



드론으로 배우는
프로그래밍 교실

Ch4-1. 디지털과 아날로그



01 가변저항에 대해서	01
가변저항이란?	02
전압 분배	03
가변저항 하드웨어 구성	04
가변저항 예제 작성하기	05
02 digital과 analog	08
아두이노의 digital과 analog	09
디지털 실습 해보기	10
Analog to Digital Converter	12
Pulse Width Modulation	13
아날로그 실습 해보기	14
03 상수에 대해서	16
상수란?	17
#define과 const	18
상수 작성 해보기	19



드론으로 배우는
프로그래밍 교실

초판발행 2016년 9월 23일
지은이 이상준 | 펴낸이 CodingBird
펴낸곳 WHIT | 주소 안산시 한양대학교55 창업보육센터 B01

Published by WHIT. Printed in Korea
Copyright © 2016 CodingBird & WHIT

이 책의 저작권은 CodingBird와 WHIT에 있습니다.
저작권법에 의해 보호를 받는 저작물이므로
무단 복제 및 무단 전재를 금합니다.

01 가변저항에 대해서



가변저항은 간단하게 저항 값을 바꿀 수 있어서 여러가지 용도로 사용되어집니다.

라디오 볼륨조절이나 티비 볼륨 조절 등에도 쓰이며, 저항 값 테스트 용도로도 많이 쓰입니다.

이러한 가변저항을 다루는 방법에 대해 알아보시다.

가변저항이란?

가변저항

가변저항은 값이 변하는 저항입니다. 저항은 전압 또는 전류값을 조절해 줄 때 사용합니다.

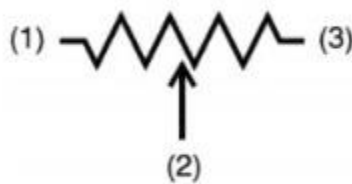
스위치의 경우 단순히 전기가 흐른다 / 흐르지 않느냐를 제어할 수 있었는데, 가변 저항에서는 전기가 흐르는 양을 제어할 수 있습니다.

이렇게 전기의 흐름(전류)가 바뀌면 그에 따라 전압이 달라지게 됩니다.



<그림1-1> 가변 저항

가변저항의 회로도에는 다음과 같이 저항 회로도에 화살표가 있는 형태입니다. (2)번의 위치에 따라 저항값이 변하게 됩니다.



<그림1-2> 가변 저항 회로도

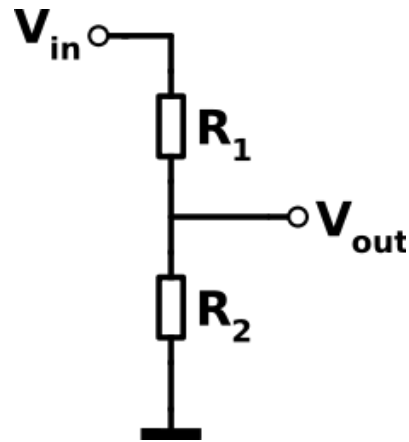
가변저항으로 저항값이 바뀔 때에는 전압 분배의 원리가 적용됩니다.

저항값이 바뀔 때 따라 중간 지점에서 확인되는 전압이 달라지게 됩니다.

달라지는 전압값을 아두이노에 확인할 수 있습니다.

전압 분배란?

전압 분배를 회로로 나타내면 다음과 같습니다.

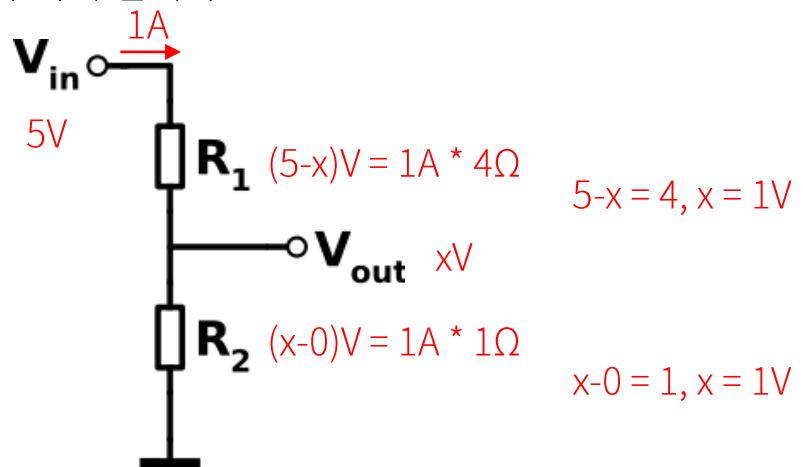


<그림1-3> 전압 분배 회로도

아두이노에서 전압은 보통 5V로 V_{in} 은 5V입니다.
이 때 R_1 과 R_2 를 합쳐서 5Ω 이라고 가정하면 전류는 1A가 됩니다. ($V = IR$)

이 때 R_1 과 R_2 의 분배에 따라서 V_{out} 이 달라지게 됩니다.

예를 들어 R_1 이 4Ω 이고 R_2 가 1Ω 인 경우에는 V_{out} 은 1이 되게 됩니다.



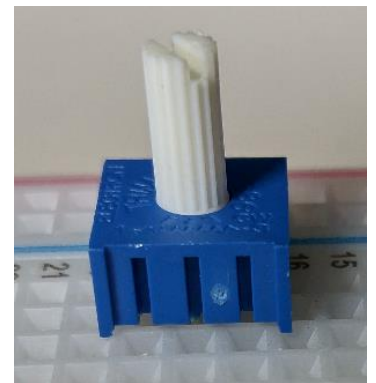
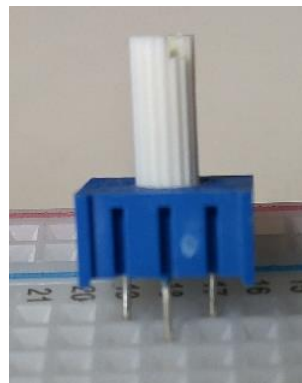
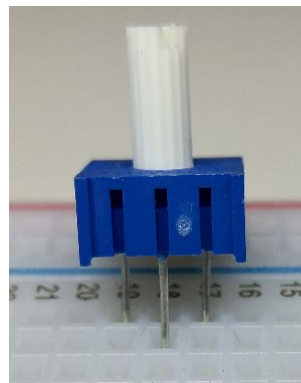
<그림1-4> 전압 분배 예시

만약 R_1 이 2이고 R_2 가 3인 경우에는 V_{out} 은 3이 되게 됩니다.

가변저항 하드웨어 구성

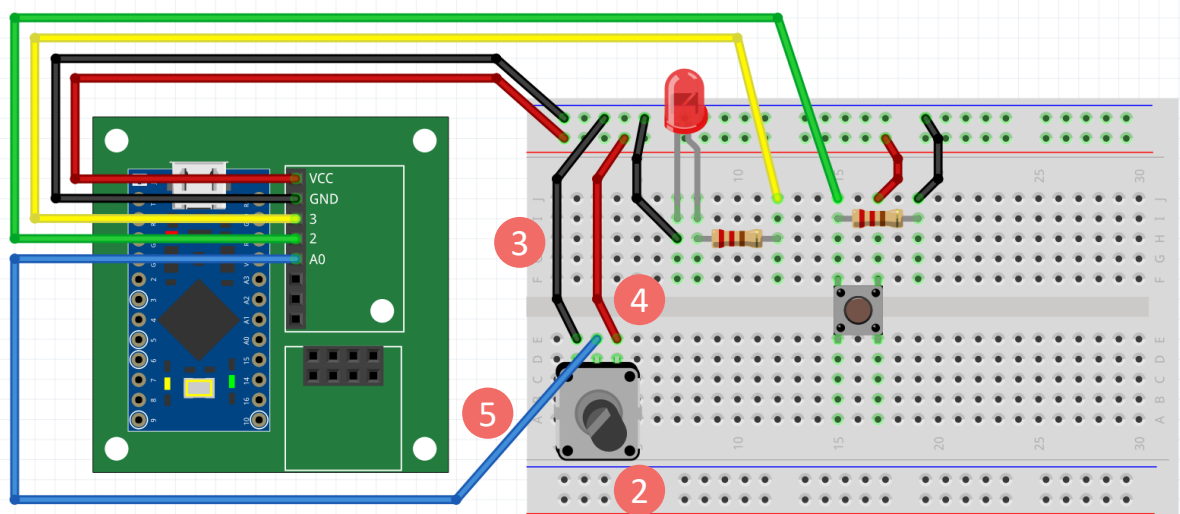
가변저항 하드웨어 구성

- 1 UBS가 연결되어 있는 아두이노를 메인 보드에 끼웁니다.(방향에 유의합니다)
- 2 가변저항을 빵판에 그림과 같이 꽂아 넣습니다.



<그림1-5> 가변저항 꽂는 법

- 3 3개의 가변 저항 다리 중 왼쪽다리를 빵판의 파란줄과 연결합니다.
- 4 3개의 가변 저항 다리 중 오른쪽 다리를 빵판의 빨간줄과 연결합니다.
- 5 3개의 가변 저항 다리 중 가운데 다리를 메인 보드의 A0핀(위에서 5번째)에 연결합니다.



<그림1-6> 베이스 보드와 가변저항 연결

꿀TIP

가변저항 연결

가변저항의 가운데 다리로 신호를 받아야 합니다.

fritzing

가변저항 예제 작성하기


가변저항 예제

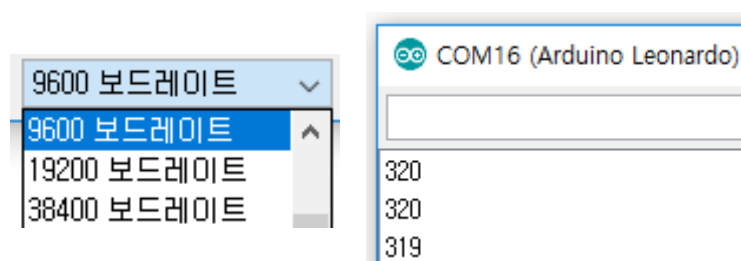
- 1 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.

ch4_1_1_variable

```
1 void setup() {
2   Serial.begin(9600);
3 }
4
5 void loop() {
6   int val = analogRead(A0);
7   Serial.println(val);
8   delay(500);
9 }
```

<그림1-7> 가변저항 예제

- 2  버튼을 눌러 시리얼 모니터를 켭니다.
- 3 보드레이트를 맞춘 후 가변저항을 돌렸을 때, 시리얼 모니터에서 값의 변화가 있는지 확인합니다.



<그림1-8> 값 확인

가변저항 해석

```
void setup() {  
  Serial.begin(9600); //시리얼 통신 시작  
}  
  
void loop() {  
  int val = analogRead(A0); //A0의 값을 받아 val에 저장  
  Serial.println(val); //val 값을 시리얼 모니터로 확인  
  delay(500); //0.5초 멈춤  
}
```



가변저항과 아날로그값에 익숙해졌다면, 아래 코드도 업로드 해 보고 어떻게 작동하는지 알아봅시다.


가변저항 활용

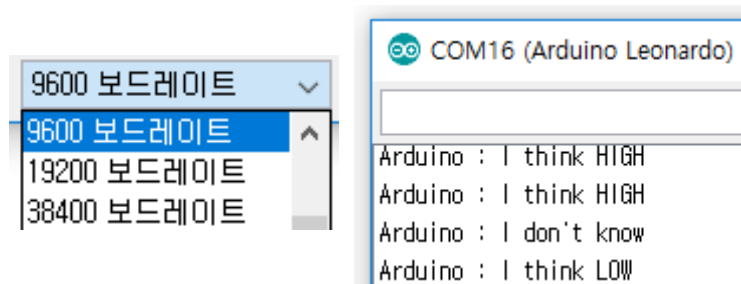
ch4_1_1_variable2

```
1 void setup() {  
2   Serial.begin(9600);  
3 }  
4  
5 void loop() {  
6   int val = analogRead(A0);  
7  
8   if (val > 614) {  
9     Serial.println("Arduino : I think HIGH");  
10  } else if (val < 614 && val > 307) {  
11    Serial.println("Arduino : I don't know");  
12  } else if (val < 307) {  
13    Serial.println("Arduino : I think LOW");  
14  }  
15  
16 // Serial.println(val);  
17 delay(500);  
18 }
```

<그림1-9> 아날로그값 활용

가변저항 활용 해석

- 1  버튼을 눌러 시리얼 모니터를 켭니다.
- 2 보드레이트를 맞춘 후 가변저항을 돌렸을 때, 시리얼 모니터에서 값의 변화가 있는지 확인합니다.



<그림1-10> 값 확인

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

void loop() {
  int val = analogRead(A0); //A0의 값을 받아 val에 저장

  if (val > 614) { //만약 val의 값이 614보다 크면
    Serial.println("Arduino : I think HIGH"); //문구 출력
  } else if (val < 614 && val > 307) { //val의 값이 614와 307 사이면
    Serial.println("Arduino : I don't know"); //문구 출력
  } else if (val < 307) { //만약 val의 값이 307보다 작으면
    Serial.println("Arduino : I think LOW"); //문구 출력
  }

  // Serial.println(val);
  delay(500); //0.5초 멈춤
}
```

실제 아두이노에서 HIGH와 LOW값을 읽어 들일 때에도 마찬가지로 일정 전압 이상이 들어오면 HIGH, 일정 전압보다 낮으면 LOW로 판단하게 됩니다.

02 digital과 analog

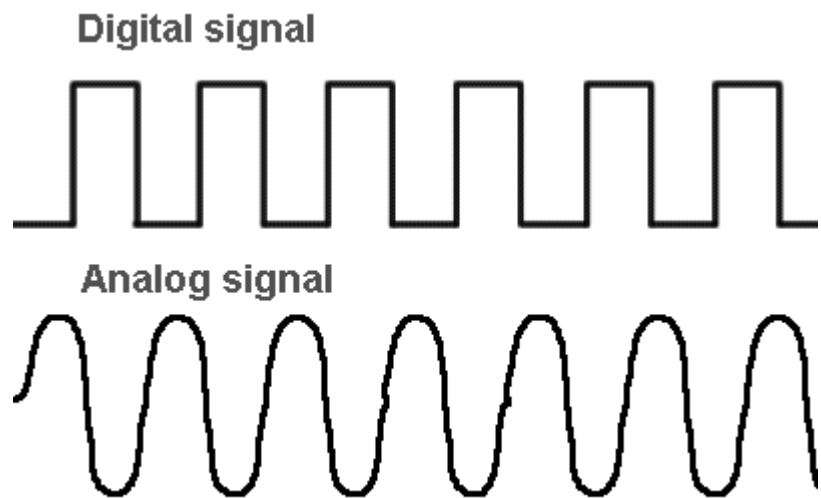


digital이란 단어를 들으면 어떤 느낌이 드나요?

digital은 현대 사회를 이루는 근간입니다. 흔히 설명할 때 analog와 대비하여 개념을 설명하게 됩니다. analog는 자연에서 흔히 발견되는 현상이며 사람의 목소리처럼 연속적으로 변하는 것이고, digital은 비연속적이며 셀 수 있는 수치로 구성되어 있습니다.

digital과 analog

digital은 0과 1을 표현할 때 주로 사용됩니다. 이와 달리 현실에서는 analog가 주로 사용됩니다. 키와 몸무게는 0과 1로 표현하기엔 부족합니다.



<그림2-1> 디지털과 아날로그

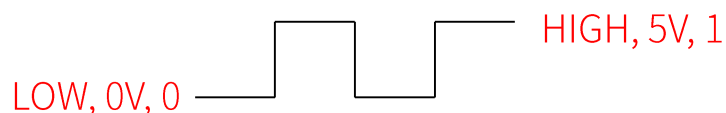
digital

digitalRead()

앞에서 배운 digitalRead() 함수는 아두이노의 특정 핀의 상태가 1인지 0인지를 읽어 들입니다. 이 때 읽어 들인 핀의 전압이 0V이면 LOW, 0으로 받아들이고 읽어 들인 핀의 전압이 5V이면 HIGH, 1로 받아들이습니다.

digitalWrite()

digitalWrite()함수는 아두이노의 특정 핀을 전압이 높은 HIGH상태로 만들거나 전압이 낮은 LOW상태로 만들 수 있습니다.



디지털 실습
해보기

- ① 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.

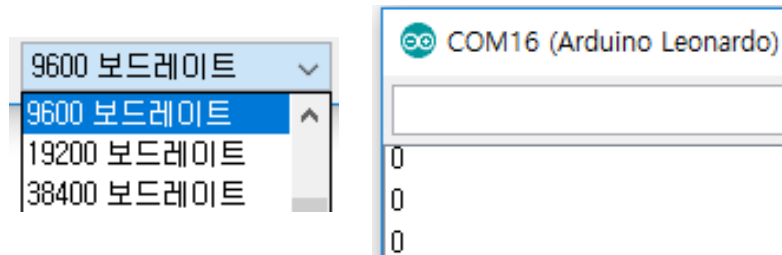
ch4_1_2_sw_led

```
1 void setup() {  
2     Serial.begin(9600);  
3     pinMode(3, OUTPUT);  
4     pinMode(2, INPUT);  
5 }  
6  
7 void loop() {  
8     int sw = digitalRead(2);  
9     Serial.println(sw);  
10    if (sw == HIGH) {  
11        digitalWrite(3, HIGH);  
12    } else {  
13        digitalWrite(3, LOW);  
14    }  
15 }
```

<그림2-2> 디지털 실습 해보기

- ②  버튼을 눌러 시리얼 모니터를 켭니다.

- 3 보드레이트를 맞추는 후 스위치를 누르고 시리얼 모니터에서 값이 변하는지, LED의 밝기 변화가 있는지 확인합니다.



<그림2-3> 값 확인

디지털 실습 해석

```
void setup() {  
  Serial.begin(9600);  
  pinMode(3, OUTPUT);  
  pinMode(2, INPUT);  
}
```

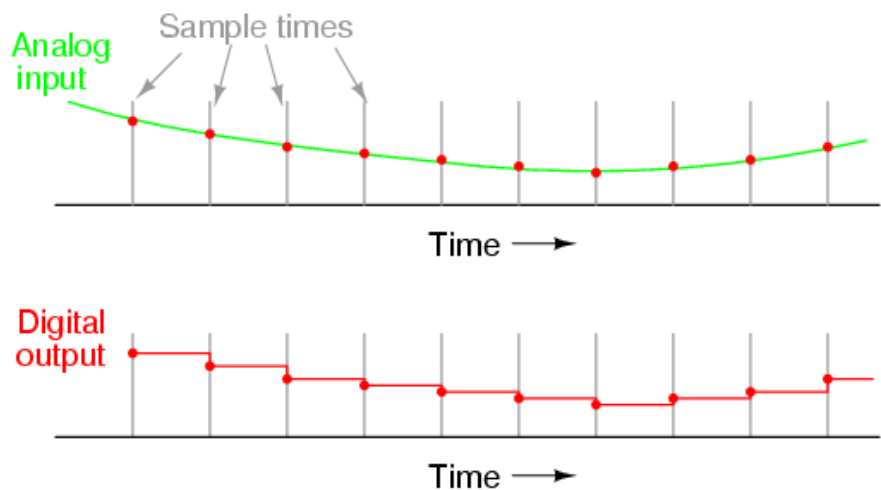
```
void loop() {  
  int sw = digitalRead(2); //2번핀의 값을 읽어서 sw에 저장  
  Serial.println(sw); //sw값 출력  
  if (sw == HIGH) { //만약 sw가 HIGH와 같다면  
    digitalWrite(3, HIGH); // LED 켜  
  } else { //그 외의 경우  
    digitalWrite(3, LOW); // LED 끄  
  }  
}
```

analog

analogRead()

아두이노에서 `analog`를 읽을 땐 A0 핀을 사용합니다. 이를 통해서 0에서부터 1023까지의 숫자를 받아들일 수 있습니다.

이 때 사용되는 개념이 ADC(Analog to Digital Converter)입니다. 쉽게 말해 아날로그 신호(0~5V의 전압)를 디지털인 0에서 1023의 숫자로 바꿔 주는 것입니다.



<그림2-4> ADC

위 그림에서처럼 연속적인 `analog`값들을 일정 구간별로 나누어 `digital`화하게 됩니다.

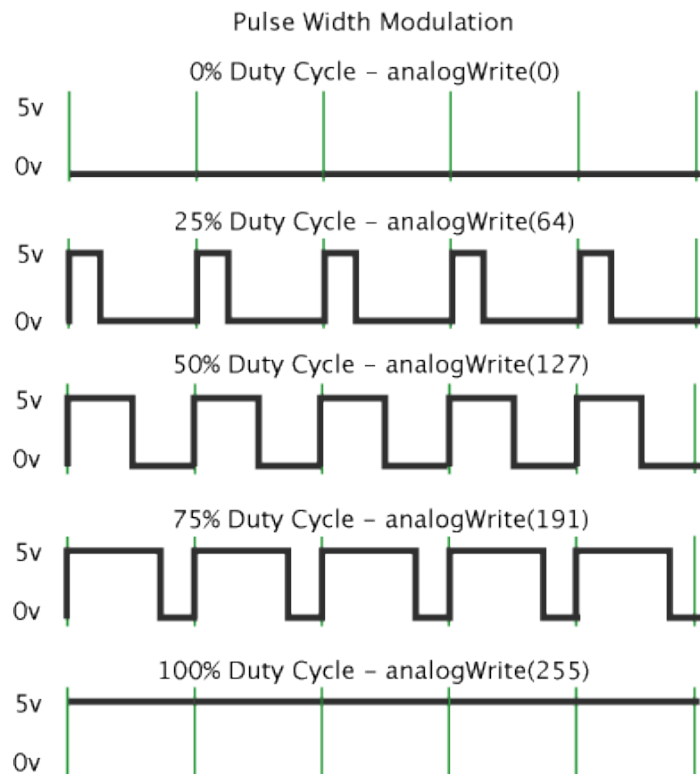
아두이노에서는 0V에서 5V까지를 1024단계로 나누어 $5/1024 = 49\text{mV}$ 단위로 `analog`값을 적용할 수 있게 됩니다.

analog

analogWrite()

`analogWrite()`은 0에서 255까지의 값을 내보낼 수 있습니다. 이를 통해 LED의 밝기를 256단계로 조절할 수 있게 됩니다.

여기선 PWM(Pulse Width Modulation)이란 개념이 적용됩니다. PWM은 펄스의 폭을 제어하여 출력을 조절하는 방식입니다.



<그림2-5> PWM

위 그림에서처럼 `analogWrite(0)`을 했을 때엔 0V가 유지됩니다.

두 번째 그림에서 `analogWrite(64)`를 하게 될 경우 64는 255의 1/4이기에 1/4만큼 5V를 주게되어 5V와 0V가 1:3 비율로 적용되어 전압이 나타나게 됩니다.


아날로그
실습
해보기

- 1 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.

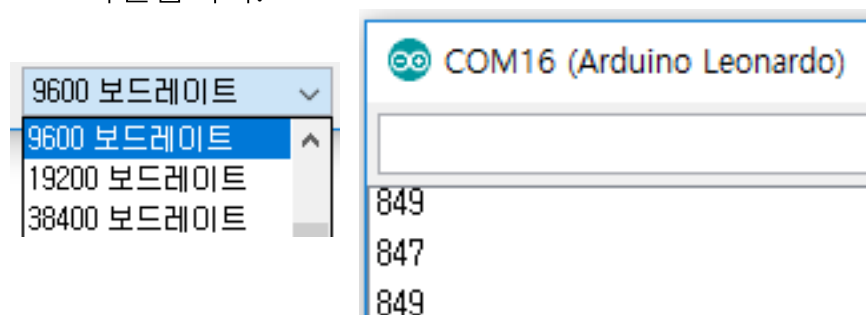
```
ch4_1_2_variable_led
```

```
1 void setup() {  
2     Serial.begin(9600);  
3     pinMode(3, OUTPUT);  
4     pinMode(2, INPUT);  
5 }  
6  
7 void loop() {  
8     int val = analogRead(A0);  
9     Serial.println(val);  
10    analogWrite(3, val / 5);  
11 }
```

<그림2-6> 아날로그 실습 해보기

- 2  버튼을 눌러 시리얼 모니터를 켭니다.

- 3 보드레이트를 맞추는 후 가변저항을 돌려서 시리얼 모니터에서 값이 변하는지, LED의 밝기 변화가 있는지 확인합니다.



<그림2-7> 값 확인

아날로그 실습 해석

```
void setup() {  
  Serial.begin(9600);  
  pinMode(3, OUTPUT);  
  pinMode(2, INPUT);  
}  
  
void loop() {  
  int val = analogRead(A0); //A0핀의 값을 받아 val에 저장  
  Serial.println(val); //val값 출력  
  analogWrite(3, val / 5); //val을 5로 나눈 값을 3번핀에 출력  
}
```

03 상수에 대해서



상수는 변하지 않는 수입니다.

아두이노에는 기본적으로 존재하는 기본 상수가 있습니다.

이러한 기본 상수에 대해 잘 모르면, 아두이노를 프로그래밍 하는데 어려움을 겪을 수 밖에 없습니다.

아두이노의 기본 상수를 알아보며 상수를 익혀봅시다.

상수란

상수는 변하는 수인 변수와는 반대로 변하지 않는 수입니다. 아두이노에는 다음과 같이 미리 정해져 있는 상수들이 존재합니다.

상수는 프로그램을 읽기 쉽게 하기 위해 쓰입니다. 단순히 1 또는 0으로 표현하면, 그 의미를 알기 어렵기 때문입니다.

```
40 #define HIGH 0x1
41 #define LOW 0x0
```

<그림3-1> 미리 정의된 HIGH

이러한 상수들은 다음과 같이 아두이노가 설치되어 있는 폴더의 **Arduino.h** 파일 안에 정의되어 있습니다.

Arduino-1.8.4\hardware\Arduino\avr\cores\Arduino

참 / 거짓

참과 거짓은 논리값들을 정할 때의 Boolean값입니다. Boolean값은 true와 false로, true는 1, false는 0입니다. 거짓은 주로 0, 참은 주로 1로 나타냅니다. 하지만 보통 0이 아닌 수는 거의 참으로 간주됩니다. 따라서, Boolean에서는 -1, -2 그리고 -200 또한 참입니다.

HIGH /
LOW

Digital Pin 값들을 HIGH, LOW로 나누기 위함입니다. HIGH는 논리값1(ON 혹은 5V)로 LOW는 논리값0(OFF 혹은 0V)로 쓰입니다.

INPUT /
OUTPUT

상수값은 또한 pinMode함수에서 digitalWrite가 INPUT인지 OUTPUT인지 결정할 때 쓰입니다. INPUT은 0, OUTPUT은 1의 값을 가집니다.

#define과 const

#define const

상수를 만드는 방법에는 두 가지가 있습니다. 바로 `#define`과 `const`입니다.

#define

`define`의 의미는 '정의하다'입니다. 예를 들어 다음과 같이

```
1 | #define LED 3
```

`#define LED 3`이라는 명령어를 적게 되면 LED라는 글자는 숫자 3으로 쓰이게 됩니다.

이러한 `#define`은 단순히 글자가 바뀌는 것처럼 작용해서 메모리를 잡아먹지 않습니다.

const

`const`는 `constant`의 약자로 '일정한'이라는 의미와 '상수'라는 의미를 갖고 있습니다.

```
2 | const int BUTTON = 2;
```

`const int BUTTON = 2;`이라는 명령어를 적게 되면 앞으로 `BUTTON`이라는 변수는 읽기 전용이 되면서 상수로 사용되게 됩니다.

만약, `const`가 쓰인 변수에 다른 값을 집어넣으려고 하면 에러가 발생합니다.

#define vs const

보통 우리가 사용하는 컴퓨터와 달리 아두이노는 메모리를 많이 갖고있지 않습니다.

변수를 사용할 때는 컴퓨터와 마찬가지로 변수의 정보가 ram에 저장되게 됩니다.

요즘 컴퓨터는 보통 4G(약4,000,000,000)의 램을 사용하는 반면 아두이노는 2k(약2,000)의 램을 사용합니다.

따라서, 메모리를 아끼기 위해 `#define`을 사용하는게 좋겠죠?


상수 작성 해보기

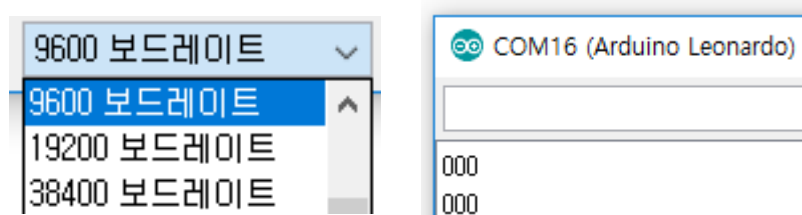
상수
작성

- 1 다음과 같이 코드를 작성하여 아두이노에 업로드합니다.

```
ch4_1_3_const
1 #define LED 3
2 const int BUTTON = 2;
3
4 void setup() {
5     Serial.begin(9600);
6     pinMode(LED, OUTPUT);
7     pinMode(BUTTON, INPUT);
8 }
9
10 void loop() {
11     if (digitalRead(BUTTON) == HIGH) {
12         Serial.print(true);
13         Serial.print(HIGH);
14         Serial.println(OUTPUT);
15         digitalWrite(LED, HIGH);
16     } else {
17         Serial.print(false);
18         Serial.print(LOW);
19         Serial.println(INPUT);
20         digitalWrite(LED, LOW);
21     }
22 }
```

<그림3-2> const 코드

- 2  버튼을 눌러 시리얼 모니터를 켭니다.
- 3 보드레이트를 맞춘 후 스위치를 눌러 시리얼모니터의 숫자가 변하는지, LED가 변하는지 확인합니다.



<그림3-3> 시리얼 통신 확인

시리얼 통신 해석

```
#define LED 3 // LED를 3으로 지정
const int BUTTON = 2; // BUTTON변수에 2를 저장 후 상수로

void setup() {
  Serial.begin(9600);
  pinMode(LED, OUTPUT); // 3번핀을 출력으로
  pinMode(BUTTON, INPUT); // 2번핀을 입력으로
}

void loop() {
  if (digitalRead(BUTTON) == HIGH) { // 만약 버튼이 눌리면
    Serial.print(true); // true값 출력
    Serial.print(HIGH); // HIGH값 출력
    Serial.println(OUTPUT); // OUTPUT값 출력 후 줄바꿈
    digitalWrite(LED, HIGH); // LED 켜
  } else {
    Serial.print(false); // false값 출력
    Serial.print(LOW); // LOW값 출력
    Serial.println(INPUT); // INPUT값 출력
    digitalWrite(LED, LOW); // LED 끄
  }
}
```

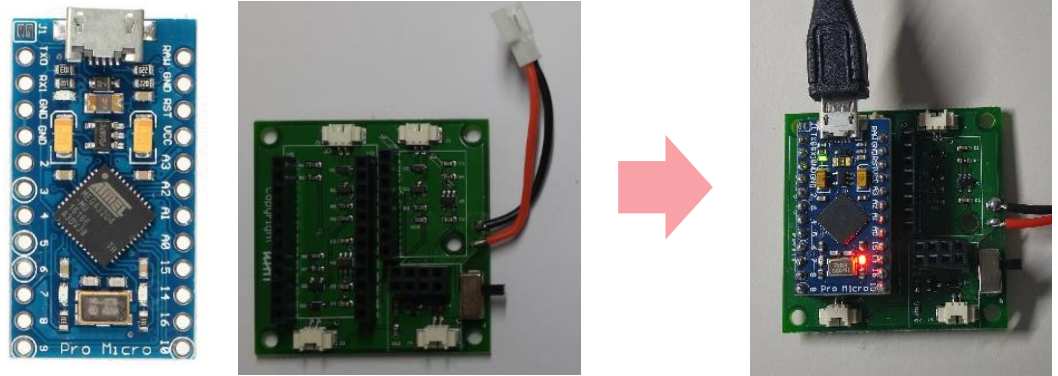


WHIT

가변저항 하드웨어 구성

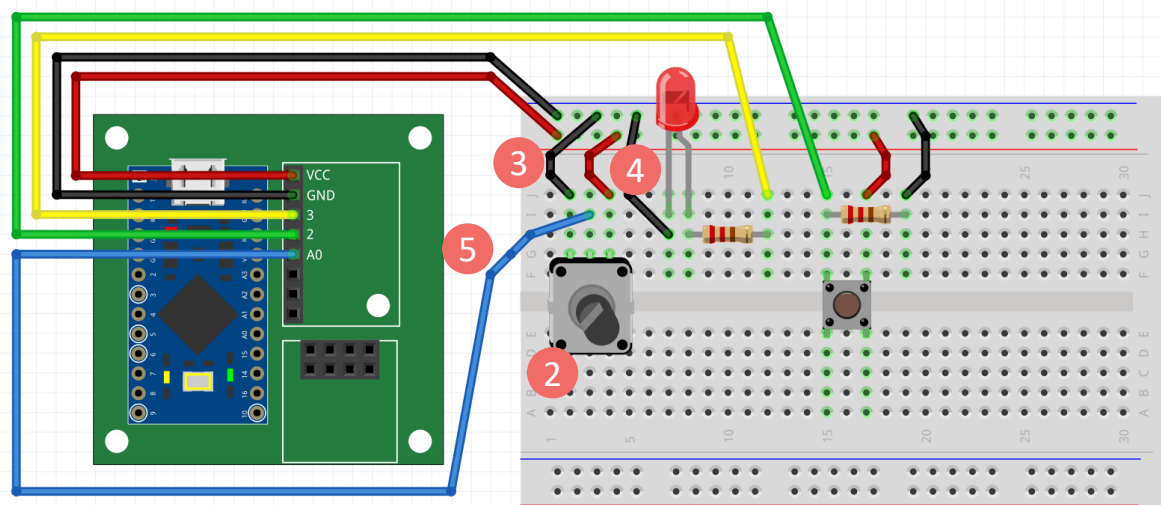
가변저항 하드웨어 구성

- 1 UBS가 연결되어 있는 아두이노를 메인 보드에 끼웁니다.(방향에 유의합니다)



<그림1-5> 아두이노와 메인 보드 연결

- 2 가변저항을 빵판에 그림과 같이 꽂아 넣습니다.
- 3 3개의 가변 저항 다리 중 왼쪽다리를 빵판의 파란줄과 연결합니다.
- 4 3개의 가변 저항 다리 중 오른쪽 다리를 빵판의 빨간줄과 연결합니다.
- 5 3개의 가변 저항 다리 중 가운데 다리를 메인 보드의 A0핀(위에서 5번째)에 연결합니다.



<그림1-6> 베이스 보드와 가변저항 연결

꿀TIP

가변저항 연결

가변저항의 가운데
다리로 신호를 받아야
합니다.

fritzing