이 있는 곳으로 흐르기 어렵기 때문입니다.
같은 0V라도 저항이 적은쪽으로 전류가 흐릅니다.

구분	스위치를 눌렀을때	스위치를 안눌렀을 때
Pull-Up	LOW	HIGH
Pull-Down	HIGH	LOW

풀업저항과 풀다운 저항을 적용한 스위치 회로에 INPUT의 값을 표로 정리하면 위와 같습니다.
일반적으로 풀다운 저항보다는 풀업 저항이 노이즈에 더 좋기 때문에 풀업저항이 많이 사용됩니다. 풀업 저항의 값은 보통 10kΩ~100kΩ까지가 가장 많이 쓰입니다.
풀업 저항의 계산법은 마이크로컨트롤러의 IO핀이 받아들일 수 있는 전류의 양에 따라 다릅니다. 예를 들어 5V 전압에 1mA의 전류가 필요하다면 옴(Ω)의 법칙(V=I x R)에 따라서

R = V / I
R = 5V / 0.001A
R = 5kΩ

5kΩ이 필요하게 됩니다.
이 값은 LED처럼 정밀할 필요가 없기때문에 너무 저항이 크거나 작지 않은 적당한 저항을 사용하시면 됩니다.

용사
알아야 할게 너무 많아요.

장로 윌프레드
양은 많지만 어려울건 없습니다. (단호)
용사님은 충분히 익히실 수 있습니다.

Quest 2. 실력 테스트

장로 윌프레드

큰일입니다. 용사님. 허니버터슬라임이 마을을 습격했다고 합니다!
아이고 이놈들아~ 내 오도독은 안된다. 용사님! 제발 마을을 지켜주세요.

Trouble 1 – OX 문제

마이크로(μ)는 10^{-6}이다.

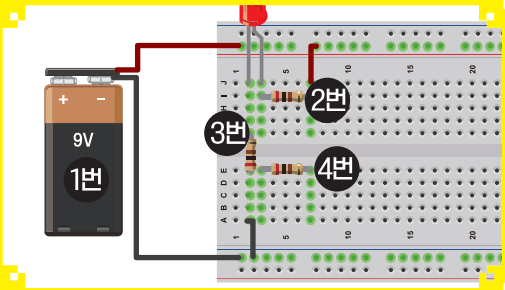
Trouble 2 – 주관식 문제

오른쪽 저항의 값은 몇 Ω인가?

Trouble 3 – 객관식 문제

다음 중 회로에 연결되어있지 않은 부품은 무엇인가?

① 배터리
② 200Ω저항
③ 100Ω저항
④ 100Ω저항



Trouble 4 – OX 문제

LED의 다리가 긴쪽이 애노드다.

Trouble 5 – 주관식 문제

정격전압 3V의 정격전류 60mA의 LED를 구동하려고한다.
입력이 12V일때 저항값을 구하시오.

Trouble 6 – 주관식 문제

아두이노의 INPUT과 VCC를 곧바로 연결했을 때 값을 일정하게
인식할 수 없는 현상을 무엇이라고 하는가?

용사

헉헉... 많았지만 다 처리한 것 같아요

장로 윌프레드

고생하셨습니다. 용사님께서 해내시리라 믿고 있었습니다 홀홀

Quest 2. 정답 체크

맞춘 문제 갯수 만큼 슬라임에 X표 해주세요.

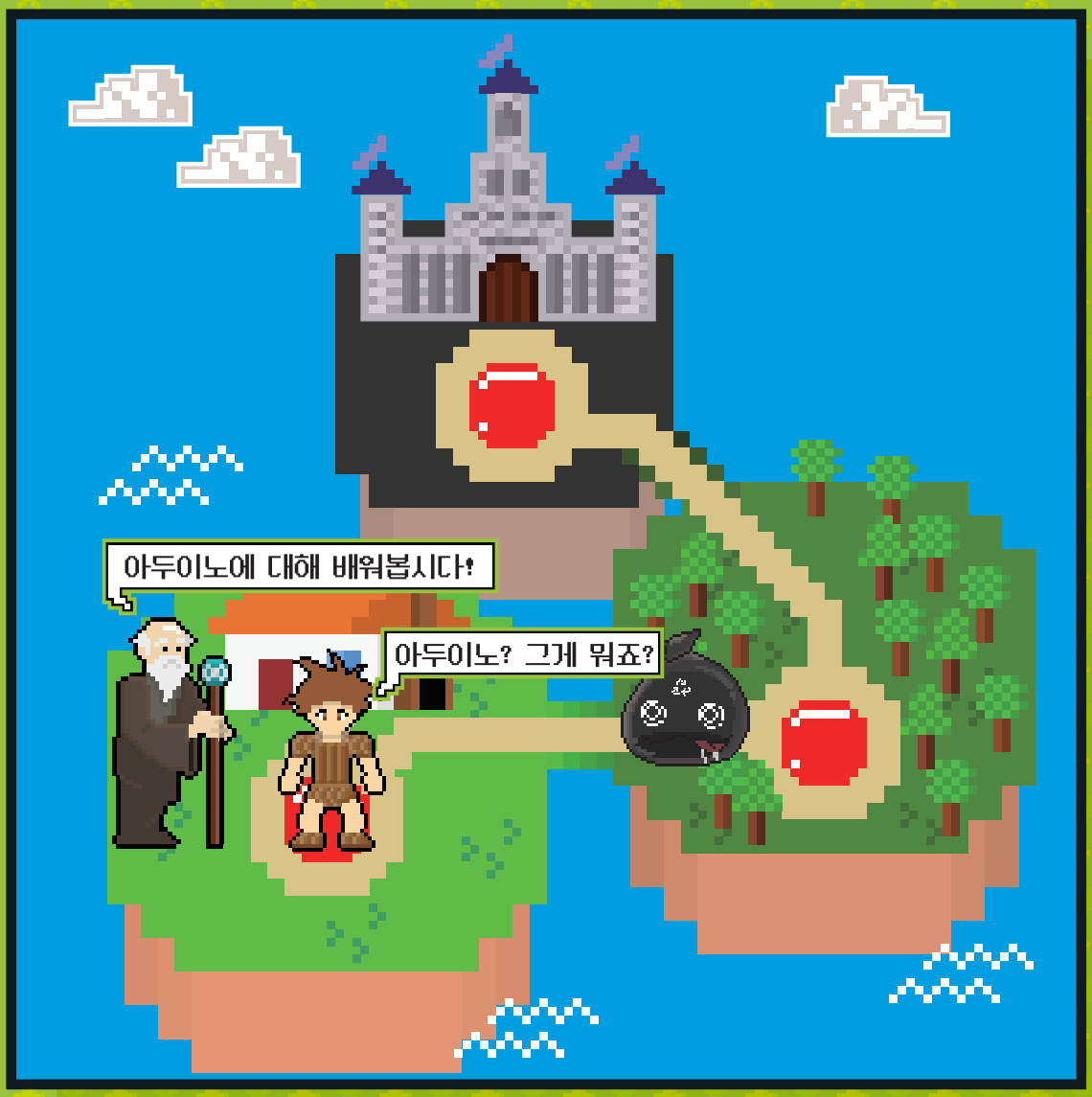
문제	정답
Trouble 1 – OX 문제	O
Trouble 2 – 주관식 문제	4.7kΩ
Trouble 3 – 객관식 문제	④
Trouble 4 – OX 문제	O
Trouble 5 – 주관식 문제	150Ω
Trouble 6 – 주관식 문제	플로팅 현상

1문제 이상 틀렸다면
이번 스테이지 처음부터 다시!

모두 맞추셨다면 다음페이지로!

용사는 『나무 갑옷』을 얻었다.

스테이지 2.1
아두이노 소개



EXP♥

2. 아두이노 시작하기

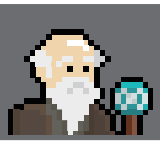
2.1. 아두이노 소개

2.1.1. 아두이노란?

아두이노(Arduino)란 2005년 이탈리아에서 하드웨어에 익숙지 않은 디자인 학교 학생들이 작품을 쉽게 제어하기 위해 개발한 개발 보드 및 개발 환경의 총칭입니다.

아두이노사(社)에서 제작된 보드뿐만 아닌 공개된 관련 자료로 아두이노 개발 환경에 호환되도록 만든 제품들을 모두 포함한 **오픈 소스 마이크로 컨트롤러 보드**(Micro Controller Board)입니다.

용사
앗 그런데 저는 아두이노가 없는데요?



장로 윌프레드
용사님 갑옷의 주머니안에 넣어두었답니다.

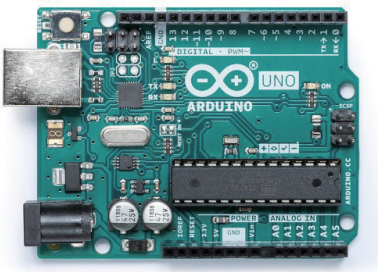
용사
앗 여기있다!

용사는 『아두이노 Uno』를 얻었다.

2.1.2. 아두이노 종류

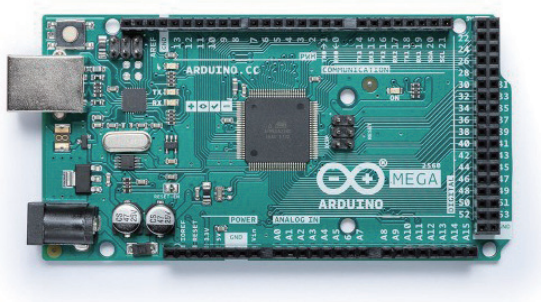
아두이노는 오픈소스 특성상 아두이노 회사에서 판매하는 정품 보드만 있는게 아니고 호환 보드 및 호환 쉴드까지 아두이노의 범주에 속하기 때문에 종류가 아주 많습니다.

그래서 일부분이지만 가장 널리 사용되는 아두이노 보드 몇 종류만 간단하게 살펴보겠습니다.



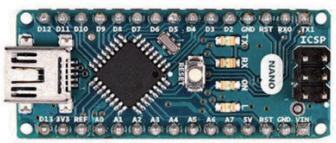
아두이노 우노 Arduino UNO

아두이노의 표준 제품, 많은 예제와 소스가 UNO를 기준으로 하고 있습니다.



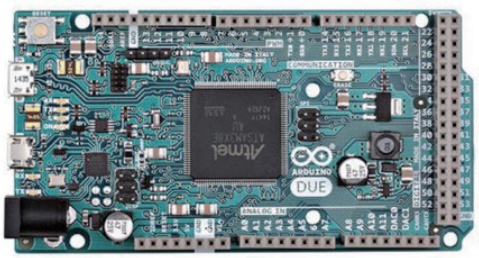
아두이노 메가 Arduino Mega

아두이노 우노에 비해 핀의 갯수와 성능을 상향시킨 보드입니다.



아두이노 나노 Arduino Nano

아두이노 우노를 축소하여 경량화, 소형화시킨 보드입니다.



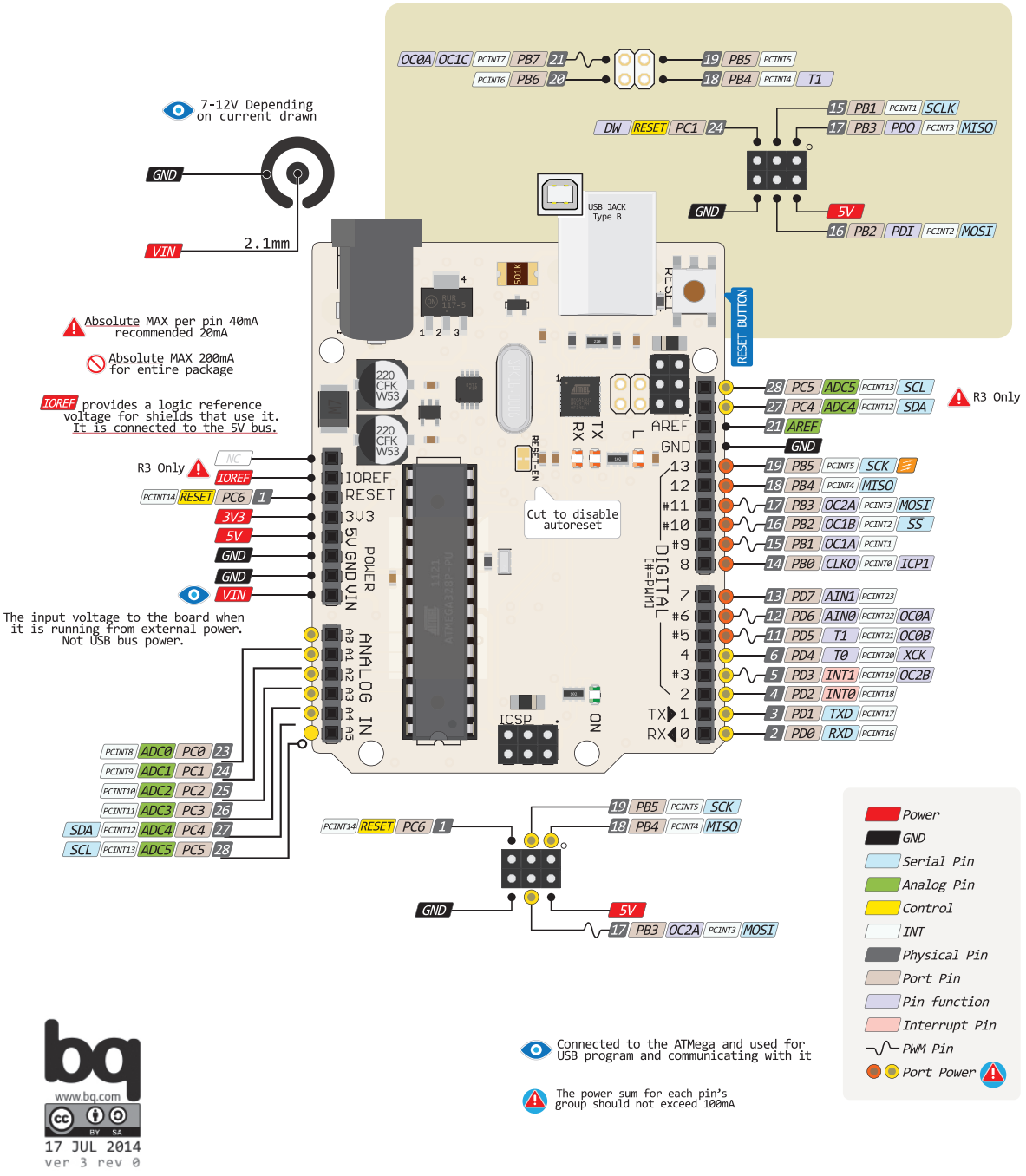
아두이노 두에 Arduino Due

ARM사의 Cortex M3를 사용한 보드로, UNO에 비하면 속도와 성능이 비약적으로 상향된 보드입니다.

용사

그러니까… 내가 갖고있는 아두이노는 아두이노 우노구나!!

2.1.3. 아두이노 우노(UNO) 핀맵

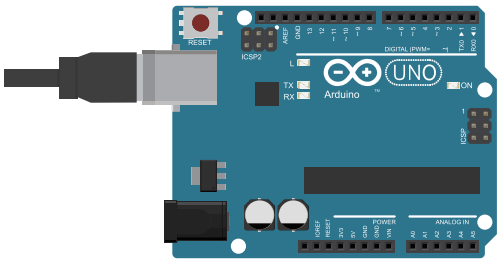


장로 윌프레드

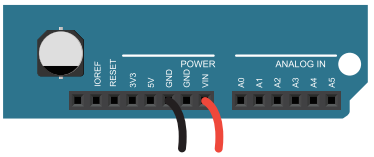
용사님이 갖고계신 아두이노 우노는 13개의 디지털 입출력 포트와 6개의 아날로그 입력 포트로 이루어져있습니다. 설명이 너무 길어지니 다른 부분들은 차차 알아가도록 하겠습니다.

2.1.4. 전원 공급하기

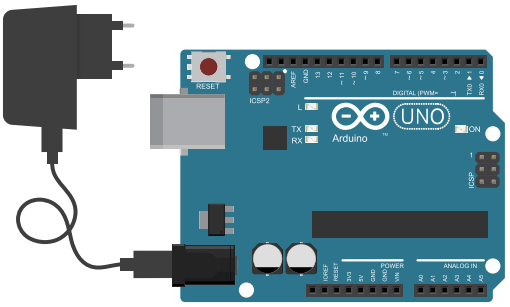
아두이노 우노는 5~12V의 전원을 인가받아 동작되며, 권장 입력 전압은 9V입니다.
전원을 공급해 주는 방법은 아래와 같습니다.



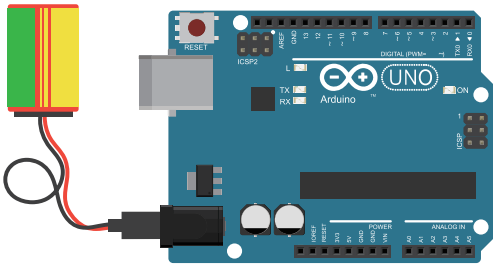
USB 단자로 전원 공급



VIN핀으로 전원 공급



DC 어댑터로 전원 공급



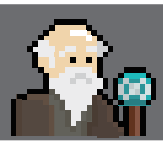
9V 건전지로 전원 공급

아두이노는 5V를 사용하는 TTL(transistor-transistor logic)방식의 보드입니다. 5V 이상의 전원을 공급하는 이유는 내장된 정류회로를 거쳐 전원이 공급되기 때문입니다.

장로 윌프레드

9V 배터리 뿐만 아니라 1.5V 배터리등의 배터리를 직렬로 연결해서
5 ~ 12V의 전압을 만들어 준다면 전원 공급이 가능하답니다. 흠흠

Quest 3. 실력 테스트



장로 윌프레드
이번에는 간단하게 쓰봉슬라임을 상대로 용사님의 기량을 한번 보겠습니다.

Trouble 1 – 객관식 문제
다음 중 아두이노 우노의 입출력 포트 갯수로 올바른 것은?
① 디지털 입출력 12 포트, 아날로그 입력 6포트
② 디지털 입력 13 포트, 아날로그 입출력 6포트
③ 디지털 입력 12 포트, 아날로그 입출력 6포트
④ 디지털 입출력 13 포트, 아날로그 입력 6포트

Trouble 2 – 주관식 문제
아두이노를 작동하기 위해 필요한 전압의 범위는?

Quest 3. 정답 체크

맞춘 문제 개수만큼 슬라임에 X 표해주세요.

문제	정답
Trouble 1 – 객관식 문제	④
Trouble 2 – 주관식 문제	5V ~12V

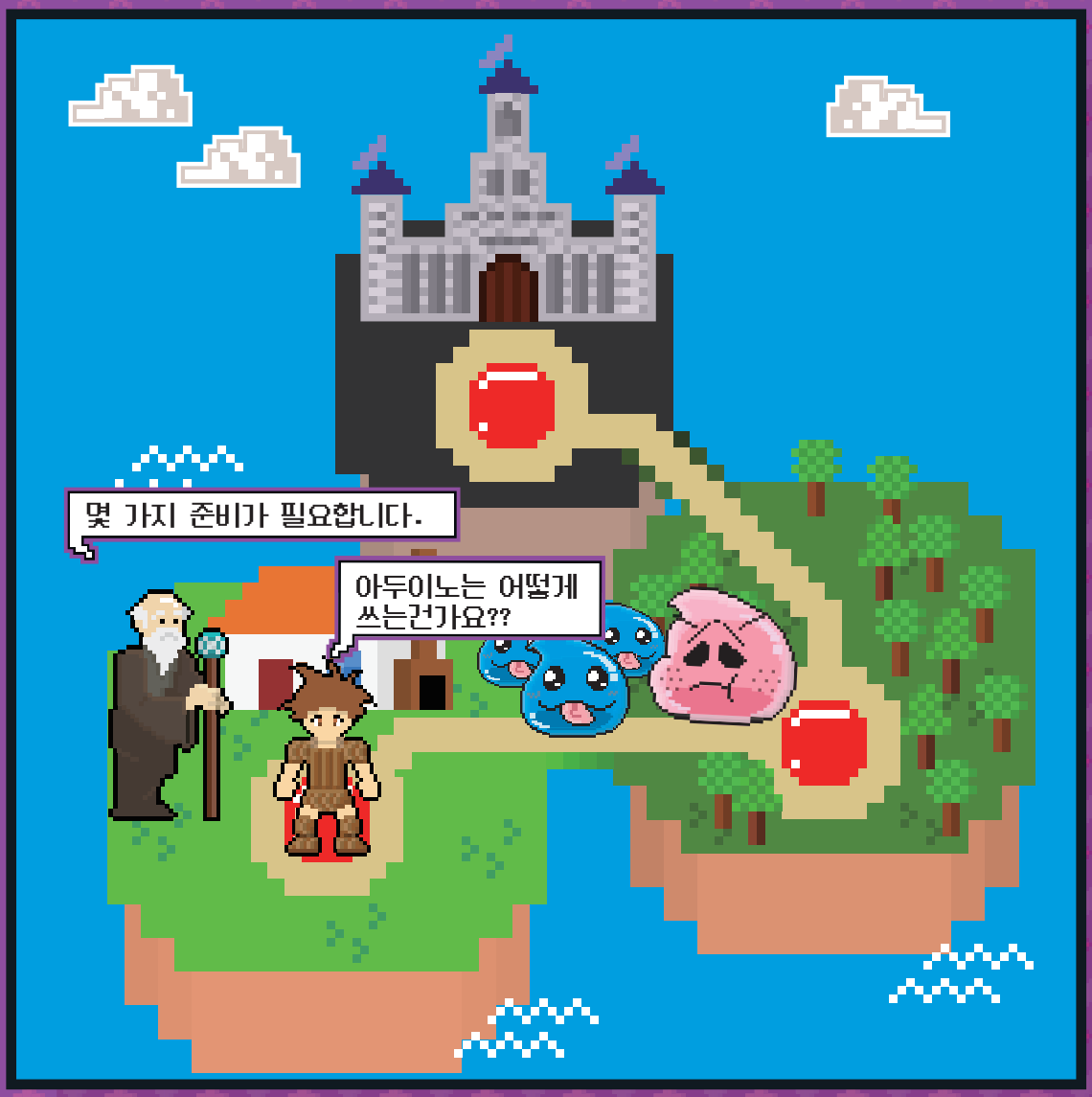
1문제 이상 틀렸다면
이번 스테이지 처음부터 다시!

모두 맞추셨다면 다음페이지로!

용사는 『나무 각반』을 얻었다.

스테이지 2.2

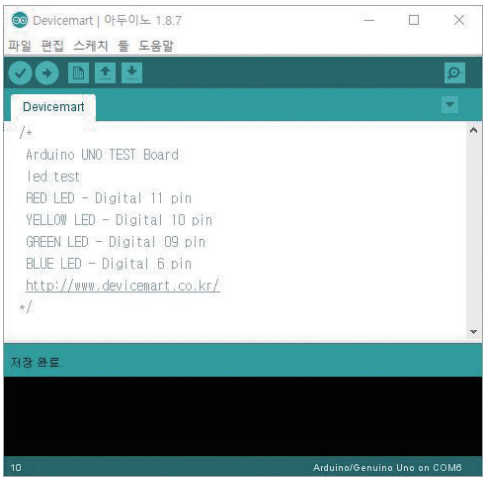
아두이노 통합개발환경(IDE)



EXP♥

2. 아두이노 시작하기

2.2. 아두이노 통합개발환경(IDE)

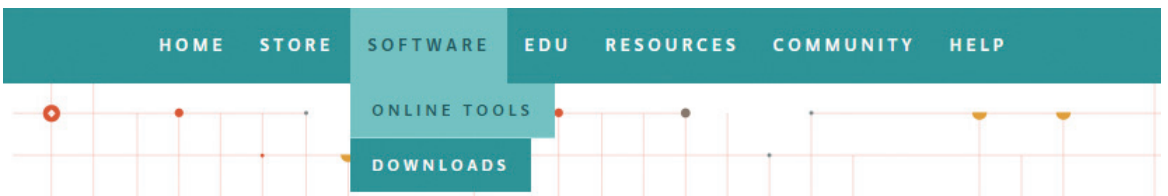


아두이노 IDE(Integrated Development Environment)는 아두이노 프로그램을 작성하고 컴파일할 수 있는 통합 개발 환경입니다. 아두이노 공식 홈페이지에서 무료로 제공하고 있어서 편리하게 사용할 수 있습니다.

장로 윌프레드

아두이노에 넣을 프로그램을 작성하고 업로드하기 위한 편리한 개발 환경입니다. 홀홀

2.2.1. 아두이노 통합개발환경(IDE) 설치



아두이노 통합개발환경(IDE)은 아두이노 공식 홈페이지(https://www.arduino.cc)에서 다운받을 수 있습니다.

Download the Arduino IDE

아두이노 공식 홈페이지에 접속하신후 **SOFTWARE ▶ DOWNLOADS**를 클릭 해주세요.

Download the Arduino IDE 부분에서 각자 운영체제에 알맞는 버전의 파일을 선택해서 다운받으시면 됩니다.

파일을 선택하시면 아두이노 소프트웨어에 대한 기부를 권하는 창이 뜹니다. 여기서는 따로 기부를 하지 않고 진행하며, JUST DOWNLOAD를 클릭하여 다운로드해주세요.

이후 설치는 운영체제에 따라 다르기 때문에 생략합니다. Window10은 마이크로소프트 스토어에서 바로 설치가 가능합니다.

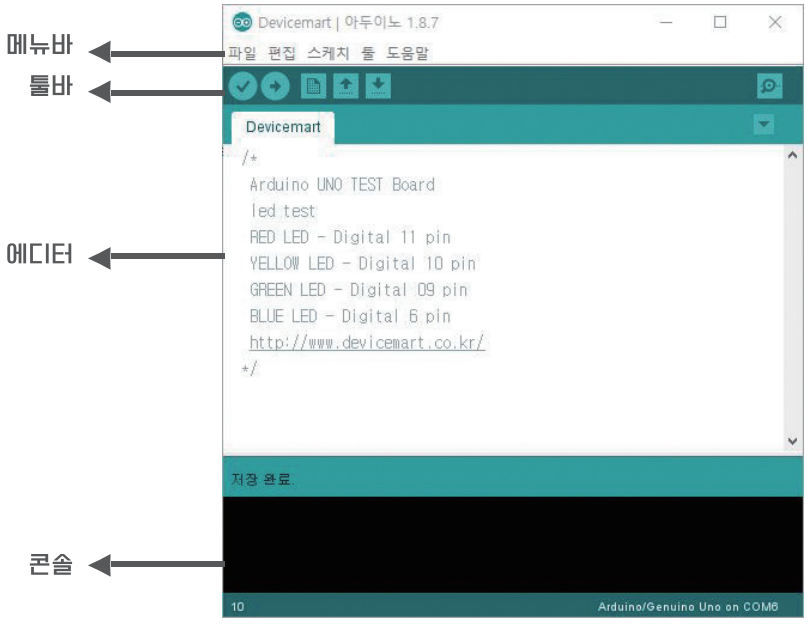
장로 윌프레드

설치는 시키는 대로만 해도 됩니다. 어렵지 않답니다. 흘흘

2.2.2. 아두이노 통합개발환경(IDE) 구조

아두이노 통합개발환경은 스케치(Sketch)라고도 부릅니다.

아두이노 스케치의 구조에 대해 알아봅시다.



스케치 프로그램은 메뉴바, 툴바, 에디터, 콘솔로 이루어져 있습니다.

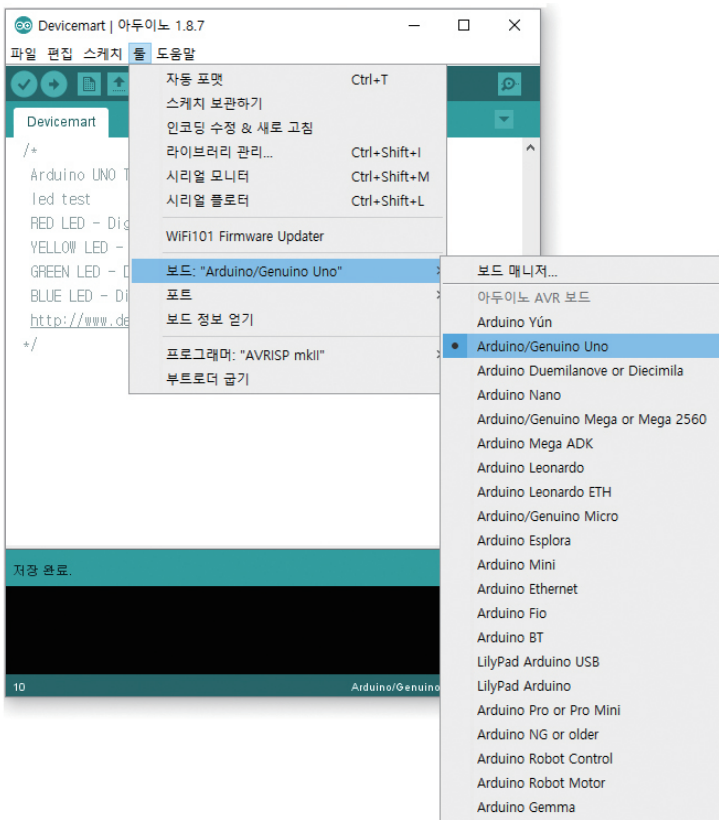
메뉴바(Menu bar)
파일, 편집, 스케치, 툴 등을 설정할 수 있습니다.

툴바(Tool bar)
왼쪽부터 순서대로 확인, 업로드, 새 파일, 열기, 저장 그리고 시리얼 모니터를 빠르게 실행할 수 있습니다.

에디터(Editor)
코드 소스를 작성하는 공간입니다.

콘솔(Console)
에러 메시지 등의 프로그램 상태를 나타내는 공간입니다.

2.2.3. 아두이노 통합개발환경(IDE) 기본 세팅



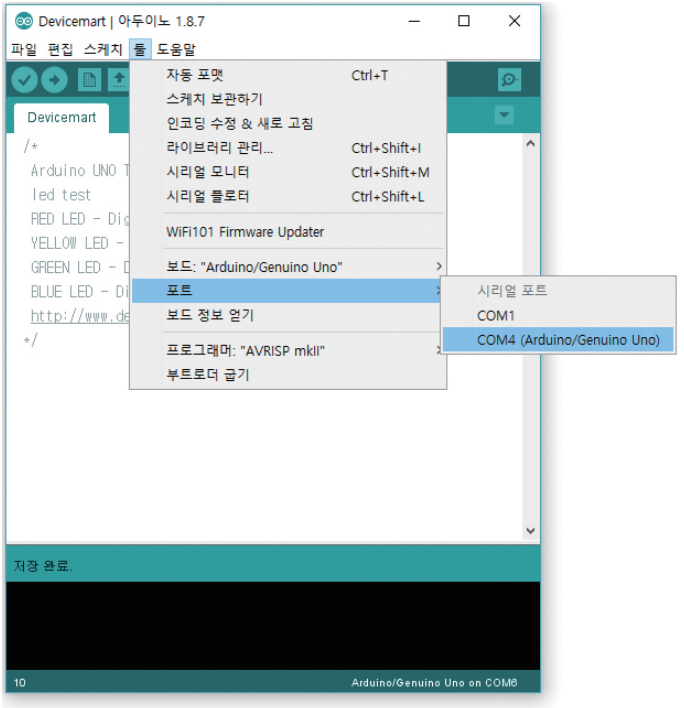
스케치(IDE)를 사용하는 방법에 대해 알아봅시다.

USB 케이블로 아두이노를 PC에 연결하고 보드(Board)와 포트(Port)를 맞춰줘야 합니다.

보드를 선택하는 방법은 [메뉴바–툴–보드–Arduino/Genuino Uno]를 선택해주면 됩니다.(아두이노 우노 기준)

다른 아두이노라면 종류에 맞는 아두이노를 선택해주시면 됩니다.

앞으로 나올 아두이노 예제는 이렇게 **컴퓨터와 아두이노를 연결한 상태**로 따라하시면 됩니다.



포트를 선택하는 방법은 [메뉴바-툴-포트]에서 아두이노에 해당하는 COM 포트를 선택해 주면 됩니다.

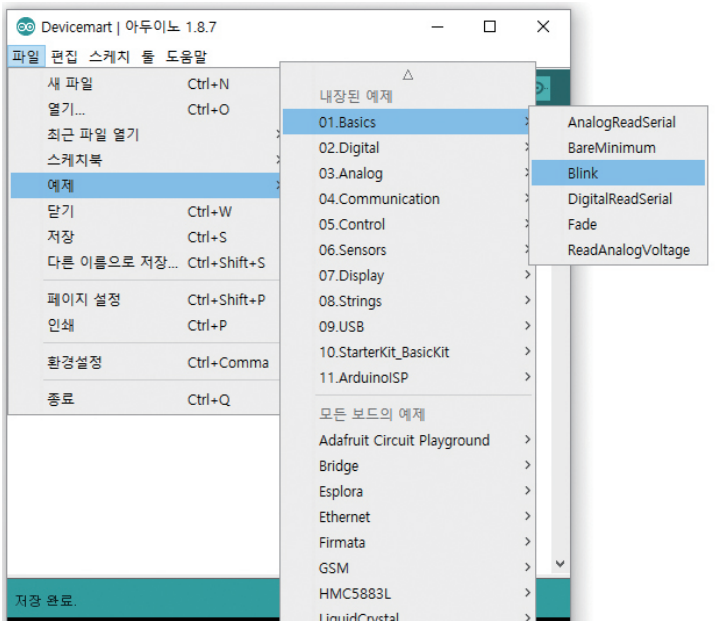
만약 정품 보드가 아닌 호환 보드는 포트 뒤에 (Arduino/Genuino Uno)가 적혀 있지 않습니다.

호환보드이더라도 올바른 COM 포트 선택을 해주셔야합니다.

어느 포트가 맞는 포트인지 모를 경우에는 아두이노에 케이블을 뺐다가 꽂았을때 노출되는 COM 포트를 선택해주시면 됩니다.

2.2.4. 아두이노 예제 실행하기

아두이노 기본 설정이 끝났으니 스케치에 기본적으로 포함되어 있는 간단한 기본 예제를 실행해 보겠습니다.



[메뉴바-파일-예제-01.Basics-Blink]를 눌러주세요.

해당 예제는 아두이노 13번핀과 연결되어 있는 LED가 1초 간격으로 깜박이는 예제입니다.

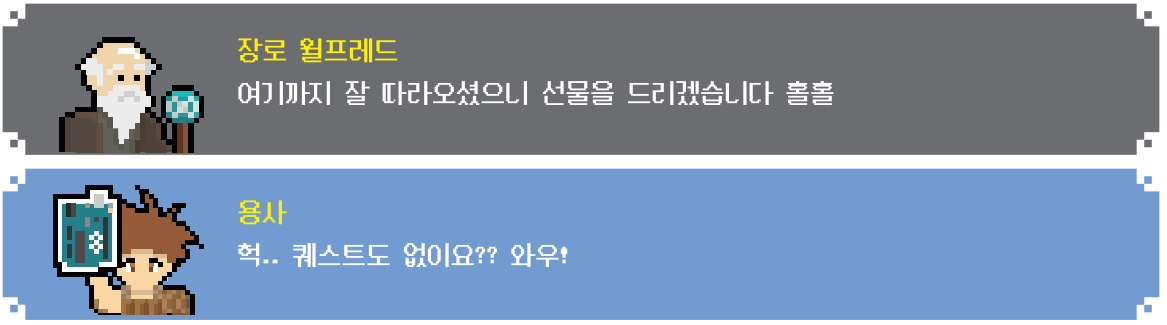
예제 소스를 열으셨다면 툴바(Tool bar)에 있는 확인버튼 버튼이미지를 눌러주세요. 소스코드에 문제가 없다면 콘솔창(Console)에서 "컴파일 완료"라고 표시됩니다.

이 작업은 사람이 읽을 수 있는 소스코드를 0과 1뿐인 기계어로 변환해주는 작업으로 컴파일이라고 합니다.

컴파일이 완료되었다면, 툴바에 있는 업로드 버튼(오른쪽 방향의 화살표 모양)을 눌러주세요.

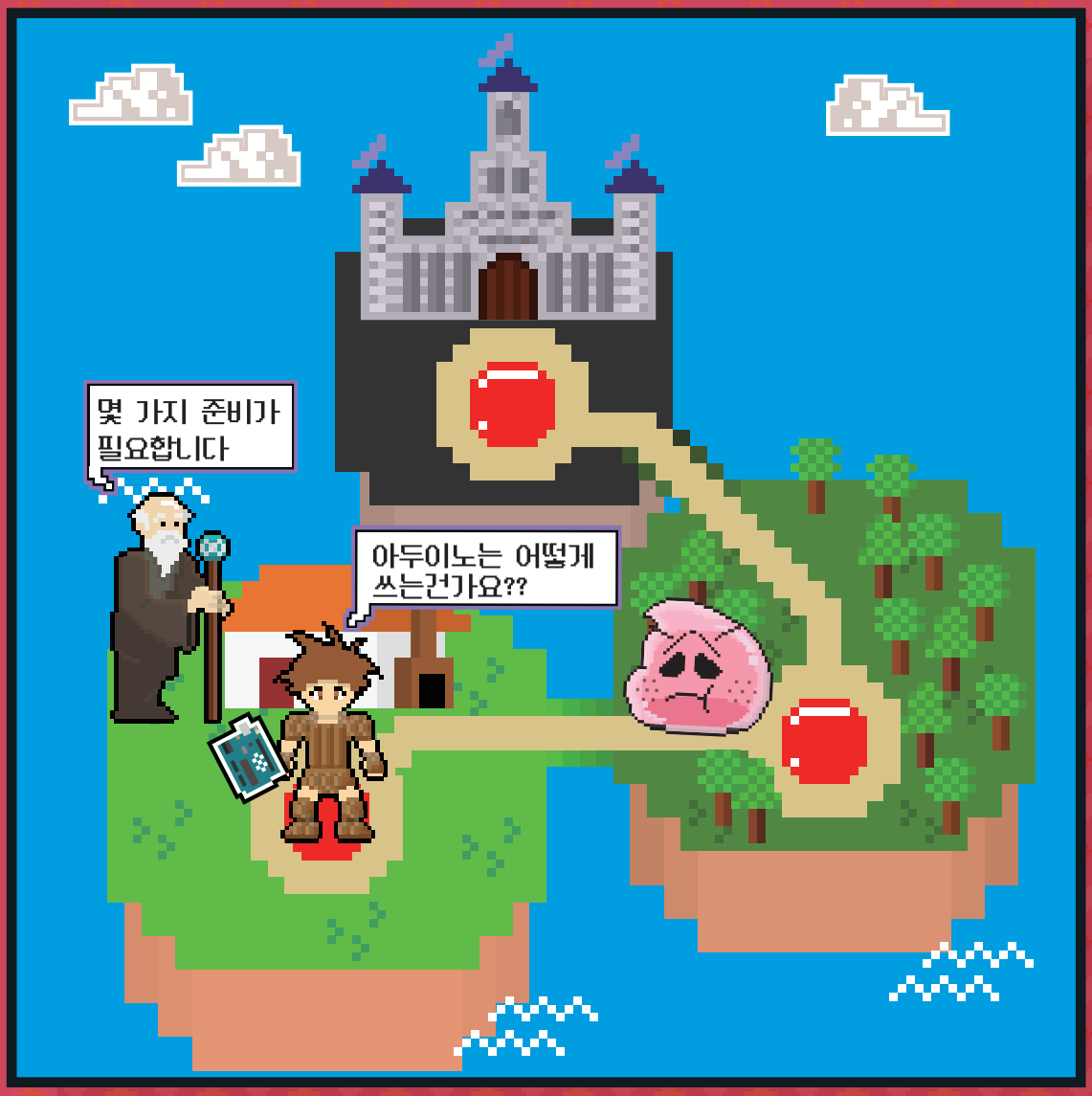
이 작업을 해야 우리가 작성한 소스코드를 아두이노에서 확인해 볼 수 있습니다.

문제가 없다면 아두이노에서 13번 LED가 1초 간격으로 On/Off를 반복합니다.



용사는 『나무 건틀릿』을 얻었다.

스테이지 2.3
아두이노 프로그래밍



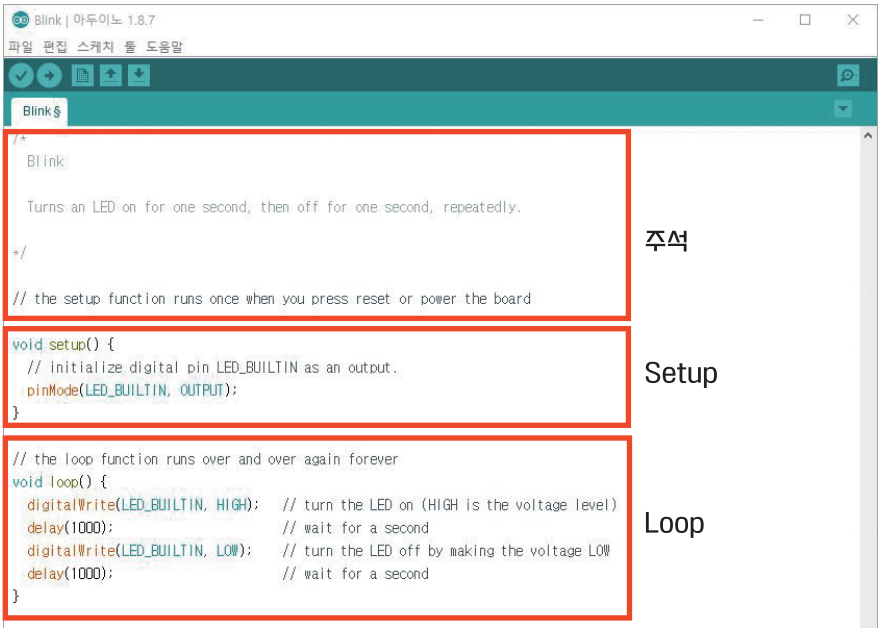
EXP♥

2. 아두이노 시작하기

2.3. 프로그래밍

2.3.1. 아두이노 프로그래밍

아두이노 프로그래밍을 하기 위해서는 C/C++ 언어의 기초 지식이 필요합니다. 아두이노는 비전공자가 쉽게 프로그래밍 할 수 있도록 개발된 보드이므로, 기초적인 프로그래밍 언어 지식만 가지고도 쉽게 코딩을 할 수 있습니다.



주석

안에 있는 코드는 컴퓨터가 읽지 못하는 문장으로, 사람들이 코드의 설명을 위해 사용하는 부분입니다. 주석처리를 하는 방법에는 두가지가 있습니다.

첫번째로는 "/*"와 "*/"로 감싸서 범위를 지정하여 문장을 쓰는 방법이 있고

/*문장*/

두번째 방법으로는 앞에 "//"를 붙여서 한줄만 주석처리 하는 방법이 있습니다.

//문장

setup()

setup() 함수는 프로그램이 시작할 때 딱 한번 실행되는 함수로, 주로 하드웨어 설정, 변수 선언 등을 할 때 사용합니다.

loop()

loop() 함수는 프로그램의 메인 함수로, 전원이 들어오는 동안은 무한 반복하는 함수 입니다.

2.3.2. 시리얼 통신

시리얼 통신(Serial Communication)은 한번에 한 비트씩 데이터를 지속적으로 주고 받는 직렬 통신을 의미합니다. USB포트를 통해 PC에서 아두이노로 소스를 업로드하는 것 또한 시리얼 통신입니다.

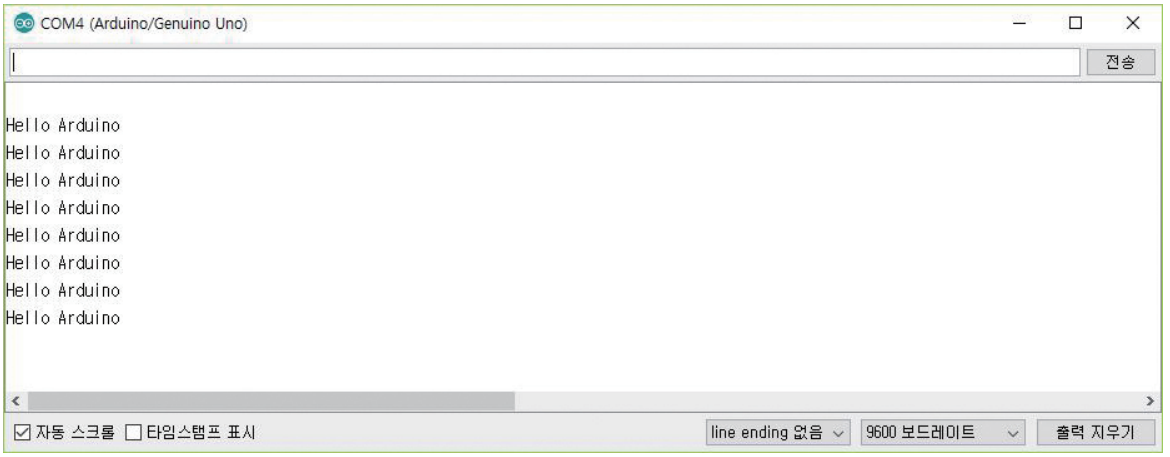
디지털 입출력 포트의 D0와 D1을 통해 통신을 합니다. 이 두 핀을 **하드웨어 시리얼**이라고 부릅니다.

시리얼 통신을 통해 아두이노와 PC 혹은 다른 장치와 통신을 할 수 있습니다. 간단한 예제를 통해 확인해 보도록 하겠습니다.

예제소스

```
/*
 시리얼 통신
*/

void setup() { //초기 설정 함수.
  Serial.begin(9600); //시리얼 통신 속도 9600으로 설정

}

void loop() { //반복 함수 프로그램의 주 내용이 들어간다.
 Serial.println("Hello Arduino");  //Hello Arduino를 출력
 delay(1000);                      //1초 동안 시간을 지연시켜라.
}
```

소스코드를 작성한 뒤 저장 후 아두이노에 업로드 시킵니다. 업로드가 완료되면 툴바 오른쪽 끝의 [시리얼 모니터] 버튼 모양을 눌러 동작을 확인할 수 있습니다.



1초마다 "Hello Arduino"라는 데이터를 출력하는 소스입니다.

Serial.begin()

시리얼 통신의 시작을 알리며 속도를 설정하는 함수입니다. 일반적으로 통신속도(Baud Rate)는 9600을 사용하며 단위는 bps(bits per second)입니다.

Serial.println()

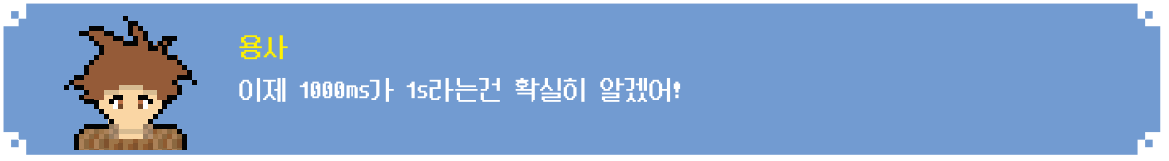
출력을 해주는 함수로, 기본적으로 출력 후 줄바꿈(₩n)을 해줍니다. 줄바꿈이 필요하지 않을 시 Serial.print() 함수를 사용하면 됩니다. 문자열을 출력할 경우에 괄호 안에 반드시 큰따옴표("")를 쓰고 그 안에 쓰셔야 합니다.

ex) Serial.println("Have a good time!");

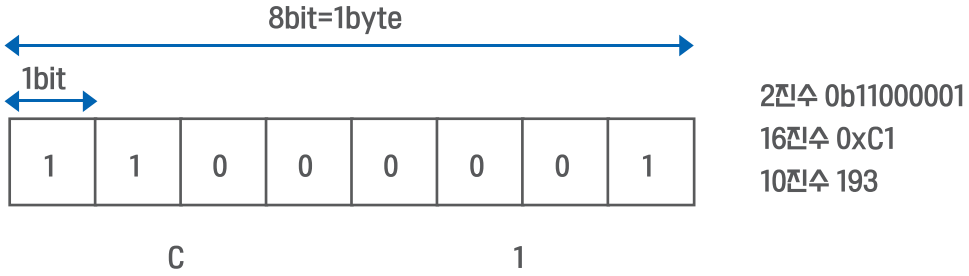
문자열 외에도 후에 배울 변수를 입력하여 변수의 값을 출력할 수도 있습니다.

delay()

시간을 지연시켜주는 함수로 () 안의 값은 ms 단위이며 1/1000초입니다. (n) 값을 넣어줌으로써 지연을 시켜줍니다.

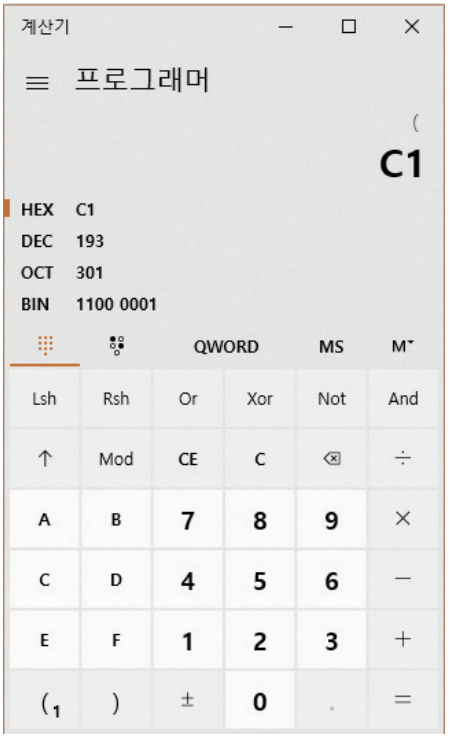


2.3.3. 2진수와 16진수



우리가 일상적에서 사용하는 수는 10진수(Decimal)로 0~9까지 10개의 숫자로 이루어져 있습니다. 하지만 디지털에서 사용하는 숫자는 2진수(Binary)인 0(LOW),1(HIGH) 두 개의 숫자로만 인식할 수 있습니다.

컴퓨터에서 1byte는 8bit입니다. 이것을 반으로 나누면 4bit가 되고, 이는 16진수(Hexadecimal)와 똑같이 0~15까지 16개의 숫자로 표현이 가능합니다. 따라서 1바이트는 0xFF처럼 16진수를 두 자릿수를 사용해서 표현할 수 있고, 2바이트는 0xFFFF처럼 16진수를 네 자릿수 사용해서 표현할 수 있습니다.



윈도우를 사용하신다면 마이크로소프트에서 제공하는 기본 계산기에서 프로그래머 버전으로 사용하시면, 10진수를 2진수와 16진수로 간단하게 변환해서 확인할 수 있습니다.

C/C++ 언어에서는 2진수는 “B”나 “0b”, 16진수는 “0x”나 “0X”를 숫자 앞에 붙임으로써 표현합니다.
예) 0b1101, 0x0F, 0b0001, 0xC5
아래의 표는 10진수에 대응하는 2진수와 16진수 변환표입니다.

10진수	2진수	16진수
0	0b00000000	0x00
1	0b00000001	0x01
2	0b00000010	0x02
3	0b00000011	0x03
4	0b00000100	0x04
5	0b00000101	0x05
6	0b00000110	0x06
7	0b00000111	0x07
8	0b00001000	0x08
9	0b00001001	0x09
10	0b00001010	0x0A
11	0b00001011	0x0B
12	0b00001100	0x0C
13	0b00001101	0x0D
14	0b00001110	0x0E
15	0b00001111	0x0F
16	0b00010000	0x10

2.3.4. 연산자

사칙연산자는 우리가 일반적으로 알고 있는 더하기, 빼기, 곱하기, 나누기입니다. 나눗셈을 응용한 나머지 연산자와 대입이 추가되어 6개의 기본적인 연산자가 있습니다.

연산자기호	기능	예시	의미
+	덧셈	a = b + c	b와 c를 더한 값을 a에 대입
–	뺄셈	a = b – c	b에서 c를 뺀 값을 a에 대입
*	곱셈	a = b * c	b와 c를 곱한 값을 a에 대입
/	나눗셈	a = b / c	b를 c로 나눈 값을 a에 대입
%	나머지	a = b % c	b를 c로 나누고 나머지 값을 a에 대입
=	대입	a = b	b의 값을 a에 대입

비교연산자(관계연산자)는 if 조건문에서 주로 사용되는 값을 비교하는 연산자입니다. 여기서 참(True)은 1을 의미하고 거짓(False)은 0을 의미합니다.

연산자기호	기능	예시	의미
〉	초과	a 〉 b	a가 b보다 크면 참, 아니면 거짓
〈	미만	a〈b	a가 b보다 작으면 참, 아니면 거짓
〉=	이상	a〉=b	a가 b보다 크거나 같으면 참, 아니면 거짓
〈=	이하	a〈=b	a가 b보다 작거나 같으면 참, 아니면 거짓
==	같을 때 참	a==b	a와b가 같으면 참, 아니면 거짓
!=	같지 않을 때 참	a!=b	a와b가 다르면 참, 아니면 거짓

논리연산자(불 연산자)는 참과 거짓을 논리 연산할 때 사용되는 연산자입니다. 참 또는 거짓을 반환합니다.

연산자기호	기능	예시	의미
&&	AND	a&&b	a와b가 모두 참이면 참, 아니면 거짓
||	OR	a||b	a와b 둘 중 하나라도 참이면 참 둘다 거짓이면 거짓
!	NOT	!a	a가 참이면 거짓, 거짓이면 참

진리표(Truth Table)는 논리연산에 사용되는 표입니다. 논리연산이 잘 이해가 안된다면 진리표를 보고 확인해볼 수 있습니다. a와 b의 값을 각각 0과 1로 주었을 때, 값을 표로 나타냈습니다.

a	b	AND(a&b)	OR(a|b)	NOT(~a)	NOT(~b)
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	1
1	1	1	1	0	0

비트연산자는 2진수에서 사용되는 연산자로, 2진수 각각의 자리를 비트(bit)로 설정하고, 비트 단위로 연산을 수행하는 것입니다. 예시로 a= 1010 1111, b=1100 0101이라고 값을 설정하고 계산을 하면, 아래 표와 같습니다.

연산자기호	기능	예시	의미
&	AND	a&b	a&b = 1000 0101
|	OR	a|b	a|b = 1110 1111
~	NOT	~a	a = 0101 0000
〈〈	왼쪽으로 1비트 이동	a〈〈3	a = 0111 1000
〉〉	오른쪽으로 1비트 이동	a〉〉2	a = 0001 1110

"|"는 한글키보드 기준으로 [shift+₩]으로 입력하실 수 있습니다.

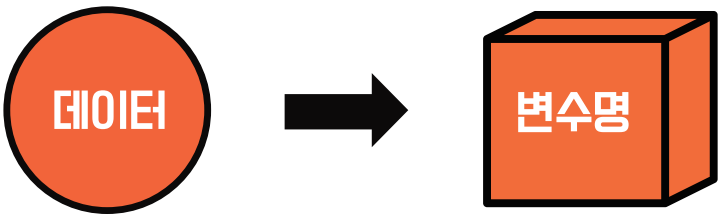
증감연산자는 증가연산자와 감소연산자로 나뉘어집니다. ++는 값을 1 증가시키는 증가연산자이고, --는 값을 1 감소시키는 감소연산자입니다. 아래 표의 i는 변수의 위치를 나타냅니다.

연산자기호	기능	예시
i++	후치 연산자	변수 i의 값을 1증가
i--	후치 연산자	변수 i의 값을 1감소
++i	전치 연산자	변수 i의 값을 1증가
--i	전치 연산자	변수 i의 값을 1감소

++와 --의 위치가 앞, 뒤의 차이에 따라 전치와 후치로 나뉘어 집니다.
전치는 변수값의 증감이 먼저 적용되고 대입됩니다. 대입이 먼저 적용되고 변수값의 증감이 적용됩니다.

```
int x;
int y;
int z;

//증가 연산자
x= 1;
y=++x; //이 경우 x는 2, y도 2이다.(전치)
y=x++; //이 경우 x는 3이지만, y는 그대로 2이다.(후치)

//감소 연산자
x=3;
y=--x; //이 경우 x는 2, y도 2이다.(전치)
y=x--; //이 경우 x는 1, y는 그대로 2이다.(후치)
```

2.3.5. 변수

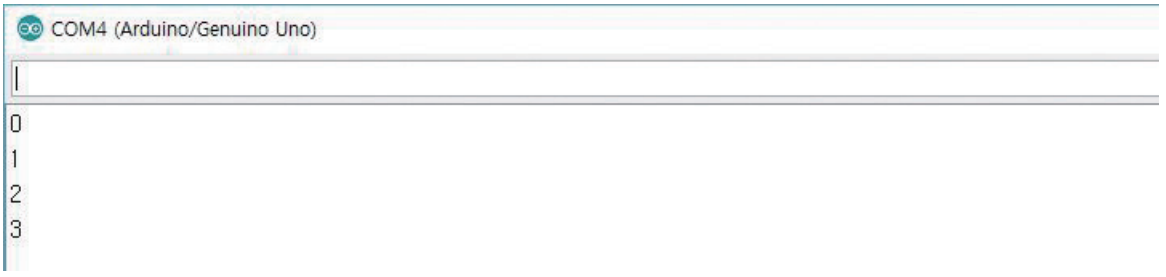


변수란 여러 값으로 변할 수 있는 숫자라는 의미입니다.
어떤 값을 임시적으로 저장했다가도 필요하면 다른 값으로 변할수 있는 미지수라고 생각하시면 됩니다.
변수를 선언해주면 변수의 이름을 통해 데이터 값을 저장할 수 있는 공간이 생성됩니다.
간단한 예제로 확인해 보겠습니다.

예제소스

```
/*
 Variable
*/
int val= 0; // int 형 변수 val을 선언하고 값을 0으로 초기화

void setup() { //초기 설정 함수.
  Serial.begin(9600); // 시리얼 통신 속도 9600으로 설정
}

void loop() { //반복 함수 프로그램의 주 내용이 들어간다.
 Serial.println(val); // 변수 val의 값을 출력
 val++; //val의 값을 1씩 증가시켜라
 delay(500);                    // 0.5초 동안 시간을 지연시켜라.
}
```

해당 예제는 val이라는 변수를 선언하고 0으로 초기화 한 후 Loop()함수가 동작할때마다 val의 값을 1씩 증가하여 출력하는 프로그램 입니다.



프로그램 출력 결과입니다.

```
int devicemart = 1;
```

변수를 선언하고 초기화할때의 형태는 아래와 같이 선언합니다.

자료형 변수이름 = 데이터;

자료형으로 변수의 총 크기와 형식을 지정해준 뒤 변수의 이름을 설정해주고 값을 초기화 해줍니다.
이 과정을 두 줄로 나눠보면,

자료형 변수이름;
변수이름 = 데이터;

이렇게 변수를 먼저 선언해주고 이후에 변수에 값을 넣어 초기화 시켜줄 수도 있습니다. 변수를 초기화 하지 않는다면 변수에 쓰레기 값(알 수 없는 값)이 들어가 있을 수 있기 때문에 사용전에는 초기화해주어야 합니다.

변수 이름은 항상 알파벳으로 시작해야 하며 이후로는 숫자나 언더바(_)를 섞어서 만들어 주시면 됩니다. 또한 대소문자를 구분합니다.

자료형		크기	값의 범위
논리형	boolean	1 비트(bit)	True 또는 False
문자형	char	1 바이트(Byte)	–128~127
정수형	byte	1 바이트(Byte)	0~255
	int	2 바이트(Byte)	–32768~32767
	short	2 바이트(Byte)	–32768~32767
	long	4 바이트(Byte)	–2147483648~2147483647
실수형	float	4 바이트(Byte)	–3.4x10^38~3.4x10^38
	double	4 바이트(Byte)	–3.4x10^38~3.4x10^38

자료형은 대소문자를 구분합니다. 위의 표와 동일하게 쓰셔야 적용됩니다. 데이터의 종류에 따라 변수의 자료형을 알맞게 써야 합니다. 너무 작으면 데이터가 못들어가고, 너무 크면 변수 저장공간인 RAM을 낭비하게 됩니다.

음수를 사용하지 않을 경우에는 자료형 앞에 "unsigned"를 붙여서 사용할 수 있습니다.
부호가 없다는 뜻으로 마이너스 부호가 없는 범위만큼 양수의 범위가 할당됩니다.
예를들어 int를 사용하여 선언한 변수는 –32768~32767 사이의 값을 가질 수 있습니다.

```
unsigned int number;
```

다만 위처럼 unsigned를 사용해 변수를 선언하면 변수 'number'의 범위는 0~65535가 됩니다.

만일 지정된 범위 외의 값이 발생하면 **오버플로우(overflow)**라는 오류가 발생합니다. 컴파일 과정에서 오류로 표현되지 않고 프로그램이 동작하는 과정에서 발생하기 때문에 파악이 어려울 수 있습니다.

예를 들어 char로 변수를 선언하여 "128"이라는 값을 넣어줬습니다. 그럼 실제로 변수를 불러왔을 때 –128이라는 값으로 출력됩니다. 반대로 해당 변수에 –129라는 값을 넣으면 127이라는 값으로 출력됩니다.

이런 현상은 2진수의 음수 표현을 다뤄야 하기 때문에 자세히 설명하기에는 어렵기 때문에 시계를 비유로 설명드리겠습니다.

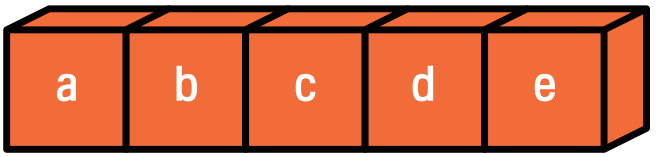
시계에 25시를 표현하려고 하면 1시로 표현할 수밖에 없고, 반대로 –1시를 표현하려고 하면 0시를 기준으로 11시로밖에 표현할 수 없는 한계가 자료형에도 동일하게 적용되기 때문입니다.

장로 윌프레드
크기와 용도에 맞게 선언하는 것이 중요하답니다 흠흠

2.3.6. 배열과 문자열

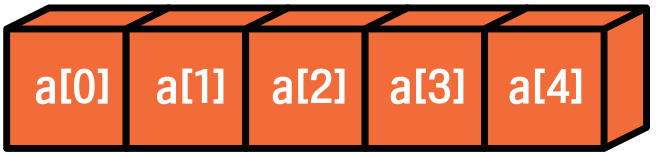
배열은 같은 자료형을 가진 여러개의 변수를 모아놓은 것입니다. 동일한 자료형의 변수를 여러개 선언해야 할 때 유용하게 사용할 수 있습니다.

```
int a, b, c, d, e; // 5개의 int형 변수 선언
```



5개의 int형 변수를 사용할 때, 이름을 각기 다르게하여 변수를 선언해야 하지만,

```
int a[5]; // int형 배열 선언
```



배열을 사용하면 a라는 이름의 배열 안에 변수를 5개 넣을 수 있습니다.

배열의 구조는 자료형, 배열의 이름, 배열의 크기로 이루어져 있습니다. 배열의 요소(인덱스,index)는 0부터 시작합니다. **배열이름[인덱스]**의 형태로 각각의 변수처럼 사용됩니다. 위 이미지에서 선언할때는 크기를 5로 선언하면 실제로는 a[0]~a[4]까지의 배열이 선언됩니다.

```
int arr[3][4]={{1,2,3,4},
               {5,6,7,8},
               {9,10,11,12}};
```

배열은 1차원 이상의 다차원 배열도 가능하며, 1차원 배열과 2차원 배열이 주로 사용합니다. 2차원 배열은 **배열이름[행 인덱스][열 인덱스]**의 형태로 각각의 변수처럼 사용할 수 있습니다. 위의 이미지처럼 2차원 배열을 선언하고 초기화하면 아래와 같이 각각의 행열에 값이 대입됩니다.


```
int a[5];
int b[]={2,4,6,8};
int c[3]={1,2,3};
char message[6]="hello";
char str1[6]="hello";
char str2[]="hello";
char str3[6]={'h','e','l','l','o'};
char str4[6]={'h','e','l','l','o','₩0'};
String stringOne ="Hello String";
```

위 예시는 배열을 선언하는 대표적인 방법들입니다.
배열 a는 초기화하지 않고 배열의 크기만 선언하는 방법입니다.
배열 b는 크기를 선언하지 않고 배열을 선언하는 방법입니다. 초기화가 반드시 필요하며, 이 경우는 컴파일러가 배열의 크기를 결정합니다. 처음에 초기화한 요소의 개수만큼만 크기가 자동으로 할당됩니다.
배열 c는 배열의 크기도 선언하고 초기화를 하는 방법입니다. 가장 기초적이고 명확한 방법입니다.

message 이후로는 문자형(char) 배열을 선언할 때입니다. 문자형 배열은 '문자열'이라고도 합니다. 문자형 배열 마지막에는 문자열이 끝나는것을 의미하는 널(null, ₩0)문자를 넣어야 하기 때문에 실제 사용할 문자열의 길이보다 배열의 크기를 1 더 크게 선택해 주어야 합니다.

str1은 message와 이름만 다른 배열입니다.
str2는 배열의 크기를 선언하지 않은 문자형 배열입니다.
str3을 보면 문자열은 {큰따옴표(" ")} 안에 정의 되고, 이 문자들은 다시 작은따옴표(' ') 안에 정의됨을 알 수 있습니다.

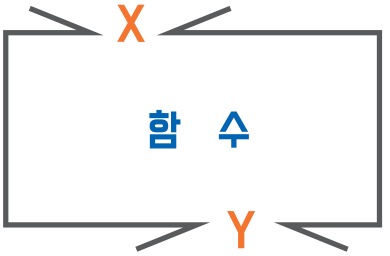
str4에서는 명시적으로 널(null)문자를 더해서 선언했습니다.

stringOne은 문자열 자료형인 String으로 선언한 문자열입니다. 문자형 배열과 역할은 같지만 선언할 때 배열처럼 대괄호와 크기지정을 하지 않고 아닌 변수처럼 선언합니다.
문자형 배열과는 달리 크기가 정해져있지 않기 때문에 나중에 더 긴 문자열을 대입하더라도 문제가 생기지 않습니다.

2.3.7. 함수

함수(function)란 어떤 값 x를 넣으면 내부적으로 어떤 과정을 통해서 결과값 y를 출력해주는 식 또는 장치입니다. 교육과정에 따라 이미 배웠다고 한다면 어려운 개념은 아닙니다.

프로그래밍에서의 함수도 위의 정의와 동일합니다. 처리하고 싶은 데이터를 함수에 넣고, 함수에서 나온 계산을 통해 y값을 반환받는 것을 함수라고 합니다.




```
자료형 함수이름(매개변수)
{
함수몸체
}
```

함수의 구조는 위와 같습니다.

자료형
함수를 호출한 곳에 반환할 값의 자료형을 알려줍니다. 반환할 값이 없으면 void를 사용하고, 반환할 값이 있을경우 return 뒤에 반환할 값과 같은 자료형의 데이터를 사용합니다.

함수명
함수의 이름을 적는 곳입니다. 변수와 작명 요령은 동일합니다.

매개변수(인자)
함수에 입력할 데이터의 유형과 크기를 할당해 놓는 곳입니다.

함수몸체
데이터를 처리하는 부분입니다. 반환 값이 있는경우 return 뒤에 적으면 호출한 곳으로 반환값이 전달됩니다. 함수몸체 내에서 선언한 변수는 함수 밖에서 호출하면 선언되어있지 않은 것으로 취급됩니다.

아두이노에는 pinMode, Serial.begin, digitalWrite, delay 등 많은 기본 함수가 있지만 차차 진행하면서 설명드리고 여기서는 사용자 정의 함수에 대해서만 설명드리겠습니다.

사용자 정의 함수란 이미 만들어진 함수를 사용하듯이 사용자가 직접 새로운 함수를 정의하여 사용하는 함수를 말합니다.

```
int sq_area(int a, int b)
{
    int area=a * b;
    return area;
}
```

사각형의 넓이를 구하는 간단한 예제함수 입니다. sq_area라는 함수에 매개변수 a와 b라는 변수로 두개의 정수값을 받을 것으로 선언합니다. 그 뒤 함수내에서 a와 b를 곱하고 변수 area에 저장한 후 return으로 반환해줍니다. 이때 a와 b의 값을 int형으로 받았으므로, 함수 자료형도 int로 해주어야 return으로 반환되는 변수 area의 값이 올바르게 전달됩니다.

C언어를 공부 해보신 분들이라면 main 함수에서 사용자 정의 함수를 호출하기 전에, 사용자 정의함수를 main 함수보다 위에 선언해야 한다고 알고 계실겁니다. 하지만 아두이노에서는 사용자 정의 함수를 main 함수 뒤에 선언하고 따로 main함수 위에 다시 선언해주지 않아도 동작합니다.

함수호출은 아래처럼 써 주시면 됩니다.

```
int num = sq_area(2, 3);
printnl(num);
```

이렇게 써 주시면 2와 3을 곱한 값이 num이라는 변수에 저장되고 시리얼모니터에 그 값을 출력합니다.

2.3.8. 조건문과 반복문

아두이노와 C언어에서 특정 조건을 주어 그 조건을 만족했을 때 실행되는 if라는 조건문이 있습니다. if문은 조건식을 지정하여 조건식을 만족한다면 중괄호 안에 있는 코드를 실행합니다.

if문은 괄호 안에 조건식이 참(True, 1)일때 실행됩니다. 중괄호가 없을 경우 if문 직후의 세미콜론(;)으로 끝나는 한줄의 코드만 실행합니다.

```
if(조건식)
{
    코드
}
```

```
if(a==1) // a가 1이면
{
  a=0; // a에 0을 대입
}
```

else문은 if 조건식을 만족하지 않을 경우 실행되는 코드입니다.

```
if(조건식)
{
  코드
}
else
{
  코드
}
```

```
if(a>1) // a가 1보다 크면
{
  a=0; // a에 0을 대입
}
else // a가 1보다 작으면
{
    a=-1; // a에 -1을 대입
}
```

else if문은 조건이 여러개 있을 때 사용합니다.
이때 마지막에 else를 사용하면 if 조건과 else if 조건에 맞지 않았을때 else 내의 코드가 실행됩니다.

```
if(조건식)
{
  코드
}
else if(if 조건식을 제외한 조건식)
{
  코드
}
else
{
  코드
}
```

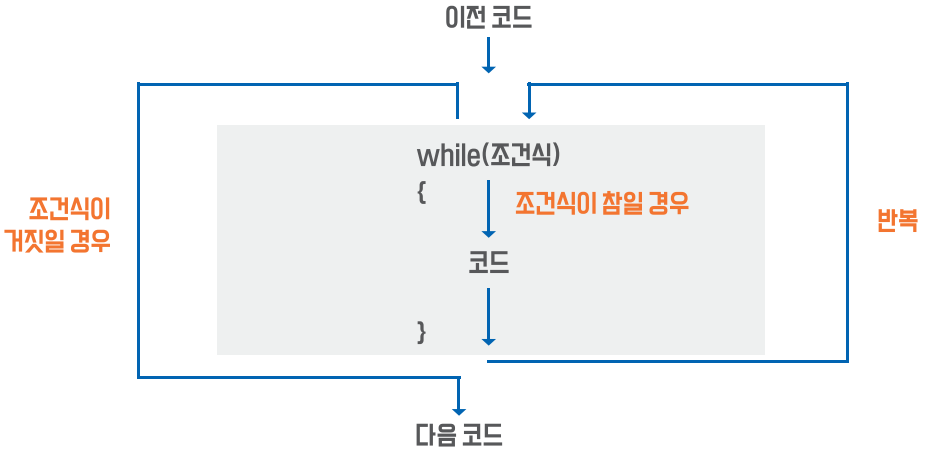
```
if(a>0) // a가 0보다 크다면
{
  a=b; // a에 b를 대입
}
else if(a==0) // a가 0이라면
{
  a=c; // a에 c를 대입
}
else // a가 0보다 작고, 0도 아니라면
{
  a=d; // a에 d를 대입
}
```

if문은 조건이 참일때 한번만 실행되는 조건문입니다. 조건이 참일때 반복하는 반복하는 반복문도 있습니다.
while 반복문은 if 조건문과 비슷하게 사용할 수 있습니다.

```
while(조건식)
{
  코드
}
```

```
while(a<10) // a가 10보다 작다면
{
  a++; // a는 1 증가
}
```

while문의 경우 조건식이 만족하는 동안에는 계속해서 수행하게 됩니다. 조건식이 변동없이 항상 참인경우 loop처럼 무한히 반복되게 되므로 주의해서 코드를 작성해주셔야 합니다.



앞서 배운 while문은 if문과 사용법이 유사하여 간결하게 사용하기 좋았습니다. 그러나 특정 횟수만 반복하기에는 다소 번거로울 수 있습니다.
이런 문제는 for 반복문을 통해 해소할 수 있습니다. for문은 특정 횟수 반복에 특화되어 있는 반복문입니

다. for 반복문은 초기값–조건식–증감식이 있는 루프 선언문과 코드가 있는 루프 본체 형식으로 구성되어 있습니다.

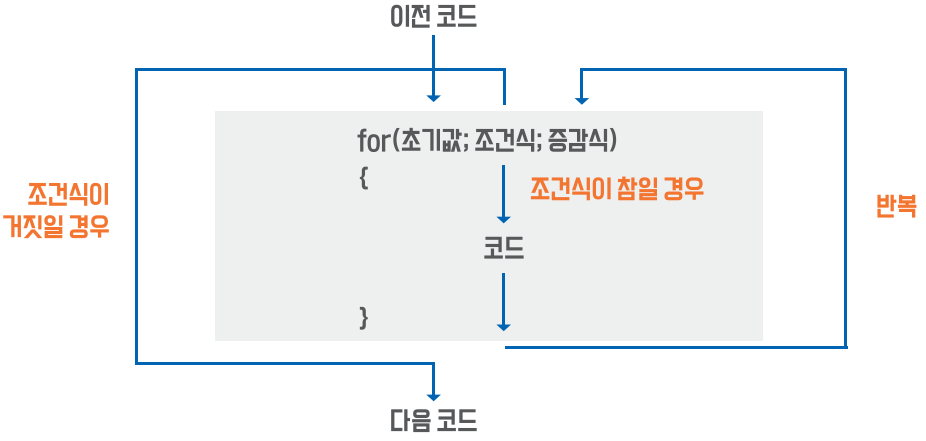
초기값 : 중괄호 안에서 사용할 변수를 초기화합니다.
조건식 : 중괄호 안쪽 코드가 실행되기 전에 참, 거짓에 대한 조건 만족 여부를 판단합니다.
증감식 : 중괄호 안쪽 코드가 실행된 후에, 루프 변수를 증감하는 용도로 사용됩니다.

for 문은 초기값이 조건을 만족할 때까지 중괄호 안에 있는 코드를 계속 반복합니다. 한번 반복하면, 증감식에 있는 식에 따라 초기값에서 선언했던 변수의 값이 변화합니다.

```
for(초기값; 조건식; 증감식)
{
  코드
}
```

```
for(i=0; i<10; i++) // i는 0부터 10보다 작을 때까지
{
  a=a/2 // a를 2로 나눈 값을 a에 대입
}
```

for 문은 초기값이 조건을 만족할 때까지 중괄호 안에 있는 코드를 계속 반복합니다. 한번 반복하면, 증감식에 있는 식에 따라 초기값에서 선언했던 변수의 값이 변화합니다.

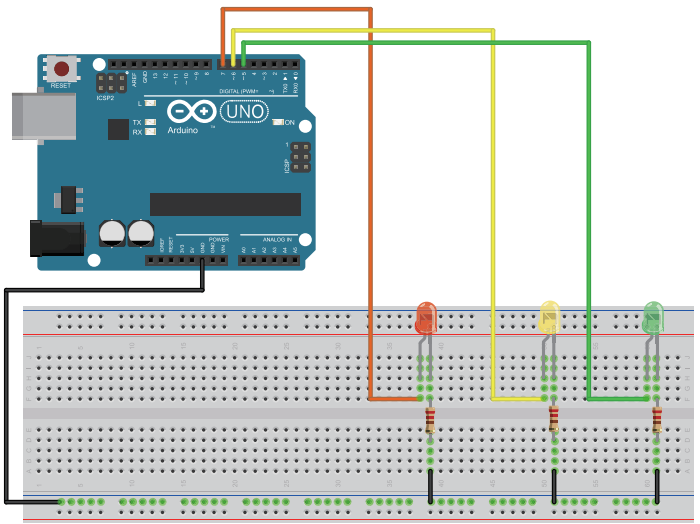


2.3.9. 디지털 입·출력

아두이노 우노에는 디지털 핀이 0~13번까지 있습니다. 0번, 1번에 해당하는 핀은 시리얼 통신의 RX(수신), TX(송신)로 기본 설정되어있고 이를 '하드웨어 시리얼'로 부른다고 앞에서 잠깐 설명드렸었습니다.

아두이노는 우노는 여기에 있는 디지털 핀으로 HIGH (+5V) 신호와 LOW (0V) 신호를 입력받거나 출력할 수 있습니다. 아쉽게도 한 핀에 입력과 출력을 동시에 사용할 수는 없습니다.

아두이노 우노 중 Atmega328칩을 사용한 보드는 각 핀당 최대 출력 전류가 40mA이고, Atmega328p 칩이 들어간 제품은 각 핀당 최대 전류가 출력 20mA입니다.



준비물	
아두이노 우노	1대
브레드보드	1개
저항 220Ω	3개
LED	3개
점퍼와이어	10가닥 내외

예제소스

```
/*
digital pin test
*/

int led1= 5; // led를 digital 핀 5번에 연결
int led2= 6; // led를 digital 핀 6번에 연결
int led3= 7; // led를 digital 핀 7번에 연결

void setup()
{
  pinMode(led2,OUTPUT); // digital 핀 6번을 출력으로 설정
  pinMode(led3,OUTPUT); // digital 핀 7번을 출력으로 설정
}

void loop()
{
```

```
  digitalWrite(led1,HIGH); // led1 on
  digitalWrite(led2,HIGH); // led2 on
  digitalWrite(led3,HIGH); // led3 on
  delay(1000); // 시간 지연 1초

  digitalWrite(led1,LOW); // led1 off
  digitalWrite(led2,LOW); // led2 off
  digitalWrite(led3,LOW); // led3 off
  delay(1000);
}
```

pinMode()

pinMode(pin, mode)

위와 같은 형식으로 선언하여 디지털 핀의 입력 또는 출력 모드를 설정할 수 있습니다. pin에는 사용할 핀에 대응되는 숫자(우노에서는 0~13)를 넣습니다. mode에는 OUTPUT(출력), INPUT(인풋), 또는 INPUT_PULLUP(인풋 풀업저항사용)을 사용할 수 있습니다.

digitalRead()

digitalRead(pin)

위와 같은 형식으로 사용하여 각 핀의 상태를 읽어올 수 있습니다. 핀모드를 통해 입력으로 설정한 핀의 값을 High인지 LOW인지 읽어와서 1또는 0으로 반환하는 함수입니다. 반환되는 값은 변수에 저장하거나 후에 나올 if문이나 반복문에 사용할 수 있습니다.

digitalWrite()

digitalWrite(pin, value)

위와 같은 형식으로 사용하여 각 핀의 신호를 출력하는 함수 입니다. pin에는 사용할 핀에 대응되는 숫자(우노에서는 0~13)를 넣습니다. mvalue값은 HIGH와 LOW가 있으며 각각 1과 0을 의미합니다.

LED가 1초동안 켜져있다가 1초동안 꺼지는 동작을 한다면 정상적인 동작입니다.

2.3.10. 아날로그 입력 & PWM 출력

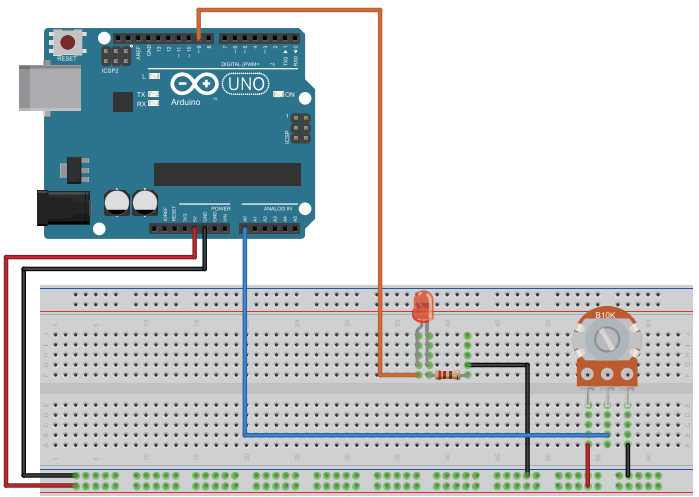
아날로그 핀은 디지털 핀과 다르게 아날로그 신호를 입력만 받을 수 있고 아날로그 전압 출력은 할 수 없습니다. 때때로 필요하다면 디지털 신호 입출력용 핀으로 사용할 수는 있지만 후에 실습해보도록 하겠습니다.

아두이노 우노는 10bit 아날로그-디지털 변환기(Analog-Digital Converter, ADC)가 내장되어 있습니다. 10bit ADC가 내장되어 있다는 말은 아두이노 우노는 아날로그 입력인 0~5V의 전압을 0~1023사이의 값으로 대응해서 값을 읽어올 수 있다는 뜻입니다. 예를 들어 0V의 경우 0의 값으로, 2.5V의 경우 512의 값으로 5V의 경우 1023의 값으로 받아들입니다.

전에 배운 PWM 출력을 통해 디지털 핀 출력을 아날로그 출력과 유사하게 제어가 가능합니다. PWM 출력의 경우 아날로그 입력과는 다르게 0~255의 값으로 출력합니다. 만약 아날로그 입력을 받아서 PWM 핀으로 출력하는 경우 아날로그 입력값을 4로 나누어서 PWM 출력으로 대응 시켜주어야 합니다.

0을 넣어주시면 듀티비 0%의 0V와 같은 동작을, 127을 넣어주시면 듀티비 50%의 2.5V와 같은 동작을, 255를 넣어주시면 듀티비 100%의 5V와 같이 출력됩니다. 물론 전에 언급 했듯 실제로는 LOW일 때와 HIGH일 때의 시간 비율이 다르고 전압이 5V와 0V 사이의 값으로 바뀌는 것은 아닙니다.

간단한 예제로 확인해보겠습니다.



준비물	
아두이노 우노	1대
브레드보드	1개
저항 220Ω	1개
가변저항	1개
점퍼와이어	10가닥 내외

예제소스

```
/*
analog pin test
led=Digital pin D9
potentiometer A0
*/

int led =9; // led를 핀 9번에 연결
void setup()
{
```

```
  pinMode(led,OUTPUT); // digital 핀 9번을 출력으로 설정
}

void loop()
{
  int input = analogRead(A0);// int 변수 input을 선언하고 값을 A0에서 읽은 값으로 초기화
  analogWrite(led, input/4); // 가변저항에서 읽어들인 값을 4로 나눈 후 9번핀으로 출력
}
```

analogRead()

analogRead(pin)

위와 같은 형식으로 선언하여 디지털 핀의 입력 또는 출력 모드를 설정할 수 있습니다.

pin에는 사용할 핀에 대응되는 핀의 이름(A0~A5)를 넣습니다.

0V~5V 범위 내의 아날로그 신호를 0~1023의 값으로 읽어오는 함수입니다.

analogWrite()

analogWrite(pin,value)

위와 같은 형식으로 사용하여 PWM 출력을 할 수 있습니다.

함수 자체내에 핀모드가 출력으로 설정되어 있어서, pinMode를 따로 설정할 필요가 없습니다.

pin에는 사용할 핀에 대응되는 핀의 이름(A0~A5)를 넣습니다.

value에는 0~255의 값을 넣어 대응되는 듀티비로 출력합니다.

디지털핀 중 ~표시가 되어있는 3번, 5번, 6번, 9번, 10번, 11번 핀으로 사용 가능합니다.

가변 저항을 돌리면 LED 밝기가 변화합니다.

Quest 4. 실력 테스트 : 먹보 슬라임

장로 윌프레드
용사님! 이웃 마을에 먹보 슬라임이 나타났다고 합니다!!
용사님께서 나서주셔야 할 때입니다!!!

용사
앗! 안 그래도 몸이 근질근질했는데!
용사의 실력을 맛봐라!

Trouble 1 – OX 문제
setup() 함수와 loop() 함수는 전원이 들어오면 무한히 반복된다.

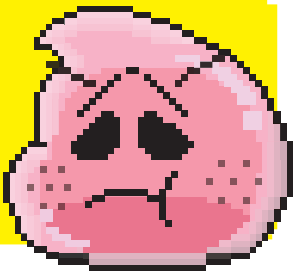
Trouble 2 – 객관식 문제
다음 중 디지털 입출력 포트의 D0와 D1핀의 기능을 명칭하는 것으로 올바른 표현은?
① 하드웨어 시리얼 ② 하드웨어 패러렐 ③ 소프트웨어 시리얼 ④ 소프트웨어 패러렐

Trouble 3 – 객관식 문제
10진수 127은 16진수로 7F입니다. C언어에서 입력하는 것으로 올바른 표현은?
① 07F ② 0x7F ③ B7F ④ 7F

Trouble 4 – 객관식 문제
다음 중 나머지를 구하는 연산자는?
① ‖ ② 〉 ③ % ④ ~

Trouble 5 – 주관식 문제
소숫점을 표현할 수 있는 변수 pi를 3.14로 초기화하여 선언하시오.

Trouble 6 – 주관식 문제
char str[ ] = “hello world!” 문자열의 크기는?



Trouble 7 – 주관식 문제

다음 함수의 매개변수 a에 12를 넣고 실행시켰을 때 변수 num의 값은?

```
int add_ten(int a)
{
  int num = a + 10;
}
```

Trouble 8 – 주관식 문제

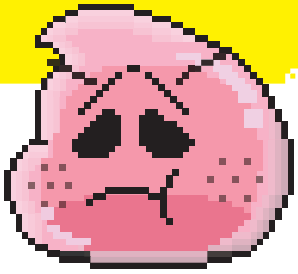
아두이노에서 사용할 수 있는 반복문에는 두가지가 있다. 두개의 반복문을 쓰시오.

Trouble 9 – 주관식 문제

내부풀업저항을 사용하는 디지털 핀 D2를 입력 모드로 선언하시오.

Trouble 10 – 주관식 문제

D3 핀에 듀티비 40%의 PWM을 출력하시오.



용사

맛이 어떠냐!

(하나도 안힘든 척)

Quest 4. 정답 체크

맞춘 문제 개수만큼 슬라임에 X 표해주세요.

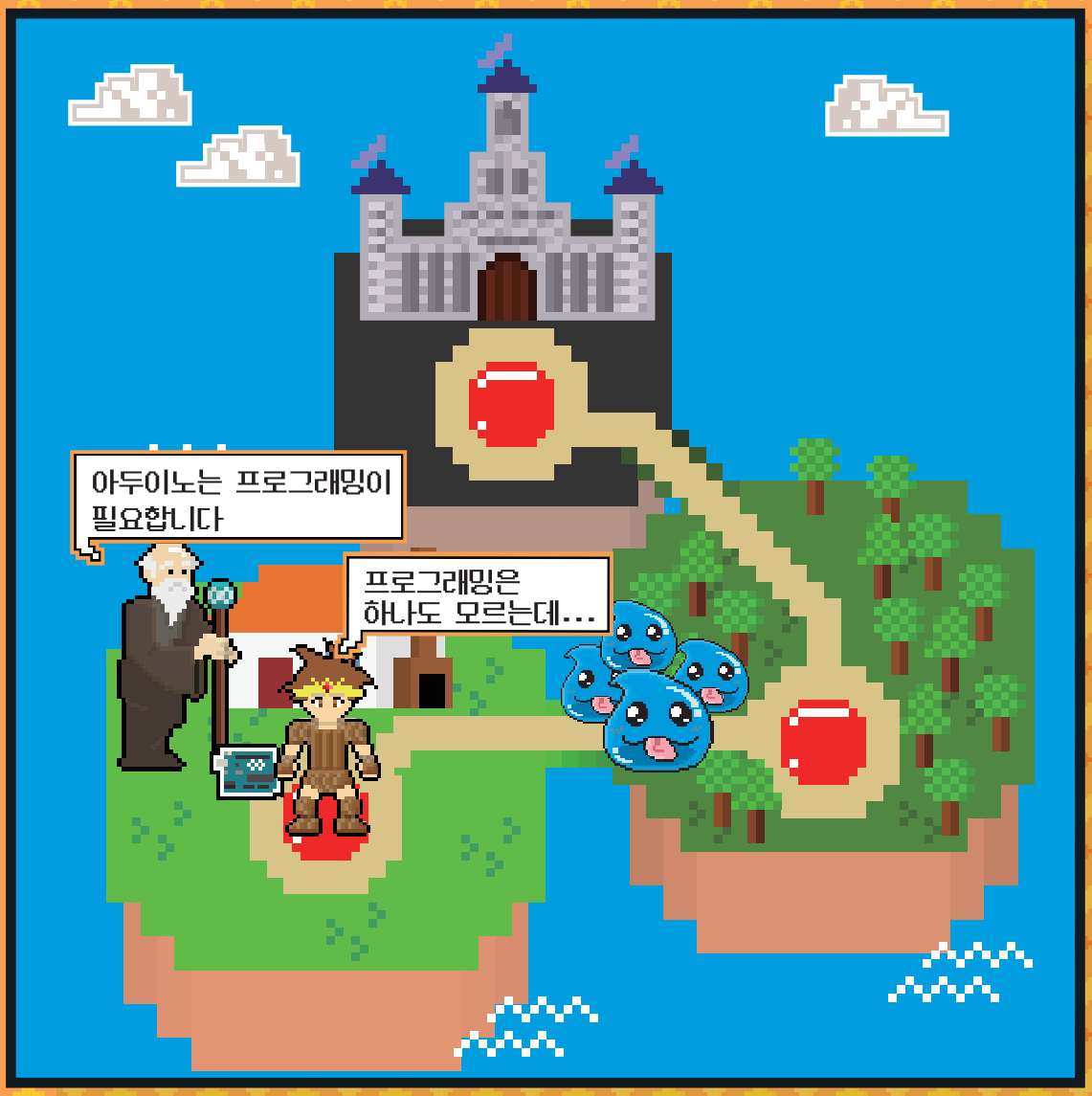
문제	정답
Trouble 1 – OX 문제	X
Trouble 2 – 객관식 문제	①
Trouble 3 – 객관식 문제	②
Trouble 4 – 객관식 문제	②
Trouble 5 – 주관식 문제	float pi = 3.14; 또는 double pi = 3.14;
Trouble 6 – 주관식 문제	13
Trouble 7 – 주관식 문제	22
Trouble 8 – 주관식 문제	while문, for문
Trouble 9 – 주관식 문제	pinMode(2, INPUT_PULLUP)
Trouble 10 – 주관식 문제	analogWrite(3,102)

1문제 이상 틀렸다면
이번 스테이지 처음부터 다시!

모두 맞추셨다면 다음페이지로!

『용사의 써클릿』을 얻었다.

스테이지 2.4
아두이노 프로그래밍



EXP♥

2. 아두이노 시작하기

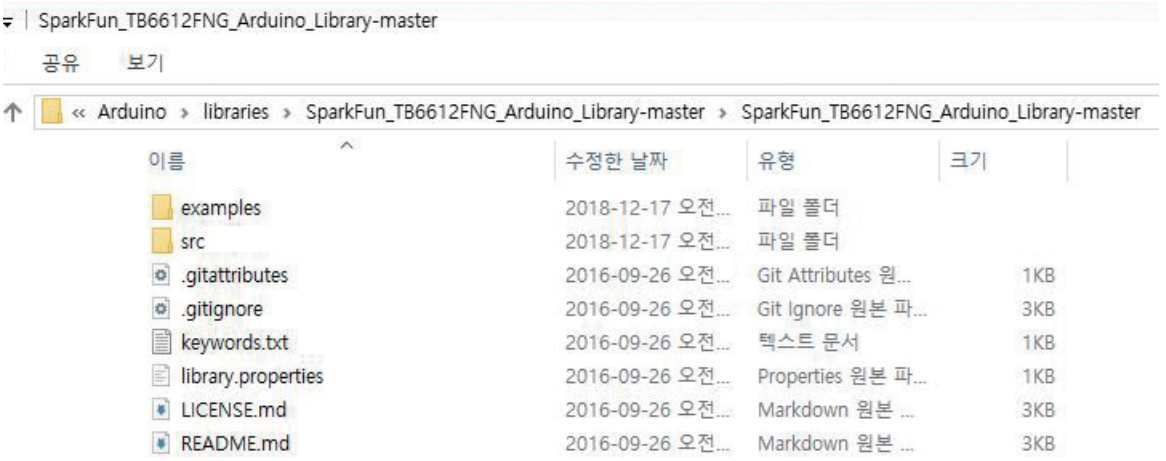
2.4. 라이브러리

2.4.1. 라이브러리 소개

아두이노를 공부하다 보면 라이브러리를 사용해야하는 상황이 자주 발생합니다. 라이브러리(Library)는 아두이노에서 센서, 모터드라이버, wifi, 이더넷 등의 특정한 모듈을 사용할 때 사용되는 함수나 상수들을 활용하기 쉽게 모아놓은 소스코드라고 할 수 있습니다.

라이브러리는 아두이노에서 제공하는 **표준 라이브러리**, 다른 사용자들이 만들어 놓은 **사용자 라이브러리**가 있습니다.

라이브러리는 주로 센서, 모터드라이버 등의 규격화 된 모듈이나 회로의 사용을 간편하게 해줍니다. 예를 들어 서보모터를 사용할 때, 직접 소스를 모두 만들기보다는 이미 제작되어 있는 라이브러리를 불러와서 원하는대로 서보모터를 제어하는 것이 시간과 노력이 적게 들면서 동일하거나 나은 결과물을 얻으실 수 있습니다.

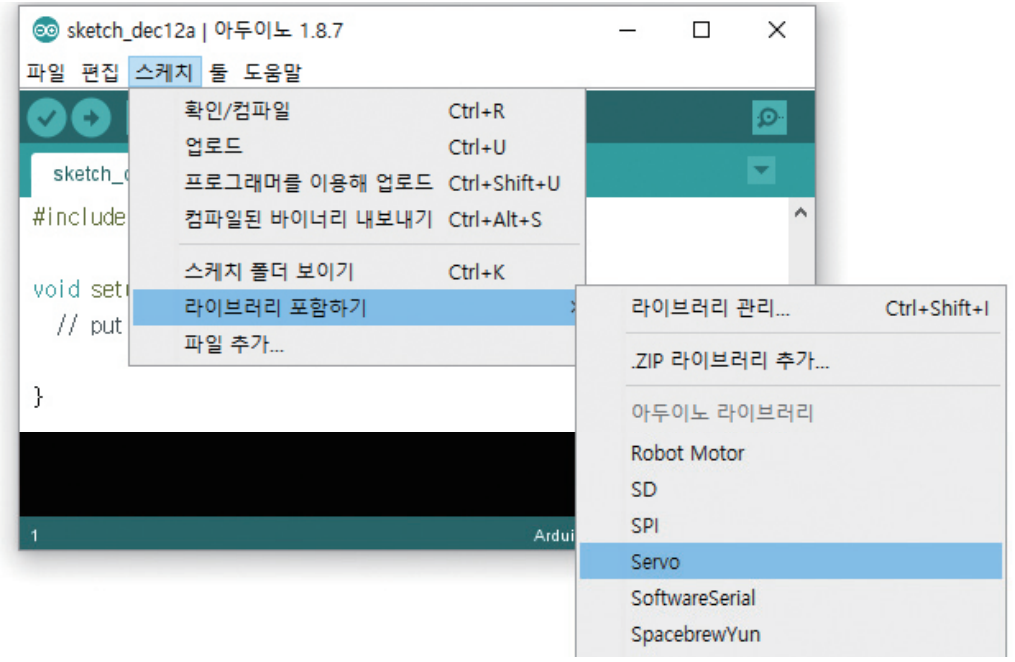


라이브러리 폴더의 구조입니다. src 폴더에 라이브러리 소스파일인 .h 파일과 .cpp 파일이 위치해 있고, example 폴더에는 라이브러리를 사용한 소스 예제가 위치해 있습니다.

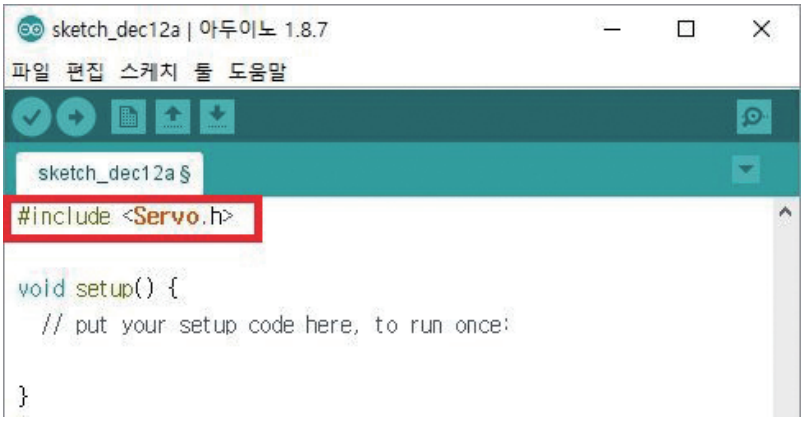
이 코드와 소스 예제를 보고 라이브러리를 사용하게 됩니다.

2.4.2. 표준 라이브러리 가져오기

아두이노에서 기본적으로 제공하는 표준 라이브러리를 사용하는 방법에 대해 알려드리겠습니다.



[메뉴바–스케치–라이브러리 포함하기]에서 필요한 라이브러리를 클릭합니다.

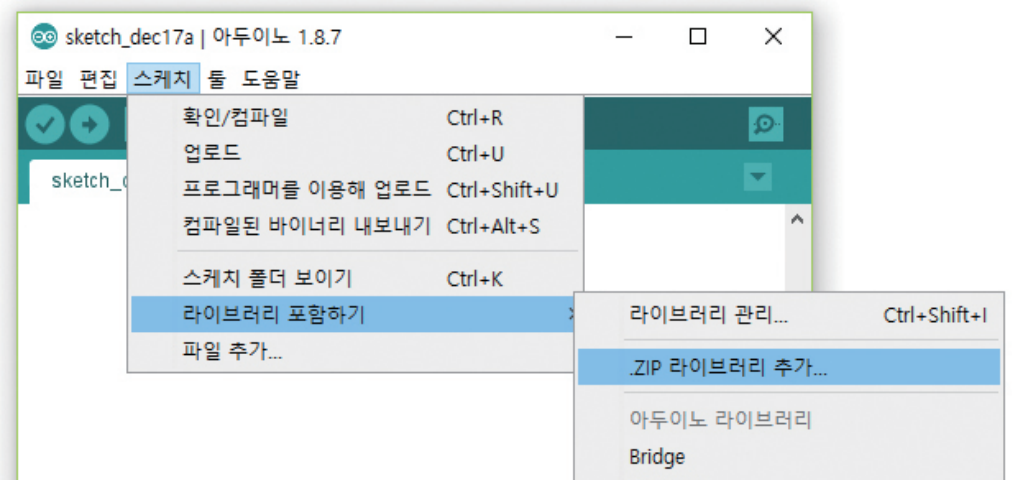


에디터창 맨위에 라이브러리 헤더파일이 추가가 됩니다. 또한 **#include 〈라이브러리 이름.h〉**을 직접 입력하셔서 추가하셔도 됩니다.

2.4.3. 사용자 라이브러리 가져오기

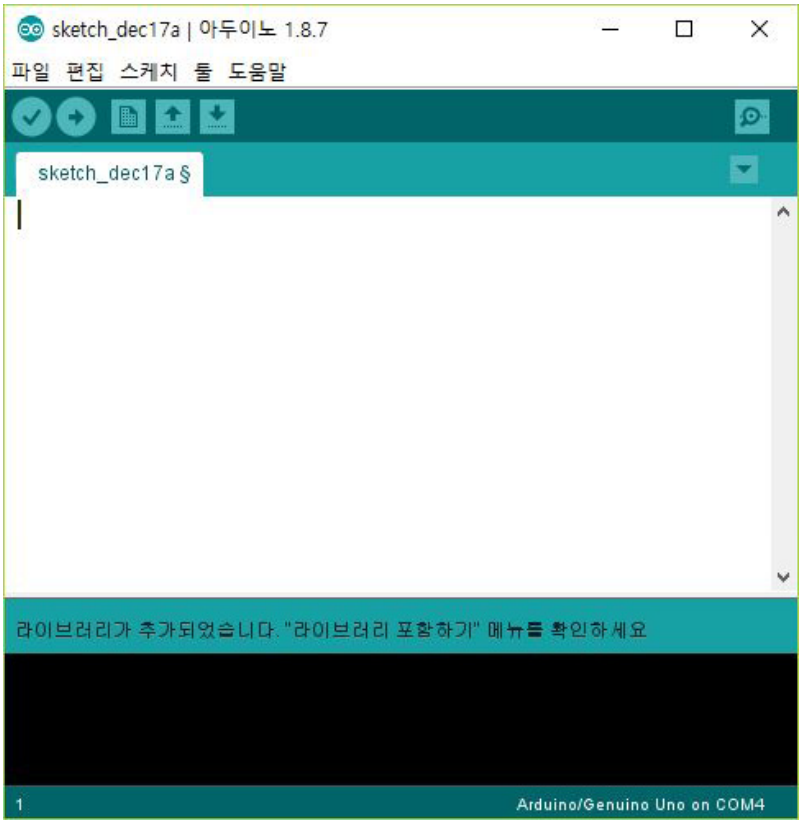
사용자 라이브러리의 경우 센서, 모터드라이버 등의 모듈에 대한 라이브러리를 제조사나 깃허브(https://github.com/), 혹은 아두이노 플레이그라운드(http://playground.arduino.cc/Main/GeneralCodeLibrary)에 검색하거나 필요한 라이브러리 이름을 구글링을 하면 찾으실 수 있습니다.

이렇게 다운받은 파일은 보통 zip 확장자의 압축파일인데, 이 라이브러리 파일을 아두이노 IDE에 추가하는 방법에 대해 알려드리겠습니다.

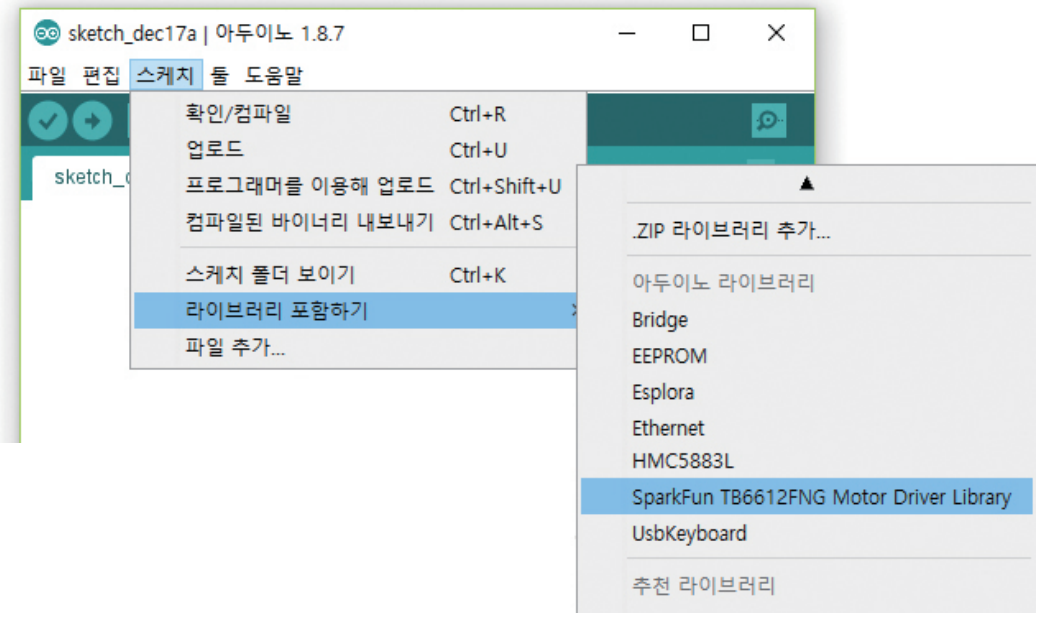


추가하려는 사용자 라이브러리를 다운받으신 후, [메뉴바–스케치–라이브러리포함하기–.ZIP 라이브러리 추가…]를 누릅니다.

탐색 창이 뜨면 다운받은 라이브러리 압축 파일을 선택해주세요.

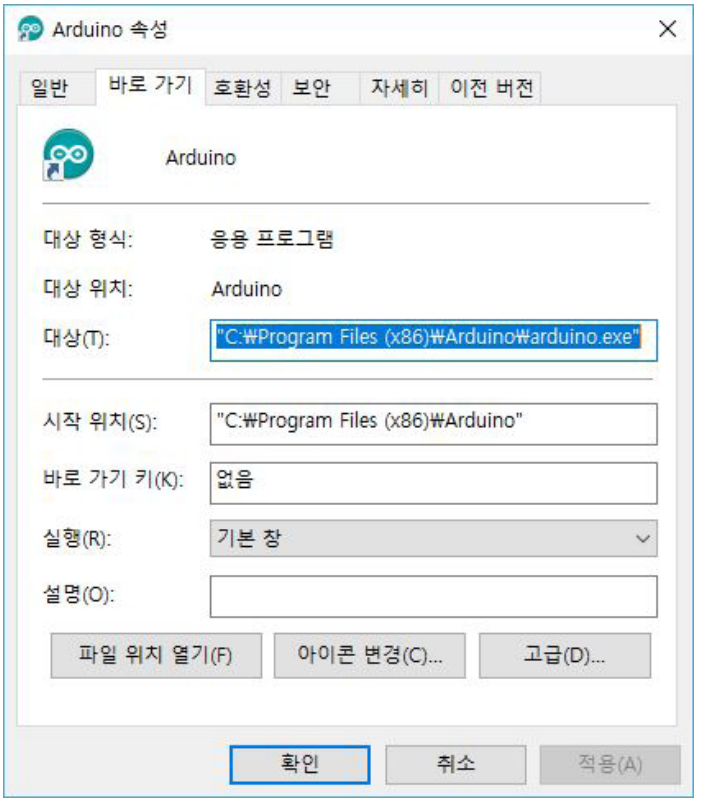


추가하시면 라이브러리가 추가되었다는 메시지가 뜹니다.

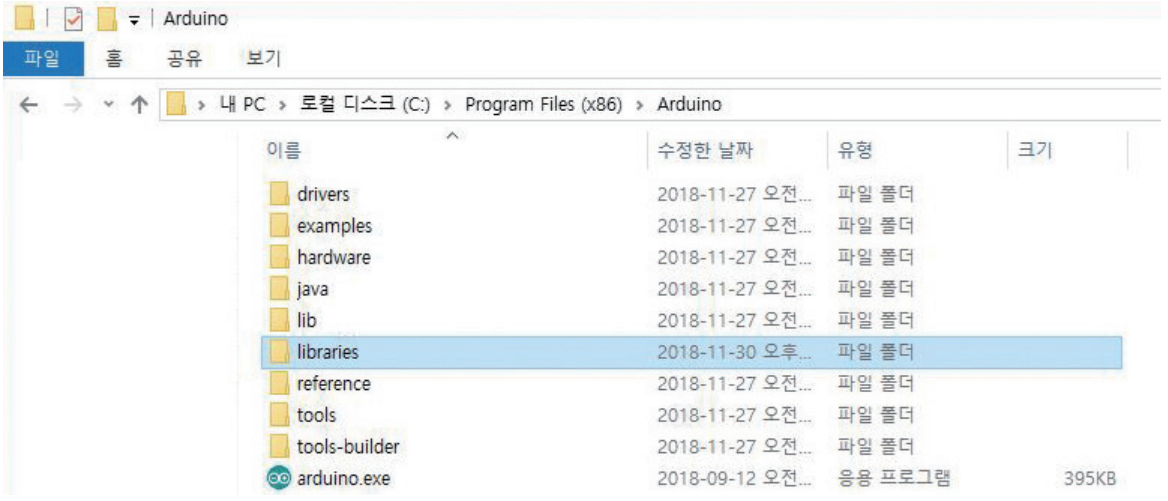


메뉴바 – 스케치 – 라이브러리 포함하기에 가시면 직접 추가한 라이브러리 목록을 볼 수 있습니다. 만약 라이브러리를 직접 수정하여야 하거나, 이 방법으로 라이브러리가 추가가 안될 때는 라이브러리를 수동으로 추가하는 방법이 있습니다.

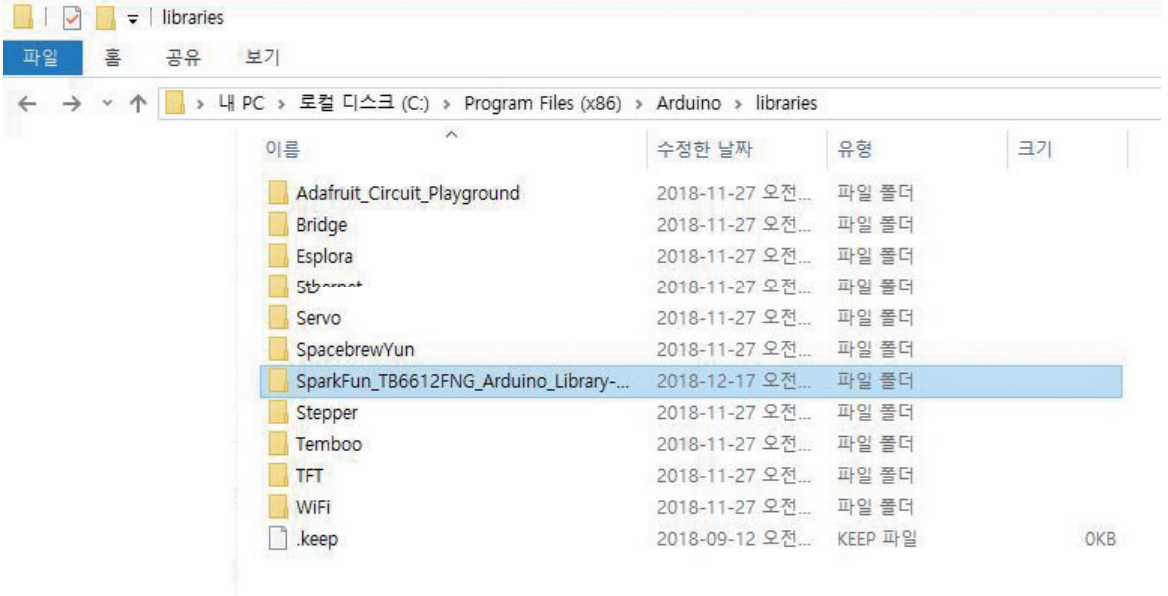
아래는 라이브러리를 수동으로 추가하는 방법입니다. 마이크로소프트 스토어로 설치하셨다면 [내 PC]에 있는 [문서]로 가시면 [Arduino]라는 폴더가 있습니다. 이외의 운영체제거나 별도로 설치하셨다면 아두이노 설치 폴더로 이동합니다.



아두이노 설치 폴더를 모르실 경우 아두이노 아이콘을 우클릭 하신 뒤 속성에 들어가시면 아두이노 설치 폴더가 있습니다.



[Arduino]폴더에서 libraries 폴더로 들어가주세요.



이 폴더에 압축을 푼 라이브러리파일을 직접 넣어주시면 됩니다.

Quest 5. 실력테스트 : 슬라임 군단의 역습-

장로 윌프레드

용사님! 지금 슬라임들이 군단을 이끌고 마을로 오고있다고 합니다.

용사

저한테 맡기세요! 얼마든지 상대해주마!

Trouble 1 – OX 문제

라이브러리는 아두이노에서 제공하는 표준 라이브러리와
다른 사용자들이 만들어 놓은 사용자 라이브러리의 두가지 종류가 있다.

Trouble 2 – 주관식 문제

기본 제공 라이브러리인 “LiquidCrystal.h”를 추가하는 코드를 작성하시오.

Trouble 3 – 객관식 문제

다음 중 아두이노 IDE에 라이브러리를 추가하는 올바른 방법은?

① [메뉴바–툴–라이브러리 관리...]
② [메뉴바–스케치–스케치 폴더 보이기]
③ [메뉴바–툴–스케치 보관하기]
④ [메뉴바–스케치–라이브러리포함하기–.ZIP 라이브러리 추가…]

Quest 5. 정답 체크

맞춘 문제 개수만큼 슬라임에 X표해주세요.

문제	정답
Trouble 1 – 객관식 문제	O
Trouble 2 – 주관식 문제	#include 〈LiquidCrystal.h〉
Trouble 3 – 주관식 문제	④

1문제 이상 틀렸다면
이번 스테이지 처음부터 다시!

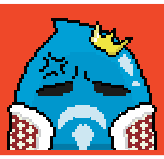
모두 맞추셨다면 다음페이지로!

용사는 『나무칼』을 얻었다.

용사
드디어 무기를 주시는군요!

장로 윌프레드
아두이노를 옳게 사용하시는 모습을 보고 드리는 겁니다.
오직 선한 일에만 사용해주세요 흠흠

Quest 6. Boss : 늪의 군주 슬라임킹



슬라임 킹
보글 부글부글 뽀글뽀글 뽀그르르!!!(내 부하들을 건드린 게 너냐!!!)

장로 윌프레드
용사님 조심하세요! 이 구역 늪의 군주인 슬라임 킹입니다.
다른 슬라임들보다 더 힘이 강한 슬라임입니다.

용사
크윽… 갑자기 강적이…

장로 윌프레드
당황하지 마시고 위에서 배운 내용을 차근차근 되짚어보세요!

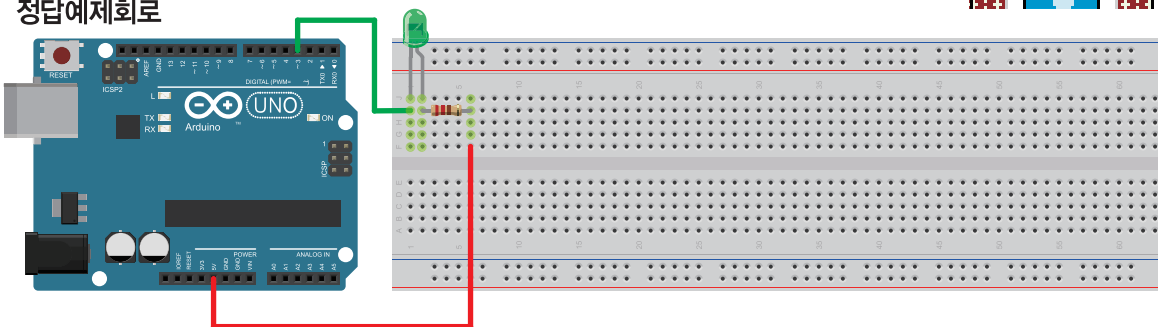
BOSS 문제
LED를 80%의 밝기로 켜고 1초에 한번씩 "Time is gold"라는 문자열을 시리얼모니터로 출력하시오.

준비물	
아두이노 우노	1대
브레드보드	1개
저항 220Ω	1개
LED	1개
점퍼와이어	10가닥 내외

Quest 6. 정답 체크

맞춘 문제 갯수 만큼 슬라임에 X표 해주세요.

정답예제회로



정답예제소스

```
void setup() // 초기 설정 함수.
{
  Serial.begin(9600); // 시리얼 통신 속도 9600으로 설정
  pinMode(3,OUTPUT); // 생략 가능, PWM 출력이 가능한 핀이라면 3번핀이 아니라도 정답입니다.
  analogWrite(3, 51); // 듀티비 20%는 255에 0.2를 곱해서 구할 수 있습니다
}

void loop() //안에 작성한 코드가 계속 반복됩니다.
{
  Serial.println("Time is gold");  // Time is gold를 출력
  delay(1000);                     // 1초 동안 시간을 지연시켜라.
}
```

※ 디지털 출력 핀을 LED의 캐소드가 아닌 에노드에 연결했다면 analogWrite()함수를 analogWrite(3, 204)로 쓰시면 회로가 다르더라도 정답이 맞습니다.

틀렸다면 스테이지 2부터 다시

모두 맞추셨다면 다음페이지로!

용사는 『아무것도』 못 얻었다.

어딘가에서 들리는 목소리

???
구해주셔서 감사합니다.

용사
누... 누구시죠? (이 기분은··· 마치 여기 처음 온 날이랑 똑같아)

엘라임 - 물의 정령왕
제 이름은 엘라임. 물의 정령왕입니다. 원래는 수정 궁전에서 머물면서
이 지역을 안전하게 다스리고 있었지만 마왕의 계략으로 인해 슬라임킹으로
둔갑당해 이용당하고 있었습니다.
이성을 잃기 직전에 용사님을 소환했는데 다행히 이제라도 뵐 수 있게 되었군요.

용사
그럼 저를 부른 게 당신이군요? 이제 해결된 것 같으니 저를 되돌려주세요.
앞으로 엄마 말도 잘 들을게요.

엘라임 - 물의 정령왕
아쉽게도 수정 궁전을 정화하지 않으면 제가 다른 차원에 영향력을
행사할 수는 없습니다. 지금은 비록 마왕 군이 점령한 마왕성이지만,
용사님께서 수정 궁전을 정화해주신다면 집으로 되돌아갈 수 있게
해 드리겠습니다.

용사
왜 항상 거절이 없는 거죠? ㅠㅠ
하지만 집에 가는 방법이 이것 뿐이라면 어쩔 수 없겠죠..?

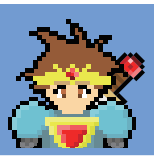
엘라임
감사합니다. 용사님께 조금이나마 보탬이 되어 드리겠습니다.

[용사의 써클릿]에서 알 수 없는 빛이 나와 몸을 휘감았다.

『나무 신발』이(가) 『판금 장화』(으)로 강화되었습니다!
『나무 갑옷』이(가) 『판금 갑옷』(으)로 강화되었습니다!
『나무 각반』이(가) 『사슬 각반』(으)로 강화되었습니다!
『나무 건틀릿』이(가) 『판금 건틀릿』(으)로 강화되었습니다!
『나무칼』이(가) 『강철검』(으)로 강화되었습니다!

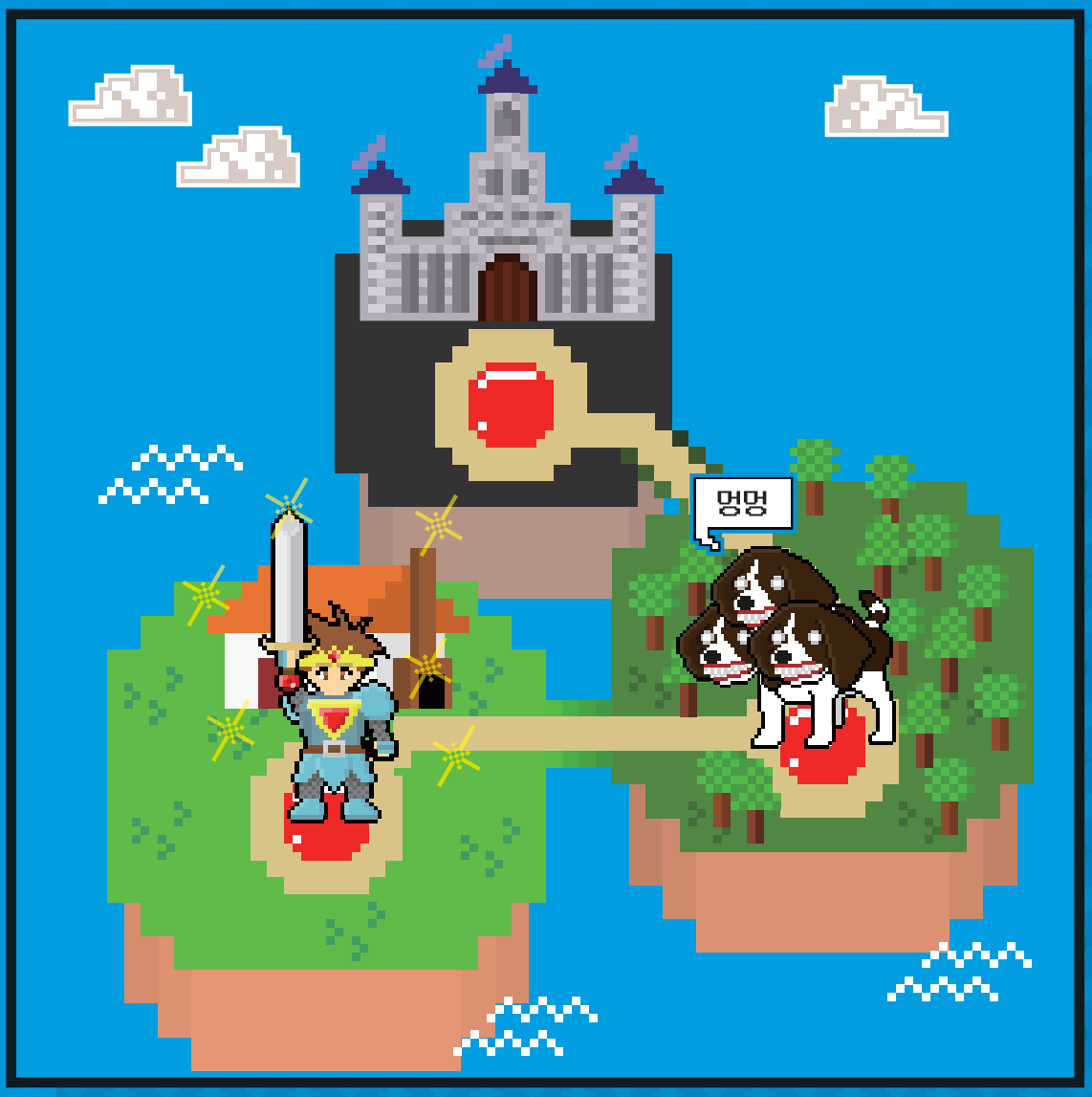
용사
음… 우물우물… 갑자기 이렇게나 많이 주면…
든든하긴 한데 너무 무거워…

장로 윌프레드
오! 용사님 정신이 드셨군요!
가문 대대로 내려오던 [초보 용사의 장비] 세트에 이런 비밀이 있었을 줄이야…
이제 마왕을 무찌르기에 손색없어 보입니다. 흠흠



용사
좋아, 당장 떠날래요!
마왕성이 어느 방향이죠?

스테이지 3.1
시리얼 통신



EXP♥

3. 아두이노 실전 연습

3.1. 통신

3.1.1. 시리얼 통신

전에 프로그래밍 파트에서 배웠던, 시리얼 통신에 대한 예제를 한번 더 실습해 보겠습니다.

아두이노 스케치의 시리얼 모니터의 구조입니다.

데이터 입력

전송할 데이터를 입력할 때 사용합니다.

데이터 출력

받은 데이터를 출력해주는 부분 입니다.

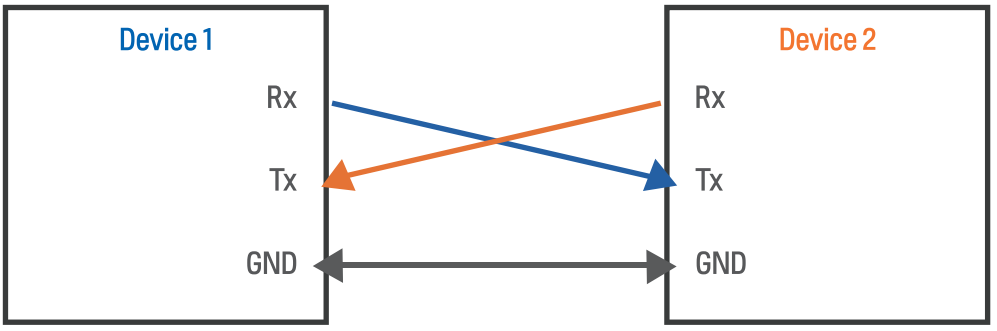
전송옵션 설정

새 줄(Newline), 캐리지 리턴(Carriage return), Both NL & CR로 이루어져 있으며,
데이터를 전송할 때 어떤 방식으로 보낼지에 대해 설정합니다.

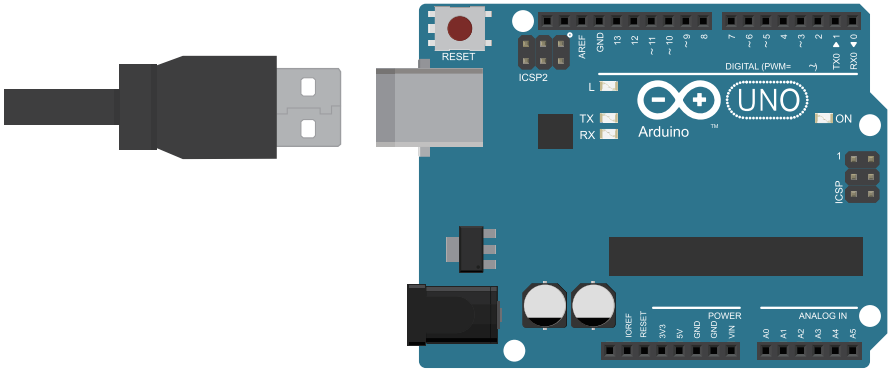
통신속도 설정

시리얼 통신을 할 때 통신속도에 대해 설정해야 합니다. 아두이노 소스상에서 설정한 통신속도와 맞춰서 주지 않으면 문자가 깨져서 읽을 수 없게 됩니다.

시리얼 통신(Serial Communication)은 한 번에 한 비트씩 데이터를 지속적으로 주고받는 직렬 통신을 의미합니다. 아두이노 – PC 통신, 아두이노 – 아두이노 통신, 아두이노 – 임베디드 통신을 할 때 사용할 수 있습니다.



Rx(Receive)는 데이터 수신을 의미하고, Tx(Transmit)는 데이터 전송을 의미합니다. 따라서 디바이스 두 개가 연결될 때, Rx와 Tx를 교차(Cross) 시켜서 연결해야 합니다. 이때 그라운드(GND) 또한 같이 연결되어야 합니다.



아래는 PC와 아두이노를 USB로 연결하여 시리얼 통신을 해보는 예제입니다. 시리얼 모니터로 문자를 입력하면 그대로 다시 출력하는 예제입니다.

예제소스

```
/*
Serial Com test
http://www.devicemart.co.kr
*/

void setup()  // 초기 설정 함수
{
  Serial.begin(9600); // 통신속도 9600bps로 시리얼 통신 시작
}
```

```
void loop() // 반복 함수 프로그램의 주 내용
{
  if(Serial.available()) // 시리얼 모니터를 통해 입력이 있다면,
  {
    char a; // 변수 a 선언
    a= Serial.read(); // 전송된 문자를 1바이트씩 읽음
    Serial.println(a); // 읽어온 문자를 그대로 출력
  }
}
```

Serial.begin()

시리얼 통신의 시작을 알리며 속도를 설정하는 함수입니다. 일반적으로 통신속도(Baud Rate)는 9600을 사용하며 단위는 bps(bits per second)입니다.

Serial.println()

Serial.println("문자열") 또는 Serial.println(변수)
위와 같은 형식으로 사용하여 문자열이나 변수의 값을 출력할 수 있습니다.
출력후 줄바꿈(₩n)을 해줍니다. 줄바꿈이 필요하지 않을시 Serial.print()함수를 사용하면 됩니다.

Serial.available()

시리얼 포트에서 읽을 수 있는 바이트 수를 가져옵니다. 이 데이터는 이미 버퍼(64바이트)에 저장되어 있는 데이터입니다. 데이터를 받지 않았다면 0을 반환합니다.

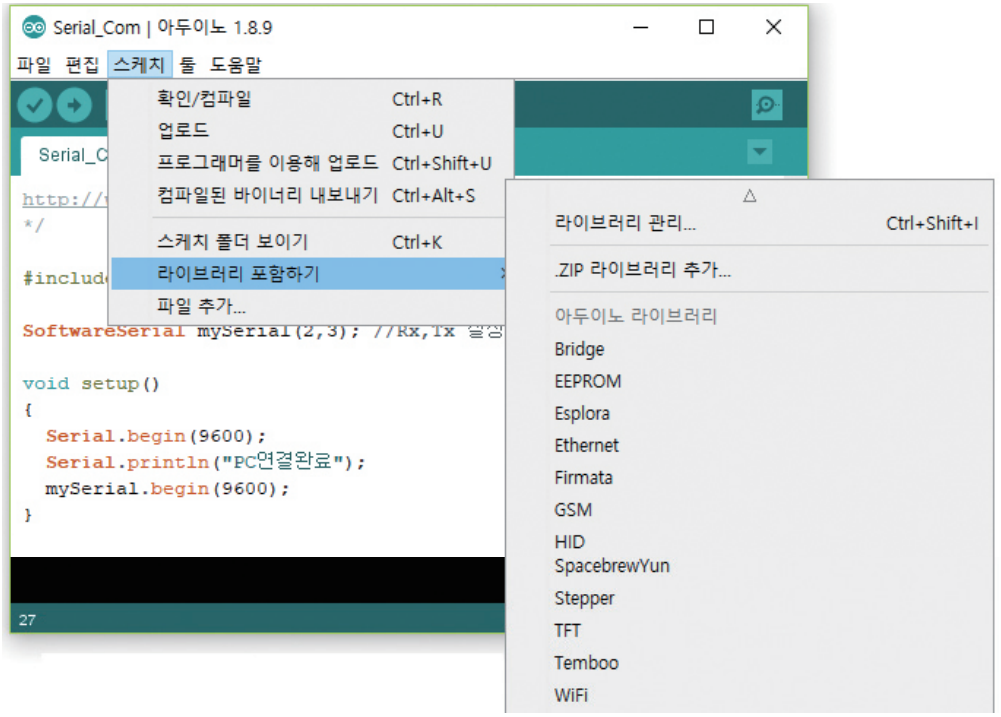
3.1.2. 소프트웨어 시리얼 통신

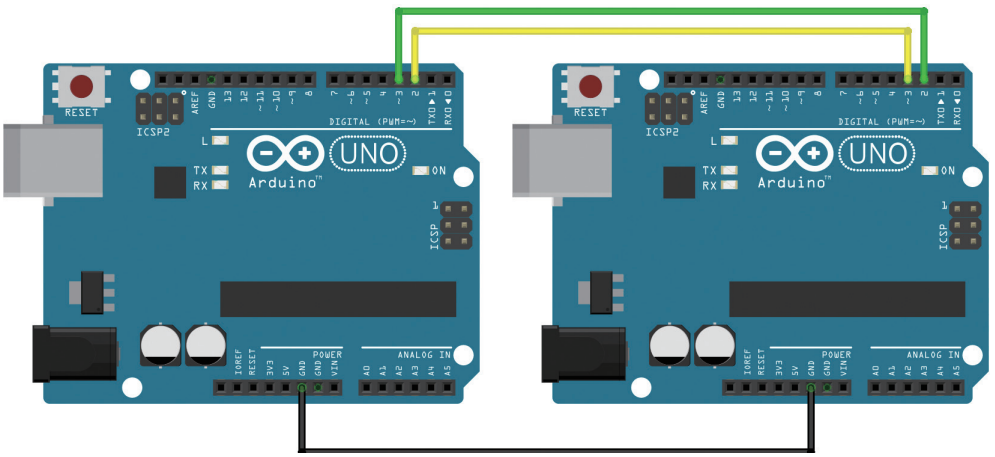
아두이노 시리얼 통신에서 업로드, 시리얼 모니터 등 PC와 통신하는 D0, D1핀을 하드웨어 시리얼(Hardware Serial)이라고 합니다. 하지만 아두이노에 하드웨어 시리얼을 통해 다른 디바이스와 연결되어 통신하고 있는 경우 아두이노로 소스코드를 업로드할 수 없습니다.

이 경우에는 다른 방법이 있습니다. 바로 소프트웨어 시리얼(Software Serial)을 사용하는 것인데요.

Software Serial 라이브러리를 추가하면 사용할 수 있습니다. 소프트웨어 시리얼을 사용하면 하드웨어 시리얼 핀 외의 다른 디지털 핀으로도 시리얼 통신을 할 수 있습니다.

소프트웨어 시리얼은 기본적으로 제공하는 표준 라이브러리입니다. 이번 실습은 소프트웨어 시리얼의 송수신을 모두 활용하기 위해 2대의 아두이노로 테스트합니다. 아두이노 보드가 한 대뿐이라면 참고만 하시고 넘어가셔도 좋습니다.





배선		
아두이노1	아두이노2	비고
GND	GND	Ground
D2	D3	mySerial Rx
D3	D2	mySerial Tx

준비물	
아두이노 우노	2대
USB 케이블	2개
브레드보드	1개
저항 220Ω	1개
LED	1개
점퍼와이어	10가닥 내외

USB 케이블은 아두이노1 (왼쪽)과 연결합니다.

예제소스

```
/*
Serial Com test2

http://www.devicemart.co.kr
*/

#include <SoftwareSerial.h> // SoftwareSerial 라이브러리 포함

SoftwareSerial mySerial(2,3); //Rx,Tx 설정

void setup()
{
  Serial.begin(9600); // PC 시리얼 통신 속도 9600으로 설정
  Serial.println("PC연결완료"); // 시리얼 통신 시작 문자 출력
  mySerial.begin(9600); // 아두이노간 시리얼 통신 속도 9600으로 설정
}
void loop()
{
  if(Serial.available()) // 시리얼 모니터를 통해 데이터 입력이 있다면
  mySerial.write(Serial.read()); // 다른 아두이노로 데이터 출력

  if(mySerial.available()) // 다른 아두이노를 통해 데이터 입력이 있다면
  Serial.write(mySerial.read()); // 시리얼 모니터로 시리얼 출력
}
```

SoftwareSerial 이름(RX핀, TX핀)

소프트웨어 시리얼 클래스 생성자입니다. 이름에 본인이 사용할 클래스 명을 지정해줍니다.
RX핀과 TX핀에는 각각의 핀에는 0번과 1번핀을 제외한 디지털 입출력 핀 번호를 넣습니다. 아날로그 입력핀으로는 동작하지 않습니다. 비교적 고차원적인 개념이기때문에 클래스에 대해서는 깊게 다루지 않겠습니다. 선언하는 방식만 암기해주시면 되시겠습니다.

이름.begin()

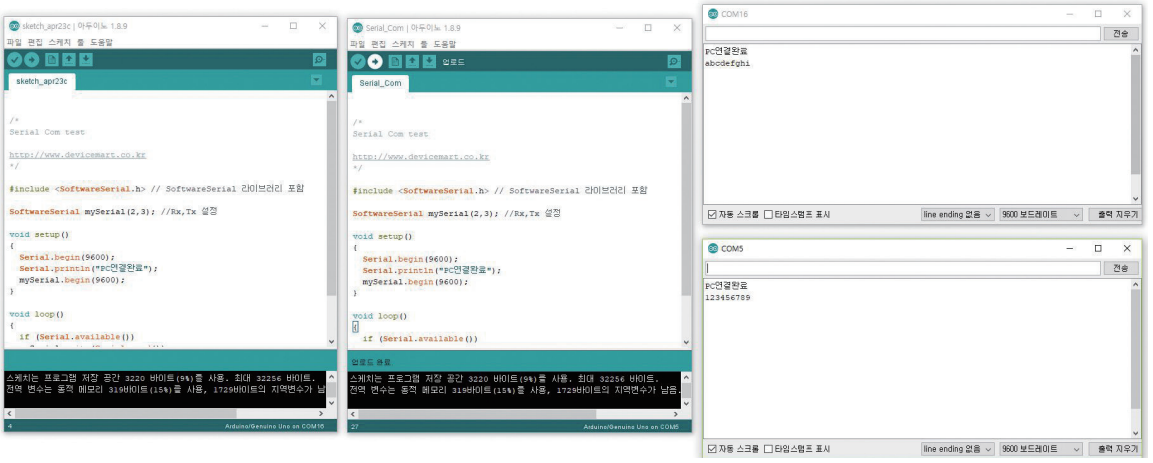
이름.begin(통신속도)
소프트웨어 시리얼 통신의 속도를 설정하고 사용을 선언하는 함수입니다. 사용법은 Serial.begin()함수와 동일합니다.

이름 .available()

소프트웨어 시리얼 포트에서 읽을 수 있는 바이트 수를 가져옵니다. 이 데이터는 이미 버퍼(64바이트)에 저장되어 있는 데이터입니다. 데이터를 받지 않았다면 0을 반환합니다.

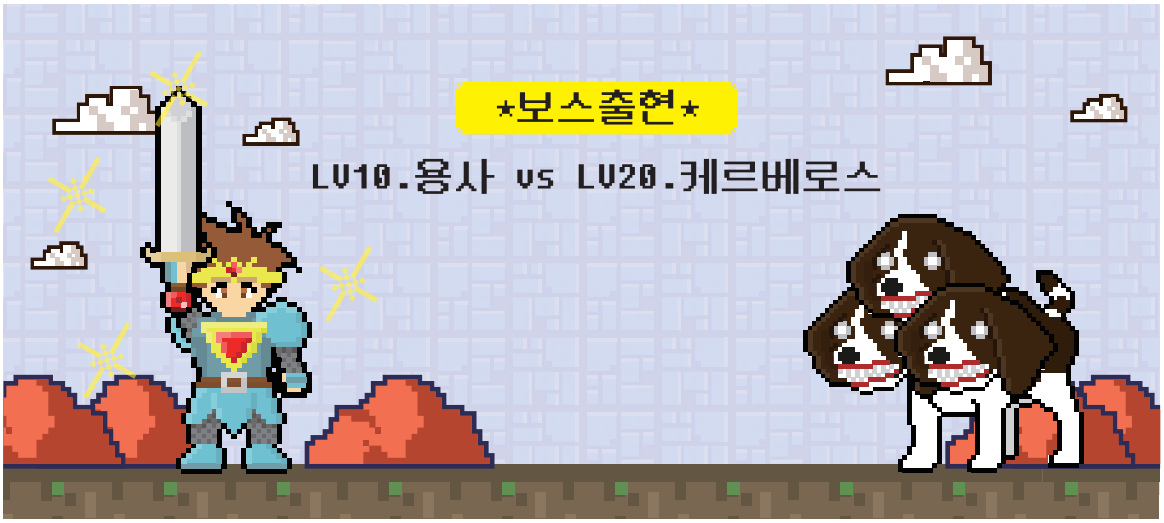
이름 .read()

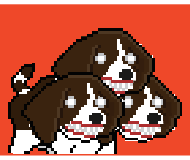
소프트웨어 시리얼 버퍼에 들어있는 데이터를 순서대로 1바이트씩 읽어오는 함수입니다. 읽어온 데이터가 없을 경우 –1을 반환합니다.



위 소스를 두 아두이노에 모두 업로드하면, PC에서 시리얼 모니터를 통해 전송한 데이터를 반대편 아두이노에 연결된 시리얼 모니터 출력창에서 확인할 수 있습니다.

Quest 7. Boss : 마왕성의 문지기





케르베로스

크르르르… 왈왈! 컹컹!!

용사

엄청 크고 무섭게 생긴 강아지다…
방심하지 말자!

BOSS 문제

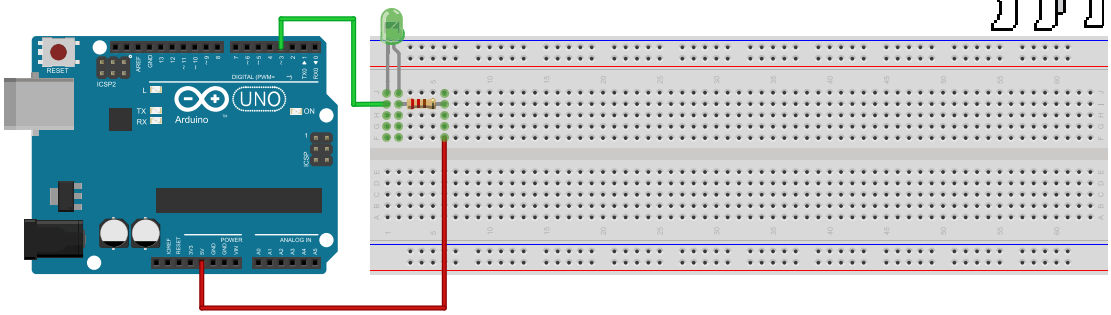
최초 구동시에는 LED가 꺼져있다가 시리얼 모니터로 데이터가 입력되면 시리얼 모니터에 "LED on"이라는 문자열을 출력하고 LED가 켜지게 하시오.

준비물	
아두이노 우노	1대
브레드보드	1개
저항 220Ω	1개
LED	1개
점퍼와이어	10가닥 내외

Quest 7. 정답 체크

정상적으로 구동되었다면 케르베로스에 X표 해주세요.

정답예제회로



정답예제소스

```
void setup() { // 초기 설정 함수.
  Serial.begin(9600); // 시리얼 통신 속도 9600으로 설정
  pinMode(3,OUTPUT); // 생략 가능, PWM 출력이 가능한 핀이라면 3번핀이 아니라도 정답입니다.
  digitalWrite(3,HIGH); // led를 꺼진 상태로 초기화합니다.
}
void loop() { // 안에 작성한 코드가 계속 반복됩니다.
  if(Serial.available()) // 시리얼 모니터를 통해 입력이 있다면,
  {
    Serial.println("LED on"); // led를 켰다는 문자열 출력
    digitalWrite(3,LOW); // led를 켭니다.
  }
}
```

※디지털 출력 핀을 LED의 캐소드가 아닌 에노드에 연결했다면 dighitalWrite(3, 출력)에서 출력의 HIGH와 LOW를 반대로 입력하셔도 정답입니다.

틀렸다면 정답예제 실습 후
스테이지 3.1 부터 다시!

모두 맞추셨다면 다음페이지로!

용사는 『시리얼 메소드』 얻었다.

스테이지 3.2

LED

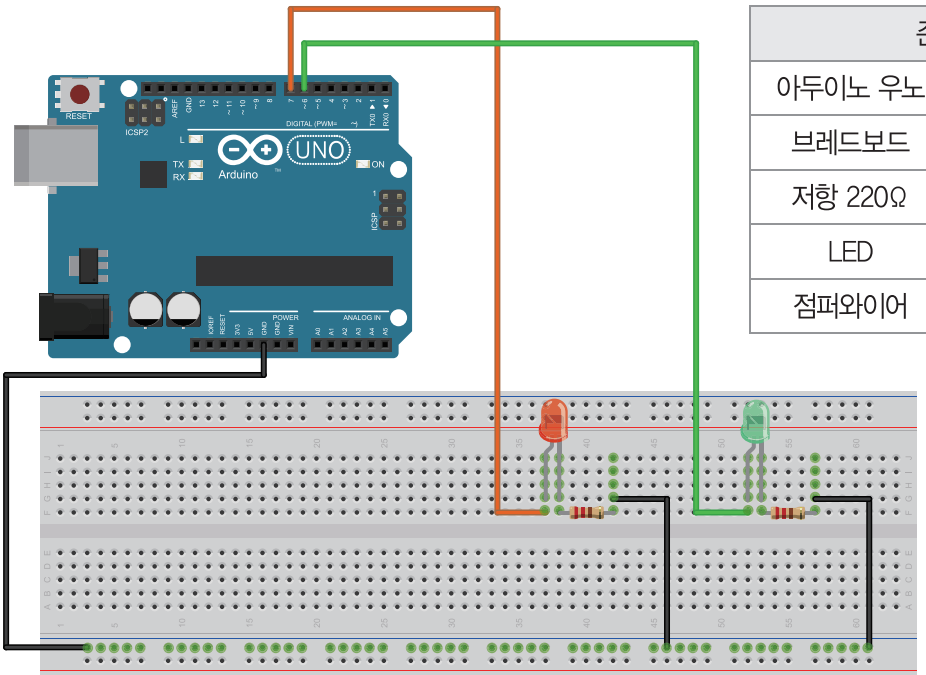
EXP♥

3. 아두이노 실전 연습

3.2. LED

3.2.1. 일반 LED

아래 예제는 시리얼 통신을 이용하여 일반 led를 껐다 켰다 할 수 있게 제어하는 예제입니다. 시리얼 모니터에 1을 입력하면 2개의 led가 on이 되고, 0을 입력하면 2개의 led가 off가 됩니다.

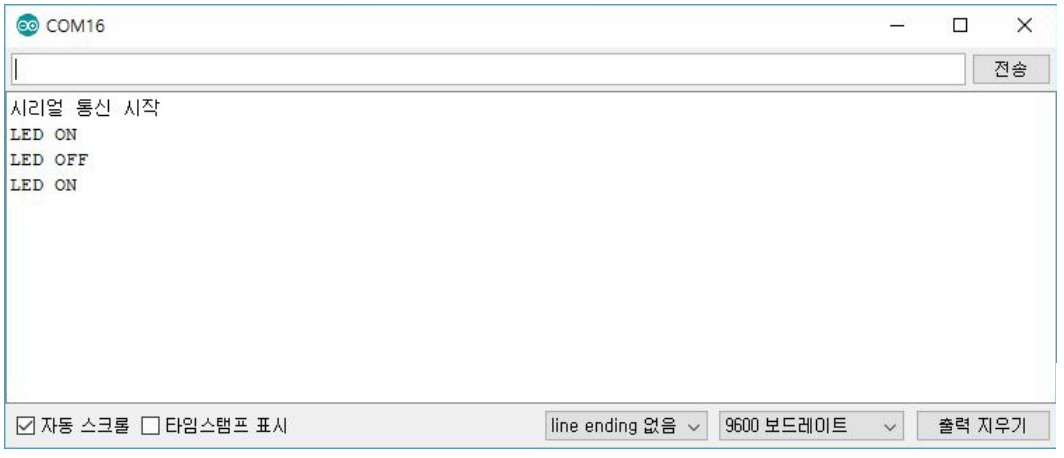


준비물	
아두이노 우노	1대
브레드보드	1개
저항 220Ω	2개
LED	2개
점퍼와이어	10가닥 내외

예제소스

```
/*
serial_led test
http://www.devicemart.co.kr
*/

int led1=6; //led1를 D6에 연결
int led2=7; //led2를 D7에 연결

void setup()  //초기 설정 함수
{
  pinMode(led1,OUTPUT); //6번에 연결된 led1를 출력모드로 설정
  pinMode(led2,OUTPUT); //7번에 연결된 led2를 출력모드로 설정
  Serial.begin(9600); //시리얼 통신 속도 9600으로 설정
  Serial.println("시리얼 통신 시작"); //시리얼 통신 시작 문자 출력
```

```
}

void loop() //반복 함수 프로그램의 주 내용
{
  if(Serial.available()) //시리얼 모니터를 통해 입력이 있다면
  {
    char a; //변수 a 선언
    a=Serial.read(); //전송된 문자를 1바이트 씩 읽음
    if(a=='1') //전송 받은 문자가 1이라면
    {
      digitalWrite(6,HIGH); //led1 on
      digitalWrite(7,HIGH); //led2 on
      Serial.println("LED ON"); //문자열 출력
    }
    if(a=='0') //전송 받은 문자가 0이라면
    {
      digitalWrite(6,LOW);  //led1 off
      digitalWrite(7,LOW);  //led2 off
      Serial.println("LED OFF"); // 문자열 출력
    }
  }
}
```



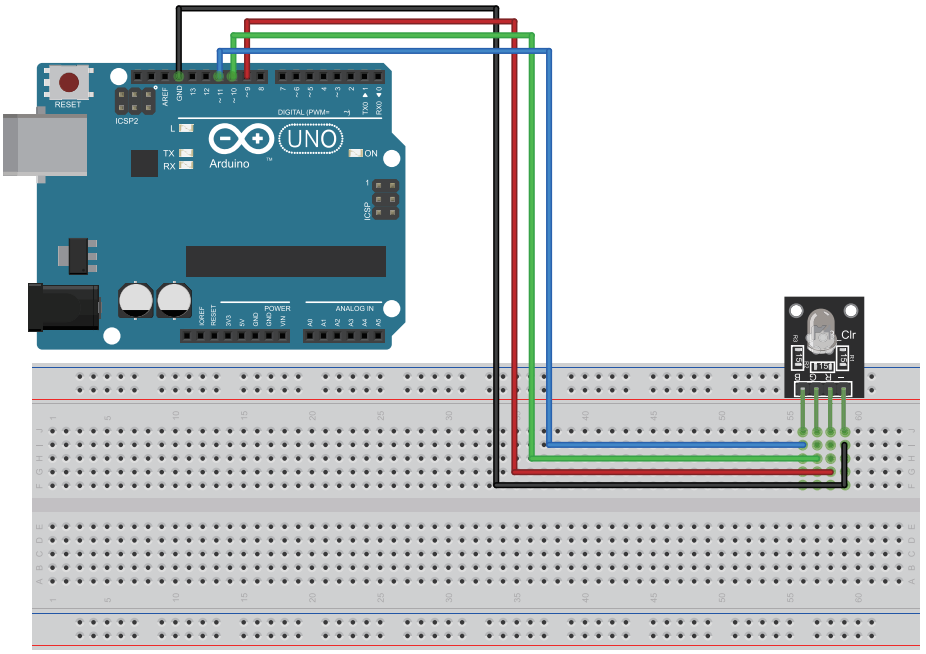
1, 0, 1 순서대로 입력했을 때의 시리얼 모니터 화면입니다. 1을 입력하면 두 개의 LED가 켜지고 0를 입력하면 두 개의 LED가 모두 꺼지는 것을 확인할 수 있습니다.

3.2.2. RGB LED 모듈

RGB모듈은 R, G, B,–(GND)의 4핀으로 이루어져 있으며, 각각 빨강(Red), 초록(Green), 파랑(Blue), 그라운드(Ground)를 뜻합니다. 즉 1개의 led로 빨강, 초록, 파랑의 색을 출력할 수 있습니다.

빨강, 초록, 파랑은 빛의 삼원색입니다. 원색이란 색을 혼합하여 모든 종류의 색을 만들 수 있는, 서로 독립적인 색을 말합니다. 아두이노에 있는 PWM 신호를 사용하게 되면 RGB핀에 각각 0~255의 값을 넣을 수 있고 그 색을 조합하면 원하는 색을 출력할 수 있습니다.

아래 예제는 아두이노와 RGB모듈을 연결하여 빨간색, 초록색, 파란색, 그리고 이 빛의 삼원색을 합쳐서 다른 색들을 출력해보는 예제 입니다.



배선		
아두이노	RGB 모듈	비고
GND	–	Ground
D9	R	빨강
D10	G	초록
D11	B	파랑

준비물	
아두이노 우노	1대
브레드보드	1개
RGB LED 모듈	1개
점퍼와이어	10가닥 내외

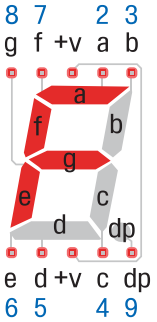
예제소스

```
/*
RGB LED
RGB LED로 색을 출력하는 예제
http://www.devicemart.co.kr/
*/

int LED1 =9; // Red를 9번에 연결
int LED2 =10; // Green을 핀 10번에 연결
int LED3 =11; // Blue를 핀 11번에 연결

void setup ()
{
  pinMode(LED1,OUTPUT);
  pinMode(LED2,OUTPUT);
  pinMode(LED3,OUTPUT);
}
void loop()
{
  digitalWrite(LED1,HIGH); // 빨간색 LED 1초동안 내보내기
  delay(1000);
  digitalWrite(LED1,LOW); // 빨간색 LED off
  delay(100);
  digitalWrite(LED2,HIGH); // 초록색 LED 1초동안 내보내기
  delay(1000);
  digitalWrite(LED2,LOW); // 초록색 LED off
  delay(100);
  digitalWrite(LED3,HIGH); // 파란색 LED 1초동안 내보내기
  delay(1000);
  digitalWrite(LED3,LOW); // 파란색 LED off
  delay(100);
  digitalWrite(LED1,HIGH); // 빨간색 LED 1초동안 내보내기
  digitalWrite(LED2,HIGH); // 초록색 LED 1초동안 내보내기
  delay(1000);
  digitalWrite(LED1,LOW); // 빨간색 LED off
  digitalWrite(LED2,LOW); // 초록색 LED off
  delay(100);
  digitalWrite(LED2,HIGH); // 초록색 LED 1초동안 내보내기
```

```
  digitalWrite(LED3,HIGH); // 파란색 LED 1초동안 내보내기
  delay(1000);
  digitalWrite(LED2,LOW); // 초록색 LED off
  digitalWrite(LED3,LOW); // 파란색 LED off
  delay(100);
  digitalWrite(LED1,HIGH); // 빨간색 LED 1초동안 내보내기
  digitalWrite(LED2,HIGH); // 초록색 LED 1초동안 내보내기
  digitalWrite(LED3,HIGH); // 파란색 LED 1초동안 내보내기
  delay(1000);
  digitalWrite(LED1,LOW); // 빨간색 LED off
  digitalWrite(LED2,LOW); // 초록색 LED off
  digitalWrite(LED3,LOW); // 파란색 LED off
  delay(100);
}
```

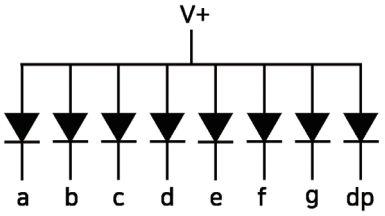
프로그램을 구동하면 RGB LED가
빨강 → **초록** → **파랑** → **노랑** → **하늘색(Cyan)** → **자주색**
순서로 1초씩 켜지는 것을 확인하실 수 있습니다.

3.2.3. FND

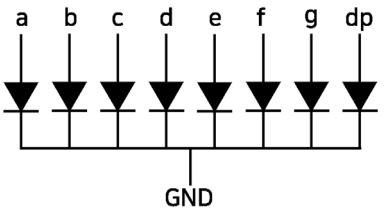


FND(Flexible Numeric Display)는 7-segment라고도 부르는 숫자를 표현하는데 사용되는 부품입니다.

내부에 LED가 8개가 들어가있고, 공통단자가 2개 있어서, 9개의 핀으로 0부터 9까지 그리고 소수점까지 표현할 수 있습니다. 전원을 VCC를 공통으로 하는지 GND를 공통으로 하는지에 따라 애노드 공통(Common Anode)타입, 캐소드 공통(Common Cathode)타입의 두가지 방식으로 분류됩니다.



애노드 공통타입



공통 캐소드

왼쪽 그림은 애노드 공통(+)타입입니다. 공통단자(Common)에 전원 5V를 입력하고, 원하는 단자에 LOW(0V)신호를 주어서 LED를 On 시킬 수 있습니다.

오른쪽 그림은 공통 캐소드(-) 타입입니다. 공통단자(Common)에 전원 GND를 입력하고, 원하는 단자에 HIGH(5V)신호를 주어서 LED를 On 시킬 수 있습니다.

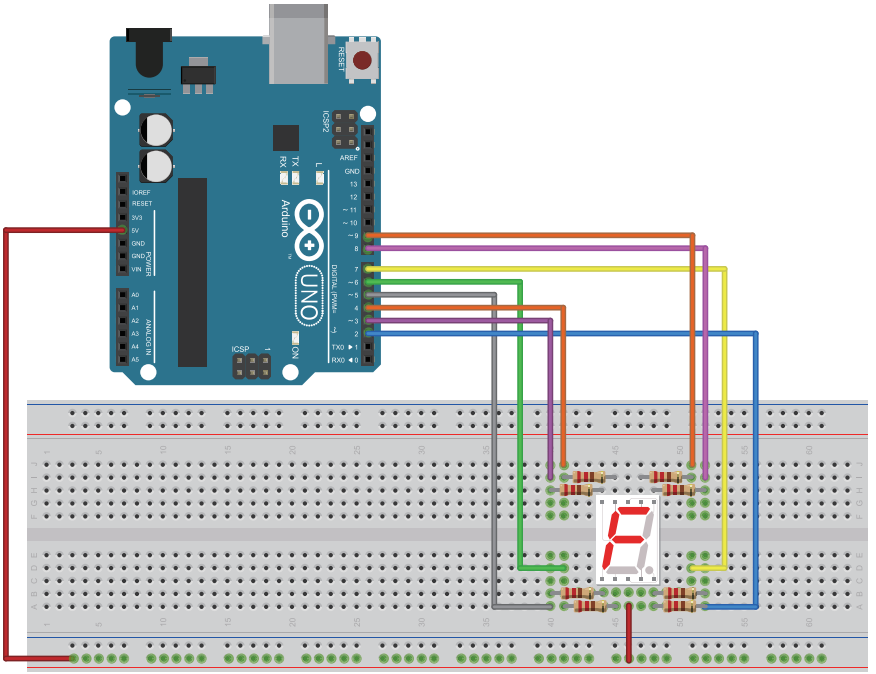
애노드 공통(Common Anode)									
숫자	dp	g	f	e	d	c	b	a	16진수
0	1	1	0	0	0	0	0	0	0xC0
1	1	1	1	1	1	0	0	1	0xF9
2	1	0	1	0	0	1	0	0	0xA4
3	1	0	1	1	0	0	0	0	0xB0
4	1	0	0	1	1	0	0	1	0x99
5	1	0	0	1	0	0	1	0	0x92
6	1	0	0	0	0	0	1	0	0x82
7	1	1	1	1	1	0	0	0	0xF8
8	1	0	0	0	0	0	0	0	0x80
9	1	0	0	1	0	0	0	0	0x90

애노드 공통 에서 숫자0은 dp를 시작으로 했을 때, 2진수로는 1100 0000이고 이 숫자를 16진수로 변환하면 0xC0입니다.

캐소드 공통(Common Cathode)									
숫자	dp	g	f	e	d	c	b	a	16진수
0	0	0	1	1	1	1	1	1	0x3F
1	0	0	0	0	0	1	1	0	0x06
2	0	1	0	1	1	0	1	1	0x5B
3	0	1	0	0	1	1	1	1	0x4F
4	0	1	1	0	0	1	1	0	0x66
5	0	1	1	0	1	1	0	1	0x6D
6	0	1	1	1	1	1	0	1	0x7D
7	0	0	0	0	0	1	1	1	0x07
8	0	1	1	1	1	1	1	1	0x7F
9	0	1	1	0	1	1	1	1	0x6F

캐소드 공통에서 숫자0은 dp를 시작으로 했을 때, 2진수로는 0111 1111이고 이 숫자를 16진수로 변환하면 0x3F입니다.

예제에 사용하는 FND는 애노드 공통 FND입니다. 아두이노로 숫자 0~9을 표시하도록 제어하는 예제입니다.



배선			
아두이노	FND	아두이노	FND
5V	V+	D6	d
D2	dp	D7	c
D3	g	D8	b
D4	f	D9	a
D5	e		

준비물	
아두이노 우노	1대
브레드보드	1개
저항 220Ω	8개
FND(Common Anode)	1개
점퍼와이어	10가닥 내외

예제소스

```
/*
7-segment (FND) 애노드 공통 타입
http://www.devicemart.co.kr/
*/
int seg[] = {2,3,4,5,6,7,8,9}; // 배열로 D2~D9까지 세그먼트 연결
byte digits[10][8] =  // 0~9까지 숫자에 대한 LED 값
{
  { 1,1,0,0,0,0,0,0 }, // 0
  { 1,1,1,1,1,0,0,1 }, // 1
  { 1,0,1,0,0,1,0,0 }, // 2
```

```
  { 1,0,1,1,0,0,0,0 }, // 3
  { 1,0,0,1,1,0,0,1 }, // 4
  { 1,0,0,1,0,0,1,0 }, // 5
  { 1,0,0,0,0,0,1,0 }, // 6
  { 1,1,1,1,1,0,0,0 }, // 7
  { 1,0,0,0,0,0,0,0 }, // 8
  { 1,0,0,1,0,0,0,0 }  // 9
};

void setup()
{
  for(int i = 0; i < 8; i++) // for 문을 사용해 핀 2~9까지 8개의 핀모드 출력으로 세팅
  {
    pinMode(seg[i], OUTPUT);
  }
}
void loop()
{
  for (int i = 0; i < 10; i++) // 이중 for문을 이용해 led로 숫자 출력
  {
    for (int j = 0; j < 8; j++)
    {
      digitalWrite(seg[j], digits[i][j]);
    }
    delay(1000); // 1초간 딜레이
  }
}
```

프로그램을 구동하시면 1초 간격으로 FND에 0에서부터 9까지의 숫자가 표시되는 동작이 반복됩니다.

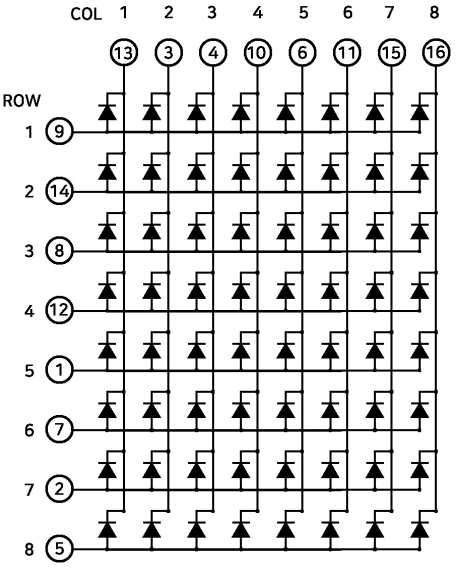
3.2.4. 도트 매트릭스

도트 매트릭스(Dot matrix)는 led matrix라고도 부르는 제품으로 주로 지하철, 버스, 가게 간판 등에서 문자와 숫자를 표현하는데 사용됩니다. 제어하기에 따라서 간단한 도형이나 이모티콘도 표현이 가능합니다.

8x8 도트 매트릭스는 내부에 led가 8행(row) 8열(column)로 64개가 들어가있고, FND와 마찬가지로 애노드 공통(Common Anode) 타입과 캐소드 공통(Common Cathode) 타입의 두 종류가 있습니다.

애노드 공통과 캐소드 공통의 구분은 행(row)을 기준으로 구분합니다.

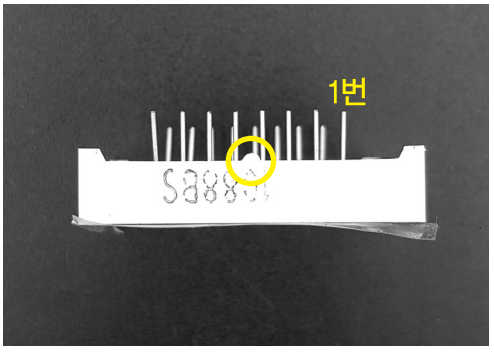
이번에 사용해 볼 애노드 공통 타입 도트 매트릭스의 내부 회로도 및 핀맵입니다.

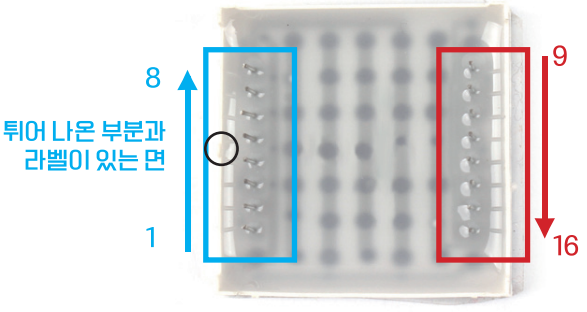


캐소드 공통 타입의 경우 내부 led의 VCC와 GND가 반대로 연결되어 있습니다.

ROW 1번부터 8번까지 VCC를 공급해 줄 수 있도록 각 LED에 공통으로 연결되어 있는 모습을 확인하실 수 있습니다.

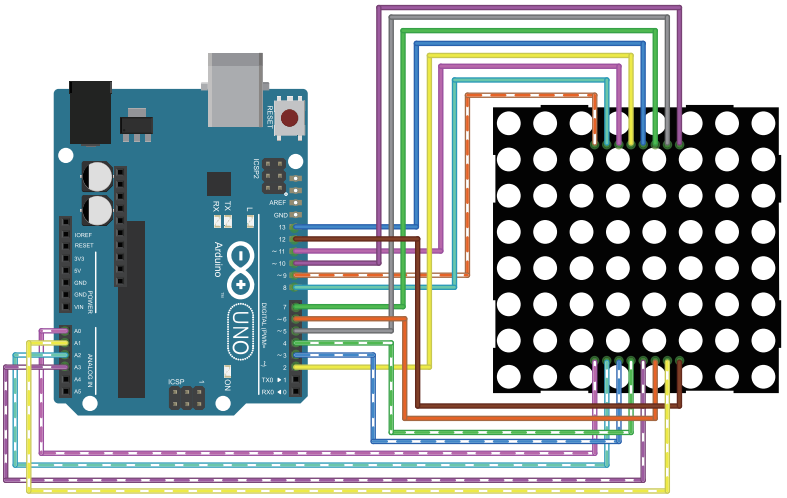
특이한 점은 핀 번호가 순서대로 되어있지 않다는 점입니다. ROW 1번은 도트 매트릭스의 9번 핀으로 시작하지만 COL 1번이 1번핀으로 시작하는 게 아닙니다. 이는 제조사마다 다를 수 있습니다. 핀 번호를 구분하는 방법 역시 제조사마다 다를 수 있지만 보편적인 방법으로 설명드리겠습니다.





좌측 사진을 보시면 부품 4면 중 한 면에 글씨가 쓰여있고 테두리 가운데가 뽈록 솟아있는 면이 있습니다. 사진처럼 핀을 위로 가게 봤을 때, 맨 오른쪽이 1번 핀입니다.

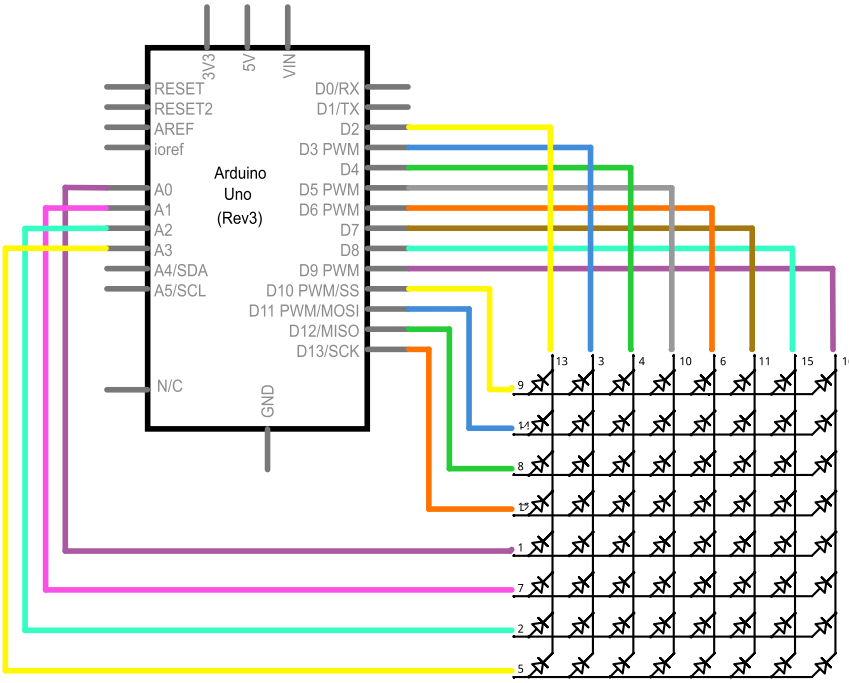
오른쪽 사진은 핀 번호의 진행입니다. 동그라미 친 부분이 뽈록 솟은 면이고 밑에 1번핀부터 8번까지 순차적으로 핀이 진행되다가 건너편 9번에서 16번까지는 반대 방향으로 번호가 증가합니다. 시계방향으로 번호를 생각하시면 됩니다.



도트매트릭스 1열(column)을 아두이노 D2핀부터 시작하여 순서대로 연결했습니다. 1행(row)은 D10부터 시작하며 모자란 디지털 단자는 A0~A3까지를 D14~17로 설정하여 사용하였습니다.

아날로그 입력 포트를 디지털 입출력으로 설정하여 사용 할 수 있다는 건 전에 짧게 언급하긴 했으나 이번에 실습으로 직접 사용해보도록 하겠습니다.

열 배선				행 배선			
아두이노	도트 매트릭스	아두이노	도트 매트릭스	아두이노	도트 매트릭스	아두이노	도트 매트릭스
D2	13	D6	6	D10	9	A0(D14)	1
D3	3	D7	11	D11	14	A1(D15)	7
D4	4	D8	15	D12	8	A2(D16)	2
D5	10	D9	16	D13	12	A3(D17)	5



준비물	
아두이노 우노	1대
브레드보드	1개
LED	1개
점퍼와이어	20가닥 내외

도트 매트릭스를 이용하여 하트 모양을 만들어 보는 예제입니다. 도트매트릭스를 제어하기 위해서는 1행 혹은 1열로만 제어가 가능합니다. 동시에 다른 행, 다른 열까지 제어하려고 하면, 모든 LED가 켜집니다.

그럼 어떻게 원하는 led만 골라서 출력을 할 수 있을까요?

0	0	0	0	0	0	0	0
0	1	1	0	0	1	1	0
1	0	0	1	1	0	0	1
1	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1
0	1	0	0	0	0	1	0
0	0	1	0	0	1	0	0
0	0	0	1	1	0	0	0

정답은 순서대로 켜는 방법입니다. 도트 매트릭스를 위에서부터 아래로 하나의 행씩 제어를 합니다.

예를 들어 위 그림처럼 1행은 아무것도 켜지않고 넘어가고, 2행은 1열 off, 2열 on, 3열 on, 4열 off, 5열 off, 6열 on, 7열 on, 8열 off 이런 식으로 순서대로 한 열을 제어합니다.

일정 속도 이상으로 on/off를 제어하게 되면 인간의 인지 한계로 인해(잔상효과)인해 모든 행의 led를 제어하는 것으로 보입니다. 실제로는 한 줄씩 8번에 거쳐 제어했지만 우리의 눈에는 하트 모양이 보이게 됩니다. 이런 식으로 도트 매트릭스를 제어하는 방법을 다이나믹(Dynamic) 구동방법이라고 합니다.

예제소스

```
/*
8x8 dot-matrix heart
http://www.devicemart.co.kr/
*/

// 행(row)열(column) 도트매트릭스 led 아두이노에 연결
// -1, -2는 행열을 1부터 시작하기 위해 더미(dummy) 핀으로 넣음
int rows[] = {-1,10,11,12,13,14,15,16,17};
int cols[] = {-2,2,3,4,5,6,7,8,9};

void setup()
{
  for (int i = 1; i < 9; i++) // 도트매트릭스 행열 출력으로 세팅
```

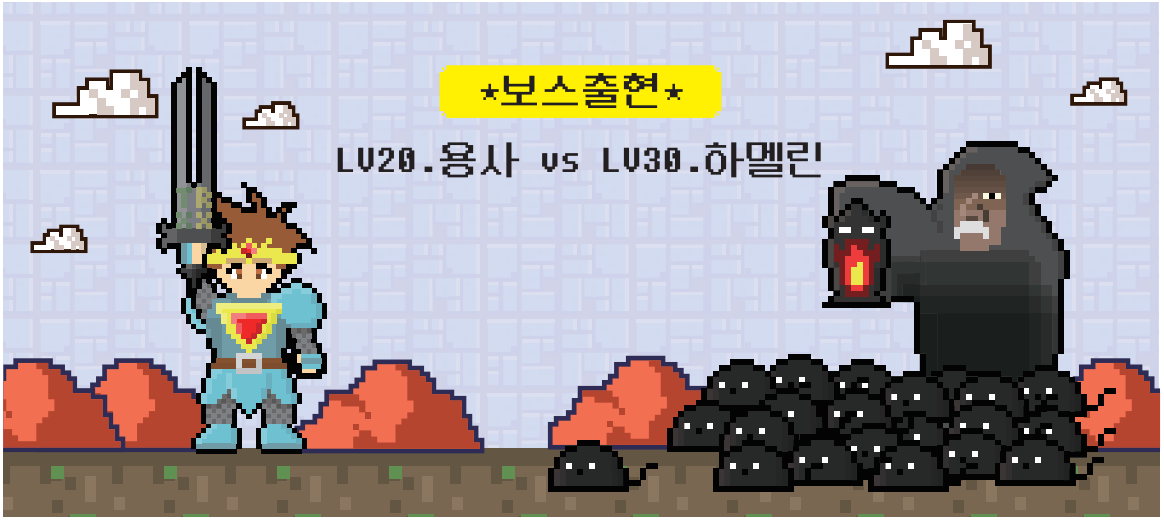
```
  {
    pinMode(cols[i], OUTPUT);
    pinMode(rows[i], OUTPUT);
  }
}

void loop()
{ // 1행엔 켜지는 led가 없으므로 생략
  clear(); // 사용자 정의함수 clear 호출, led 초기화
  digitalWrite(rows[2],HIGH); // 2행 led on
  digitalWrite(cols[2],LOW);
  digitalWrite(cols[3],LOW);
  digitalWrite(cols[6],LOW);
  digitalWrite(cols[7],LOW);
  delay(2);
  clear();
  digitalWrite(rows[3],HIGH); // 3행 led on
  digitalWrite(cols[1],LOW);
  digitalWrite(cols[4],LOW);
  digitalWrite(cols[5],LOW);
  digitalWrite(cols[8],LOW);
  delay(2);

  clear();
  digitalWrite(rows[4],HIGH); // 4행 led on
  digitalWrite(cols[1],LOW);
  digitalWrite(cols[8],LOW);
  delay(2);

  clear();
  digitalWrite(rows[5],HIGH); // 5행 led on
  digitalWrite(cols[1],LOW);
  digitalWrite(cols[8],LOW);
  delay(2);

  clear();
  digitalWrite(rows[6],HIGH); // 6행 led on
  digitalWrite(cols[2],LOW);
```

```
  digitalWrite(cols[7],LOW);
  delay(2);

  clear();
  digitalWrite(rows[7],HIGH);  // 7행 led on
  digitalWrite(cols[3],LOW);
  digitalWrite(cols[6],LOW);
  delay(2);
  clear();
  digitalWrite(rows[8],HIGH);  // 8행 led on
  digitalWrite(cols[4],LOW);
  digitalWrite(cols[5],LOW);
  delay(2);
}

void clear() // 사용자 정의함수 clear 선언
{
  for(int i = 1; i < 9 ; i++)  // LED 초기화 2중 for문
  {
    digitalWrite(rows[i],LOW);
}
```

clear()

사용자 정의 함수로 임의로 만든 함수입니다. for문을 이용하여 모든 행열의 led를 제어하여 초기화 시키는 함수입니다. clear()를 사용해서 함수를 호출할 수 있습니다.

Quest 8. Boss : 사천왕 제 4석 쥐몰이꾼 하멜린



하멜린
용케도 여기까지 왔군… 킬킬킬
하지만 이 앞은 한발자국도 지나갈 수 없을거다… 킬킬킬

용사
비켜주세요!! 집으로 돌아가야해요!

BOSS 문제

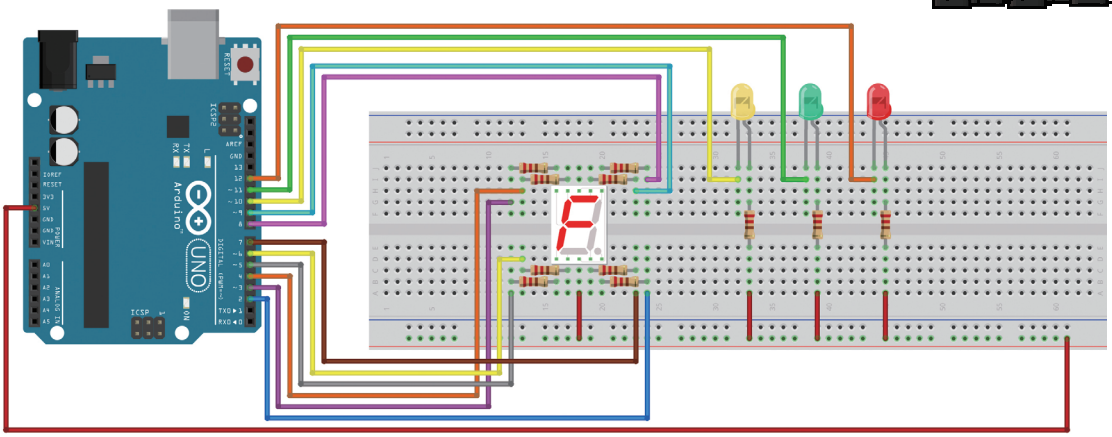
아두이노에 LED 3개와 FND 1개를 연결합니다.
시리얼 모니터에 1을 입력하면 LED가 1개 켜지고, FND에 숫자 1을 표시합니다.
시리얼 모니터에 2를 입력하면 LED가 2개 켜지고, FND에 숫자 2을 표시합니다.
시리얼 모니터에 3을 입력하면 LED가 3개 켜지고, FND에 숫자 3을 표시합니다.
시리얼 모니터에 0을 입력하면 LED가 모두 꺼지고, FND에 숫자 0을 표시합니다.
LED 색상은 상관 없습니다.
위처럼 동작하도록 회로를 구성하고 프로그래밍하여 동작하시오.

준비물	
아두이노 우노	1대
브레드보드	1개
저항 220Ω	11개
LED	3개
FND	1개
점퍼와이어	10가닥 내외

Quest 8. 정답 체크

정상적으로 구동되었다면 하멜린에 X표 해주세요.

정답예제회로



FND 배선			
아두이노	FND	아두이노	FND
5V	V+	D6	d
D2	dp	D7	c
D3	g	D8	b
D4	f	D9	a
D5	e		

LED 배선	
아두이노	LED
D10	노랑 LED
D11	초록 LED
D12	빨강 LED

정답예제소스

```
int led1=12; // led1를 D12에 연결
int led2=11; // led2를 D11에 연결
int led3=10; // led3을 D10에 연결

int seg[] = {2,3,4,5,6,7,8,9}; // 배열로 D2~D9까지 세그먼트 연결
byte digits[4][8] =  // 1~3까지 숫자에 대한 LED 값
{
 { 1,1,0,0,0,0,0,0 }, // 0
 { 1,1,1,1,1,0,0,1 }, // 1
 { 1,0,1,0,0,1,0,0 }, // 2
 { 1,0,1,1,0,0,0,0 }  // 3
```

```
};

void setup()
{
 for(int i = 0; i < 8; i++) // for문을 사용해 핀 2~9까지 8개의 핀모드 출력으로 세팅
 {
   pinMode(seg[i], OUTPUT);
 }

// 초기 세팅
 for (int i = 0; i < 8; i++)     // for문으로 숫자 0 출력
     {
       digitalWrite(seg[i], digits[0][i]);
     }

 pinMode(led1, OUTPUT); // led1 출력세팅
 pinMode(led2, OUTPUT); // led2 출력세팅
 pinMode(led3, OUTPUT); // led3 출력세팅
 digitalWrite(led1, HIGH); // led1 off로 초기화
 digitalWrite(led2, HIGH); // led2 off로 초기화
 digitalWrite(led3, HIGH); // led3 off로 초기화
 Serial.begin(9600); // 시리얼 통신 속도 9600으로 설정
 Serial.println("시리얼 통신 시작"); // 시리얼 통신 시작 문자 출력
}

void loop() // 반복 함수 프로그램의 주 내용
{

 char a; // 변수 a 선언
 if(Serial.available()) // 시리얼 모니터를 통해 입력이 있다면
 {
   a=Serial.read(); // 전송된 문자를 1바이트씩 읽음
   if(a == '0') // 전송받은 문자가 0이라면
   {
     digitalWrite(led1, HIGH); // led1 off
     digitalWrite(led2, HIGH); // led2 off
     digitalWrite(led3, HIGH); // led3 off
     for (int i = 0; i < 8; i++)     // for문으로 숫자 0 출력
```

```
      {
        digitalWrite(seg[i], digits[0][i]);
      }
    }
    if(a == '1') // 전송받은 문자가 1이라면
    {
      digitalWrite(led1, LOW); // led1 on
      digitalWrite(led2, HIGH); // led2 off
      digitalWrite(led3, HIGH); // led3 off
      for (int i = 0; i < 8; i++)    // for문으로 숫자 1 출력
      {
        digitalWrite(seg[i], digits[1][i]);
      }
    }
    if(a == '2') // 전송받은 문자가 2라면
    {
      digitalWrite(led1, LOW); // led1 on
      digitalWrite(led2, LOW); // led2 on
      digitalWrite(led3, HIGH); // led3 off
      for (int i = 0; i < 8; i++)    // for문으로 숫자 2 출력
      {
        digitalWrite(seg[i], digits[2][i]);
      }
    }

  if(a == '3') // 전송받은 문자가 3이라면
    {
      digitalWrite(led1, LOW); // led1 on
      digitalWrite(led2, LOW); // led2 on
      digitalWrite(led3, LOW); // led3 on
      for (int i = 0; i < 8; i++)    // for문으로 숫자 3 출력
      {
        digitalWrite(seg[i], digits[3][i]);
      }
    }
  }
}
```

※ 디지털 출력 핀을 LED의 캐소드가 아닌 에노드에 연결했다면 dighitalWrite()의 HIGH와 LOW를 반대로 입력하셔도 정답입니다.

틀렸다면 정답예제 실습 후
스테이지 3.2 부터 다시!

모두 맞추셨다면 다음페이지로!

용사
시리얼 통신 쯤이야 이젠 어렵지 않아!

하멜린
시리얼 메소드가 있을 줄이야…
크윽… 분하다…

용사는 『디지털라이트 메소드』를 얻었다.

용사
디지털 출력, 기초인만큼 정말 유용한 것 같아!!

스테이지 3.3

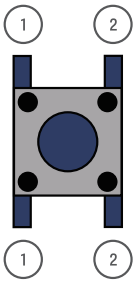
스위치

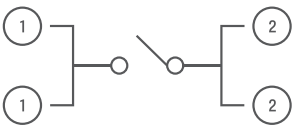
EXP♥

3. 아두이노 실전 연습

3.3. 스위치

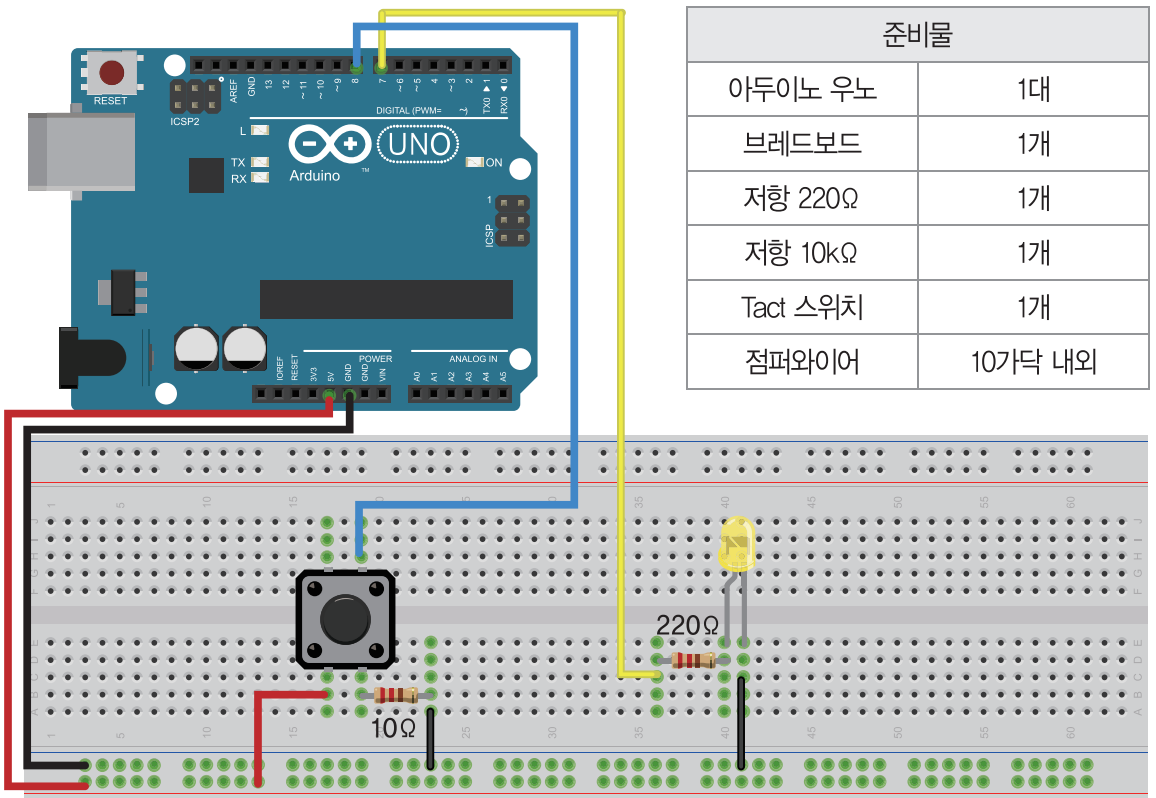
3.3.1. 택트 스위치





택트 스위치(Tact switch)는 아두이노에서 입력을 받을 때 제일 기초적으로 사용되는 스위치로 푸쉬 버튼 스위치(Push button switch)라고도 누르기 전에는 세로줄로 각각 연결되어 있다가, 스위치를 누르고 있을 때는 4개의 핀이 모두 연결이 되는 구조입니다.

이번 실습은 Tact 스위치로 입력을 받고, led를 on/off하는 예제입니다. 플로팅현상을 방지하기 위해 스위치에 풀다운 저항을 연결했습니다.



준비물	
아두이노 우노	1대
브레드보드	1개
저항 220Ω	1개
저항 10kΩ	1개
Tact 스위치	1개
점퍼와이어	10가닥 내외

예제소스

```
/*
switch test
switch=8
led=7
http://www.devicemart.co.kr/
*/
int sw= 8; // 스위치를 digital 핀 8번에 연결
int led= 7; // led를 digital 핀 7번에 연결

void setup()
{
  pinMode(sw, INPUT); // digital 핀 8번을 입력으로 설정
  pinMode(led, OUTPUT); // digital 핀 7번을 출력으로 설정
}

void loop()
{
  if(digitalRead(sw) == 0) // 8번핀의 입력 값을 읽어서 입력값이 0이면
  {
    digitalWrite(led, LOW); // led off
  }
  else // 8번핀의 입력 값을 읽어서 입력값이 1이면
  {
    digitalWrite(led, HIGH); // led on
  }
  delay(100); // 시간 지연 0.1초
}
```

digitalRead()

digitalRead(pin)

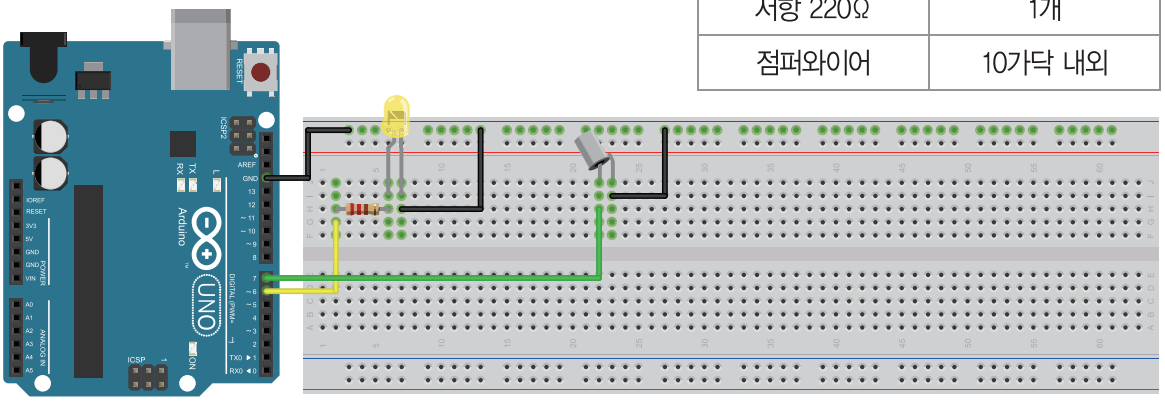
위와 같이 사용합니다. pin에는 값을 불러올 디지털핀 번호를 기입합니다. 핀모드를 통해 입력으로 설정한 핀의 값을 HIGH인지 LOW인지 읽어와서 1 또는 0으로 반환하는 함수입니다. 이 값을 변수에 저장하거나 다른 값과 비교할 수 있습니다.

3.3.2. 볼 스위치

볼(ball)스위치는 말그대로 안에 작은 공(ball)이 들어있어서 평소에는 연결(on)되어 있다가, 기울이게되면 공이 굴러 가게 되서 스위치 off가 되는 제품입니다.

기울임을 감지하거나 진동, 충격 감지의 용도로 사용됩니다.

준비물	
아두이노 우노	1대
브레드보드	1개
볼 스위치	1개
LED	1개
저항 220Ω	1개
점퍼와이어	10가닥 내외

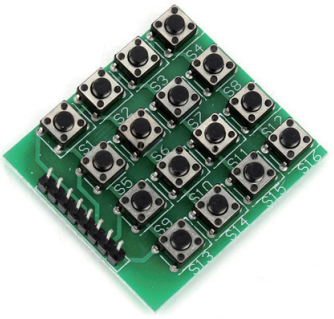


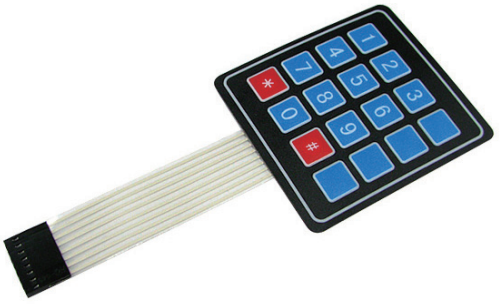
예제소스

```
/*
볼 스위치 값을 받아 led 제어
http://www.devicemart.co.kr/
*/

int led = 6; // D6에 led 연결
int ball = 7; // D7에 볼 스위치 연결

void setup()
{
  pinMode(led, OUTPUT); // led 출력 모드로 설정
  pinMode(ball, INPUT_PULLUP); // ball switch는 입력 모드설정, 내부 풀업저항 사용
}

void loop()
{
  if (digitalRead(ball)) // 평소에는 led off
  {
    digitalWrite(led, LOW);
  }
```

```
  else // ball switch가 기울어지면 led on
  {
    digitalWrite(led, HIGH);
  }
}
```

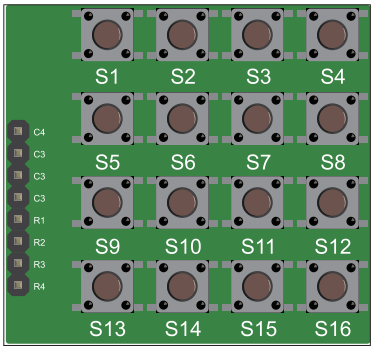
3.3.3. 4×4 키 패드

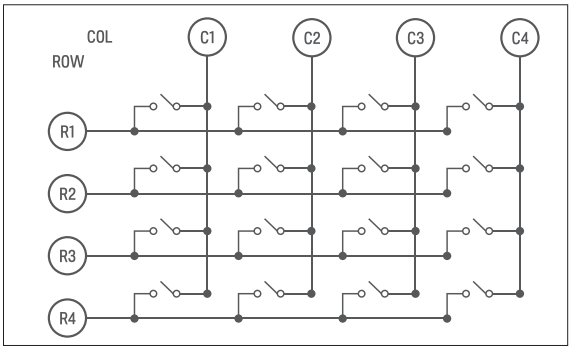




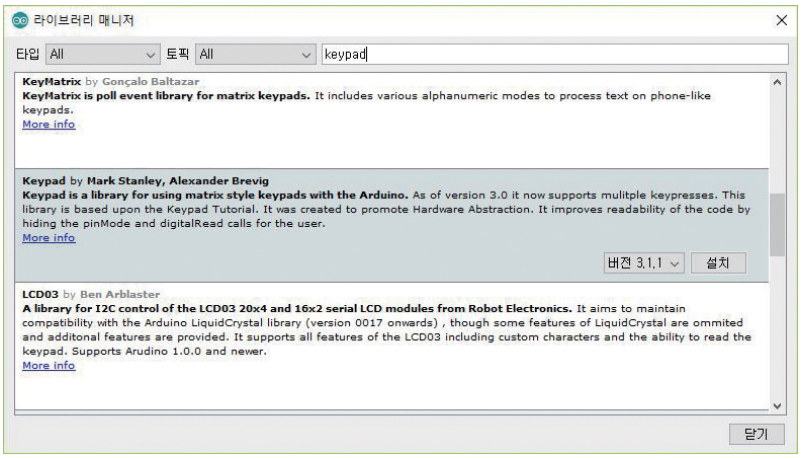
일상생활에서 현관문 또는 금고에서 비밀번호를 누를수 있는 도어락을 보시면 키패드가 내부에 장착되어 있습니다.

택트 스위치로 이루어진 키패드 혹은 위 사진과 같이 문자가 프린팅되어 있는 멤브레인 방식도 있습니다.



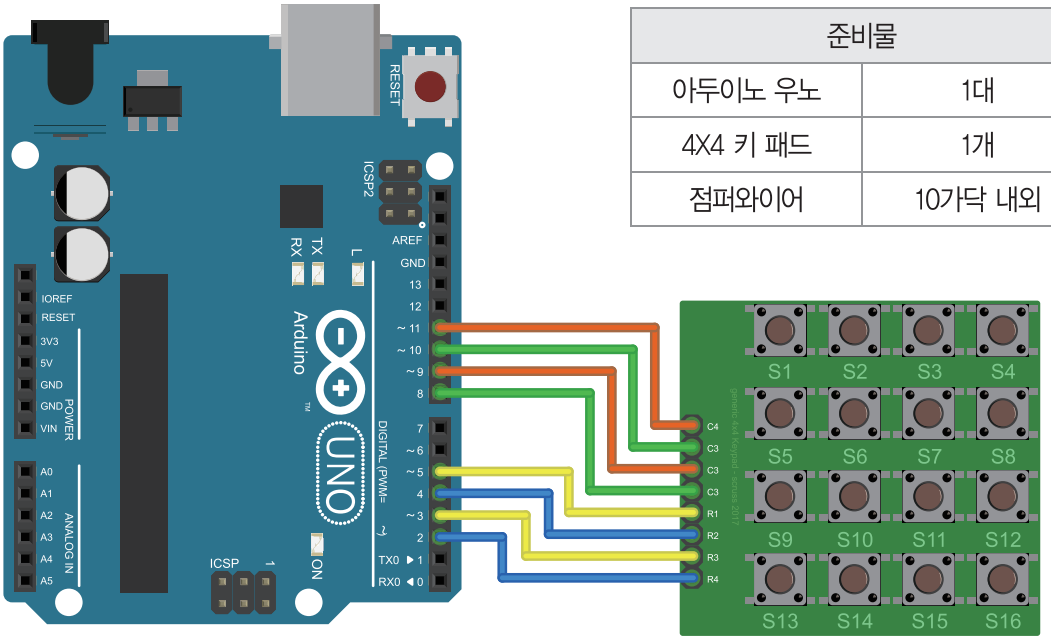


내부적으로는 16개의 스위치 회로가 행열로 배치되어 있습니다. 위와 같은 형태로 배치되어 있습니다. 이전에 배웠던 도트매트릭스와 같은 구조입니다. 출력장치였던 LED 대신 입력장치인 스위치로 바뀐 구조입니다.



키패드를 조금 더 편하게 사용하기 위해서는 라이브러리가 필요합니다.
[아두이노 IDE – 툴 – 라이브러리 관리에서 keypad]를 검색하시고 다운로드합니다.

준비물	
아두이노 우노	1대
4X4 키 패드	1개
점퍼와이어	10가닥 내외

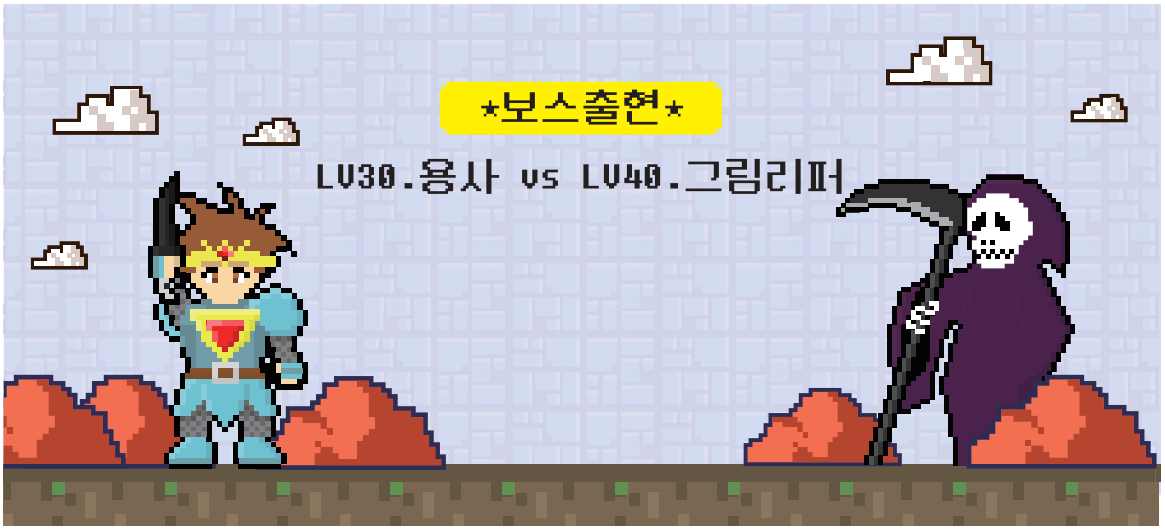


예제소스

```
/*
4x4 keypad를 사용해서 시리얼 모니터로 출력하는 예제
keypad 라이브러리가 필요합니다.
http://www.devicemart.co.kr/
*/

#include <Keypad.h> // keypad 라이브러리 포함
```

```
const byte ROWS = 4; // 행의 갯수
const byte COLS = 4; // 열의 갯수
char myKeys[ROWS][COLS] = // 배열로 각 버튼을 숫자나 문자로 대응
{
 {'0','1','2','3'},
 {'4','5','6','7'},
 {'8','9','A','B'},
 {'C','D','E','F'}
};

byte rowPins[ROWS] = {5, 4, 3, 2}; // 행(Raw)가 연결된 아두이노 핀 번호
byte colPins[COLS] = {8, 9, 10, 11}; // 열(column)가 연결된 아두이노 핀 번호

// keypad 객체 인스턴스화 Keypad(makeKeymap(userKeymap), row[], col[], rows, cols)
Keypad customKeypad = Keypad( makeKeymap(myKeys), rowPins, colPins, ROWS, COLS);

void setup()
{
  Serial.begin(9600); // 통신속도 9600으로 시리얼 통신 시작
}
void loop()
{
  // getKey()는 키패드에서 입력된 값이 있을경우 반환합니다.
  char customKey = customKeypad.getKey();
  if(customKey)  // 입력된 값이 있으면, 출력
  {
    Serial.println(customKey);
  }
}
```

Quest 9. Boss : 사천왕 제 3석 그림리퍼



하멜린

스~켈켈켈켈~ 나약한 하멜린 녀석... 당해버렸군.
너도 곧 하멜린이 있는 곳으로 보내주마!

용사

비켜주세요!!

BOSS 문제

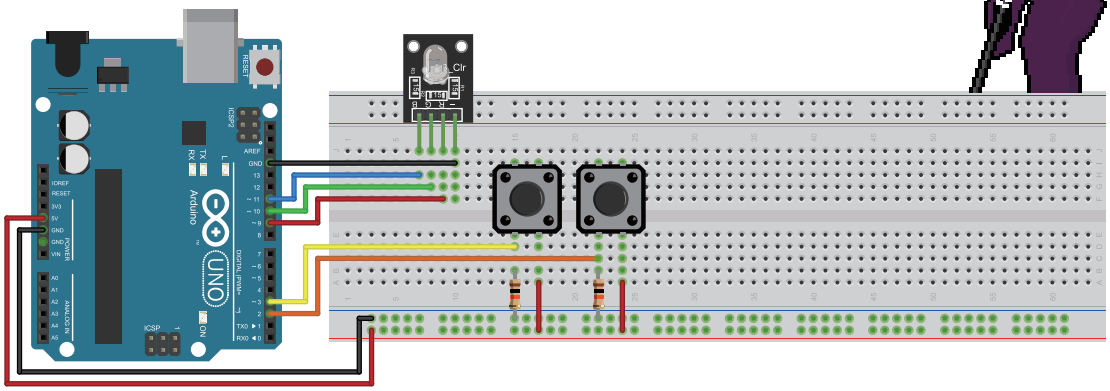
스위치 두개와 RGB LED 모듈로 구성된 회로를 구성합니다.
스위치1을 누르면 누를 때마다 RGB LED가 '빨강 노랑 초록 하늘 파랑 자주' 순서로 변합니다.
스위치2를 누르면 누를 때마다 RGB LED가 '자주 파랑 하늘 초록 노랑 빨강' 순서로 변합니다.
위와 같이 동작하는 회로를 구성하고 프로그래밍하시오.

준비물	
아두이노 우노	1대
브레드보드	1개
RGB LED 모듈	1개
저항 10kΩ	2개
택트 스위치	2개
점퍼와이어	10가닥 내외

Quest 9. 정답 체크

정상적으로 구동되었다면 그림리퍼에 X표해주세요.

정답예제회로



RGB LED		
아두이노	RGB 모듈	비고
GND	–	Ground
D9	R	빨강
D10	G	초록
D11	B	파랑

정답예제소스

```
int LED1 = 9; // Red를 9번에 연결
int LED2 = 10; // Green을 핀 10번에 연결
int LED3 = 11; // Blue를 핀 11번에 연결
int sw1 = 3; // 스위치1을 digital 핀 3번에 연결
int sw2 = 2; // 스위치2를 digital 핀 2번에 연결
int key_count = 0; // 스위치 눌린 횟수 초기화

void setup()
{
  pinMode(LED1, OUTPUT); // LED 출력 설정
  pinMode(LED2, OUTPUT); // LED 출력 설정
  pinMode(LED3, OUTPUT); // LED 출력 설정
  pinMode(sw1, INPUT); // 스위치 입력 설정
  pinMode(sw2, INPUT); // 스위치 입력 설정
```

```
  digitalWrite(LED1, HIGH); // 빨강LED 켜짐으로 초기설정
  digitalWrite(LED2, LOW); // 초록LED 꺼짐으로 초기설정
  digitalWrite(LED3, LOW); // 파랑LED 꺼짐으로 초기설정
}
void loop()
{
  if(digitalRead(sw1) == 1) // 스위치1의 입력 값을 읽어서 입력값이 1이면
  {
    key_count++; // 키 카운트 1 증가
  }
  if(digitalRead(sw2) == 1) // 스위치2의 입력 값을 읽어서 입력값이 1이면
  {
    key_count--; // 키 카운트 1 감소
  }
  if(key_count < 0) key_count = 5; // 스위치 눌린 횟수가 0보다 작을 때 5로
  if(key_count > 5) key_count = 0; // 스위치 눌린 횟수가 5보다 클 때 0으로
  rainbow(key_count); // 해당 키 카운트에 해당하는 led를 켜는 사용자 함수
  delay(200);                       // 짧은 시간지연
}

void rainbow(int kc)
{
  if(kc == 0) // 빨강
  {
    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, LOW);
  }
  else if(kc == 1) // 노랑
  {
    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, HIGH);
    digitalWrite(LED3, LOW);
  }
  else if(kc == 2) // 초록
  {
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, HIGH);
```

```
    digitalWrite(LED3, LOW);
  }
 else if(kc == 3) // 하늘
  {
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, HIGH);
    digitalWrite(LED3, HIGH);
  }
  else if(kc == 4) // 파랑
  {
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, HIGH);
  }
  else if(kc == 5) // 자주
  {
    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, HIGH);
  }
}
```

※회로와 코드가 정답예시와 동일할 필요는 없습니다. 문제의 동작과 동일하게만 동작한다면 정답입니다.

틀렸다면 정답예제 실습 후
스테이지 3.3 부터 다시!

모두 맞추셨다면 다음페이지로!

용사는 『디지털리드 메소드』를 얻었다.

용사
디지털 입력과 디지털 출력은 함께 사용하면 유용하네!

스테이지 3.4
포텐셔미터

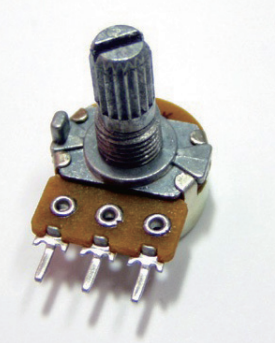
EXP♥

3. 아두이노 실전 연습

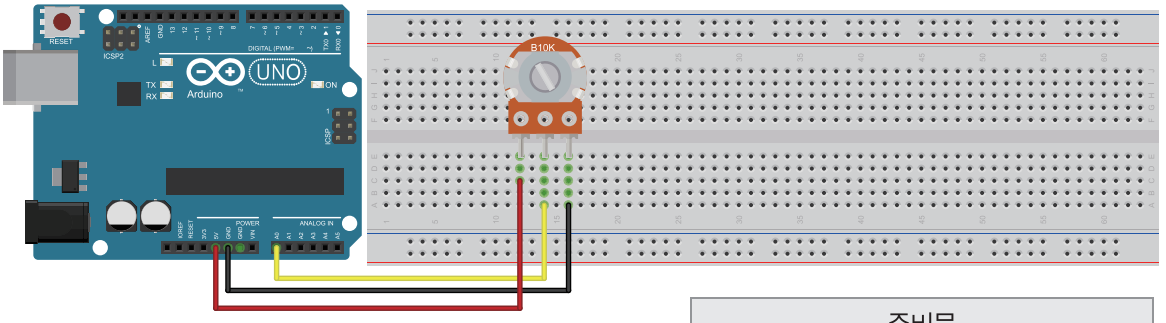
3.4. 포텐셔미터

3.4.1. 가변저항

가변저항(Potentiometer)은 전자회로에서 저항값을 임의로 바꿀 수 있는 저항입니다. 내부에 저항이 들어있고, 핀이 3개로 나뉘어져 있습니다. 저항은 두께와 길이에 비례하기 때문에 양쪽 가변저항의 노브를 돌리게 되면, 1–2번 사이의 저항값, 2–3번 사이의 저항값을 변화 시킬 수 있습니다. 또한 1–3번의 저항값을 측정하면 전체 가변 저항의 값을 알 수 있습니다.



이번 실습은 가변저항의 값을 아날로그 입력으로 아두이노로 받은 후 저항값으로 변환해서 시리얼 모니터로 출력하는 실습입니다.



준비물	
아두이노 우노	1대
브레드보드	1개
가변저항 10kΩ	1개
점퍼와이어	10가닥 내외

예제소스

```
/*
potentiometer_value
가변저항으로 값을 시리얼 모니터로 출력하는 예제
http://www.devicemart.co.kr/
*/
```

```
#define max_R 10000 // 가변저항의 총 저항값 정의

void setup()
{
  Serial.begin(9600); // 9600bps로 시리얼 통신 시작
  Serial.println("Serial Com start"); // 시리얼 통신 시작 문자 출력
}

void loop()
{
  int val=analogRead(A0); // 아날로그 A0에서 값을 읽어서 변수 val에 저장, 전압에 따라 0 ~ 1023
의 범위
  int resistor=map(val,0,1023,0,max_R); // map함수를 사용해서 아날로그값을 저항값으로 변환

   Serial.print("Resistor value:");
   Serial.print(resistor); // 저항값 출력
   Serial.println("ohm");
   delay(200);
}
```

#define

#define 치환할_이름 치환할_값

원하는 이름으로 값을 정의할 수 있는 전처리기입니다. 치환할 이름에 원하는 이름을 넣고 치환할 값에는 원하는 값을 넣습니다.

이번 실습에서는 10kΩ 가변저항을 사용하기 때문에 10,000이라는 값을 max_R로 선언했습니다.

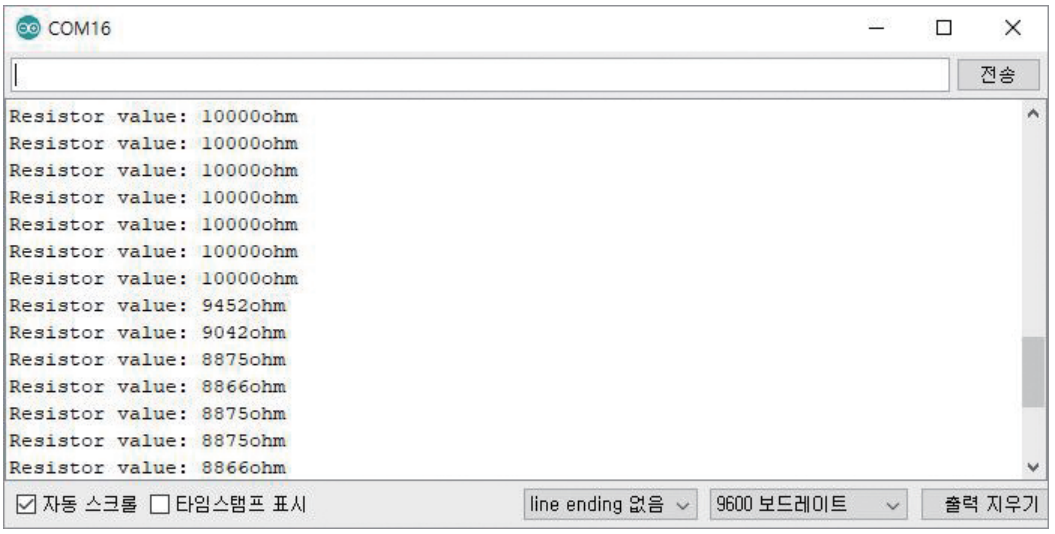
map()

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

특정 범위의 숫자를 다른 범위의 숫자로 변환하는 함수입니다.

예를들어 0~10의 숫자 중 3이라는 숫자를 100~200이라는 범위의 숫자로 변환하면 130이 됩니다.

value는 변환하고 싶은 값(위 예시의 3), fromLow와 fromHigh는 value가 속한 최소/최대 범위(위 예시의 0~10), toLow와 toHigh는 변환하고 싶은 최소/최대 범위(위 예시의 100~200)로 기입하여 사용하시면 됩니다.



실행 결과입니다.

정상적으로 구동된다면 0.2초 간격으로 가변저항에 인가된 전압을 입력받아서 현재 얼마만큼의 저항값으로 조절되어있는지 시리얼모니터로 출력해줍니다.

용사

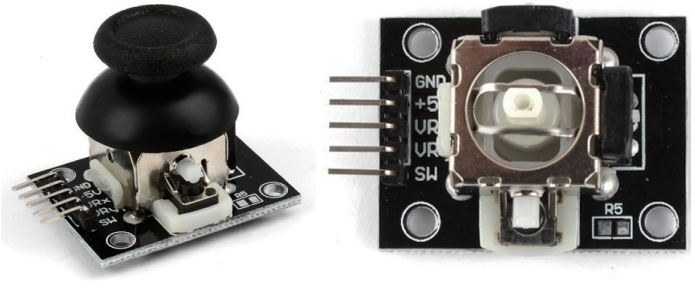
전압으로 저항값을 알 수 있다니 유용하네!

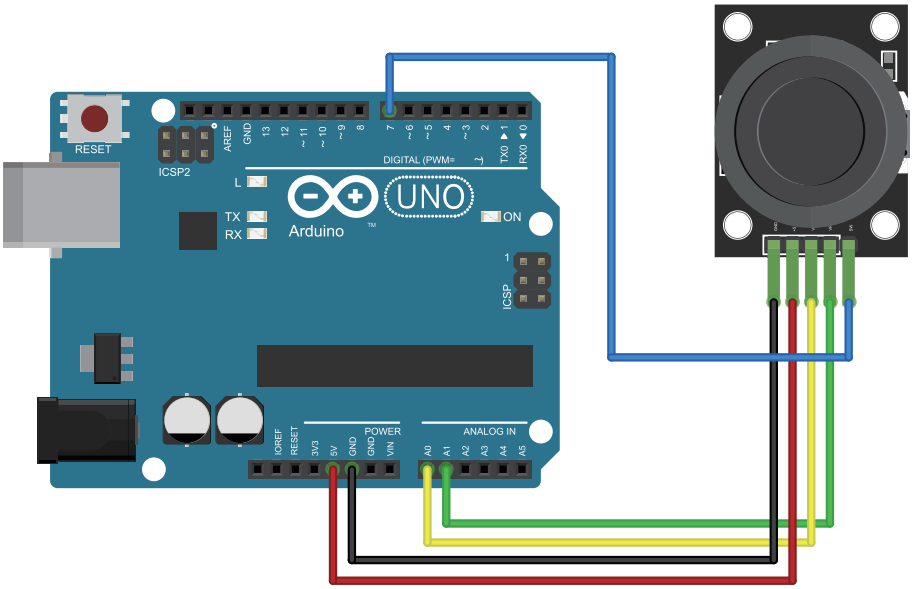
3.4.2. 조이스틱

RC카 조종기, 게임기 등에서 쓰이는 조이스틱 모듈입니다.

내부에 가변저항이 있어서, x축과 y 축의 데이터를 읽어올 수 있습니다. 모듈에 따라 푸시 스위치가 있을 수도 있는데 푸시 스위치 내장형은 디지털 신호의 푸시스위치 입력도 받을 수 있습니다.

조이스틱의 변화하는 값을 시리얼 모니터로 확인할 수 있는 예제입니다.





배선		
아두이노	조이스틱 모듈	비고
GND	GND	Ground
5V	+5V	VCC
A0	VRX	X축
A1	VRY	Y축
D7	SW	푸쉬스위치

준비물	
아두이노 우노	1대
조이스틱 모듈	1개
점퍼와이어	10가닥 내외

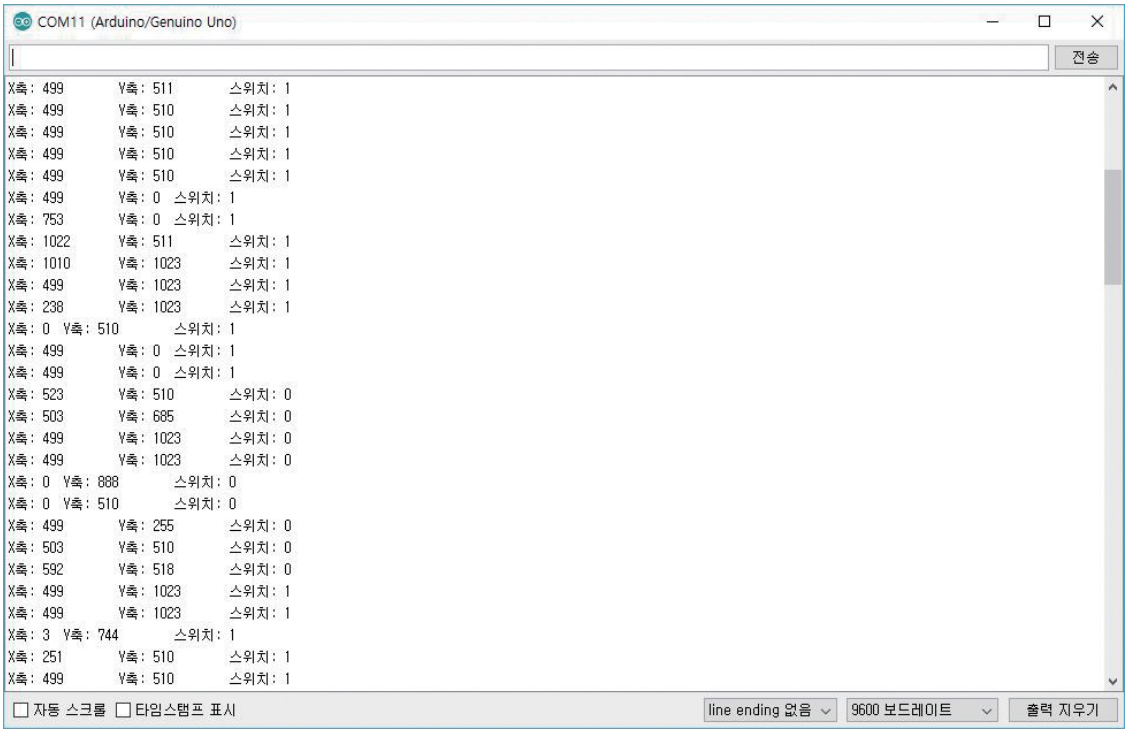
예제소스

```
/*
joystick 값을 시리얼 모니터로 출력하는 예제
http://www.devicemart.co.kr/
*/
int sw=7; // 조이스틱 푸쉬 스위치를 D7번에 연결

void setup()
{
  Serial.begin(9600); // 통신속도 9600bps로 시리얼 통신 시작
  pinMode(sw, INPUT_PULLUP); // 스위치를 입력으로 세팅하고 내부 풀업저항을 사용함
  delay(1000); // 1초 후에 시작
}
void loop()
{
```

```
  Serial.print("X축:");
  Serial.print(analogRead(A0)); // X축 값 출력
  Serial.print("\t");
  Serial.print("Y축:");
  Serial.print(analogRead(A1)); // Y축 값 출력
  Serial.print("\t");
  Serial.print("스위치:");
  Serial.println(digitalRead(sw)); // 스위치 값 출력
  delay(100);
}
```

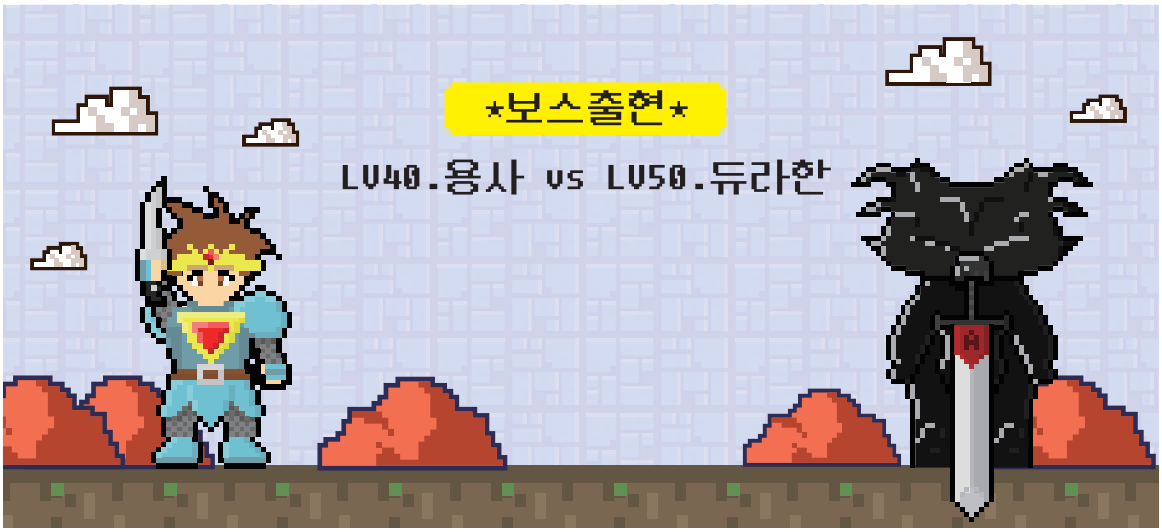
\t

이스케이프 시퀀스(Escape sequence)라고 하는 특수한 문자형 명령어 중 하나입니다.
키보드의 'Tab'키를 눌렀을때 생기는 공백이 입력됩니다.
println에서 언급했듯이 "\n"도 이스케이프 시퀀스이며 출력시 줄바꿈이 됩니다.



정상적으로 구동되었다면 0.1초마다 X축과 Y축의 ADC값과 스위치의 눌림 여부가 시리얼 모니터로 출력됩니다.

Quest 10. Boss : 사천왕 제 2석 듀라한



듀라한
용사여, 그대의 실력을 시험해 보겠소.

용사
얼굴이 없는데 목소리가 어디서 나오는거에요??

BOSS 문제

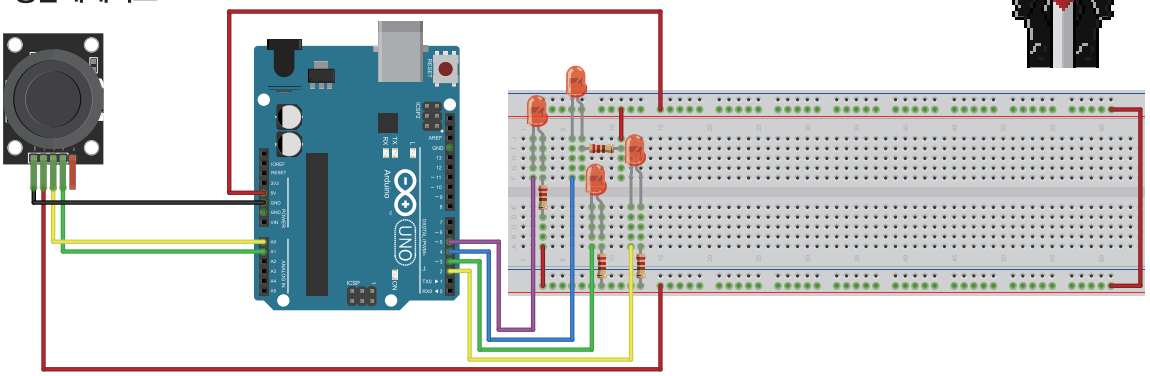
LED 4개를 ❖모양으로 배치합니다.
조이스틱을 기울이면 기울인 방향에 해당하는 LED가 들어오도록 회로를 구성하여 구동하시오.

준비물	
아두이노 우노	1대
브레드보드	1개
저항 220Ω	4개
LED	4개
조이스틱 모듈	1개
점퍼와이어	10가닥 내외

Quest 10. 정답 체크

정상적으로 구동되었다면 듀라한에 X표해주세요.

정답예제회로



정답예제소스

```
int LED_R = 2; // 오른쪽 LED를 D12에 연결
int LED_D = 3; // 아래쪽 LED를 D11에 연결
int LED_U = 4; // 위쪽 LED을 D10에 연결
int LED_L = 5; // 왼쪽 LED을 D10에 연결

void setup()
{
  pinMode(LED_R, OUTPUT); // 오른쪽 LED 출력 설정
  pinMode(LED_D, OUTPUT); // 아래쪽 LED 출력 설정
  pinMode(LED_U, OUTPUT); // 위쪽 LED 출력 설정
  pinMode(LED_L, OUTPUT); // 왼쪽 LED 출력 설정

  digitalWrite(LED_R, HIGH); // 오른쪽 LED 꺼짐으로 초기설정
  digitalWrite(LED_D, HIGH); // 아래쪽 LED 꺼짐으로 초기설정
  digitalWrite(LED_U, HIGH); // 위쪽 LED 꺼짐으로 초기설정
  digitalWrite(LED_L, HIGH); // 왼쪽 LED 꺼짐으로 초기설정
}

void loop()
{
  int val_ud = analogRead(A0); // 조이스틱 상하 값 변수에 저장
  int val_rl = analogRead(A1); // 조이스틱 좌우 값 변수에 저장
  if(val_ud < 256)
  {
    digitalWrite(LED_U, LOW); // 위쪽 LED 켜짐
    digitalWrite(LED_D, HIGH); // 아래쪽 LED 꺼짐
```

```
  }
  else if(val_ud > 768)
  {
    digitalWrite(LED_U, HIGH); // 위쪽 LED 꺼짐
    digitalWrite(LED_D, LOW); // 아래쪽 LED 켜짐
  }
else
  {
    digitalWrite(LED_U, HIGH); // 위쪽 LED 꺼짐
    digitalWrite(LED_D, HIGH); // 아래쪽 LED 꺼짐
  }

  if(val_rl < 256)
  {
    digitalWrite(LED_R, LOW); // 오른쪽 LED 켜짐
    digitalWrite(LED_L, HIGH); // 왼쪽 LED 꺼짐
  }
  else if(val_rl > 768)
  {
    digitalWrite(LED_R, HIGH); // 오른쪽 LED 꺼짐
    digitalWrite(LED_L, LOW); // 왼쪽 LED 켜짐
  }
  else
  {
    digitalWrite(LED_R, HIGH); // 오른쪽 LED 꺼짐
    digitalWrite(LED_L, HIGH); // 왼쪽 LED 꺼짐
  }
  delay(200);                    // 짧은 시간지연
}
```

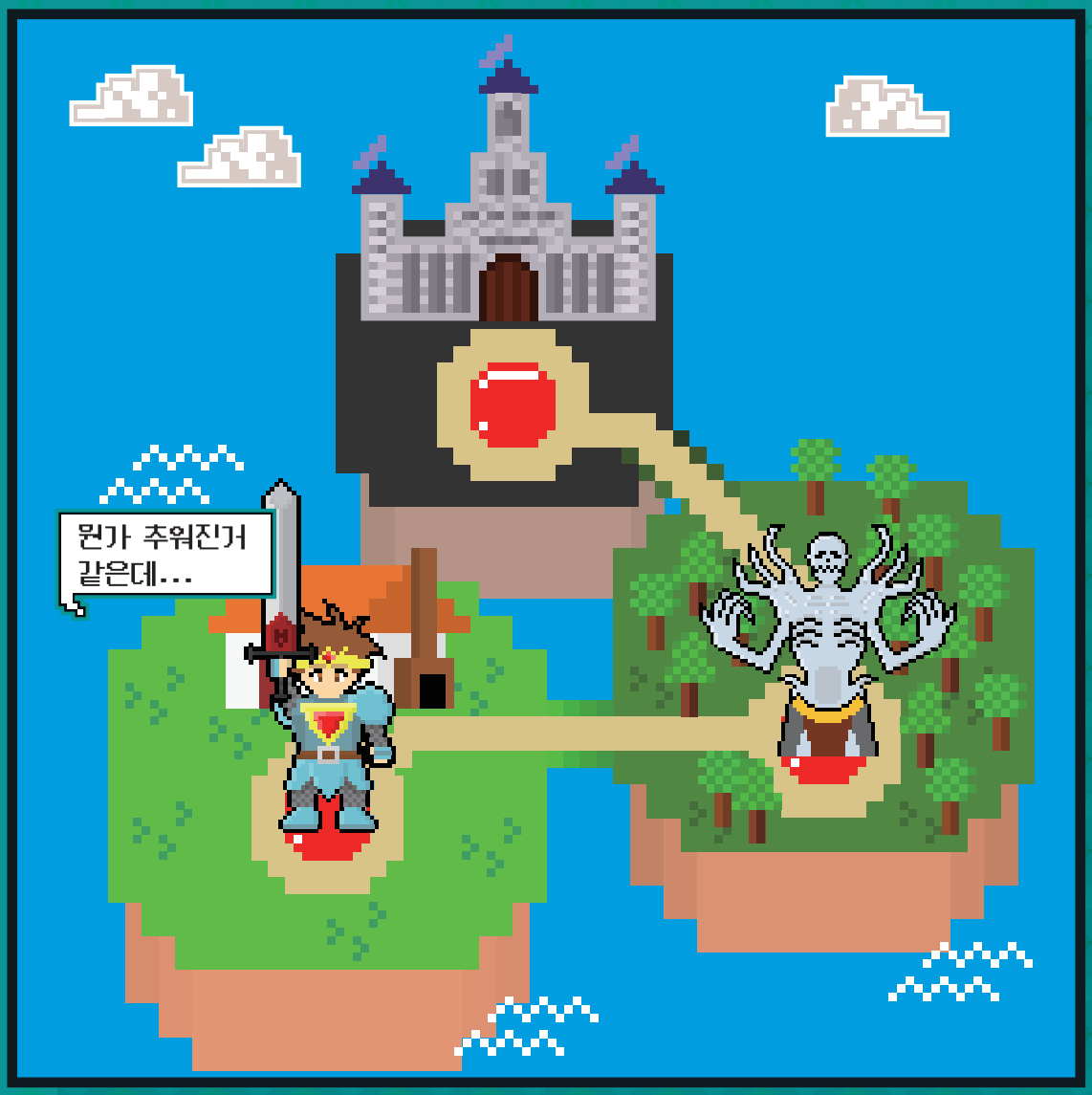
※회로와 코드가 정답예시와 동일할 필요는 없습니다. 문제의 동작과 동일하게만 동작한다면 정답입니다.

틀렸다면 정답예제 실습 후
스테이지 3.4 부터 다시!

모두 맞추셨다면 다음페이지로!

용사는 『아날로그리드 메소드』를 얻었다.

스테이지 3.5
온도

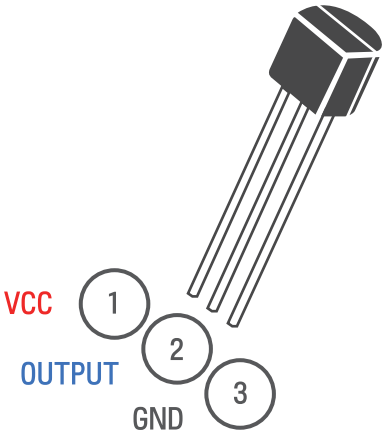


EXP♥

3. 아두이노 실전 연습

3.5. 온도

3.5.1. 온도센서



간단한 온도센서 중 하나인 LM35입니다.

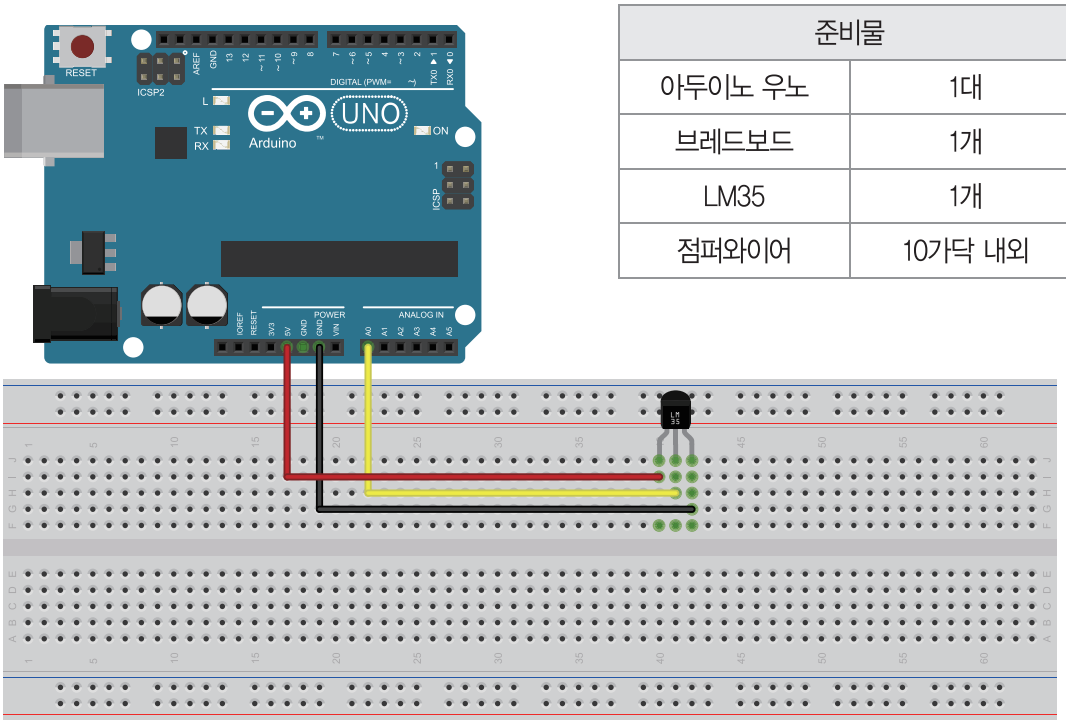
전원, 출력, 그라운드로 이루어진 3핀의 다리가 있습니다.

소자를 보시면 평평한 면과 볼록한 면이 있는데 평평한 면을 기준으로 왼쪽이 1번(VCC)입니다.

온도에 따라 출력 전압이 달라지는 소자입니다. 0℃일 때 0V를 출력하고 1℃ 올라갈 때마다 10mV씩 전압이 증가합니다. 이를 아래와 같이 공식으로 바꿀 수 있습니다.

온도(℃) = (5.0 × LM35 출력값 × 100) / 1024.0

위 공식을 사용해서 1초마다 섭씨온도값을 구하는 실습을 해보겠습니다.

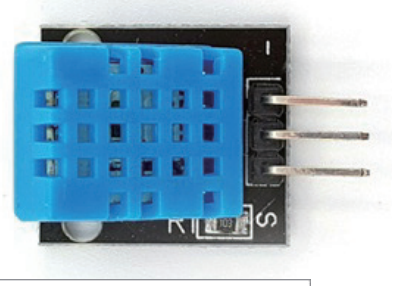


준비물	
아두이노 우노	1대
브레드보드	1개
LM35	1개
점퍼와이어	10가닥 내외

예제소스

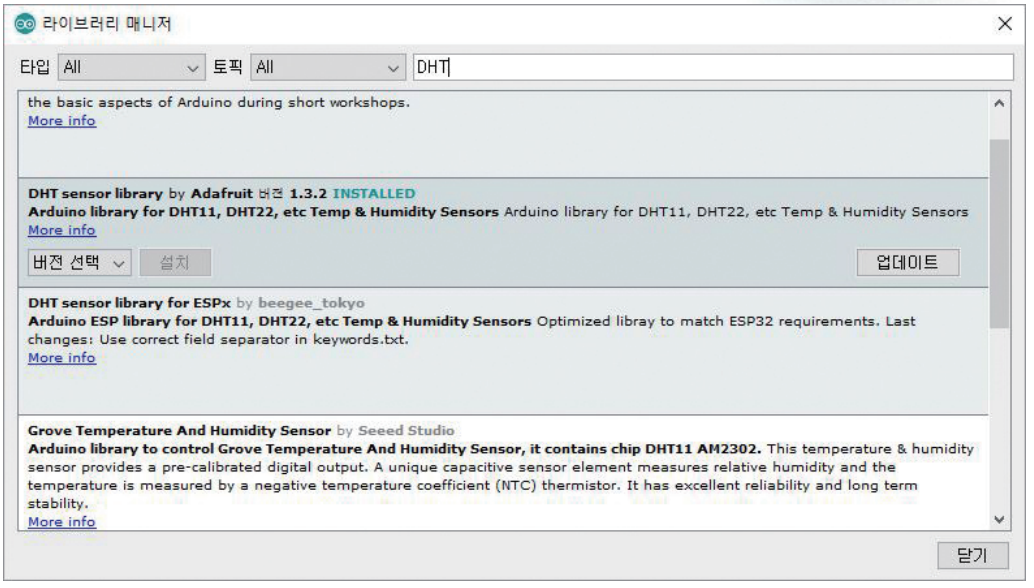
```
/*
LM35 온도센서 출력 예제
http://www.devicemart.co.kr/
*/
void setup()
{
  Serial.begin(9600);  // 통신속도 9600으로 통신 시작
}
void loop()
{
  int temp = analogRead(A0);  // 온도 값을 읽고 변수에 저장
  int value = 5.0*temp*100.0/1024; // 센서값 섭씨온도로 변환 (LM35 계산 공식에 의함)
  Serial.print(value);
  Serial.println("℃");
  delay(1000);
}
```

3.5.2. 온습도센서

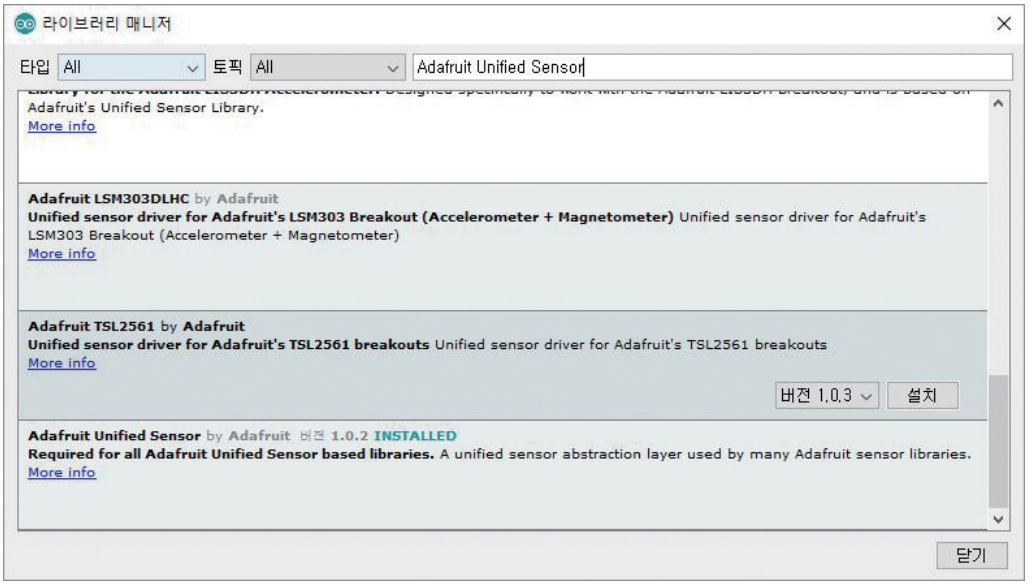


아두이노에서 온습도 관련 프로젝트를 할 때 많이 사용되는 DHT11 온습도 센서모듈입니다.

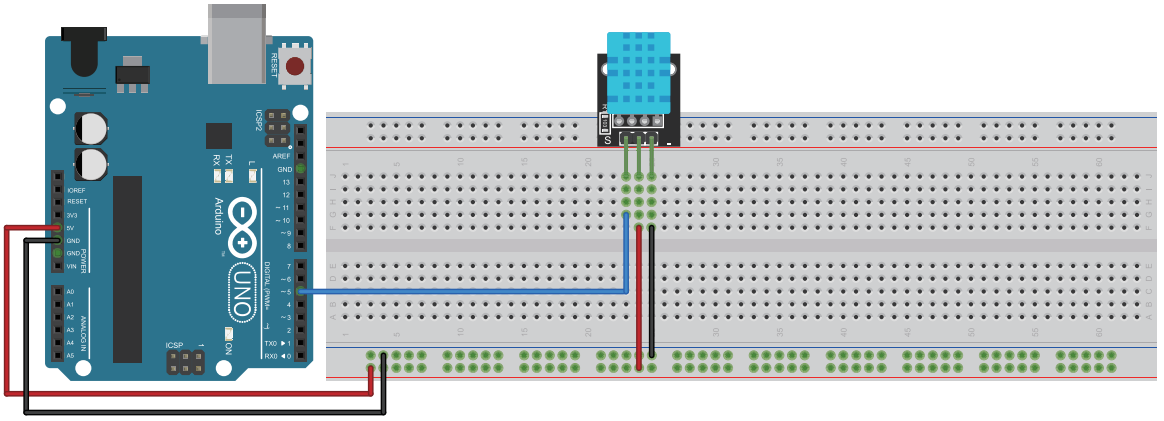
라이브러리를 사용해서 온도값과 습도값 둘 다 얻을 수 있는 센서 모듈입니다. 통신 라이브러리와 센서 라이브러리, 두 개의 라이브러리를 사용합니다.



[메뉴바 – 툴 – 라이브러리 관리]에서 DHT sensor library by Adafruit를 다운로드해 주세요.



[메뉴바 – 툴 – 라이브러리 관리]에서 Adafruit Unified Sensor by Adafruit를 다운로드해 주세요.
DHT11 온습도 센서를 사용하여 온도와 습도를 시리얼 모니터로 출력 받아보는 실습입니다.

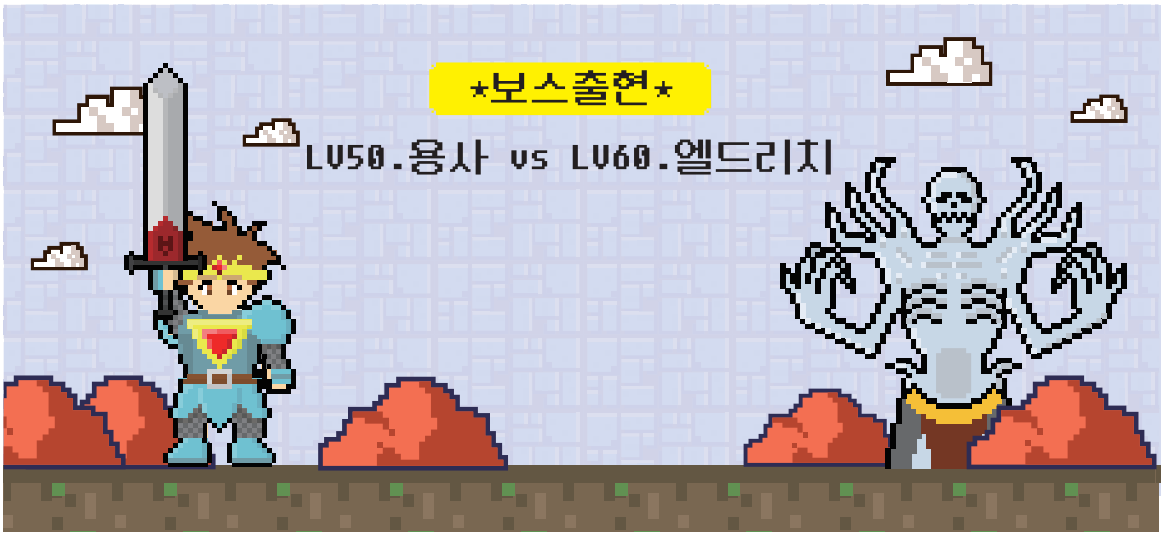


준비물	
아두이노 우노	1대
브레드보드	1개
DHT11	1개
점퍼와이어	10가닥 내외

예제소스

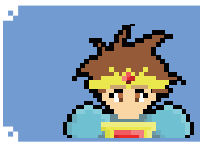
```
/*
온습도 센서 예제
http://www.devicemart.co.kr/
*/
#include "DHT.h" // DHT 라이브러리 호출

#define DHTPIN 5      // 온습도 센서가 5번에 연결
#define DHTTYPE DHT11    // DHT11 온습도 센서 사용

DHT dht(DHTPIN, DHTTYPE); // DHT 설정 (5,DHT11)

void setup()
{
  Serial.begin(9600); // 통신속도 9600으로 통신 시작
  Serial.println("DHT11 test!"); // 문자 출력
}

void loop()
{
  int h = dht.readHumidity(); // 습도값을 h에 저장
  int t = dht.readTemperature(); // 온도값을 t에 저장
  Serial.print("Humidity: "); // 문자열 출력
  Serial.print(h); // 습도값 출력
  Serial.print("% ");
  Serial.print("Temperature: ");
  Serial.print(t); // 온도값 출력
  Serial.println("℃");
  delay(2000);
}
```

Quest 11. Boss : 사천왕 제 1석 엘드리치



엘드리치

...어리석은 자여... 너의 시간은 곧 얼어붙을 것이다...



용사

으... 오싹하고 추워...

BOSS 문제

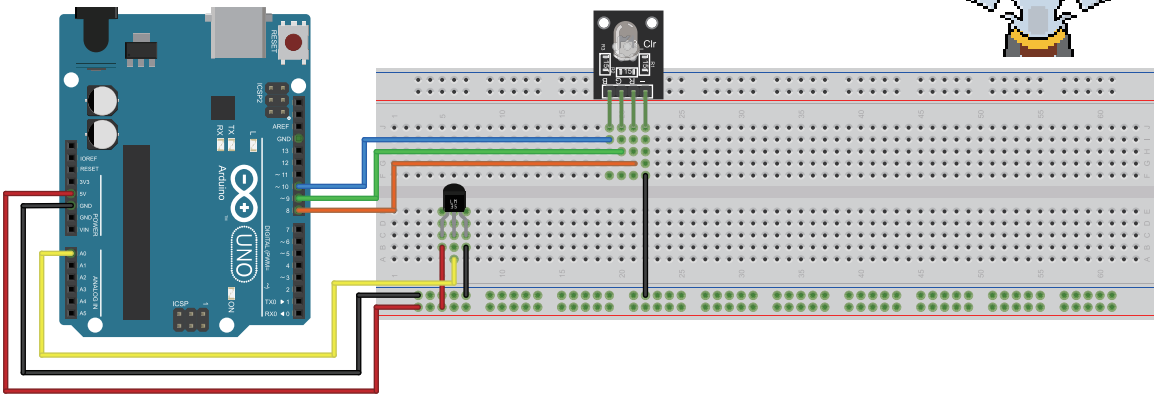
온도를 0.5초 간격으로 시리얼 모니터에 출력합니다.
30℃ 초과일 때 LED에 빨간불, 20℃ 미만일 때 LED에 파란불,
20~30℃일 때 초록불이 들어오게 회로를 구성하여 구동하시오.

준비물	
아두이노 우노	1대
브레드보드	1개
LM35	1개
RGB LED	1개
점퍼와이어	10가닥 내외

Quest 11. 정답 체크

정상적으로 구동되었다면 엘드리치에 X표 해주세요.

정답예제회로



정답예제소스

```
int LED1 = 8; // Red를 8번에 연결
int LED2 = 9; // Green을 핀 9번에 연결
int LED3 = 10; // Blue를 핀 10번에 연결

void setup()
{
  Serial.begin(9600);  // 통신속도 9600으로 통신 시작

  pinMode(LED1, OUTPUT); // LED 출력 설정
  pinMode(LED2, OUTPUT); // LED 출력 설정
  pinMode(LED3, OUTPUT); // LED 출력 설정

  digitalWrite(LED1, LOW); // 빨강LED 꺼짐으로 초기설정
  digitalWrite(LED2, LOW); // 초록LED 꺼짐으로 초기설정
  digitalWrite(LED3, LOW); // 파랑LED 꺼짐으로 초기설정
}

void loop()
{
  int temp = analogRead(A0); // 온도센서 값을 변수에 저장
  int cels = 5.0*temp*100.0/1024; // 섭씨온도로 변환하여 변수에 저장

  if(cels > 30) // 온도가 30도 초과일 때
```

```
  {
    digitalWrite(LED1, HIGH); // 빨강LED 켜짐으로 초기설정
    digitalWrite(LED2, LOW); // 초록LED 꺼짐으로 초기설정
    digitalWrite(LED3, LOW); // 파랑LED 꺼짐으로 초기설정
  }
  else if(cels < 20)  // 온도가 20도 미만일 때
  {
    digitalWrite(LED1, LOW); // 빨강LED 꺼짐으로 초기설정
    digitalWrite(LED2, LOW); // 초록LED 꺼짐으로 초기설정
    digitalWrite(LED3, HIGH); // 파랑LED 켜짐으로 초기설정
  }
else // 온도가 20~30도 사이일 때
  {
    digitalWrite(LED1, LOW); // 빨강LED 꺼짐으로 초기설정
    digitalWrite(LED2, HIGH); // 초록LED 켜짐으로 초기설정
    digitalWrite(LED3, LOW); // 파랑LED 꺼짐으로 초기설정
  }
  Serial.print(cels);
  Serial.println("℃");
  delay(500);
}
```

※회로와 코드가 정답예시와 동일할 필요는 없습니다. 문제의 동작과 동일하게만 동작한다면 정답입니다.

틀렸다면 정답예제 실습 후
스테이지 3.5부터 다시!

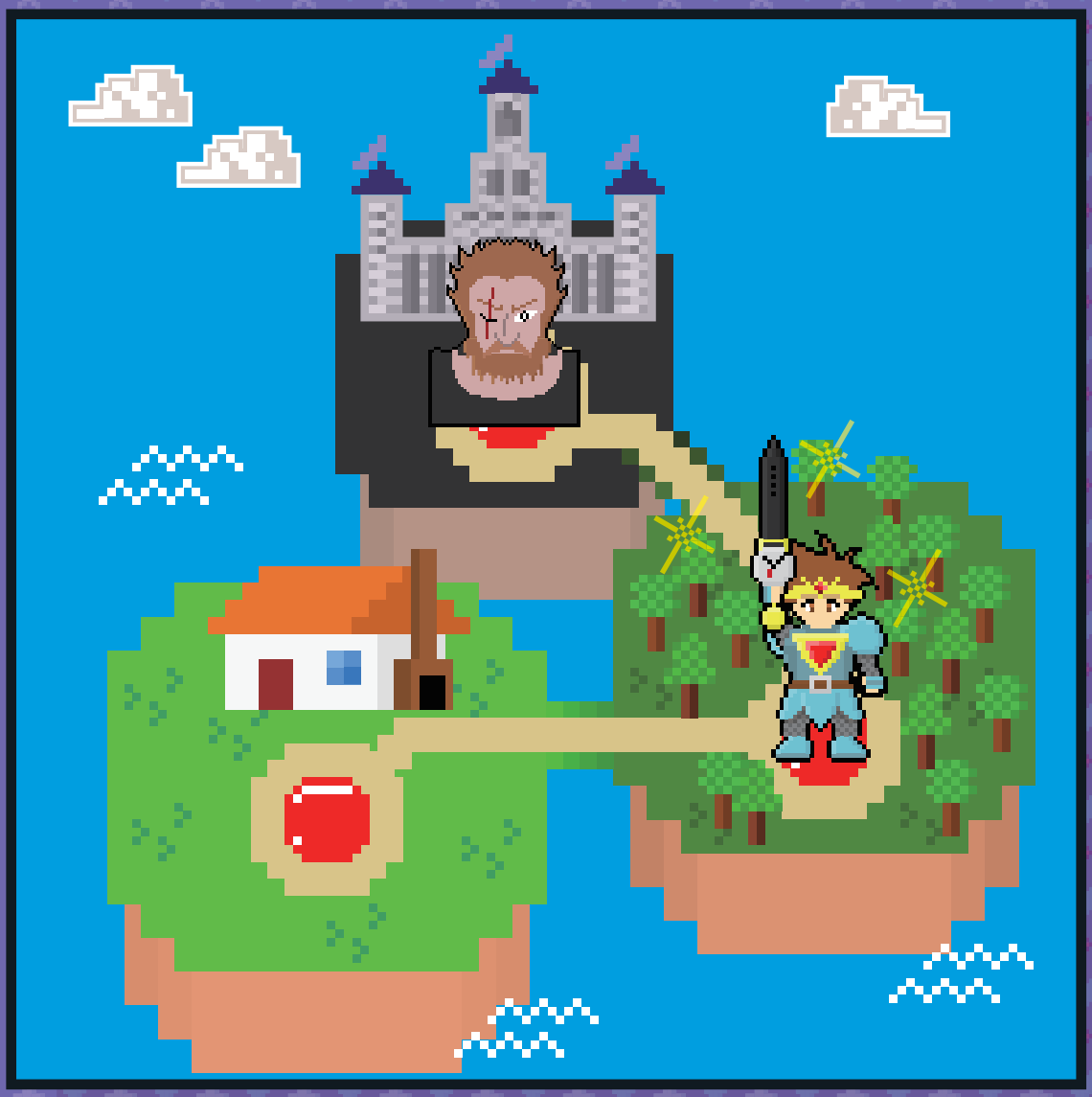
모두 맞추셨다면 다음페이지로!

용사는 『딜레이 메소드』를 얻었다.

용사
둠칫 둠칫 두둠칫 타이밍이 더 완벽해 진 것 같아!

스테이지 3.6

빛



EXP♥

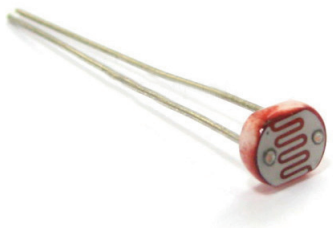
3. 아두이노 실전 연습

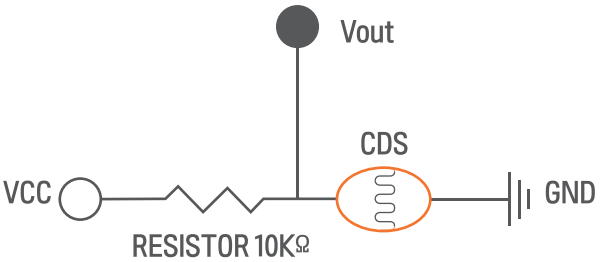
3.6. 빛

3.6.1. 조도센서

조도센서란 밝기(조도)를 측정할 수 있는 센서입니다. 조도센서는 여러 가지 종류가 있지만 여기서 다루게 되는 센서는 황화카드뮴(cds)을 소재로 만든 cds 센서입니다. cds 광도전셀 이라고도 합니다.

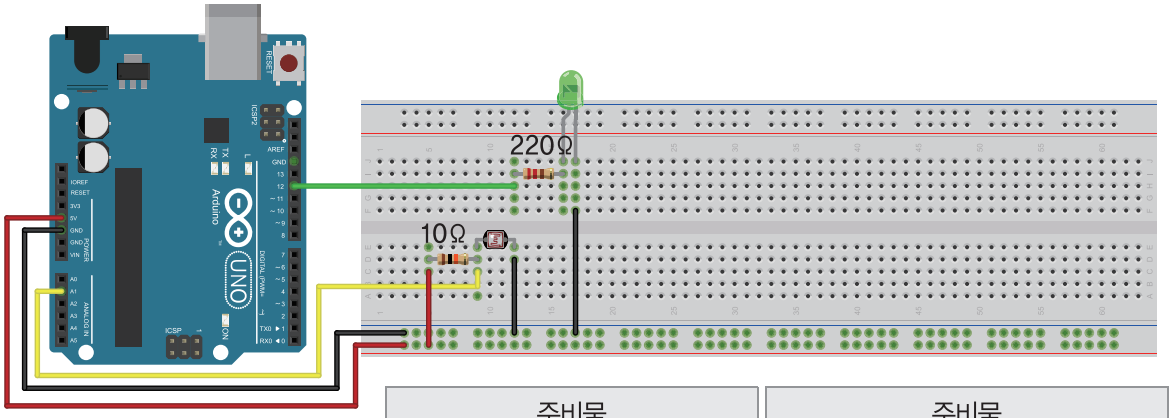
일상생활에서 날이 저물면 켜지는 가로등이나 자동차의 헤드라이트에 많이 쓰입니다. 조도센서는 극성이 없고, 빛의 양 따라 전도율이 달라집니다. 밝을수록 전도율이 높아져 저항이 낮아지고, 어두울수록 전도율이 낮아져서 저항이 높아집니다.





핀이 2개뿐이기 때문에 센서의 값을 받기 위해서는 저항을 이용해서 전압 분배를 해줘야 합니다. 조도센서 내부의 저항의 변화로 인해 Vout을 통해 아두이노로 입력되는 전압은 변하게 되고 이를 아날로그 입력 핀을 통해 값을 얻습니다.

조도센서와 LED를 사용한 실습입니다. 가로등처럼 어두워지면 LED의 불이 켜집니다. 센서의 사양에 따라 특정 전압일 때 밝기가 어느 정도인지 각각 다르기 때문에 ADC 중간값인 512를 기점으로 밝다 어둡다를 구분했습니다.



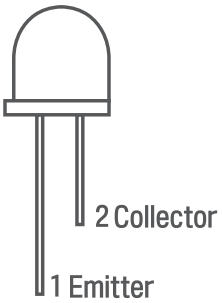
준비물		준비물	
아두이노 우노	1대	저항 10kΩ	1개
브레드보드	1개	LED	1개
조도센서	1개	점퍼와이어	10가닥 내외
저항 220Ω	1개		

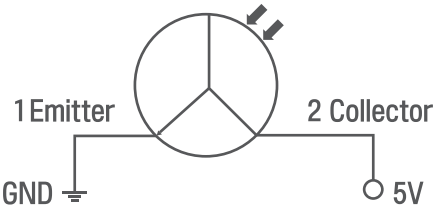
예제소스

```
/*
조도센서 LED 예제
어두우면(아날로그값 중간 이하) led의 불이 켜지는 예제
http://www.devicemart.co.kr/
*/

int led = 12;        // 12번핀 led를 변수 led에 대입

void setup()
{
  pinMode(led,OUTPUT);  // led를 출력모드로 설정
  Serial.begin(9600);  // 속도 9600으로 통신 시작
}

void loop()
{
  int val= analogRead(A1);  //  A1 아날로그 값을 읽어서 val에 저장
  Serial.println(val);   // val 값을 출력

  if(val<512)  // 아날로그 값이 중간 미만이라면 LED를 켜라
  {
    digitalWrite(led,HIGH);
  }
  else // 아날로그 값이 중간 이상이라면 LED를 꺼라
  {
    digitalWrite(led,LOW);
  }
  delay(100);
}
```

3.6.2. 불꽃감지 센서

불꽃 센서는 불꽃의 스펙트럼에 해당하는 파장에 반응하는 포토 트랜지스터입니다. 불꽃에서 나오는 빛(적외선)을 받으면 콜렉터(C)에서 에미터(E)로 전류가 흐르게 되어 출력값이 증가하게 되는 소자입니다.

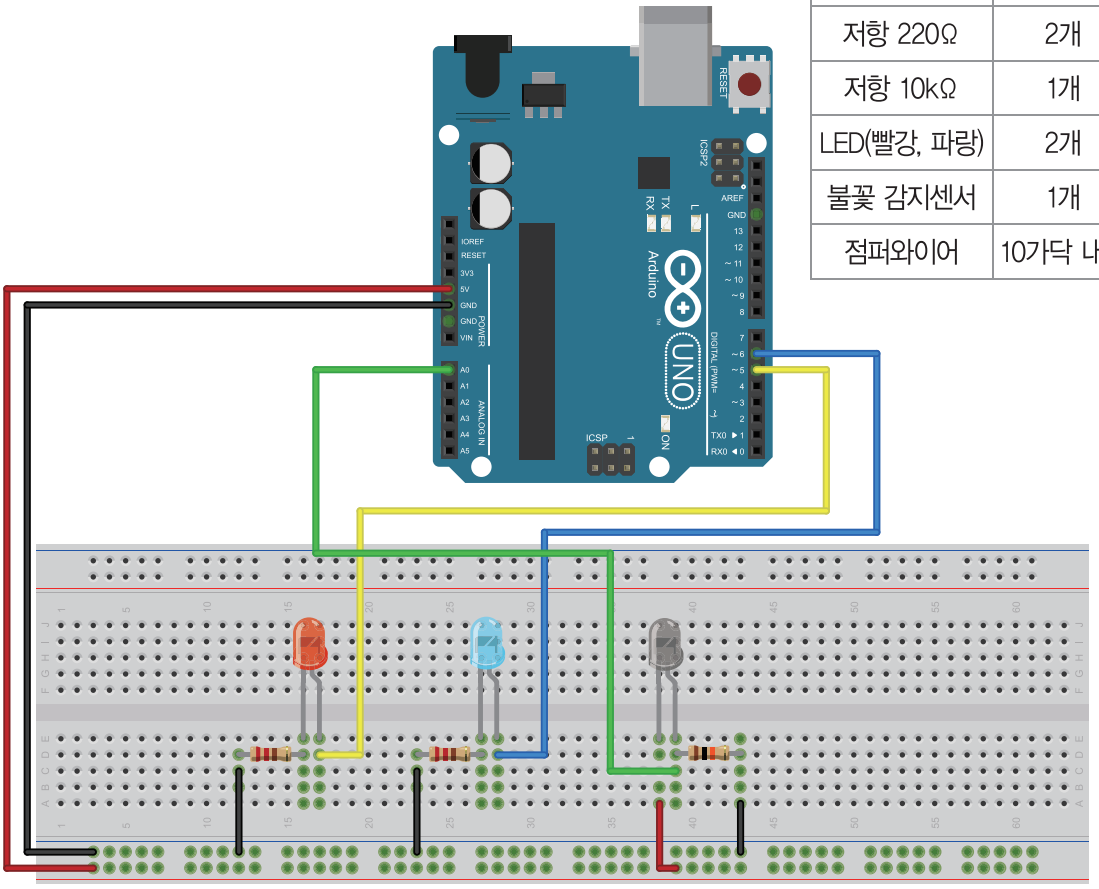




이번 실습은 화재경보기를 만들어봅시다.

평소에는 파란 LED가 켜져있다가 불꽃감지 센서가 불꽃을 감지하면 파란 LED는 꺼지고 빨간 LED가 켜집니다.

준비물	
아두이노 우노	1대
브레드보드	1개
저항 220Ω	2개
저항 10kΩ	1개
LED(빨강, 파랑)	2개
불꽃 감지센서	1개
점퍼와이어	10가닥 내외



예제소스

```
/*
불꽃센서로 테스트 예제
http://www.devicemart.co.kr/
*/

int led1 = 5;  // led1 5번에 연결(빨강 LED)
int led2 = 6;    // led2 6번에 연결(파랑 LED)
int val;

void setup()
{
  pinMode(led1, OUTPUT);   // led 출력설정
  pinMode(led2, OUTPUT);
  pinMode(flame,INPUT);    // 불꽃 감지센서 입력설정
  Serial.begin(9600);     // 통신속도 9600으로 통신 시작
}

void loop()
{
  val = analogRead(A0);  // 불꽃 감지센서에서 값을 읽음
  Serial.println(val);     // 불꽃 감지센서 입력값 출력
  // 불꽃 감지센서 입력값이 800이상이면 (불꽃을 감지하면)
  // led1 on , led2 off
  if(val >= 800) // 아날로그값 800은 조정 가능*
  {
    digitalWrite(led1,HIGT);
    digitalWrite(led2,LOW);
  }
  else // 불꽃 감지센서 입력값이 800이하이면 반대
  {
    digitalWrite(led1,LOW);;
    digitalWrite(led2,HIGH)
  }
}
```

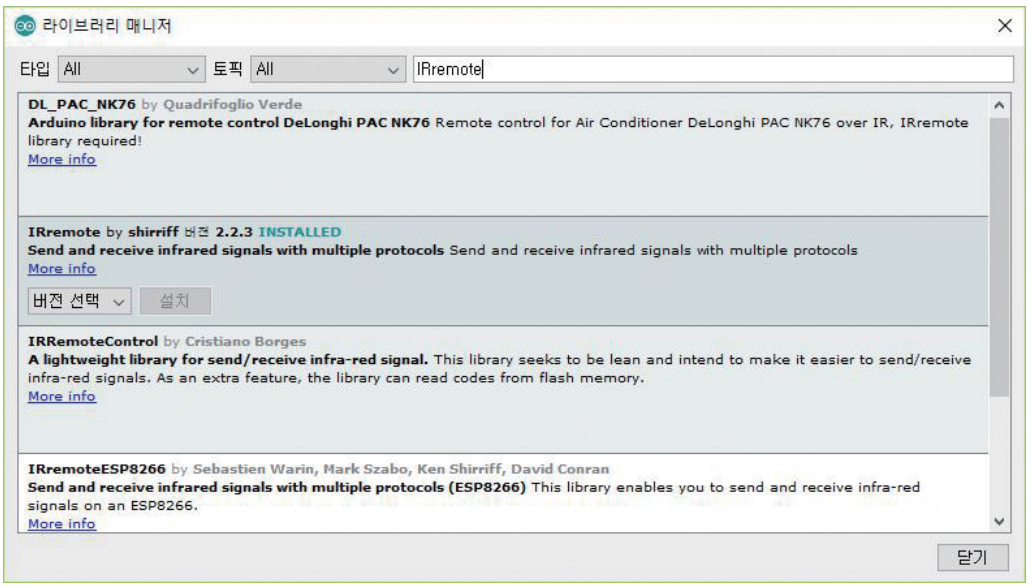
loop문 안에 있는 "if(val >= 800)"의 800이란 수치는 센서 감도에 따라 높거나 낮게 수정해주시면 됩니다.

3.6.3. IR 리모컨

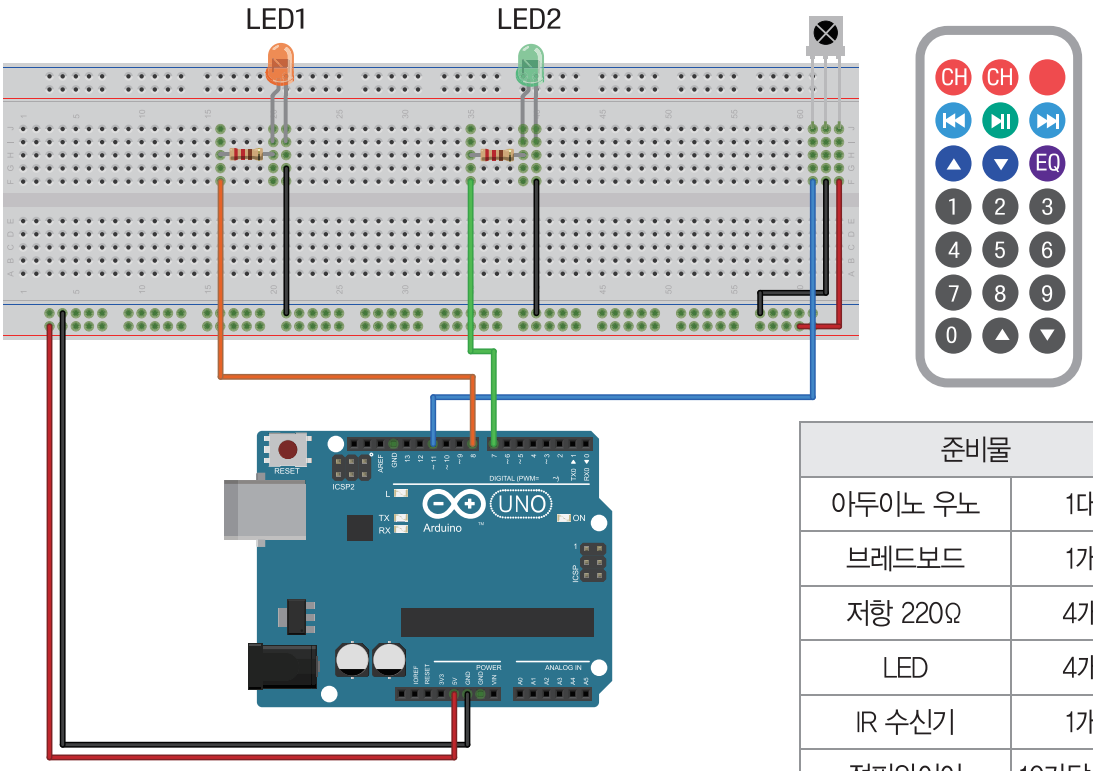
적외선 리모컨의 적외선(Infrared, IR)은 가시광선보다 파장이 길어서 눈에 보이지 않는 빛입니다. 부품 가격이 저렴하고 호환성도 좋기 때문에 TV나 에어컨 등의 가전기기의 원격 제어용으로 많이 사용됩니다.

오른쪽 사진은 적외선 수신기입니다. IR receiver라고도 합니다.

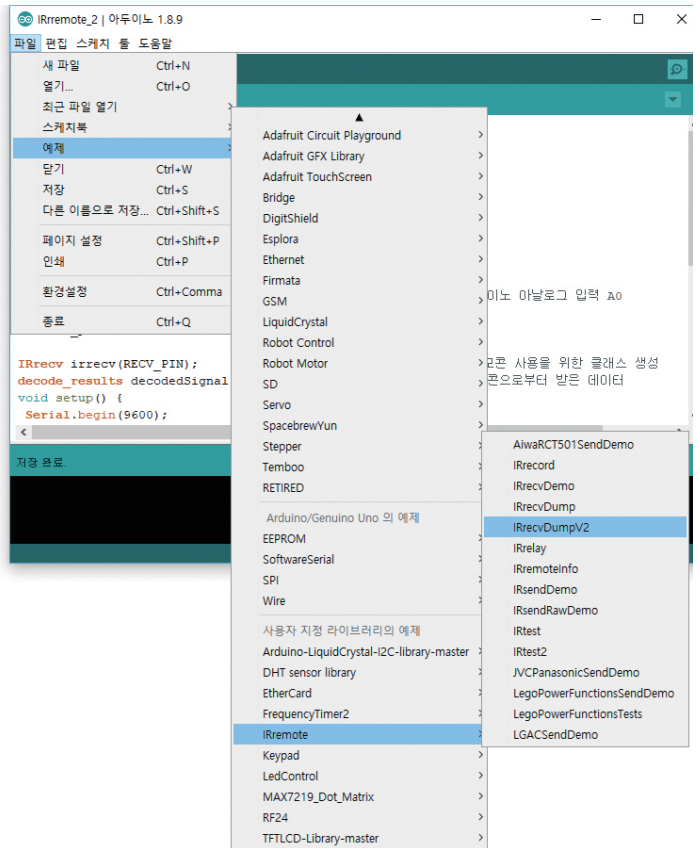
IR 리모컨의 신호를 확인해보고, 더 나아가 신호에 따라 LED를 작동시키는 실습을 해봅시다. 이를 위해서는 우선 IRremote 라이브러리가 필요합니다.



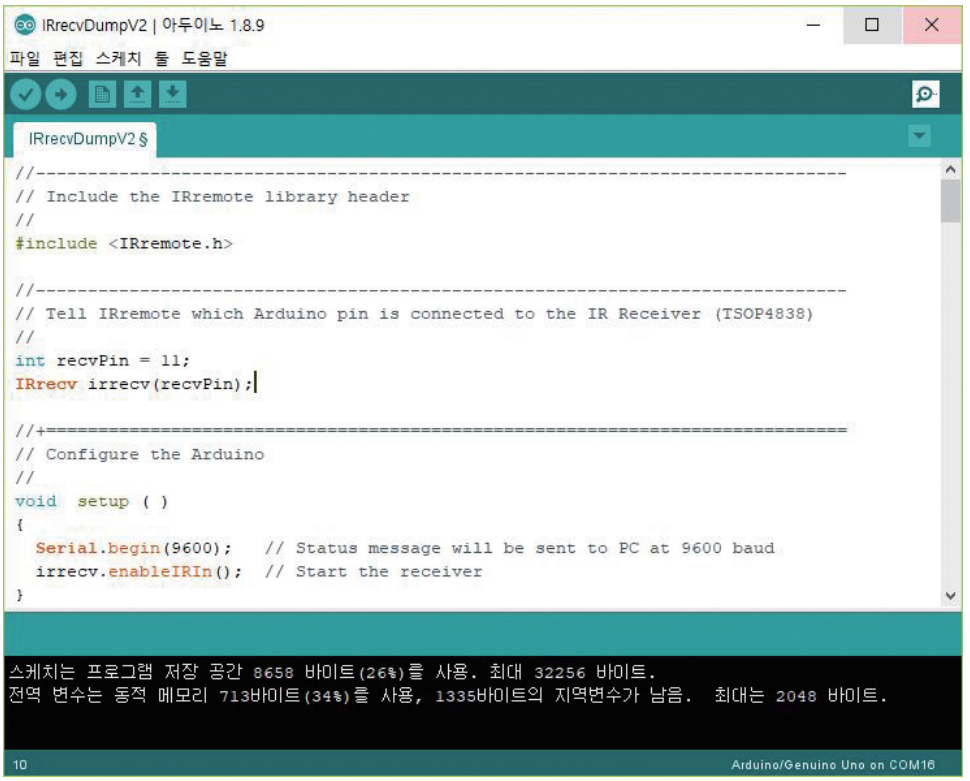
[메뉴바 – 툴 – 라이브러리 관리]에서 IRremote를 다운로드해 주세요.



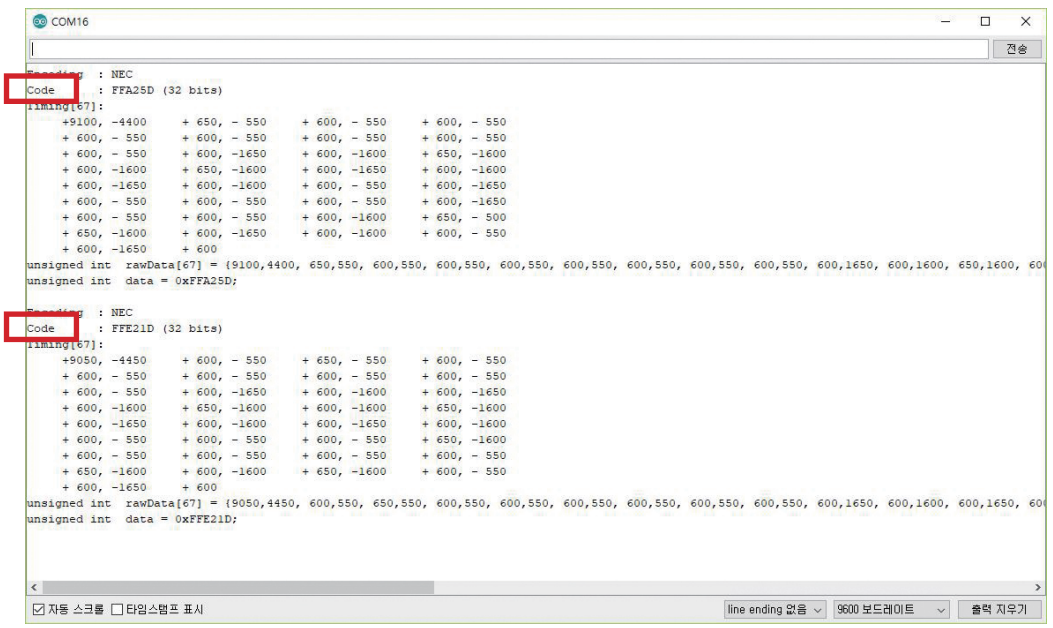
준비물	
아두이노 우노	1대
브레드보드	1개
저항 220Ω	4개
LED	4개
IR 수신기	1개
점퍼와이어	10가닥 내외



먼저 IR리모컨의 적외선 신호를 확인하기 위해, 왼쪽 사진의 [메뉴바 – 파일 – 예제 – IRremote – IRrecvDumpV2]을 실행해줍니다.



위 예제를 아두이노에 업로드하고 시리얼 모니터를 실행하면 아래와 같이 리모컨의 버튼을 누를 때마다 전송되는 데이터를 확인하실 수 있습니다.



통신 프로토콜에 대한 많은 정보가 있지만 저희가 알고 싶은 데이터는 16진수로 나와있습니다. 0xFFA25D, 0xFFE21D처럼 Code라고 되어있는 곳에 표시된 16진수 코드입니다. 각 버튼마다 고유한 코드를 갖고있습니다.

그럼 이제, 이 정보를 가지고 IR리모컨의 버튼으로 led를 on/off하는 예제를 실행시켜 보겠습니다.

예제소스

```
/*
리모컨으로 적외선 센서에 값을 넣고, led를 제어
http://www.devicemart.co.kr/
*/
#include <IRremote.h>

int RECV_PIN = 11; // IR 수신기 D11에 연결
int LED1 = 7; // LED1은 7번핀에 연결
int LED2 = 8; // LED2는 8번핀에 연결

IRrecv irrecv(RECV_PIN); // IR리모컨 사용을 위한 클래스 생성
decode_results decodedSignal; // IR리모컨으로부터 받은 데이터

void setup()
{
  Serial.begin(9600); // 시리얼 통신 시작
  irrecv.enableIRIn(); // 리모컨 수신 시작
  pinMode(LED1, OUTPUT); // LED 출력모드로 설정
  pinMode(LED2, OUTPUT); // LED 출력모드로 설정
}

void loop()
{
  if(irrecv.decode(&decodedSignal) == true) // IR 수신값이 있는지 판단
  {
    Serial.println(decodedSignal.value);

    if(decodedSignal.value == 0xFFA25D) // 리모컨의 ch- 버튼이 눌리면 LED 1 ON
    {
      digitalWrite(LED1, HIGH);
    }
    if(decodedSignal.value == 0xFFE21D) // 리모컨의 ch+ 버튼이 눌리면 LED 1 OFF
    {
      digitalWrite(LED1, LOW);
```

```
    }

   if(decodedSignal.value == 0xFF22DD) // 리모컨의 prev버튼이 눌리면 LED 2 ON
    {
      digitalWrite(LED2, HIGH);
    }
    if(decodedSignal.value == 0xFFC23D) // 리모컨의 play버튼이 눌리면 LED 2 OFF
    {
      digitalWrite(LED2, LOW);
    }
    irrecv.resume();   // IR 다음 데이터 수신
  }
}
```

Quest 12. Boss : 흉악광 발로르

발로르

감고있는 이 눈이 떠지기 전에 나를 쓰러뜨려라

용사

네??? 갑자기??

못 지나가게 막는게 아니라 쓰러뜨리라구요??

BOSS 문제

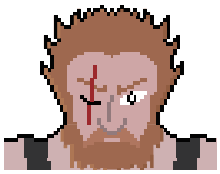
조도센서의 값을 0.5초마다 시리얼모니터에 출력하고,

감지되는 밝기가 밝을수록 LED도 밝아지는 회로를 구성하고 구동하시오.

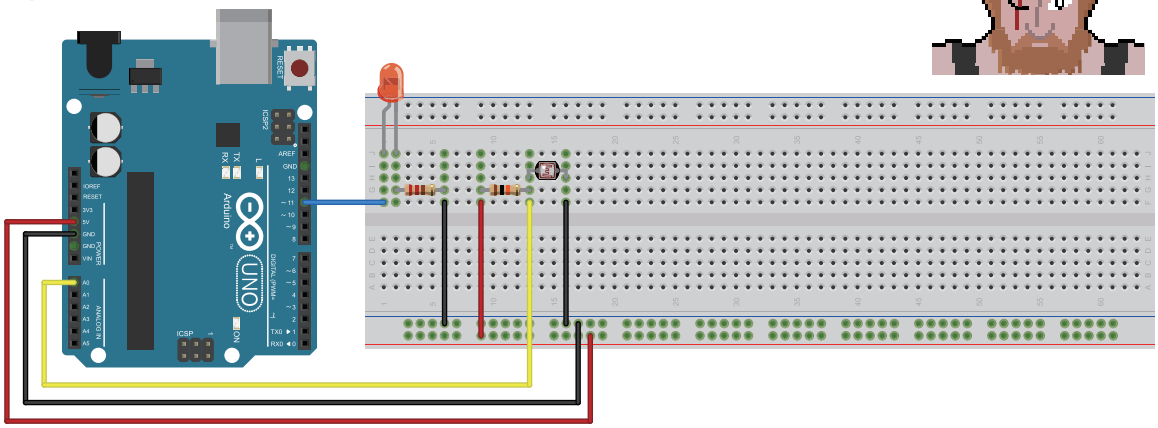
준비물	
아두이노 우노	1대
브레드보드	1개
저항 220Ω	1개
LED	1개
조도 센서	1개
점퍼와이어	10가닥 내외

Quest 12. 정답 체크

정상적으로 구동되었다면 발로르에 X표 해주세요.



정답예제회로



정답예제소스

```
int led = 11;        // LED를 11번핀에 연결

void setup()
{
  pinMode(led,OUTPUT);  // LED를 출력모드로 설정
  Serial.begin(9600);  // 속도 9600으로 통신 시작
}

void loop()
{
  int val = analogRead(A0);  //  A0 아날로그 값을 읽어서 val에 저장
  Serial.println(val);     // val 값을 출력

  // 조도센서의 val  아날로그 값의 범위는 0~1023, 밝을수록 저항이 낮아져서 아날로그 값은 낮아짐
  // led는 pwm값의 범위는 0~255, pwm이 높을수록 밝아지기 때문에, map 함수를 사용하여 변환
  int brightness = map(val,0,1023,255,0);
  analogWrite(led, brightness);
  delay(500);
}
```

※회로와 코드가 정답예시와 동일할 필요는 없습니다. 문제의 동작과 동일하게만 동작한다면 정답입니다.

발로르

이 저주받은 몸에서 자유롭게 해줘서 고맙군
하지만 남은 녀석들은 만만치 않을것이다

용사

???

틀렸다면 정답예제 실습 후
스테이지 3.6부터 다시!

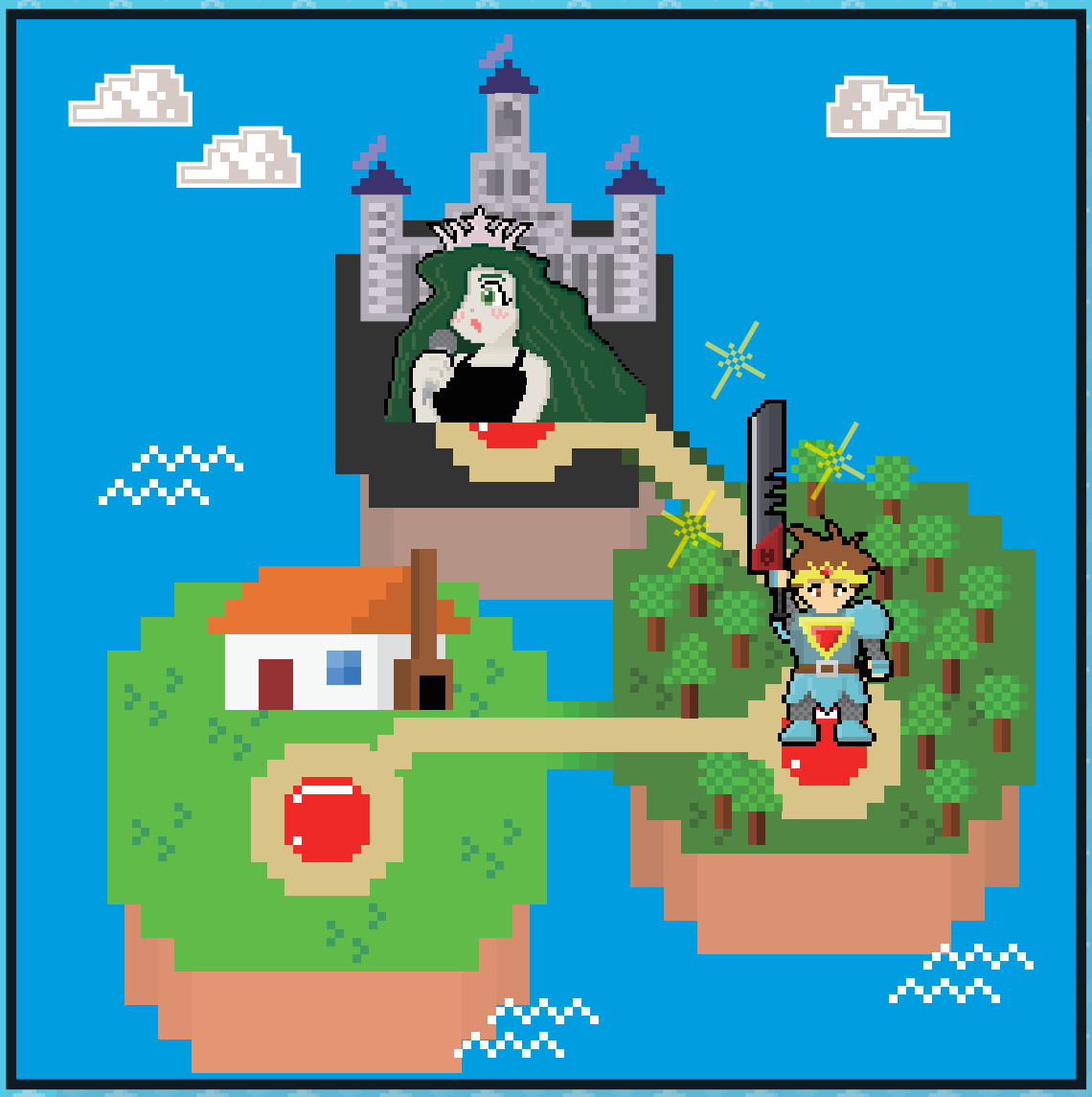
모두 맞추셨다면 다음페이지로!

용사는 『아날로그라이트 메소드』를 얻었다.

용사

힘조절...이랄까?

스테이지 3.7
소리

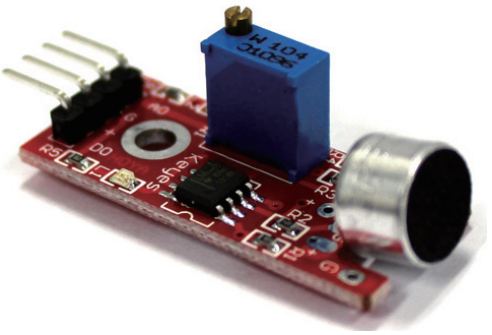


EXP♥

3. 아두이노 실전 연습

3.7. 소리

3.7.1. 사운드 센서



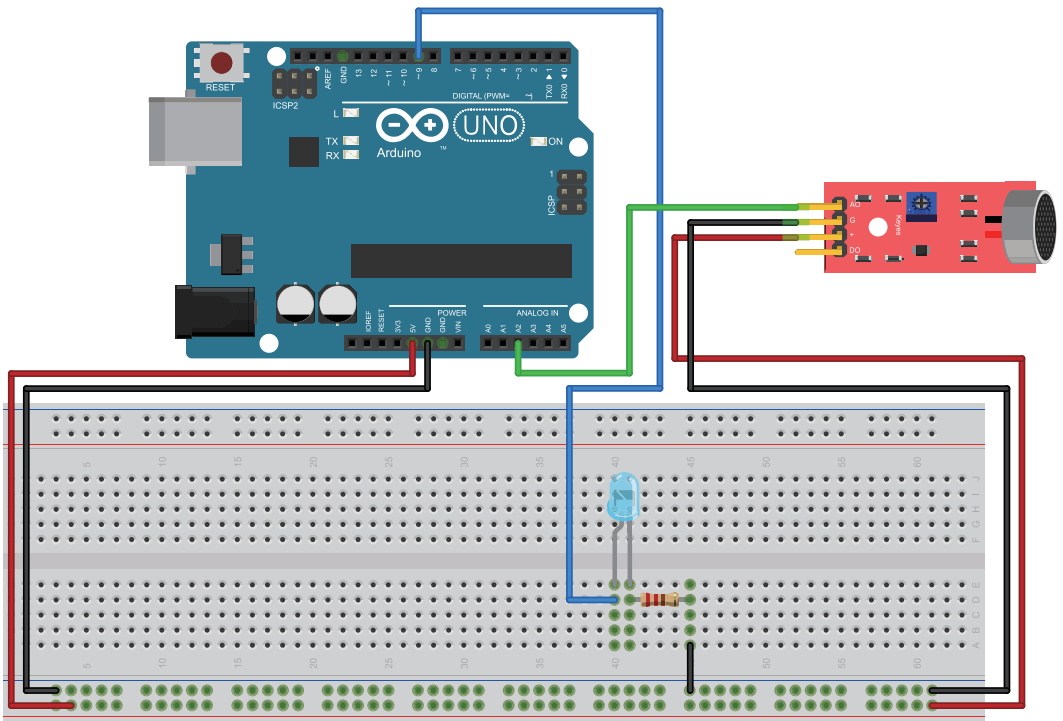
사운드 센서는 소리감지 센서라고도 불립니다. 소리는 소리의 높낮이와 소리의 크기로 나눌 수 있습니다.

소리는 공기의 떨림인데, 이 떨림의 진동수가 높으면 높은 음입니다. 반대로 진동수가 낮으면 낮은 음입니다. 소리의 크기는 진동수와 관련이 없고 진폭이 클수록 음량이 크고 진폭이 적을수록 음량이 작습니다.

이번에 다룰 센서는 소리의 크기와 관련된 센서입니다. 장착된 마이크를 통해서 소리의 크기를 아날로그 신호나 디지털 신호로 출력해 줍니다.

디지털 신호는 소리가 특정 크기 이상일 때 HIGH로 출력됩니다. 사운드 센서에 내장된 가변저항을 조절하여 디지털 신호의 소리 크기 기준을 정해줄 수 있습니다.

이번 실습에서는 감지되는 소리가 클수록 LED의 밝기가 밝아지도록 해보겠습니다.



준비물		준비물	
아두이노 우노	1대	LED	1개
브레드보드	1개	사운드센서 모듈	1개
저항 220Ω	1개	점퍼와이어	10가닥 내외

예제소스

```
/*
sound_detect with a led
소리 감지 모듈을 사용해서 led를 제어하는 예제
http://www.devicemart.co.kr/
*/
int led = 9; // LED를 9번핀에 연결

void setup()
{
  pinMode(led,OUTPUT); // led를 출력으로 설정
  Serial.begin(9600); // 통신속도 9600bps로 시리얼 통신을 시작
}

void loop()
{
  int value = analogRead(A2); // 소리감지  센서값을 읽어서 변수 value에 저장
   /*map()함수 사용 map(value, fromLow, fromHigh, toLow, toHigh)
  변환할 수, 현재 범위 값의 하한, 현재 범위 값의 상한, 목표 범위값의 하한, 목표 범위값의 상한*/
  //읽어들인 값으로 LED의 밝기를 구하고, LED의 밝기를 새로고침 합니다.
  int adjust = map(value,0,1024,0,255);
  analogWrite(led,adjust);
  Serial.println(value);//센서값을 시리얼 모니터로 출력
  delay(100);
}
```

정상적으로 동작한다면 소리가 클 수록 LED가 밝아지는 것을 확인하실 수 있습니다.

3.7.2. 버저

Buzzer는 외래어 표기법에 따르면 '버저'가 올바른 표현이지만 국내에서는 '부저'라는 발음으로 더 자주 사용됩니다. 비퍼(beeper)라고도 부릅니다.

위 사진처럼 생긴 버저는 일반 스피커와는 다르게 전기적 신호를 주었을 때 수축하거나 확장하는 피에조 효과를 이용해서 공기를 진동 시켜 소리를 만들어 냅니다. 이런 부품을 피에조 버저라고 합니다.

피에조 버저는 제어 방식에 따라 2종류로 분류됩니다. 능동버저(active buzzer)와 수동버저(passive buzzer)입니다.

능동 버저는 내부에 내장된 회로가 있어서 전원을 연결하기만 해도 내부에서 특정 주파수를 발생시켜서 소리가 나게 됩니다.

수동 버저는 내장된 회로가 없어서 전원을 인가해도 소리가 나지 않고, 주파수를 발생시켜 입력해주어야만 소리가 출력됩니다.

능동 버저와 수동 버저를 구분하는 방법은 두 가지 정도가 있습니다. 위에서 설명했듯이 가장 확실한 방법은 전원을 주었을 때 소리의 발생 유무를 확인하는 것입니다. 또 다른 방법은 제조사에 따라 다를 수 있지만 다리 길이가 같다면 수동 버저, 다리 길이가 다르다면 능동 버저입니다.

버저는 극성이 있는 제품으로 부품 상단의 (+) 마크 쪽에 전원이나 제어핀을 연결해주시면 됩니다.

피에조 수동 버저 만으로는 소리의 크기를 제어할 수 없습니다. 필요하다면 저항을 통해 음량을 줄일 수는 있습니다.

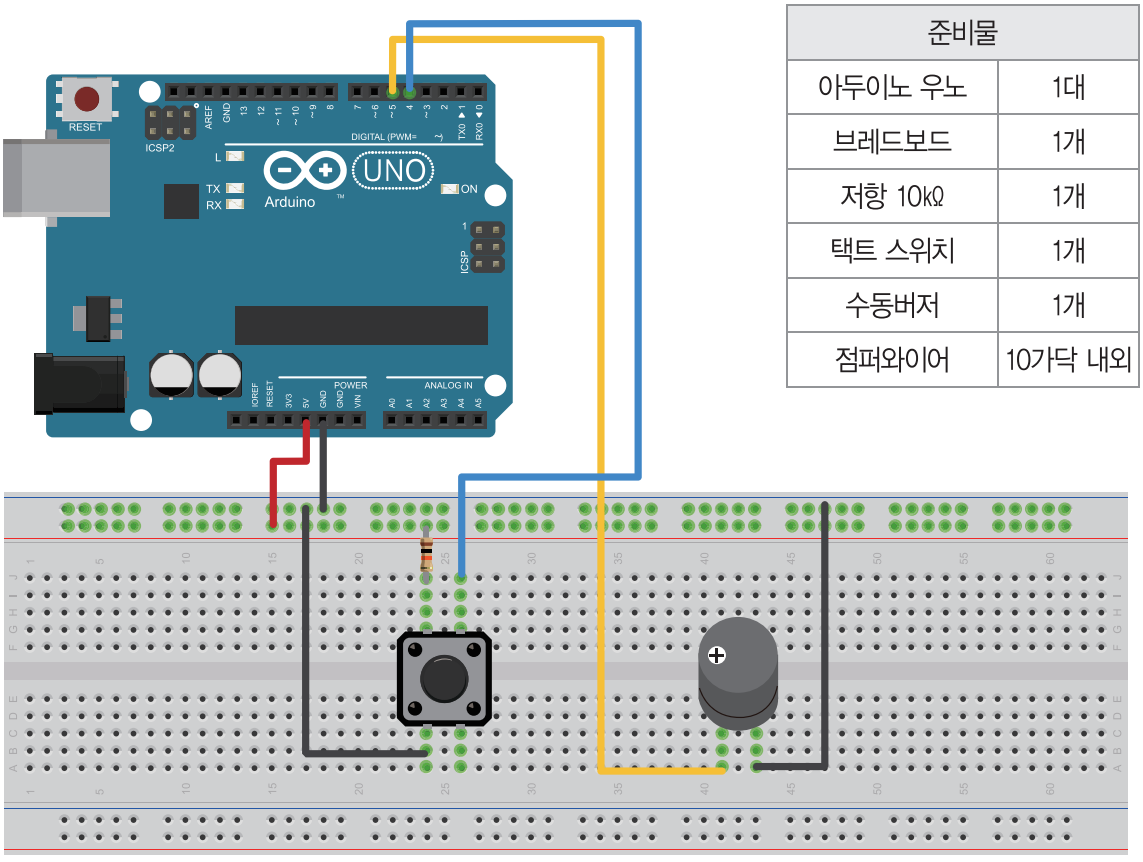
아래 표와 같은 주파수를 버저에 입력해주면 다양한 높낮이의 소리를 출력할 수 있습니다. 이 주파수는 수동버저와 능동버저 모두에 해당합니다.

옥타브 / 코드	C(도)	C#	D(레)	D#	E(미)	F(파)	F#	G(솔)	G#	A(라)	A#	B(시)
1	33	35	37	39	41	44	46	49	52	55	58	62
2	65	69	73	78	82	87	93	98	104	110	117	123
3	131	139	147	156	165	175	185	196	208	220	233	247
4	262	277	294	311	330	349	370	392	415	440	466	494
5	523	554	587	622	659	698	740	784	831	880	932	988
6	1047	1109	1175	1245	1319	1397	1480	1568	1661	1760	1865	1976
7	2093	2217	2349	2489	2637	2794	2960	3136	3322	3520	3729	3951
8	4186	4435	4699	4978								

단위는 1초에 몇 번 진동하는지 표기하는 단위인 Hz(헤르츠)고 소수점은 반올림했습니다.

사운드 센서에서 언급했듯이 진동수가 높을수록 높은 음이 출력됩니다.

이제 스위치를 누르면 '도레미파솔라시도' 소리를 내보는 실습을 해보도록 하겠습니다.



준비물	
아두이노 우노	1대
브레드보드	1개
저항 10kΩ	1개
택트 스위치	1개
수동버저	1개
점퍼와이어	10가닥 내외

예제소스

```
/*
스위치-버저 테스트
http://www.devicemart.co.kr/
*/
int buzzer = 5; // 버저를 핀 5번에 연결
int sw = 4; // 스위치를 핀 4번에 연결

// '도레미파솔라시도' 음계 주파수 배열로 선언
int melody[]={262, 294, 330, 349, 392, 440, 494, 523};

void setup ()
{
  pinMode(buzzer,OUTPUT); //  핀 5번을 출력 설정
  pinMode(sw,INPUT_PULLUP); // 핀 4번을 내부풀업으로 입력 설정
}
void loop()
```

```
{
  if(digitalRead(sw)==0)
  {
    for(int a=0; a<8; a++)
    {
      tone(buzzer,melody[a],400); // 0.4초동안 음계 출력
      delay(500); // 0.5초 대기
    }
   }
  else
  {
    noTone(5);    // 피에조 버저 OFF
  }
}
```

tone()

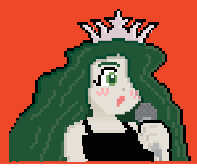
tone(출력 핀번호, 주파수, 유지시간)

출력 핀 번호에는 PWM 출력이 가능한 핀을 사용합니다.

주파수는 위의 표를 참고 하셔서 작성해주시면 됩니다. 유지시간은 말 그대로 소리를 얼마나 길게 출력할 것인지를 말하며, 유지시간은 밀리 초(ms)단위로 기록하시면 됩니다. 이 함수는 한 번에 1개의 핀만 제어할 수 있으며, 이미 tone 함수를 사용해 연주하고 있다면, 다른핀에 tone 함수를 적용해도 작동하지 않습니다.

delay 함수를 통해 유지시간이 끝나기를 기다렸다가 순서대로 연주해야합니다.

Quest 13. Boss : 음치 세이렌



세이렌

딩딩딩~ 굿모닝~ 딩딩딩~ 빠빠빠 빠빠 빱빠빠빠 굿모닝~♪
빠빠빠 빠빠 빱빠빠빠 굿모닝~♪ 빠빠빠 빠빠 빱빠빠빠 뷰리풀데이~♪
빠빠빠빠 이츠 어 뷰리풀데이~

용사

으… 완전 시끄러워,,,!
노래가 소음공해잖아…

BOSS 문제

상어가족 첫 소절을 무한히 연주하는 회로를 구성하고 구동하시오.

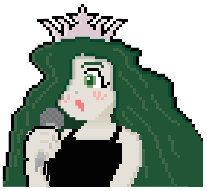
계이름 : [레미 솔솔 솔솔솔솔솔 레미 솔(쉼표) 솔솔솔솔솔 레미 솔(쉼표) 솔솔솔솔솔 솔솔 파# 파#(쉼표)]

유지시간 : [400 400 200 200 200 100 300 100 200 200 200 100 30 200 100 300 100 200 200 200 100 30 200 100 300 100 200 200 200 200 200 3000]

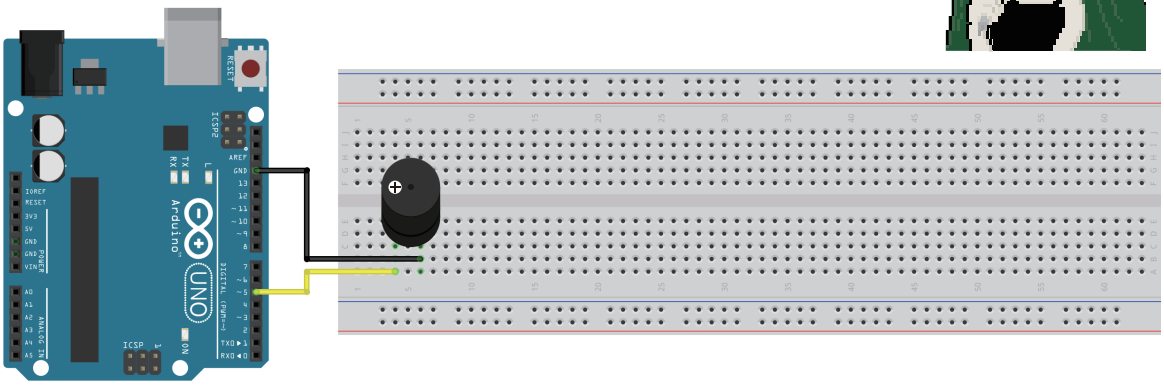
준비물	
아두이노 우노	1대
브레드보드	1개
수동버저	1개
점퍼와이어	10가닥 내외

Quest 13. 정답 체크

정상적으로 구동되었다면 세이렌에 X표 해주세요.



정답예제회로



정답예제소스

```
int buzzer = 5; // 버저를 핀 5번에 연결

//쉼표
int rest =0;

// 계이름을 순서대로 배열로 선언
int melody[] = {294, 330, 392, 392, 392, 392, 392, 392, 392, 294, 330, 392, rest,
                // 레   미   솔   솔   솔   솔   솔   솔   솔   레   미   솔
                // 아   기   상   어   뚜   루   룻   뚜   루   귀   여   운
                  392, 392, 392, 392, 392, 294, 330, 392, rest,
                // 솔   솔   솔   솔   솔   레   미   솔
                // 뚜   루   룻   뚜   루   바   다   속
                  392, 392, 392, 392, 392, 392, 392, 370, 370, rest};
               // 솔   솔   솔   솔   솔   솔   솔   파#  파#
               // 뚜   루   룻   뚜   루   아   기   상   어

// 연주 유지시간을 배열로 선언
int damper[] = {400, 400, 200, 200, 200, 100, 300, 100, 200, 200, 200, 100, 30,
                200, 100, 300, 100, 200, 200, 200, 100,  30,
                200, 100, 300, 100, 200, 200, 200, 200, 200, 3000};

void setup ()
```

```
{
  pinMode(buzzer,OUTPUT); //  핀 5번을 출력 설정
}

void loop()
{
  for(int a=0; a<32; a++)
  {
    tone(5,melody[a],damper[a]); // 각 음계와 유지시간으로 버저 연주
    delay(damper[a]+100); // 유지시간+100ms 만큼 대기하여 연주 박자 유지
  }
}
```

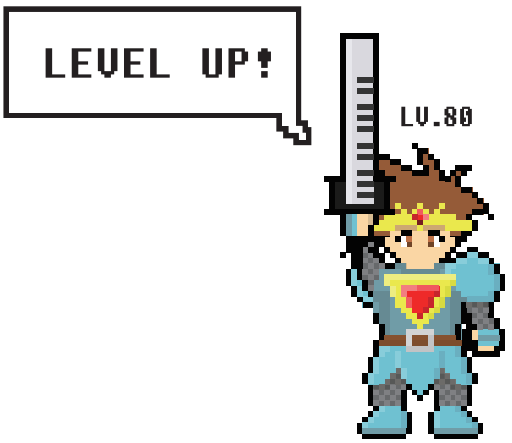
※회로와 코드가 정답예시와 동일할 필요는 없습니다. 문제의 동작과 동일하게만 동작한다면 정답입니다.

틀렸다면 정답예제 실습 후
스테이지 3.7부터 다시!

모두 맞추셨다면 다음페이지로!

세이렌
상어다! 뚜루루뚜루~ 도망쳐! 뚜루루뚜루~

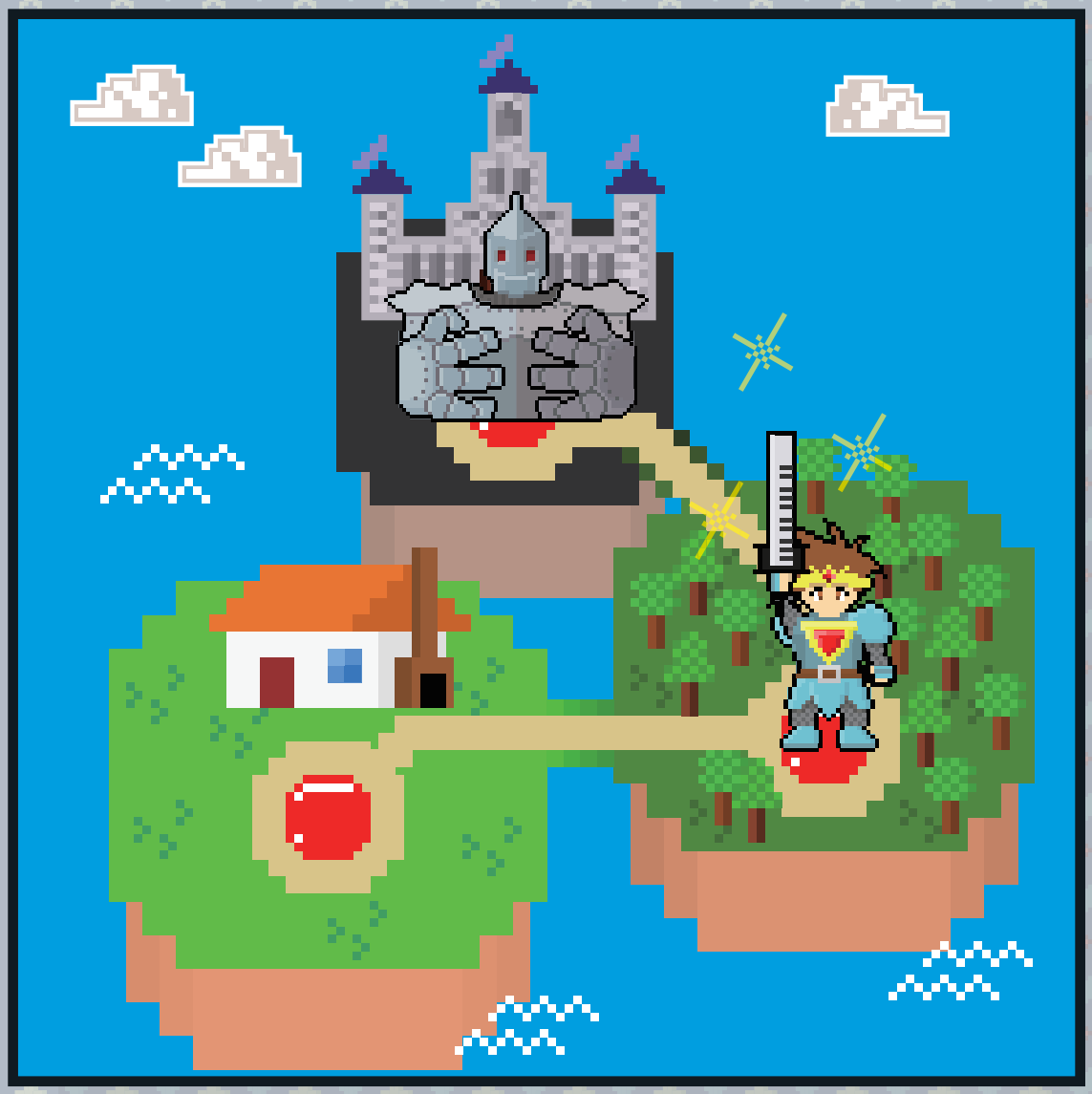
용사
훨씬 듣기 좋네ㅎㅎ 이제 정말 얼마 안남았다!



용사는 『톤 메소드』를 얻었다.

스테이지 3.8

출력



EXP♥

3. 아두이노 실전 연습

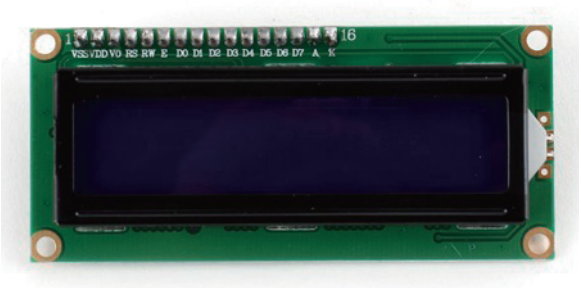
3.8. 출력

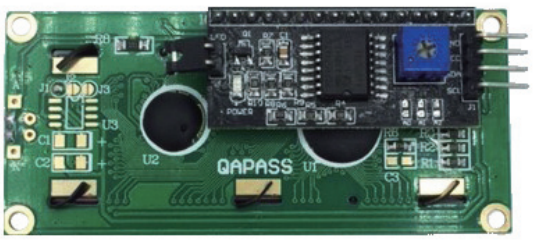
3.8.1. LCD

LCD(Liquid Crystal Display)는 액정표시장치라고도 부릅니다. LCD는 TV나 휴대폰에 주로 사용되는 제품입니다. 스스로 빛을 내지 않기 때문에 후면에 백라이트(Back Light), 전면에 액정을 두어 액정이 전기 신호에 따라 빛을 차단하거나 통과시키는 방식으로 문자나 그래픽을 표시합니다.

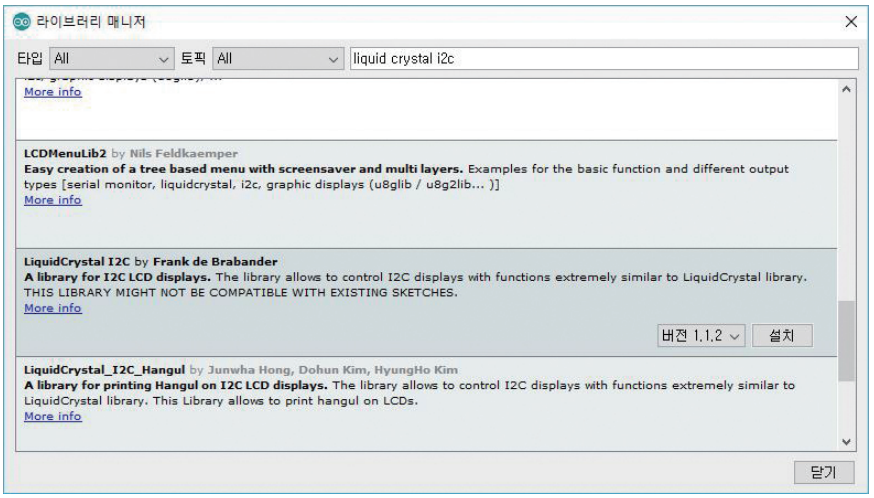
다양한 규격이 있으나 주로 16×2 Character LCD를 아두이노와 함께 사용합니다. 이전에는 LCD 모듈에 I2C 모듈없이 LCD의 디지털 포트를 그대로 사용하였고, 때문에 입출력 핀을 많이 필요로 하여 효율이 좋지 않았습니다.

현재는 I2C 모듈이 부착되어있는 LCD가 많이 보급되었고, 이번 실습에서도 I2C LCD 모듈을 사용합니다. I2C 통신을 통해 제어가 가능하기 때문에 4개의 핀만으로 LCD 제어가 가능합니다.

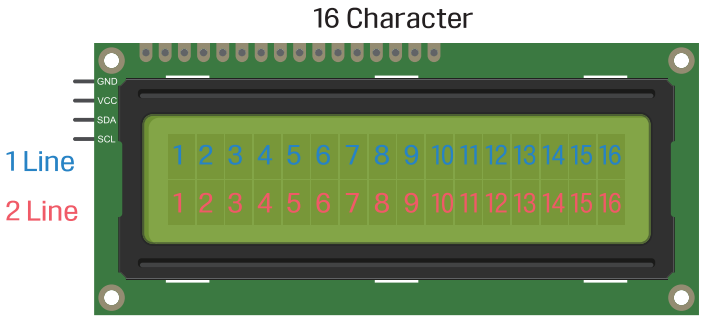




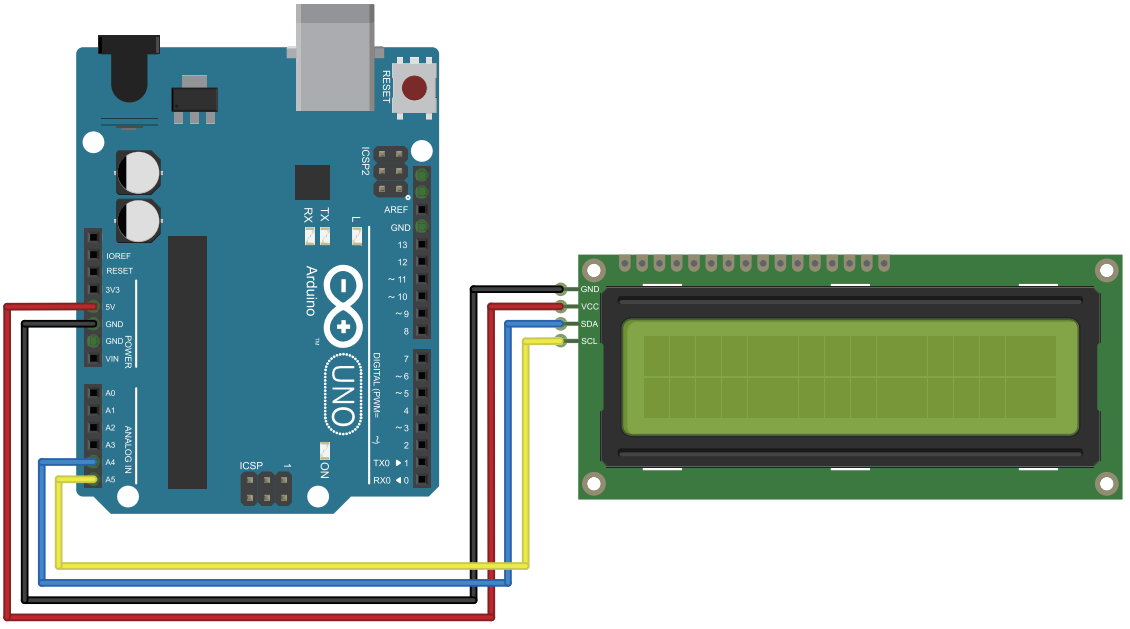
뒷면에 있는 점퍼는 백라이트 ON을 위해서 연결되어있고, 가변저항을 통해서 문자의 선명도를 조절합니다. I2C LCD를 아두이노에서 간편하게 사용하기 위해 LiquidCrystal_I2C 라이브러리를 설치하겠습니다.



[메뉴바 – 툴 – 라이브러리 관리에서 LiquidCrystal_I2C]를 설치합니다.



시리얼 모니터에 16자 이하의 영문/숫자를 입력하면 LCD에 똑같이 표시하는 실습입니다.
라이브러리 특성상 아쉽게도 LCD 한글 출력은 지원되지 않습니다.



배선		
아두이노	조이스틱 모듈	비고
GND	GND	Ground
5V	VCC	Vcc
A4	SDA	Serial Data
A5	SCL	Serial Clock

준비물	
아두이노 우노	1대
I2C LCD 모듈	1개
점퍼와이어	10가닥 내외

예제소스

```
/*
lcd 시리얼 테스트
http://www.devicemart.co.kr/
*/
#include <Wire.h> // I2C 통신을 위해 Wire 라이브러리 사용
#include <LiquidCrystal_I2C.h> // LiquidCrystal_i2c 라이브러리 사용

// 객체이름을 lcd로 선언 I2C 주소값을 0x27로 설정, 16 문자 2줄 디스플레이
// 만약 lcd가 출력이 안된다면, 주소값을 0x3F로 수정
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup()
{
  lcd.begin(); // lcd 시작
  lcd.backlight(); // lcd 백라이트 on
  Serial.begin(9600); // 속도 9600bps로 시리얼 통신시작
  Serial.println("LCD통신 시작"); // 시리얼 통신 시작 문자 출력
}

void loop()
{
  if (Serial.available()) // 시리얼 통신으로 문자열이 도착한다면
  {
    delay(10); // 메세지가 모두 도착하도록 잠시 대기
    lcd.clear(); // lcd 내용 삭제(초기화)
    // 받은 문자들을 시리얼 모니터와 lcd에 출력
    while (Serial.available() > 0)
    {
      char val=Serial.read(); // 받은 문자열을 val에 저장
      if(val>16)
      {
        Serial.print(val); // val 값을 시리얼 모니터에 출력
        lcd.print(val);  // val 값을 lcd에 출력
      }
      else
      {
        Serial.print("16자 이내로 입력해주세요");
```

```
        lcd.print("plz 16char below");
      }
    }
  }
}
```

시리얼 모니터에 숫자/문자/공백이 포함된 16자 이하의 글자를 입력하면 LCD에 표시되야 정상적인 동작입니다. LCD 관련 함수들은 아래와 같이 사용하시면 됩니다.

```
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

위 문장처럼 객체이름을 lcd로 선언했을 때의 함수명입니다. lcd는 다른 임의의 객체이름으로 변경 가능합니다.

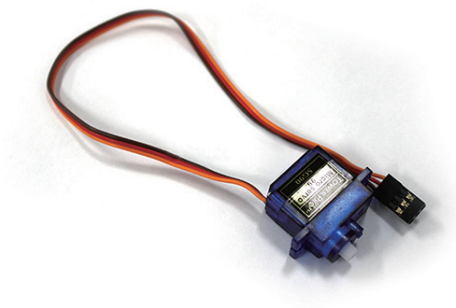
예) 객체이름을 lcd로 선언 => lcd.begin(), 객체이름을 device로 선언 => device.begin()

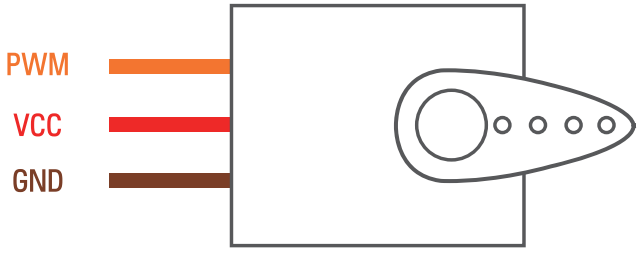
함수	기능
lcd.begin()	lcd 사용을 시작
lcd.clear()	lcd 화면의 모든 내용 삭제
lcd.backlight()	lcd 백라이트 on
lcd.nobacklight()	lcd 백라이트 off
lcd.display()	lcd 내용 표시
lcd.noDisplay()	lcd 내용 숨김
lcd.setCursor(col,row)	lcd 커서를 col,row좌표로 이동 (0부터시작)
lcd.cursor()	lcd 커서 표시
lcd.noCursor()	lcd 커서 숨김
lcd.home()	lcd 커서의 위치를 0,0 좌표로 이동
lcd.blink()	lcd 커서를 깜박임
lcd.noBlink()	lcd 커서를 깜박이지 않음
lcd.write(data)	lcd 화면에 data를 출력
lcd.print(data)	lcd 화면에 data를 출력
lcd.scrollDisplayRight()	lcd 내용을 우측으로 1칸 이동
lcd.scrollDisplayLeft()	lcd 내용을 좌측으로 1칸 이동
lcd.autoscroll()	lcd 내용을 자동으로 스크롤
leftToRight()	lcd 문자 표시를 좌에서 우로 설정
rightToLeft()	lcd 문자 표시를 우에서 좌로 설정

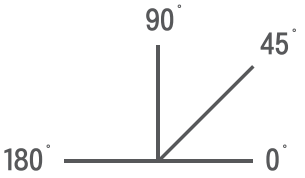
3.8.2. 서보모터

서보(Servo)는 "추종한다, 따른다"의 뜻을 가지고 있는 용어입니다. 사용자가 원하는 속도나 각도를 지정해주면 그대로 동작하는 모터를 서보모터라고 합니다.

아두이노에서 주로 사용하는 서보모터는 오른쪽 사진과 같이 3pin으로 구성되어 있습니다. 대표 모델인 SG-90은 0~180도로 동작하지만 0~360도로 동작하는 연속 회전이 가능한 서보모터도 있습니다.







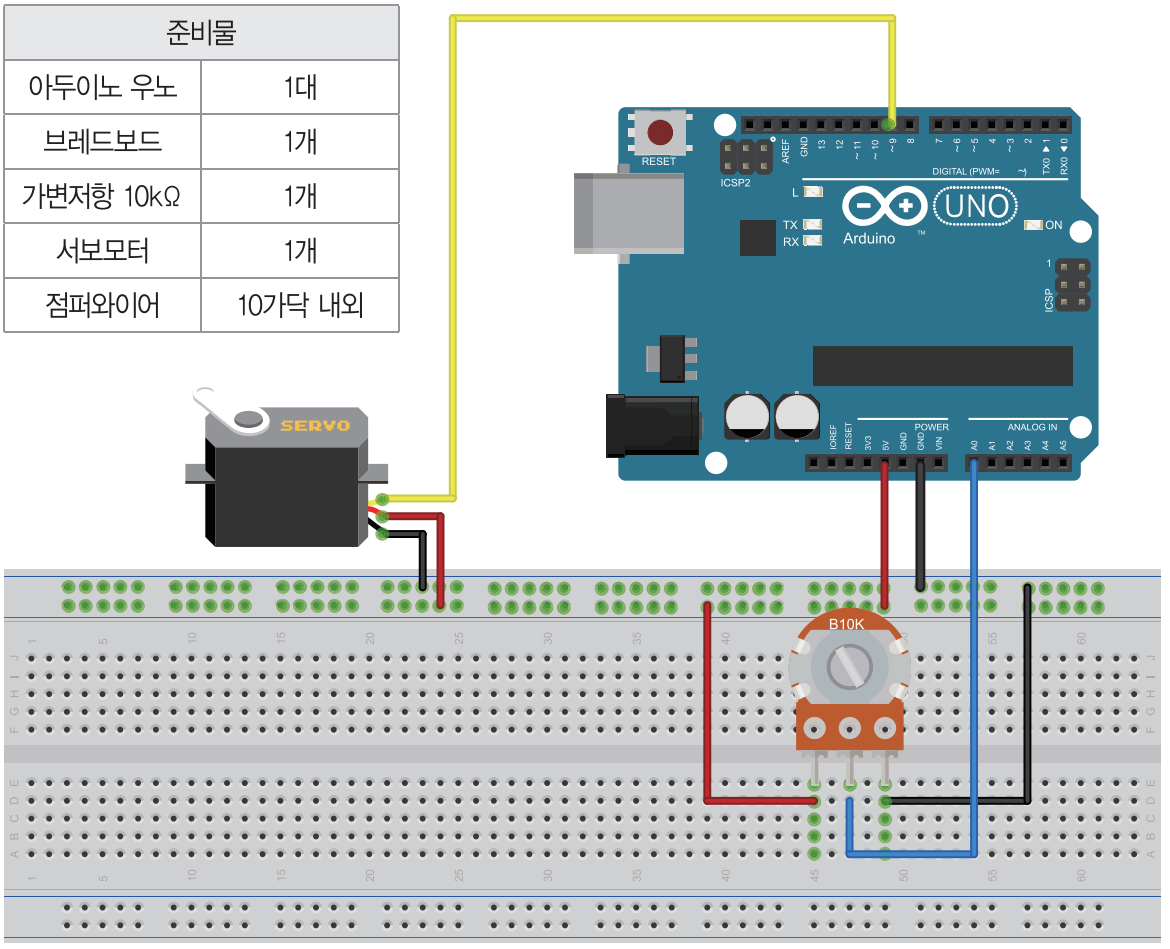
PWM 핀의 경우 주황, 노랑, 흰색 등 제조사마다 조금씩 다릅니다.

GND 핀의 경우 갈색 혹은 검정색일 수 있습니다.

서보모터 출력단에 토크(회전 힘)가 많이 필요하게 되면 부하에 따라 높은 전류를 요구합니다. 아두이노 우노는 USB로 전원을 공급할 때 5V 핀에서 최대 500mA 출력이 가능하고 9V 이상의 외부전원 공급 시 최대 800mA의 전류를 출력할 수 있습니다.

이 이상의 전류를 요구하면 아두이노에 전원 공급이 원활하지 않아 작동중에 꺼졌다 켜지는 현상이 발생하거나 보드나 모터가 고장 날 수 있습니다. 이렇게 큰 부하를 사용하는 경우에는 서보모터 전원은 외부에서 파워서플라이나 배터리 등을 이용하여 별도의 전원을 공급해야 합니다.

준비물	
아두이노 우노	1대
브레드보드	1개
가변저항 10kΩ	1개
서보모터	1개
점퍼와이어	10가닥 내외



예제소스

```
/*
서보모터 테스트
http://www.devicemart.co.kr/
*/
#include <Servo.h> // 서보모터 라이브러리 사용

Servo myservo;  // 서보모터 객체를 myservo로 생성

int potentiometer = A0;  // 가변저항 A0에 연결
int val;     // val 변수 선언

void setup()
{
  myservo.attach(9);  // 서보모터 9번핀에 연결
}
```

```
void loop()
{
  val = analogRead(potentiometer); // 가변저항에서 읽은 아날로그 값을 변수 val에 저장

  // map 함수를 이용해 아날로그값 0~1023을 0~180으로 조정한 후 val에 저장
  val = map(val, 0, 1023, 0, 180);
  myservo.write(val); // val 값에 따라 서보모터 이동
  delay(15); // 서보모터가 움직일 시간 15ms 딜레이
}
```

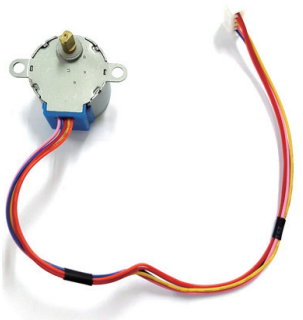
myservo.attach()

servo.attach(pin number,min,max)의 형식으로 사용됩니다. myservo 객체에 서보 모터 핀을 연결 하는 함수입니다. pin number는 pwm 출력이 가능한 핀을 선택해 주시면 됩니다.
min과 max의 의미는 서보의 최대 각도에 해당하는 펄스폭입니다. 예제소스와 같이 핀만 선언하고 min과 max는 생략할 수 있습니다. min과 max를 생략하게 되면 기본으로 최소 544, 최대 2400 마이크로초(us)로 설정됩니다.

myservo.write()

servo.write(angle)의 형식으로 사용됩니다. myservo객체에 서보 모터의 각도값을 넣어주는 함수입니다. angle에 원하는 각도 값을 넣어주면 서보모터가 해당 값으로 움직입니다.
만약 360도 회전이 가능한 연속 회전 서보모터일 경우 각도가 아닌 서보의 속도와 방향을 설정합니다. 0은 정방향 최고속도, 180은 역방향 최고속도, 90은 정지입니다.

3.8.3. 스테퍼모터



스테퍼(stepper) 모터는 스텝(step)모터, 스테핑(stepping) 모터라고도 부릅니다.

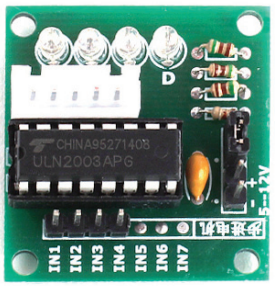
스테퍼모터는 펄스신호에 의해 작동하는 모터입니다. 입력 펄스 수와 모터의 회전 각도가 정확히 비례합니다. 이 점을 이용하면 펄스제어를 통해 모터를 특정 각도 단위로 정밀하게 제어할 수 있습니다.

Lead Wire Color	CW Direction (1–2 Phase)							
	1	2	3	4	5	6	7	8
4 Orange	1	1						1
3 Yellow		1	1	1				
2 Pink				1	1	1		
1 Blue						1	1	1

서보모터는 모터드라이버가 내장되어 있기 때문에 아두이노에 그대로 연결하여 사용했지만 DC모터, 스테퍼모터, BLDC모터 등의 모터를 사용할 때는 모터드라이버를 통해 연결해야 합니다.

아두이노 ▶ 모터드라이버 ▶ 모터

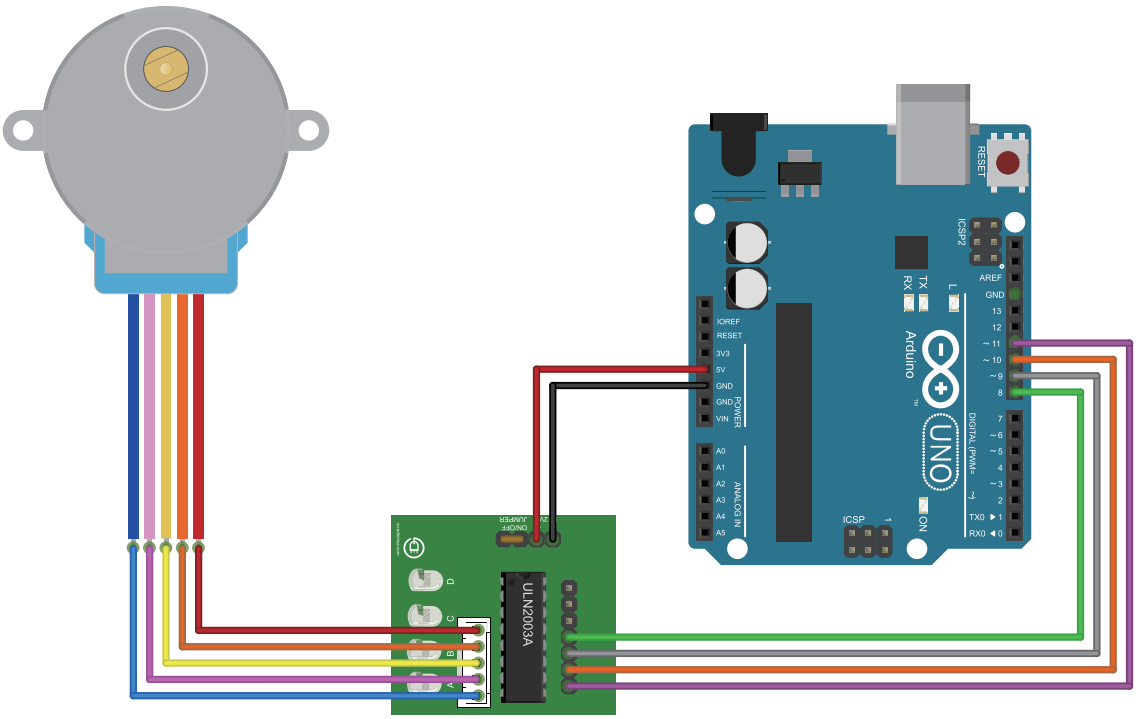
아두이노와 같은 컨트롤러에서 모터드라이버를 제어하고, 모터드라이버가 모터를 제어합니다. 스테퍼모터에는 스테퍼모터 드라이버를 사용합니다.



스테퍼모터 드라이버 중 하나인 ULN2003 모터드라이버 모듈입니다. 아두이노에서 모터드라이버로 펄스를 발생시켜주면 모터드라이버에서 모터를 동작시켜줍니다.

ULN2003 모터 드라이버의 전원은 5~12V입니다. 사진의 오른쪽 저항 아래 점퍼소켓 밑에(사진상 우측) 핀헤더가 2핀이 있는데 +, –핀을 통해 전원을 공급합니다.

점퍼 소켓이 꽂혀 있으면 모터드라이버가 사용(enable) 상태이며, 점퍼를 빼면 모터드라이버가 미사용(disable) 상태로 설정됩니다.



배선			
아두이노	UNL2003 모듈	아두이노	UNL2003 모듈
D8	IN1	D11	IN4
D9	IN2	5V	+
D10	IN3	GND	–

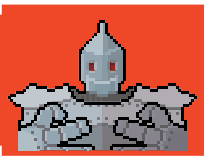
준비물	
아두이노 우노	1대
28BYJ-48 스테퍼 모터	1개
UNL2003 드라이버 모듈	1개
점퍼와이어	10가닥 내외

예제소스

```
/*
Stepper motor
스테퍼모터와 드라이버는 각각 28BYJ-48과 ULN2003을 사용
http://www.devicemart.co.kr/
*/
#include <Stepper.h> // 스테퍼모터 라이브러리 사용

const int steps = 2048; // 모터의 스텝수  2048=1바퀴, 1024=반바퀴

// myStepper 객체 생성(스텝 수, IN4, IN3, IN2, IN1)
// IN1~4까지 각각 아두이노 D8~D11번에 연결
Stepper myStepper(steps,11,9,10,8);

void setup()
{
  myStepper.setSpeed(15); // 스테퍼모터 속도(rpm)설정, 스텝 수와 연결되어 있음
  Serial.begin(9600); // 9600bps로 시리얼통신시작
}

void loop()
{
  Serial.println("시계방향");
  myStepper.step(steps); // 시계방향으로 스텝 수만큼 이동
  delay(500); // 모터 정지 후 잠시 대기
  Serial.println("시계반대방향");
  myStepper.step(-steps); // 반시계방향으로 스텝 수만큼 이동
  delay(500); // 모터 정지 후 잠시 대기
}
```

모터가 정방향 한 바퀴, 역방향 한 바퀴 도는 동작을 반복한다면 정상 구동입니다. 15rpm으로 구동하기 때문에 1바퀴를 도는데 4초(60초÷15rpm)가 걸리게 됩니다.

Quest 14. Boss : 극악력 탈로스



탈로스

삐빅. 올라오느라. 고생했다. 휴먼.
그러나. 인간. 시대에. 끝이. 도래했다.

용사

비켜라 깡통로봇!

BOSS 문제

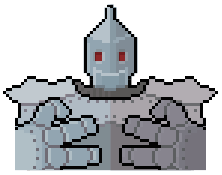
스테퍼모터와 map 함수를 이용하여 한 바퀴를 60등분하여 시계의 초침처럼 1초에 6° 이동 후 대기하는 회로를 구성하고 구동하시오.

※ 연속해서 회전하지 않고 초침처럼 6°마다 잠깐 멈췄다 움직여야 하며, 6° 회전과 멈추는 시간을 포함하여 1초일 때에만 정상 구동입니다.

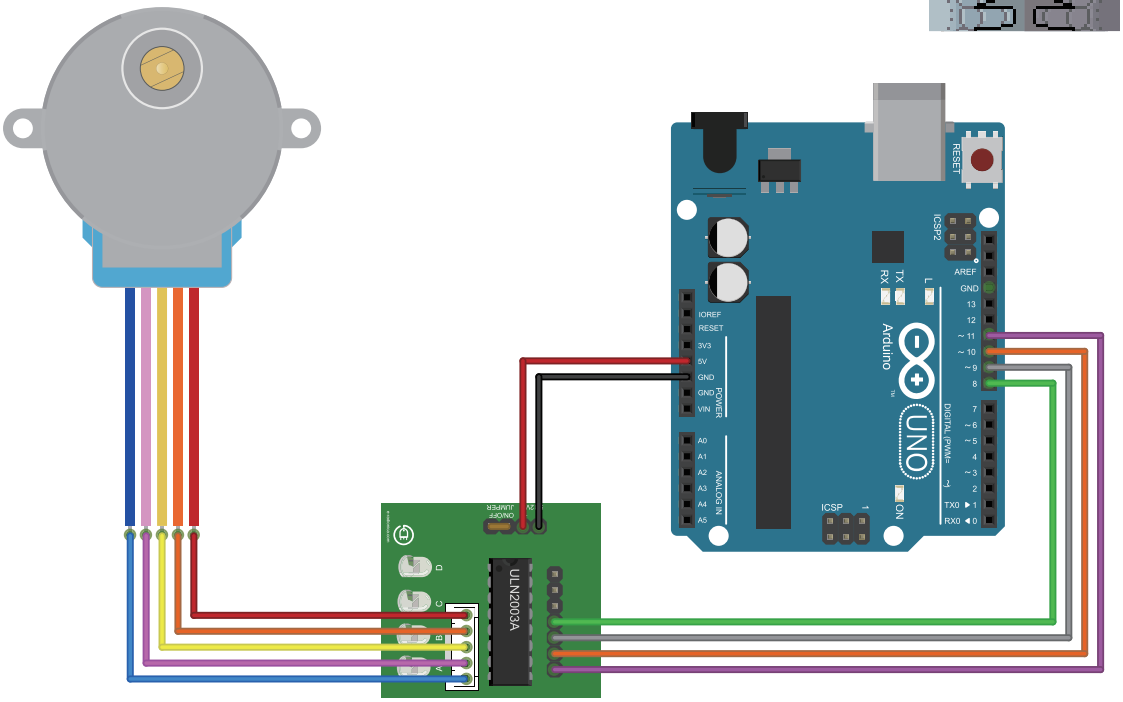
준비물	
아두이노 우노	1대
28BYJ-48 스테퍼 모터	1개
UNL2003 드라이버 모듈	1개
점퍼와이어	10가닥 내외

Quest 14. 정답 체크

정상적으로 구동되었다면 탈로스에 X표 해주세요.



정답예제회로



정답예제소스

```
#include <Stepper.h> // 스테퍼모터 라이브러리 사용

const int steps = 2048; // 모터의 스텝수  2048=1바퀴, 1024=반바퀴

// myStepper 객체 생성(스텝 수, IN4, IN3, IN2, IN1)
// IN1~4까지 각각 아두이노 D8~D11번에 연결
Stepper myStepper(steps,11,9,10,8);

void setup()
{
  myStepper.setSpeed(10); // 스테퍼모터 속도(rpm)설정, 스텝 수와 연결되어 있음
}

void loop()
{
```

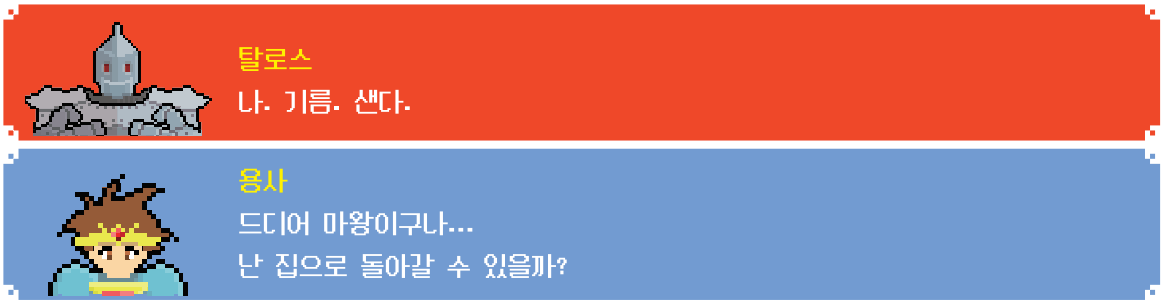
```
  int one_sec = map(1,0,60,0,steps);
  myStepper.step(one_sec); // 시계방향으로 스텝 수만큼 이동, 10rpm은 6도 이동시 0.1초 소요
  delay(900); // 0.9초 대기
}
```

※회로와 코드가 정답예시와 동일할 필요는 없습니다. 문제의 동작과 동일하게만 동작한다면 정답입니다.

10rpm은 1분에 10번의 회전을 합니다.
1회전은 60초÷10회전=6이 되므로 6초가 걸린다는 것을 알 수 있습니다.
초시계는 회전할 때 6도씩 회전하여 60번 멈춰야하는데, 0도에서 6도까지 가는데 걸리는 시간은 6초÷60번=0.1이 나옵니다. 즉 0도에서 6도까지 0.1초가 걸립니다.
딜레이를 사용하여 0.9초를 대기하여 1초를 맞춰주면 1초에 6도씩 회전하는 시계의 초침이 완성됩니다.

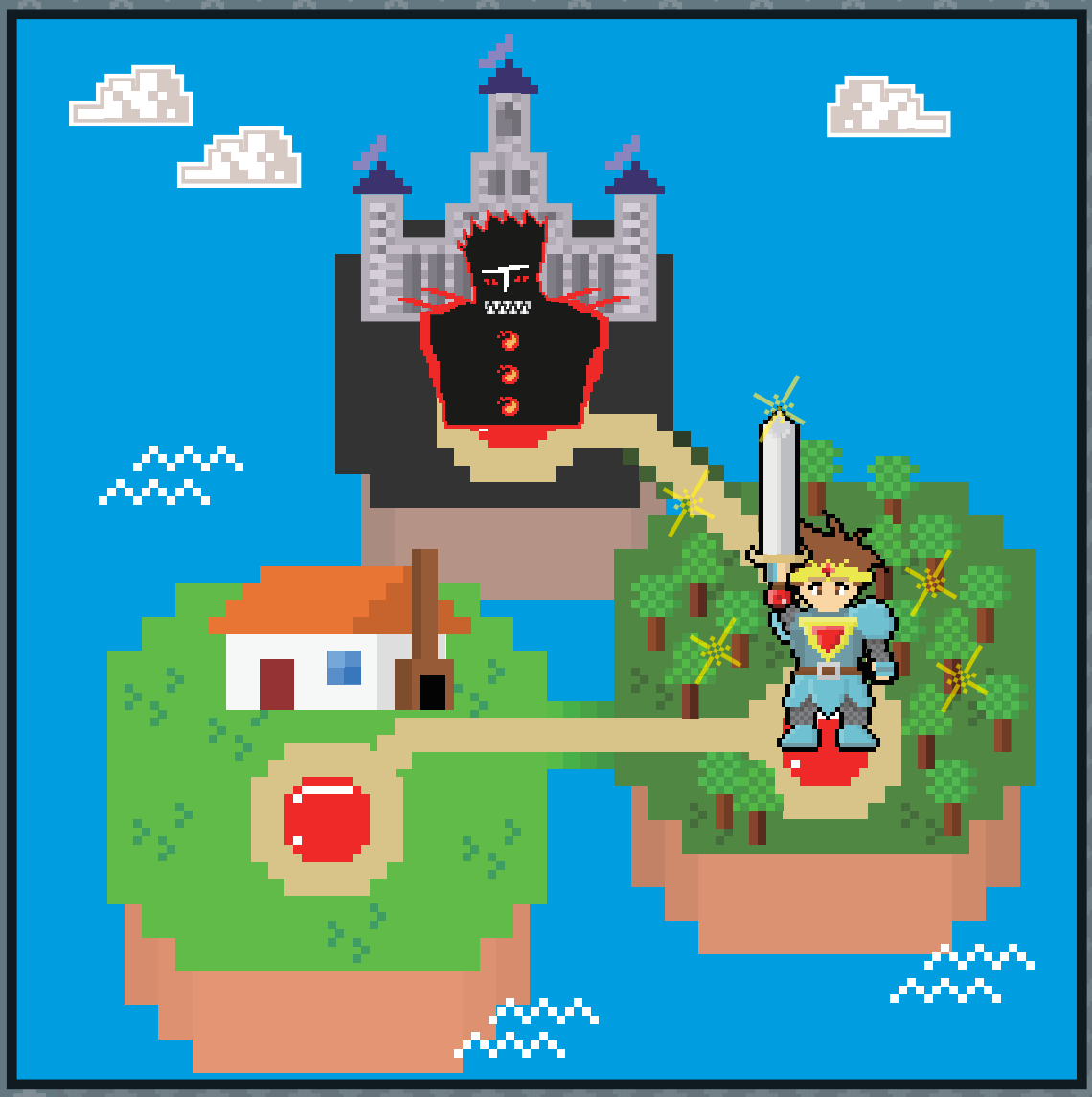
틀렸다면 정답예제 실습 후
스테이지 3.8부터 다시!

모두 맞추셨다면 다음페이지로!



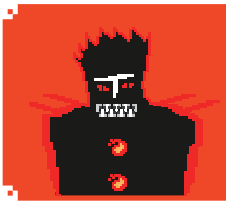
용사는 『맵 메소드』를 얻었다.

스테이지 4
응용 문제



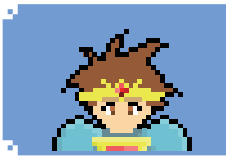
EXP♥

Quest 15. Boss : 마왕 프란츠



마왕 프란츠

나는 마왕 프란츠다.
사천왕과 삼악군단장을 모두 쓰러뜨린 실력은 인정한다.
특별히 내 부하가 되어 마왕군을 이끌 기회를 주겠다.



용사

그럴 일은 없어!
난 너를 쓰러뜨리고 우리집으로 돌아갈거야!

FINAL BOSS 문제

아래와 같이 동작하도록 회로를 구성하고 구동하시오.

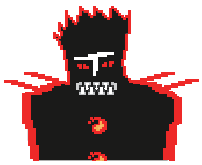
① 온도를 측정하여 일정 시간마다 시리얼 모니터에 출력한다.
② 출력하는 시간 간격은 포텐셔미터로 조정하며 최소 0.5초에서 최대 2초 간격으로 조절 가능하다.
③ 초기에는 LED가 켜져있고 이 때는 시리얼 모니터에 섭씨(℃)로 온도가 출력된다.
④ 스위치를 누르고 있을 때는 LED가 꺼지며 시리얼 모니터에 화씨(℉)로 온도가 출력된다.

LM35 센서값 섭씨(℃) 변환 공식 : 5.0 × 센서값 × 100.0 ÷ 1024

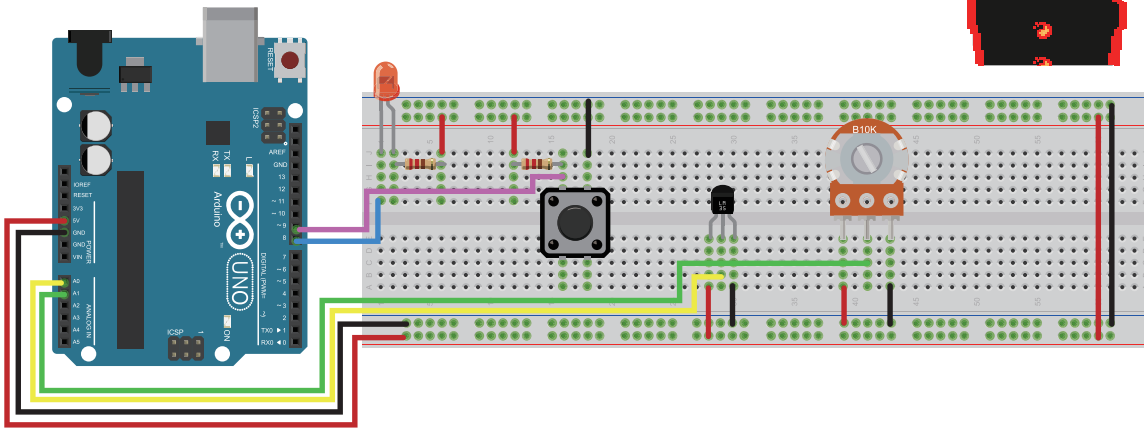
화씨온도 구하는 공식 : (섭씨온도 × 9/5) + 32

Quest 15. 정답 확인

정상적으로 구동되었다면 마왕에 X표 해주세요.



정답예제회로



정답예제소스

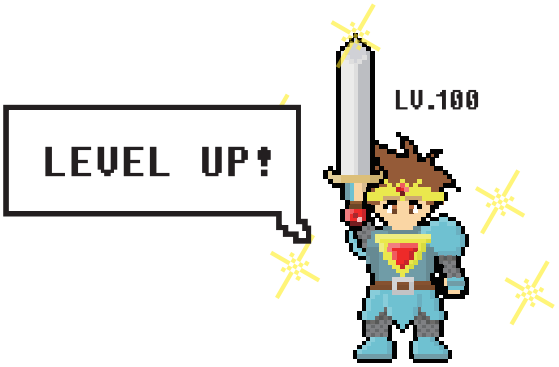
```
int led = 8;  // led를 9번핀에 연결
int sw = 9;  // 스위치를 8번핀에 연결

void setup()
{
  Serial.begin(9600);  // 통신속도 9600으로 통신 시작
  pinMode(led,OUTPUT);  // LED를 출력모드로 설정
  pinMode(sw,INPUT);  // LED를 출력모드로 설정
  digitalWrite(led, LOW);  // LED를 켜진상태로 초기화
}

void loop()
{
  int temp = analogRead(A0);  // 온도 값을 읽고 변수에 저장
  int temp_c = 5.0*temp*100.0/1024;  // 센서값 섭씨온도로 변환
  int temp_f = (temp_c*9/5)+32;  // 섭씨온도 화씨온도로 변환
  int key_scan = digitalRead(sw);  // 스위치 값을 읽고 변수에 저장
  int term = analogRead(A1);  // 가변저항 값을 읽고 변수에 저장
  term = map(term,0,1023,500,2000);  // 가변저항 값을 0.5에서 2초 범위로 변환

  if(key_scan == 1)  // 스위치가 눌리지 않았을 때
  {
```

```
    digitalWrite(led, LOW);   // LED를 킨다
    Serial.print(temp_c);   // 섭씨온도 출력
    Serial.println("℃");
  }
  else   // 스위치가 눌리고 있을 때
  {
    digitalWrite(led, HIGH);   // LED를 끈다
    Serial.print(temp_f);   // 화씨온도 출력
    Serial.println("℉");
  }
  delay(term);   // 가변저항에 의해 조절된 출력시간 간격
}
```

※ 회로와 코드가 정답예시와 동일할 필요는 없습니다. 문제의 동작과 동일하게만 동작한다면 정답입니다.

틀렸다면 정답예제 실습 후
스테이지 3부터 다시!

모두 맞추셨다면 다음페이지로!



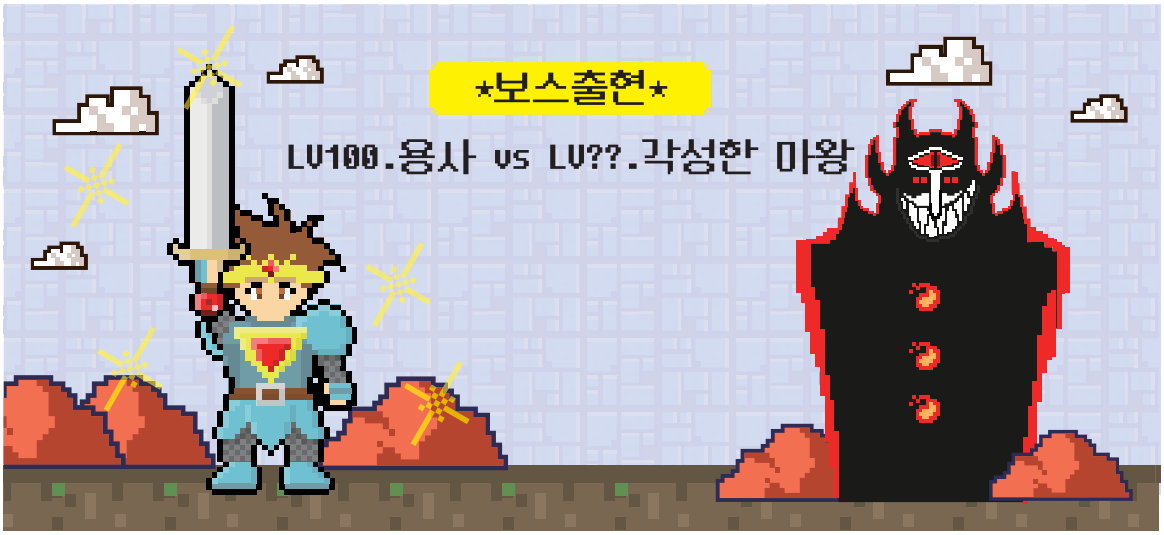
용사
헉...헉... 해치웠나?

마왕 프란츠
이대로... 이대로 끝날수는 없다....
내 모든 힘을 다해 널 쓰러뜨려주겠다!!

각성한 마왕 프란츠
이제 협상은 없다...
각오해라! 진정한 힘을 보여주마!

용사
역시 강해... 하지만 내 운명은 내가 정한다..!
비키지 않는다면 뚫고 지나가면 돼!

Final Boss : 각성한 마왕 프란츠



FINAL BOSS 문제

아래와 같이 동작하도록 회로를 구성하고 구동하시오.

① 리모컨의 1번부터 9번까지 눌렀을 때 버저에서 각각 [도, 미, 솔#, 라, 시, 높은도, 높은레, 높은레#, 높은미]의 음이 0.1초간 출력된다.

② LCD에 현재 출력되고 있는 음을 [Do, Mi, Sol#, La, Si, High Do, High Re, High Re#, High Mi]의 문자열로 출력한다.

③ 989895764, 1245, 2356, 989895764, 1245, 2654 순서대로 리모컨을 눌러 연주한다.

버저와 IR 리모컨은 타이머 2 인터럽트를 사용하기 때문에 컴파일하면 에러가 출력됩니다.

이 때 아두이노 라이브러리 폴더(스테이지 2.6.3 참고) ▶ [z3t0-Arduino-IRremote-2dc97b2] 폴더 ▶ [boarddefs.h]를 메모장을 이용하여 아래처럼 수정하여 저장한 뒤 컴파일해야 합니다.

```
#else
// Arduino Duemilanove, Diecimila, LilyPad, Mini, Fio, Nano, etc
// ATmega48, ATmega88, ATmega168, ATmega328
    //#define IR_USE_TIMER1    // tx = pin 9
    #define IR_USE_TIMER2      // tx = pin 3
```

수정 전

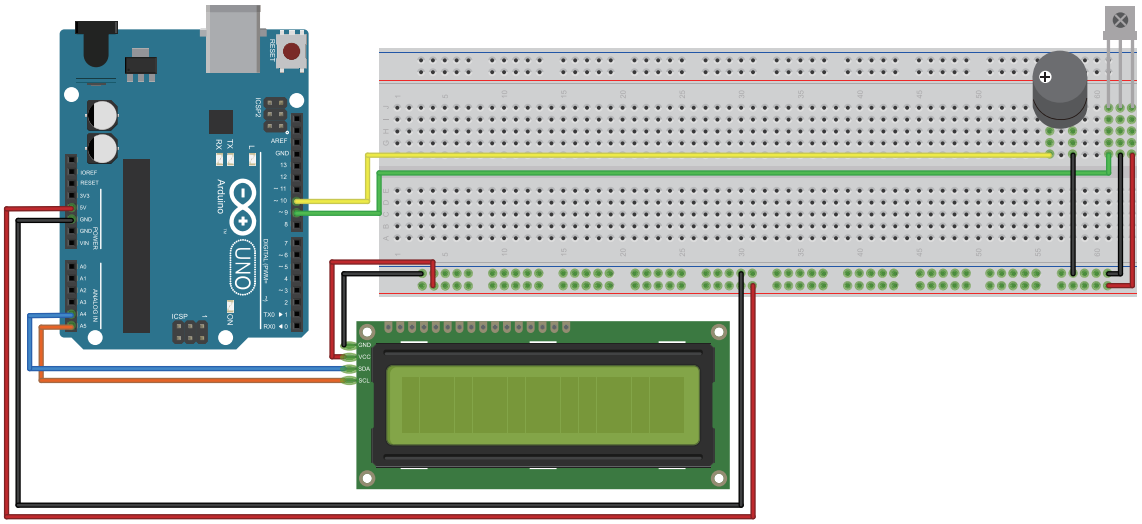
```
#else
// Arduino Duemilanove, Diecimila, LilyPad, Mini, Fio, Nano, etc
// ATmega48, ATmega88, ATmega168, ATmega328
    #define IR_USE_TIMER1    // tx = pin 9
    //#define IR_USE_TIMER2      // tx = pin 3
```

수정 후

Final Boss. 정답 확인

정상적으로 구동되었다면 마왕에 X표 해주세요.

정답예제회로



정답예제소스

```
#include <Wire.h> // I2C 통신을 위해 Wire 라이브러리 사용
#include <LiquidCrystal_I2C.h> // LiquidCrystal_i2c 라이브러리 사용
#include <IRremote.h>

int RECV_PIN = 9; // IR 수신기 D9에 연결
int buzzer = 10; // 버저를 핀 5번에 연결
int melody[]={262, 330, 415, 440, 494, 523, 587, 622, 659};

// 객체이름을 lcd로 선언 I2C 주소값을 0x27로 설정, 16 문자 2줄 디스플레이
// 만약 lcd가 출력이 안된다면, 주소값을 0x3F로 수정
LiquidCrystal_I2C lcd(0x27, 16, 2);

IRrecv irrecv(RECV_PIN); // IR리모컨 사용을 위한 클래스 생성
decode_results decodedSignal; // IR리모컨으로부터 받은 데이터

void setup()
{
  lcd.begin(); // lcd 시작
```

```
  lcd.backlight(); // lcd 백라이트 on
  irrecv.enableIRIn(); // 리모컨 수신 시작
}

void loop()
{
  if(irrecv.decode(&decodedSignal) == true) // IR 수신값이 있는지 판단.
  {
    if(decodedSignal.value == 0xFF30CF) // 1번이 눌렸을 때
    {
      tone(buzzer,melody[0],100);
      lcd.clear(); // lcd 초기화
      lcd.print("Do");
    }
 if(decodedSignal.value == 0xFF18E7) // 2번이 눌렸을 때
    {
      tone(buzzer,melody[1],100);
      lcd.clear(); // lcd 초기화
      lcd.print("Mi");
    }
    if(decodedSignal.value == 0xFF7A85) // 3번이 눌렸을 때
    {
      tone(buzzer,melody[2],100);
      lcd.clear(); // lcd 초기화
      lcd.print("Sol#");
    }
    if(decodedSignal.value == 0xFF10EF) // 4번이 눌렸을 때
    {
      tone(buzzer,melody[3],100);
      lcd.clear(); // lcd 초기화
      lcd.print("La");
    }
    if(decodedSignal.value == 0xFF38C7) // 5번이 눌렸을 때
    {
      tone(buzzer,melody[4],100);
      lcd.clear(); // lcd 초기화
      lcd.print("Si");
    }
```

```
    if(decodedSignal.value == 0xFF5AA5) // 6번이 눌렸을 때
    {
      tone(buzzer,melody[5],100);
      lcd.clear(); // lcd 초기화
      lcd.print("High Do");
    }

 if(decodedSignal.value == 0xFF42BD) // 7번이 눌렸을 때
    {
      tone(buzzer,melody[6],100);
      lcd.clear(); // lcd 초기화
      lcd.print("High Re");
    }
    if(decodedSignal.value == 0xFF4AB5) // 8번이 눌렸을 때
    {
      tone(buzzer,melody[7],100);
      lcd.clear(); // lcd 초기화
      lcd.print("High Re#");
    }
    if(decodedSignal.value == 0xFF52AD) // 9번이 눌렸을 때
    {
      tone(buzzer,melody[8],100);
      lcd.clear(); // lcd 초기화
      lcd.print("High Mi");
    }
    delay(100); // 0.1초 대기
    irrecv.resume();  // IR 다음 데이터 수신
  }
}
```

※회로와 코드가 정답예시와 동일할 필요는 없습니다. 문제의 동작과 동일하게만 동작한다면 정답입니다.

틀렸다면 정답예제 실습 후
스테이지 처음부터 다시!

모두 맞추셨다면 다음페이지로!

용사
힘든 싸움이었어...
이제 정말 집으로 가는 건가?