



드론으로 배우는
프로그래밍 교실

1h. 아두이노란?



01 아두이노와 친해지기



아두이노는 하드웨어를 다루기 쉽게 해주는 관련 도구 및 개발 환경입니다. 아두이노 이전에는 하드웨어에 대해 쉽게 접근하기 어려웠는데, 아두이노의 등장으로 손쉽게 하드웨어 제품을 만들 수 있게 되었습니다. 납땜할 필요 없이 핀을 이용하여 쉽게 하드웨어를 제작하고, 레지스터 설정이 필요 없이 쉽게 프로그래밍이 가능합니다. 아두이노와 함께 DIY세계로 빠져 봅시다.

아두이노란?

아두이노
정의

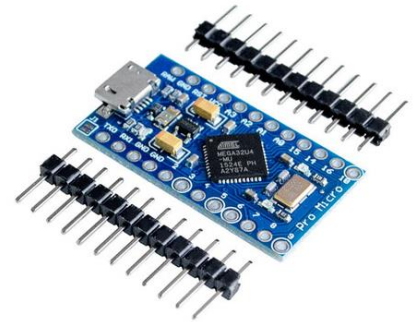
아두이노는 오픈 소스를 기반으로 한 단일 보드 마이크로컨트롤러로 완성 된 보드(상품)와 관련 개발 도구 및 환경을 말합니다.

쉽게 말해 마이크로 컨트롤러를 간편히 사용할 수 있게 제공되는 키트입니다.

아래 사진은 아두이노의 한 종류인 아두이노 우노와 아두이노 프로 마이크로입니다.



<그림1-7> 아두이노 우노



<그림1-8> 아두이노 프로 마이크로

링크 : <https://www.youtube.com/watch?v=Q4YeUspAjuU>

아두이노
DIY

아두이노는 마이크로컨트롤러를 간편히 제어할 수 있는 환경을 제공해주어 많은 사람들에게 환영을 받았습니다.

아두이노 DIY도 굉장히 많이 있어서, 여러분 또한 여러분의 아이디어를 현실로 만들어 낼 수 있을 겁니다.

꿀TIP

DIY란?

Do It Yourself의 줄임말로, 스스로 만들어 내는 것을 의미합니다.



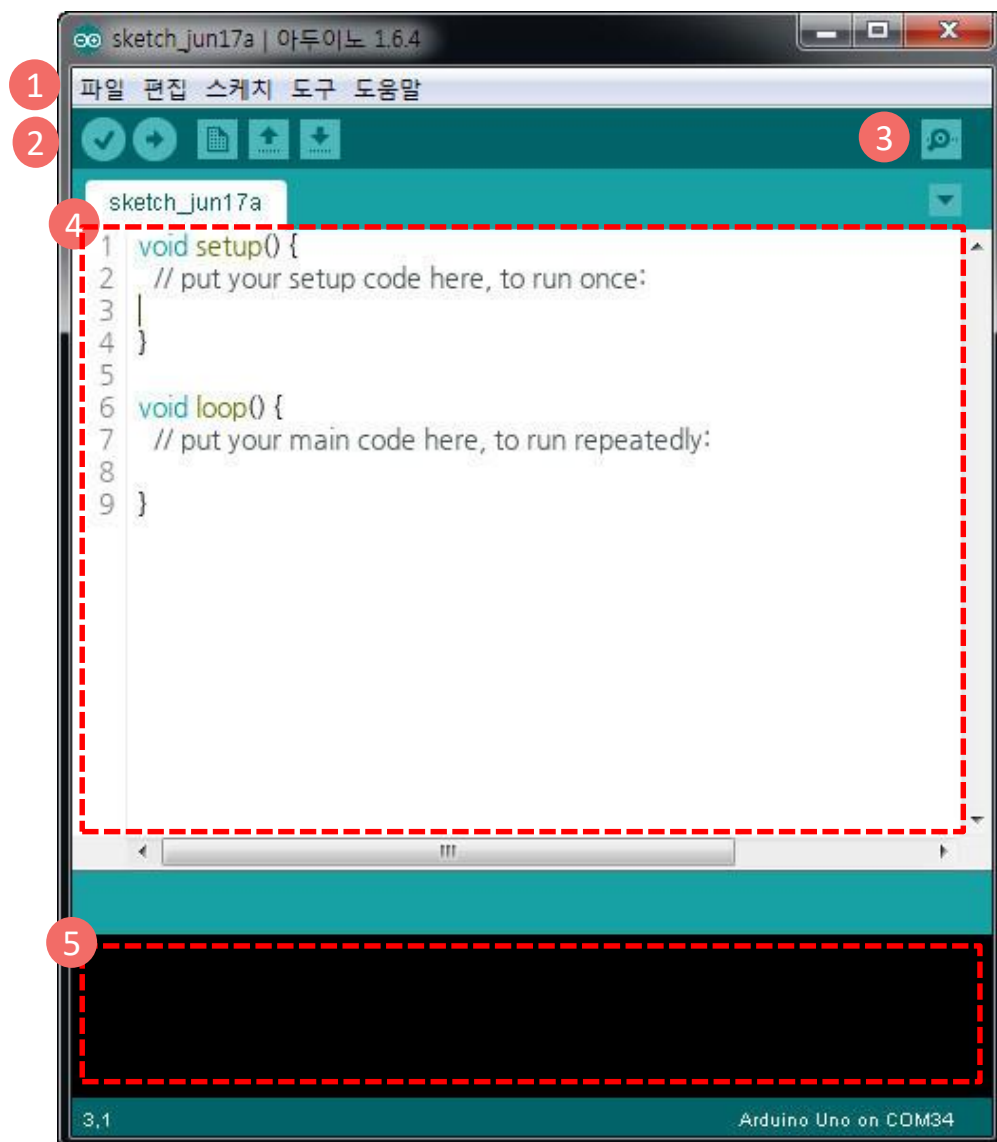
링크 : https://www.youtube.com/watch?v=t_Za7G38YFU

<그림1-9> 아두이노 RC카



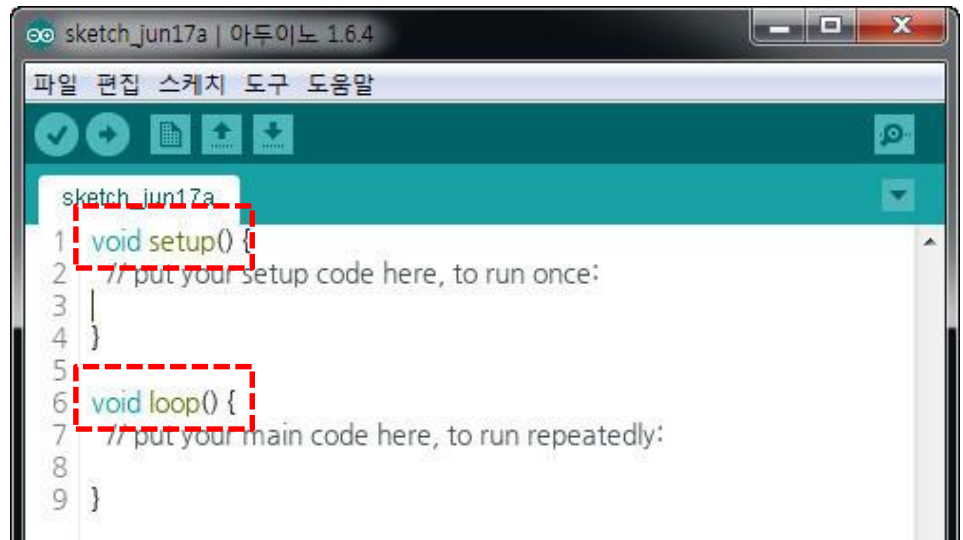
<그림1-10> 키위 드론

스케치 구성



<그림2-2> 스케치 구성

- 1** 메뉴 바 : 파일을 불러오거나 저장하고, 설정 등을 할 수 있습니다.
- 2** 툴 바 : 코드를 컴파일하고 보드에 업로드 합니다.
- 3** 시리얼 모니터 : 아두이노와 메시지를 주고 받을 수 있습니다.
- 4** 텍스트 에디터 창 : 프로그램을 만들 수 있습니다.
- 5** 콘솔 창 : 업로드 결과 및 에러 등을 확인할 수 있습니다.

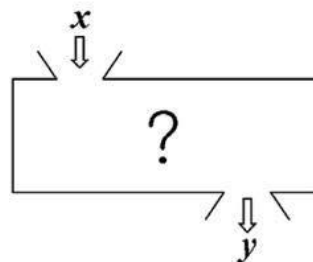
아두이노
기초 함수

<그림3-1> setup()과 loop()

Setup()과 loop()는 아두이노의 기초 함수입니다.

함수란?

함수는 입력에 대해 어떤 처리를 거쳐 나오는 출력의 과정을 의미합니다.



<그림3-2> 함수의 입력, 처리, 출력

함수는 코드의 뭉텅이로 이름을 가지고 있고, 명령어로 구성된 블록입니다. 함수는 자신의 이름이 불렸을 때 실행됩니다.

이미 만들어져 있는 함수도 있고, 우리가 함수를 만들 수도 있습니다. 함수는 반복되는 작업을 수행할 수 있고, 코드를 깔끔하게 볼 수 있게 해 줍니다.

함수 구조

예를 들어 덧셈을 하는 함수가 있다면 입력으로 두 숫자를 받은 후 더하기 처리를 하여 결과를 출력합니다.

함수 이름
add

정수 a, b 입력

정수 출력

```
int add(int a, int b){  
    return a+b;  
}
```

a, b의 합 출력

<그림3-3> 함수의 구조

setup()과 loop() 앞에 있는 void는 아무것도 출력하지 않는다는 뜻입니다.

함수 사용

실제로 함수를 사용할 경우 함수의 이름과 입력 값을 괄호안에 적으면 됩니다.

```
sketch_feb02a  
1 void setup() {  
2   // put your setup code here, to run once:  
3   add(4, 5);  
4 }  
5  
6 void loop() {  
7   // put your main code here, to run repeatedly:  
8  
9 }  
10  
11 int add(int a, int b){  
12   return a+b;  
13 }  
14
```

<그림3-4> 함수의 실제 사용

기타

```

1 void setup() {
2   // put your setup code here, to run once:
3 }

```

<그림3-5> 프로그래밍 기본 상식

- ◆ { } 는 중괄호로 함수의 범위를 지정해 줍니다. 만약, 중괄호가 없을 경우 어디까지가 함수의 끝인지 알 수 없게 됩니다.
중괄호는 { 로 시작하여 }로 끝이 나는데, 항상 쌍을 이룹니다. 만약 중괄호가 한 짝 밖에 없다면 에러가 나게 됩니다.
- ◆ // 는 문장 주석입니다. 한 줄의 문장을 주석으로 처리하여 명령어처럼 실행되지 않게 합니다.
예를 들어
// 이렇게 되어 있으면 이 문장은 실행되지 않습니다.
- ◆ /* ... */ 는 블록 주석입니다. /*으로 시작하여 */로 끝나는데, 이 범위안에 있는 코드는 실행되지 않습니다. 코드에 대한 해석이나 의견을 적을 때 사용하게 됩니다.
/* 이렇게 블록 주석으로 쌓여 있으면
문장이 명령어처럼 실행되지 않습니다. */
- ◆ ; 는 세미콜론입니다. 세미콜론은 명령어 문장의 끝을 나타냅니다. 예를 들어 변수 x를 지정하고 x에 5를 집어 넣는 명령을 할 경우
int x = 5;
와 같이 문장을 작성하게 되는데, 세미콜론을 빼먹는다면 문장의 끝이 어딘지 모르게 돼 에러가 납니다.

02 아두이노와 LED



아두이노에 프로그램을 어떻게 전달할까요?

아두이노와 컴퓨터를 연결해주는 것은 하드웨어적으로는 USB케이블이 있고, 소프트웨어적으로는 드라이버가 있습니다.

아두이노라는 장치를 컴퓨터에서 인식하기 위해서는 드라이버가 깔려 있어야 합니다.

USB연결, 보드 및 포트 설정

USB 연결

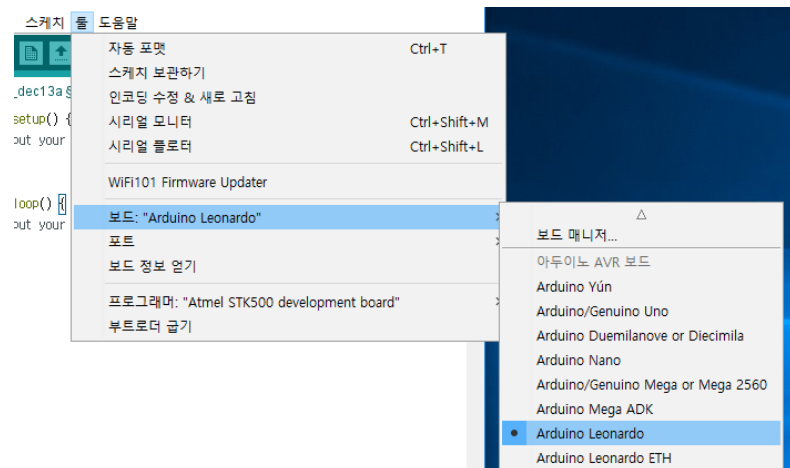
- 1 아두이노와 컴퓨터를 USB케이블을 통해 연결합니다.



<그림1-1> 아두이노 USB 연결

보드 설정

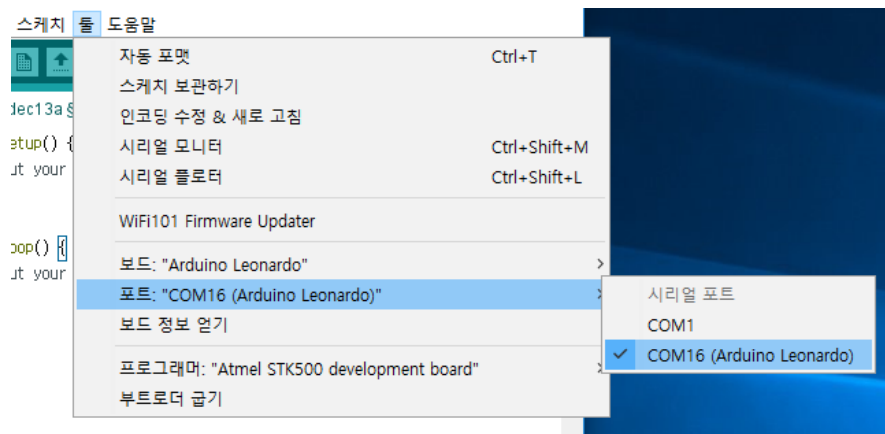
- 2 메뉴 바의 도구(툴) – 보드 – Arduino Leonardo를 선택합니다.



<그림1-2> 아두이노 보드 선택

포트 설정

- 3 메뉴 바의 도구 – 포트 – 아두이노의 COM을 선택합니다.



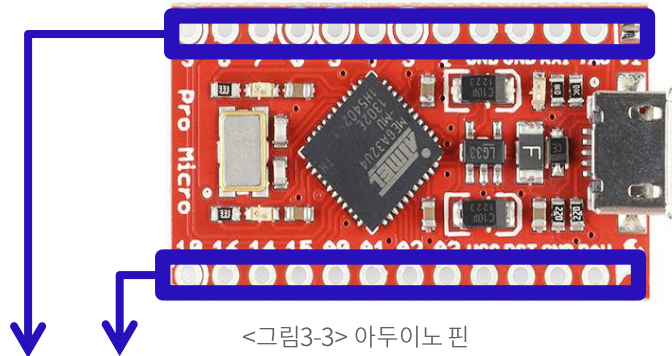
<그림1-3> 아두이노 포트 선택

- 4 업로드를 클릭하여 프로그램을 아두이노에 넣습니다.



아두이노
핀 구성

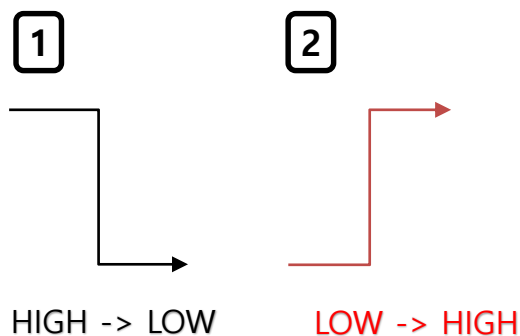
아두이노의 좌우로 매겨져 있는 번호를 아두이노의 핀이라고 합니다. 이 핀을 통해서 전류를 내보내 모터를 제어하거나, 전압을 읽어서 센서 값을 받아 들입니다.



<그림3-3> 아두이노 핀

아두이노의 핀을 통해서 원하는 명령을 내릴 수 있습니다. 크게 나누면 핀을 통해서 입력을 받거나 출력을 내보냅니다.

핀은 크게 digital pin과 analog pin으로 구분할 수 있는데, digital pin의 경우 LOW인 상태와 HIGH인 상태를 가집니다. LOW는 전압이 낮은 상태이고 HIGH는 전압이 높은 상태입니다.



<그림3-4> LOW와 HIGH인 상태

예를 들어

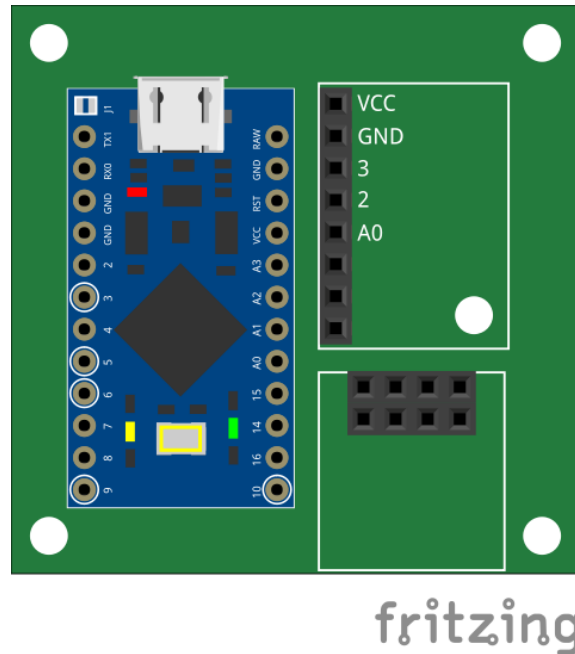
digitalWrite(3,HIGH);

와 같은 명령어의 경우 3번 핀을 HIGH로 만들어 전압을 높은 상태로 출력하라는 뜻입니다.

베이스 보드 핀 구성

베이스 보드

베이스보드는 내부적으로 아두이노의 핀들과 연결되어 있습니다. 그 중에서 우리는 VCC, GND, 3, 2, A0 핀을 사용할 것입니다.



<그림3-5> 아두이노와 베이스 보드 연결

베이스보드 핀 특징

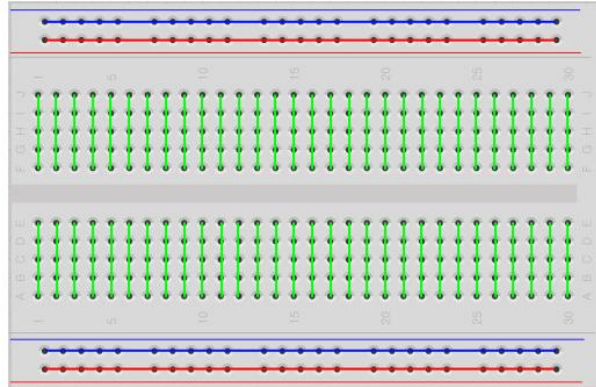
베이스보드의 핀들은 아두이노의 핀들과 연결되어 있고, 이들은 각각의 특징을 가지고 있습니다.

- VCC는 5V의 전압을 유지합니다.
- GND는 0V의 전압을 유지합니다.
- 3번핀은 디지털 입출력과 아날로그 출력으로 사용 가능합니다
- 2번핀은 디지털 입출력으로 사용 가능합니다.
- A0핀은 아날로그 값을 읽어 들일 수 있습니다.

핀과 관련된 내용은 뒷장에서 차근차근 배워볼 것입니다. 지금은 베이스보드에 이런 핀들이 있구나 정도만 알아두면 충분합니다.

브레드보드

브레드보드



<그림3-6> 브레드보드

브레드보드는 빵판이라고도 불리며, 시제품을 만드는 데 쓰는 재활용할 수 있는 무 땀납 장치입니다.

중간의 **초록 선들** 구역은 세로로 연결되어 있습니다. 세로로 5칸 씩 구멍이 있고 부품 영역이라고 합니다.

위의 5칸과 아래의 5칸은 서로 전기가 통하지 않습니다.

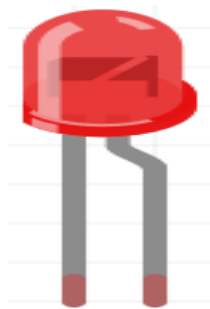
파란 선 : GND(접지선)입니다.

빨간 선 : VCC(전압)입니다.

초록 선 : 세로로 같은 핀을 나타냅니다.

LED

발광 다이오드, LED(Light Emitting Diode)



<그림3-7> LED

긴 쪽 : +

발광 다이오드는 순방향으로 전압을 가할 때 발광하는 반도체 소자입니다.

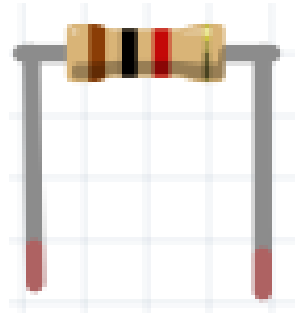
전류량에 비례해서 빛이 발생합니다.

긴 쪽을 +(전압이 높은 부분)과 연결해야 합니다.

저항이란?

저항

저항



<그림3-8> 탄소 피막 저항

저항은 전류의 흐름을 방해하는 정도를 나타내는 물리량입니다.

저항의 주된 용도는

- 1) 전류량 제어
- 2) 전압의 분배입니다.

옴의 법칙

옴의 법칙

$$I = \frac{V}{R}$$

<그림3-9> 옴의 법칙

옴의 법칙은 전압과 전류, 저항 사이의 관계를 나타낸 식입니다.

전압이 일정하다는 가정 하에 저항이 커지면 전류가 작아지게 됩니다.

저항의 종류

저항의 종류



<그림3-10> 권선 저항



<그림3-11> 탄소 피막 저항



<그림3-12> 가변 저항

종류	설명&용도	장점&단점
권선 저항	도선을 길게 만든 저항	단순한 구조 고주파 회로에서 잡음 발생
탄소 피막 저항	DIP타입으로 제작 피막의 흠으로 저항크기 조절	저렴한 가격 온도에 따른 저항 값 변화가 크고 노이즈有
가변 저항	저항 값이 변하는 저항, 라디오 등 볼륨조절기에 사용됨	다양한 용도로 쓰일 수 있습니다.

<그림3-13> 저항 종류 비교

04 스위치



스witch는 사용자가 눌렀을 때와 아닐 때를 구분하여 각각의 경우에 따라 다른 명령을 실행할 수 있게 해줍니다.

여러분의 컴퓨터 본체에도 스위치가 달려있고, 핸드폰에도 스위치가 달려 있습니다.

스위치의 작동 원리에 대해 살펴봅시다.

스위치란?

스위치
정의

스witch는 전기 회로를 이었다 끊었다 하는 장치로 보통 전등, 라디오, 텔레비전 따위의 전기 기구를 손으로 올리고 내리거나 누르거나 틀어서 작동하는 부분을 일컫는다.

출처: 네이버 백과사전



<그림1-1> 스위치 회로도

스witch는 쉽게 말해 연결해주는 장치입니다. 사용자의 선택에 따라 연결할 수도 끊을 수도 있습니다.

스위치
종류

스witch는 크게 탭트 스위치, 슬라이드 스위치, 푸쉬 스위치, 훅 스위치, 멀티웨이 스위치로 나뉩니다. 훅 스위치는 보통 전화기에 사용되어 무게에 따라 스위치가 동작하고, 멀티웨이 스위치는 보통 조이스틱에 사용됩니다.



<그림1-2> 탭트 스위치



<그림1-3> 슬라이드 스위치



<그림1-4> 푸쉬 스위치



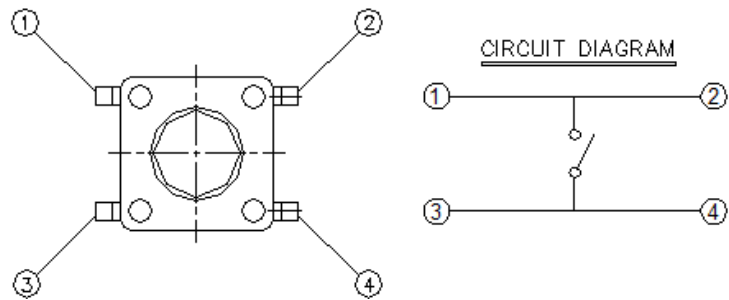
<그림1-5> 훅 스위치



<그림1-6> 멀티웨이 스위치

택트 스위치

택트 스위치는 가장 흔한 스위치로 버튼 혹은 key라고도 불립니다.



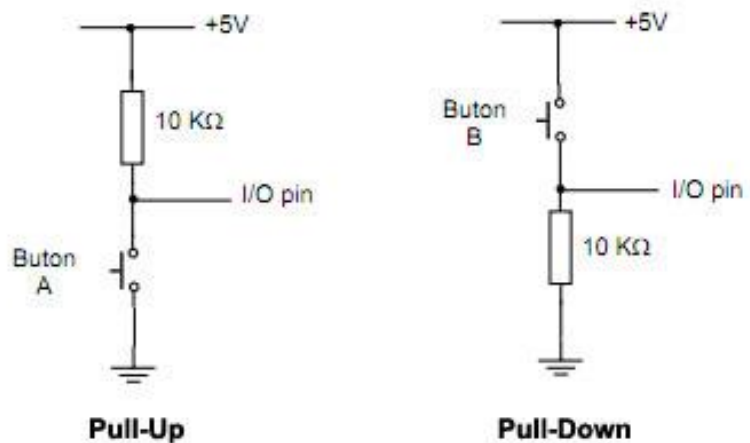
<그림1-7> 스위치 회로도

우리가 사용할 택트 스위치는 다리가 4개로, 1번과 2번이 기본적으로 연결되어 있고, 3번과 4번이 기본적으로 연결되어 있습니다.

이 때, 스위치를 누르게 되면 1번과 4번 혹은 1번과 3번을 연결할 수 있게 됩니다.

풀업 풀다운

아두이노와 스위치의 연결에는 풀업 방식과 풀다운 방식이 있습니다. 저항이 위에 있으면 풀 업, 아래 있으면 풀 다운입니다.



<그림1-8> 풀업 풀다운

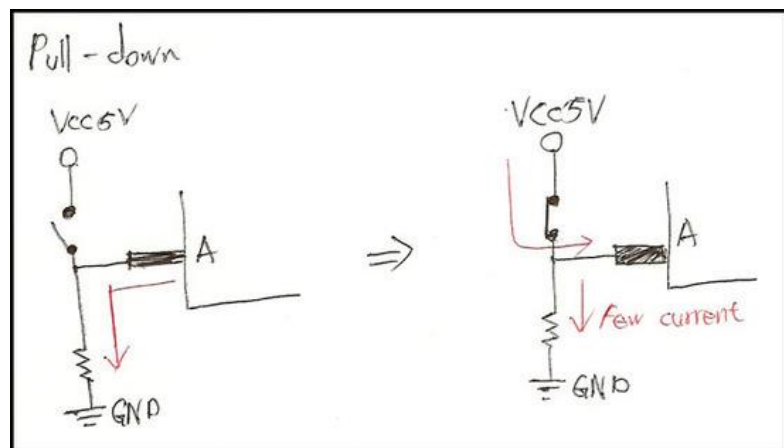
두 가지 방식 중 우리는 풀다운 방식을 사용할 것입니다.

풀다운 방식

스위치는 보통 두 선의 연결을 이어주거나 끊어줄 때 쓰이기도 하지만, 아두이노와 함께 쓰일 때는 아두이노 핀에 입력을 줄 때 쓰이기도 합니다.

아두이노의 핀은 디지털 핀과 아날로그 핀으로 나뉘고, 디지털핀은 받아들이는 전압이 높으면 1, 낮으면 0으로 입력을 받습니다.

풀다운방식을 사용하면 스위치를 누르지 않았을 경우 0, 스위치를 누를 경우 1이 입력으로 들어가게 됩니다.



<그림1-9> 풀다운 저항

위 그림과 같이 풀다운 방식으로 회로를 구성하여, 스위치가 눌리지 않았을 때는 입력핀이 GND와 연결되어 전압이 낮은 상태(LOW)로 유지됩니다.

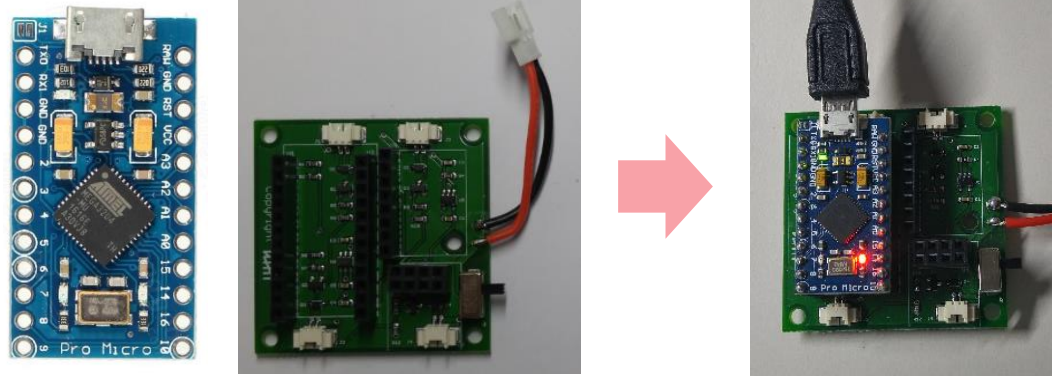
스위치가 눌릴 때는 VCC가 입력핀과 연결되어 입력핀은 VCC와 같이 전압이 높은 상태(HIGH)로 유지됩니다.

이를 통하여 `digitalRead(2)`를 했을 때 상태가 HIGH인지 LOW인지 구분하여 스위치의 눌림을 감지할 수 있게 됩니다.

스위치 입력 받기 - 연결

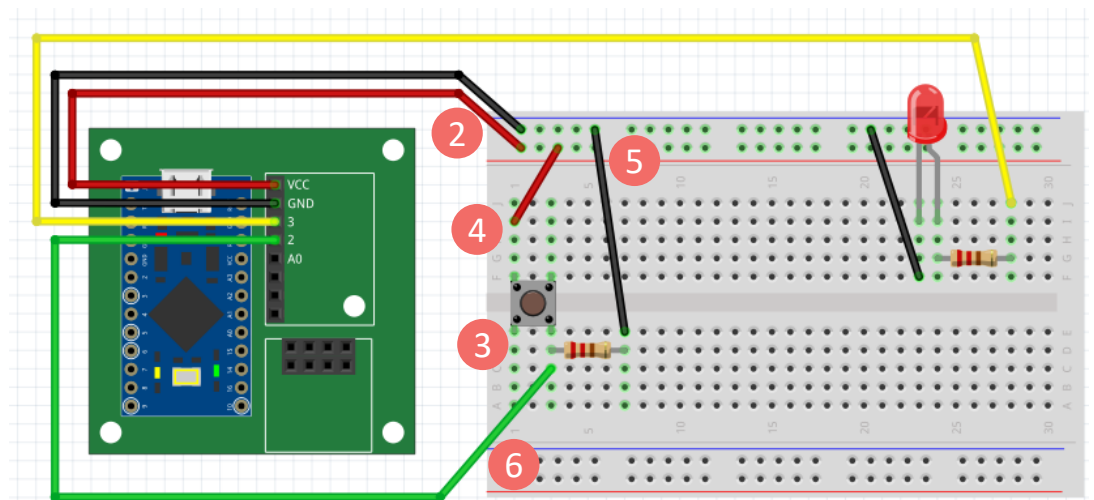
스위치 입력 받기 연결

- 1 아두이노를 베이스 보드에 끼우고 USB를 연결합니다.(방향에 유의합니다)



<그림1-10> 아두이노와 베이스 보드 연결

- 2 베이스 보드의 VCC핀(위에서1번째)을 빵판의 빨간줄에 꽂습니다.
- 3 스위치와 저항을 베이스보드에 그림과 같이 꽂아 넣습니다. (세로로 같은 라인에 꽂아야 합니다.)
- 4 스위치의 좌측상단 다리와 빵판의 빨간줄을 연결합니다.
- 5 저항의 오른쪽 끝을 빵판의 파란줄과 연결합니다.
- 6 베이스보드의 2번핀(위에서 4번째)을 스위치와 저항이 같이 연결된 부분에 연결합니다.



<그림1-11> 베이스 보드와 스위치 연결

스위치 입력 받기 - 코드

스위치 입력 받기 코드

- 1 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.


```
ch3_4_1_sw_serial
1 void setup() {
2   Serial.begin(9600);
3   pinMode(2, INPUT);
4 }
5
6 void loop() {
7   if(digitalRead(2) == HIGH){
8     Serial.println("원하는 문구 적기");
9   }
10 }
```

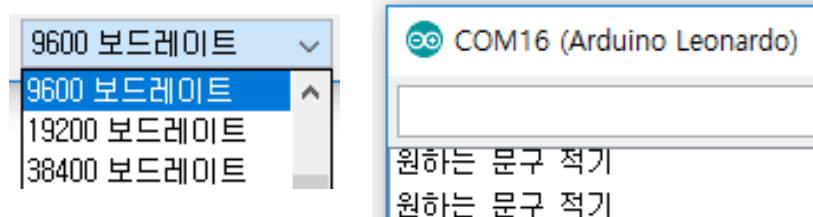
<그림1-12> 스위치 입력 받기

스위치 입력 받기 코드 해석

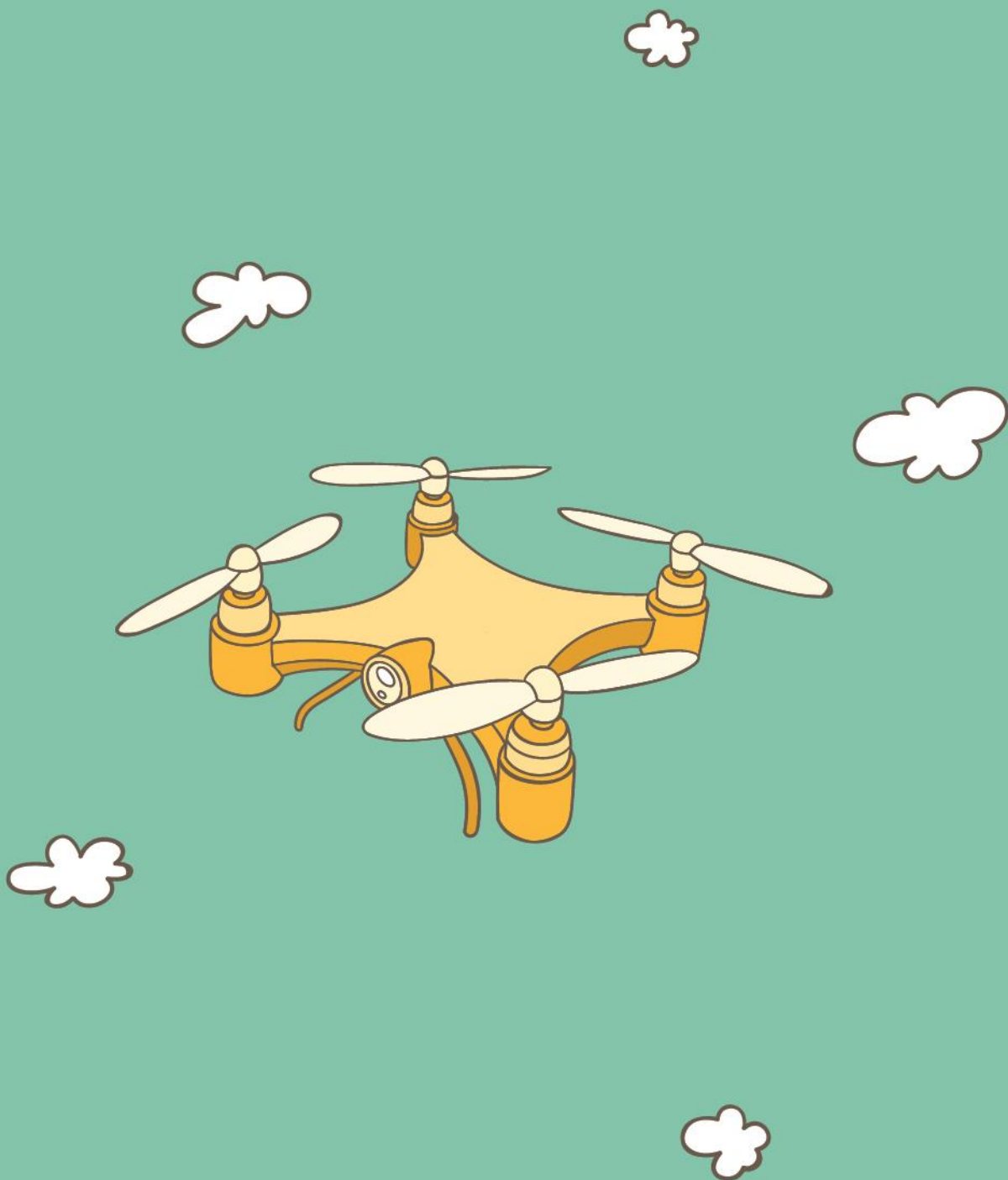
```
void setup() {
  Serial.begin(9600);
  pinMode(2, INPUT); //2번 핀을 입력으로 설정
}
```

```
void loop() {
  if(digitalRead(2) == HIGH){//만약 2번핀이 HIGH이면 실행
    Serial.println("원하는 문구 적기"); //원하는 문구 출력
  }
}
```

- 2  버튼을 눌러 시리얼 모니터를 켭니다.
- 3 보드레이트를 맞춘 후 스위치를 누르고 시리얼 모니터에서 문구가 뜨는 것을 확인합니다.



<그림1-13> 문구 확인



WHIT

LED 깜빡이기 예제 작성하기

LED
깜빡이기

- ① 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.

```
ch3_2_1_blink
1 void setup() {
2   pinMode(3, OUTPUT);
3 }
4
5 void loop() {
6   digitalWrite(3, HIGH);
7   delay(1000);
8   digitalWrite(3, LOW);
9   delay(1000);
10 }
```

<그림2-2> LED Blink 예제

```
void setup(){
  pinMode(3, OUTPUT); //3번 핀을 출력으로 설정
}

void loop()
{
  digitalWrite(3, HIGH); //3번 핀을 HIGH로 출력
  delay(1000);           // 1초간 멈춤
  digitalWrite(3, LOW);  //3번 핀을 LOW로 출력
  delay(1000);           // 1초간 멈춤
}
```

꿀TIP

프로젝트 저장

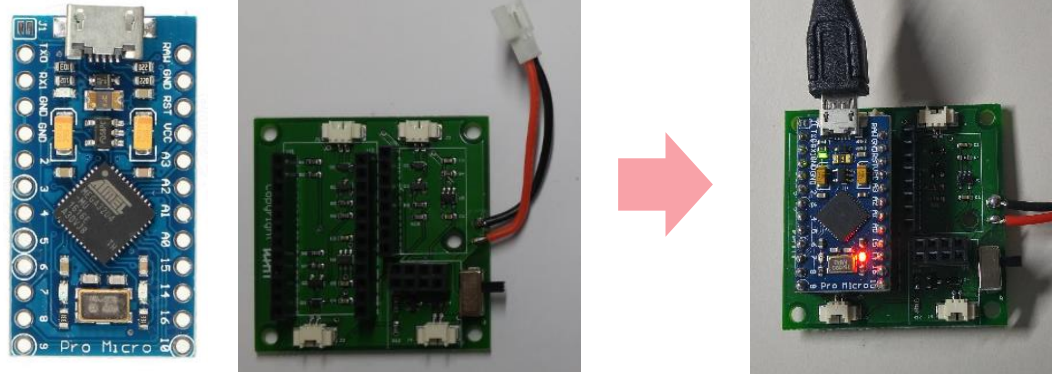
업로드 시 프로젝트를
저장할 지 물어보는데,
저장을 해 두는게
좋습니다.

LED
깜빡이기
해석

LED 하드웨어 구성

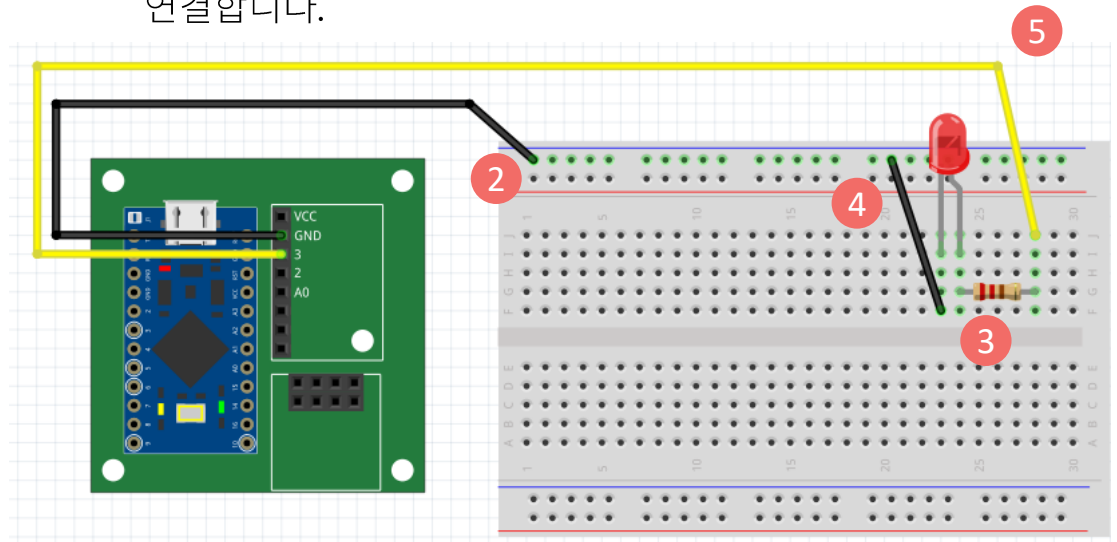
최종 구성

- 1 아두이노를 베이스 보드에 끼우고 USB를 연결합니다.(방향에 유의합니다)



<그림3-1> 아두이노와 베이스 보드 연결

- 2 베이스 보드의 GND핀(위에서2번째)을 빵판의 파란줄에 꽂습니다.
- 3 LED와 저항을 베이스보드에 그림과 같이 꽂아 넣습니다. (세로로 같은 라인에 꽂아야 합니다.)
- 4 LED의 -(짧은 쪽)와 빵판의 파란줄을 연결합니다.
- 5 베이스보드의 3번핀(위에서 3번째)을 저항의 한 쪽에 연결합니다.



<그림3-2> 베이스 보드와 LED 연결

- 6 만약, LED가 깜빡이지 않으면, 업로드가 되었는지, 선 연결이 잘 되었는지 확인합니다.

꿀TIP

LED 연결

LED는 다리가 긴 쪽이 +입니다.
전기는 +에서 -로 흐릅니다.

03 Serial 통신



Serial 통신은 아두이노 통신의 기초입니다.

아두이노는 Serial통신 방식을 사용하여 컴퓨터와 의사소통을 하게 됩니다.

아두이노에 프로그램을 업로드할 때 뿐 아니라, 아두이노에 들어있는 데이터를 컴퓨터 상에서 확인할 때에도 Serial통신을 사용하게 됩니다.

Serial통신을 숙지하여 아두이노와 즐겁게 이야기 해 봅시다.

시리얼 통신이란?

Serial
통신

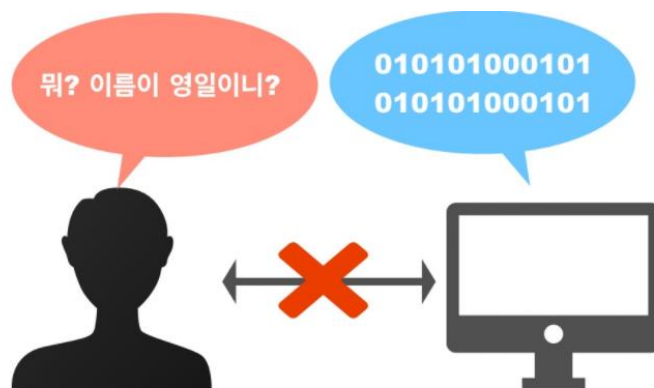
아두이노와 대화를 하려면 어떻게 해야 할까요?
컴퓨터와 의사소통을 하기 위해선 모니터, 키보드, 마우스 같은 도구가 필요합니다.

아두이노에서는 주로 시리얼통신이라는 방법을 통해 아두이노와 정보를 주고 받을 수 있습니다.

Serial
통신

Serial 통신은 영어 해석 그대로 직렬 통신으로, 한번에 한 비트씩 보내는 통신 방식입니다.

아두이노는 한국어를 알아듣지 못하기 때문에, 모든 정보를 0과 1의 비트 단위로 보내주어야 합니다.



<그림1-1> 사람과 아두이노의 언어 차이

Serial
통신

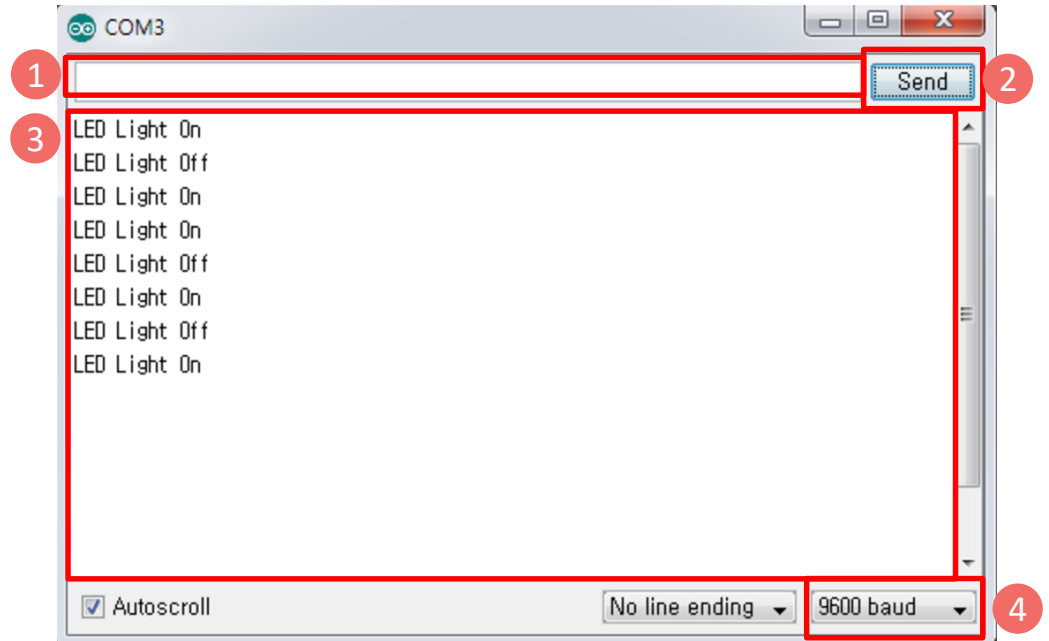
우리가 직접 모든 데이터를 0과 1로 바꿀 순 없습니다.
이런 작업을 편하게 해 주는게 시리얼 통신 함수입니다.

함수	설명
Serial.begin()	Baudrate를 정하여 통신을 알립니다.
Serial.available()	아두이노 버퍼에서 64byte까지 읽어오며 읽어올 byte가 없을 시 -1을 반환합니다.
Serial.read()	아두이노 버퍼에서 1byte만큼 데이터를 읽은 뒤 삭제하며 읽어올 byte가 없을 시 -1을 반환합니다.
Serial.write(val)	val 안 데이터를 TX핀을 통해 송신 후, 데이터의 크기를 반환합니다.

시리얼 모니터



시리얼 통신으로 주고 받는 데이터는 시리얼 모니터를 통해서 확인할 수 있습니다.

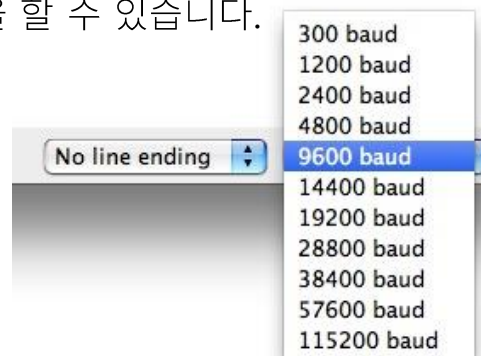


<그림1-2> 시리얼 모니터

- ① 텍스트 입력 창 : 아두이노로 보낼 데이터를 입력할 수 있습니다.
- ② 전송 : 아두이노로 데이터를 전송합니다.
- ③ 콘솔 창 : 아두이노로부터 받은 데이터를 확인할 수 있습니다.
- ④ 보드레이트 : 아두이노와의 통신 속도를 정할 수 있습니다.

보드레이트 (Baudrate)

보드레이트는 시리얼 통신을 할 때의 통신 속도를 나타냅니다. 보내는 쪽과 받는 쪽에서 속도가 같아야지만 서로 통신을 할 수 있습니다.



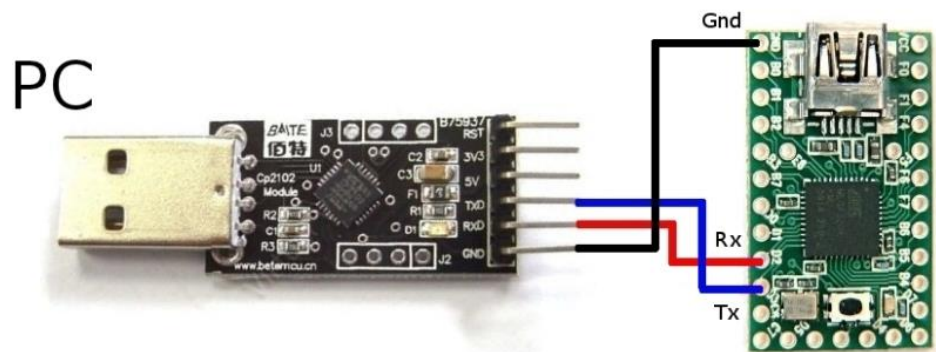
<그림1-3> 시리얼 모니터의 보드레이트 설정

HW Serial SW Serial

HW Serial 통신 특징

아두이노의 시리얼 통신에는 하드웨어 시리얼과 소프트웨어 시리얼이 있는데, 우리는 하드웨어 시리얼만 다룰 것입니다.

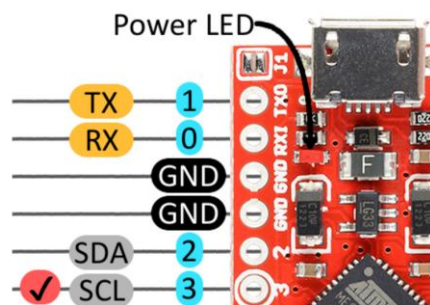
- 기본적으로 1:1 통신 방식입니다.
- 한 번에 한 비트씩 통신하는 직렬 통신입니다. (여러 비트를 동시에 보내는 병렬 통신과 상반된 개념)
- 시리얼 통신에는 USB 또는 RX, TX선이 사용됩니다.
- 시리얼 통신 시 디지털 0, 1번 핀은 사용 불가합니다.



<그림1-4> USB, RX, TX 연결

0, 1번 핀 사용불가

시리얼 통신을 할 때에는 아두이노의 0번 핀과 1번 핀을 사용할 수 없습니다. 그 이유는 바로 0번 핀과 1번 핀이 RX, TX와 같은 핀을 사용하기 때문입니다.



<그림1-5> 0, 1번 핀과 RX, TX 중첩


Serial통신 실습하기

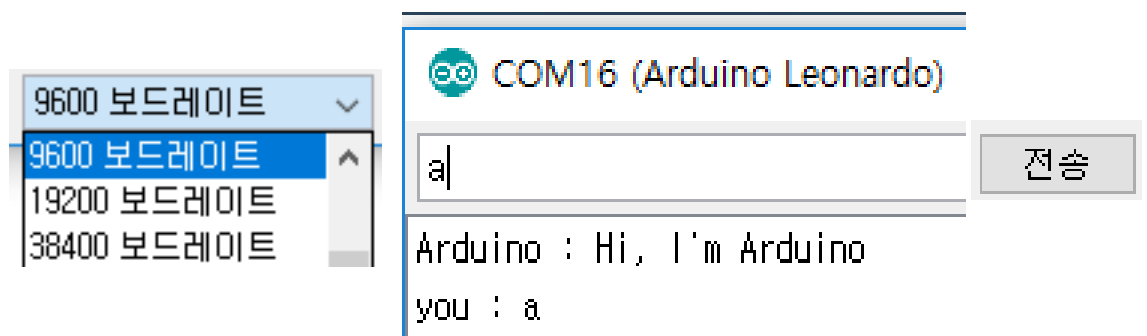
시리얼 통신 해보기

- 1 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.

```
ch3_3_1_serial
1 void setup() {
2   Serial.begin(9600);
3 }
4
5 void loop() {
6   if (Serial.available()) {
7     Serial.write("Arduino : Hi, I'm Arduino");
8     Serial.write('\n');
9
10    Serial.write("you : ");
11    Serial.write(Serial.read());
12    Serial.write('\n');
13  }
14 }
```

<그림1-6> 시리얼 통신 코드

- 2  버튼을 눌러 시리얼 모니터를 켭니다.
- 3 보드레이트를 맞춘 후 원하는 알파벳을 적고 전송 버튼을 클릭합니다.



<그림1-7> 시리얼 통신 확인

- 4 되돌아온 문구를 확인합니다.

시리얼 통신 해석

```
void setup() {  
  Serial.begin(9600); //9600의 보드레이트로 시리얼 통신 실행  
}  
  
void loop() {  
  if (Serial.available()) { //만약 사용자의 입력이 있다면  
    Serial.write("Arduino : Hi, I'm Arduino"); //문구 출력  
    Serial.write('\n'); //줄 바꿈 표시  
  
    Serial.write("you : "); //문구 출력  
    Serial.write(Serial.read()); //사용자의 입력을 출력  
    Serial.write('\n'); //줄 바꿈  
  }  
}
```

시리얼 통신으로 아두이노와 대화를 해봤습니다. 감이 좀 잡히시나요??
시리얼 통신은 아두이노에 들어있는 데이터를 확인하거나 에러를 체크할 때 유용하게 쓰입니다.

아두이노야 말해봐

> 아두이노가 원하는 말을 할 수 있도록 코딩해보세요!!

```
ch3_3_1_serial2  
1 void setup() {  
2   Serial.begin(9600);  
3 }  
4  
5 void loop() {  
6   if (Serial.available()) {  
7     Serial.write("Arduino : 원하는 말을 적어보세요");  
8     Serial.write('\n');  
9   }  
10 }
```

<그림1-8> 아두이노야 말해봐