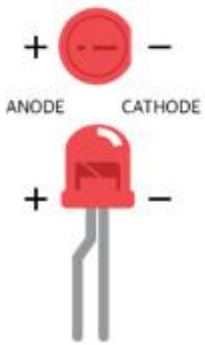
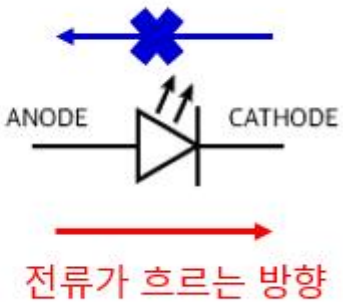
《4. 디지털 출력 - LED 제어하기(1)》

1. LED

아두이노를 다루면서 저항 다음으로 가장 많이 사용하게 되는 소자가 LED이다.
LED란 Light Emitting Diode의 약자로 빛을 뿜어내는 반도체라는 뜻이다.
LED의 가장 큰 특징은 기존 전기 구동 발광체에 비하여 열을 적게 발생한다는 것과 열손실로 낭비되는 에너지가 적고 반영구적인 수명이라는 것이다.





LED를 연결할 때 LED 몰딩 안을 자세히보면 LED 소자의 다리와 연결된 쇠판이 보일 것이다.
일반적으로 쇠판 작은 쪽이 리드선이 길고 애노드(ANODE)이고 VCC(전원)에 연결해야 한다.
쇠판이 큰 쪽이 리드선이 짧고 캐소드(CATHODE)라 하고 GND(그라운드)에 연결해야 한다.
LED는 극성소자이기 때문에 전류는 한 방향으로만 흐르기 때문에 반대로 연결하면 LED가 켜지지 않는다.
LED를 사용할 때는 저항과 항상 함께 사용해야 한다. LED마다 최대 전압과 전류가 정해져있다.
LED를 켤 때 저항을 사용하지 않으면 LED가 허용하는 최대 전류 값을 초과하여 LED가 고장날 수도 있다.

- LED 구성요소
다이오드는 한 방향으로만 전류가 흐를 수 있고 전류가 흐를 때 다이오드 양단은 기본적으로 정전압 특성을 보인다.
즉, 저항처럼 전류에 비례하는 전압강하가 아닌, 전류가 얼마나 흐르느냐에 무관하게 일정한 전압강하를 보이게 된다.
그래서 다이오드의 전류-전압 그래프를 보면 전압이 0부터 증가하여 어느 지점까지는 전류가 거의 흐르지 않다가
그 선을 넘으면서 전류가 거의 수직으로 급격하게 증가하게 되는데, 마치 약간의 전압이 걸린 채로 단락된 것과 비슷하다.
물론, 실제적인 다이오드는 전류가 증가함에 따라 아주 약간의 전압증가가 따르게 되며 전류가 증가하면 할수록 LED의 밝기가 밝아지다가 전류가 어느 한도 이상으로 증가하게 되면 지나친 발열로 소자가 파손되기도 한다.

LED의 사양에는 항상 몇 V에 몇 mA라는 식으로 전류와 전압이 함께 표시된다.
제조업체에 따라 약간씩은 다르지만 적색의 다이오드는 대개 1.6V 정도의 전압에서 동작하는 반면에
청색이나 백색은 그 보다 높은 3.5V 정도에서 동작하며, 대략 20mA 정도의 정격전류에서 동작을 한다.
정격 전류란 소자의 성능이 최대한 발휘되면서 연속적으로 동작시켜도 안정성이 보장되는 전류를 의미한다.
LED는 정격 동작전압보다 조금이라도 높은 전압이 걸리면 과전류가 흐르면서 파손될 수 있다.
3V 20mA의 LED에 3.2V 전압을 연결하면 다이오드의 특성상 정격 전류보다 훨씬 많은 전류가 흐르면서 파손된다.

LED는 직렬로 저항을 연결하여 사용하는 것이 정석이다.
예를 들면, 3V 20mA 정격의 LED를 5V 전원으로 구동하고자 하는 경우라면, 정격을 초과하는 전압이 2V가 되므로,
20mA의 전류에서 2V의 초과 전압을 감당하게끔 R=2V/20mA=100옴의 저항을 직렬로 연결하여 사용한다.
이 저항은 기술적으로 표현하면 전압원을 전류원으로 바꾸어주는 transconductance의 역할을 하는 것이다.
예를 들어, LED+직렬저항 양단의 전원전압이 변화한 경우, LED의 전압은 거의 일정하기 때문에 직렬저항 양단의 전압만 변화하게 되어 변화된 전압에 비례하는 만큼 전류가 변화하여 LED의 밝기를 조절할 수 있게 된다.

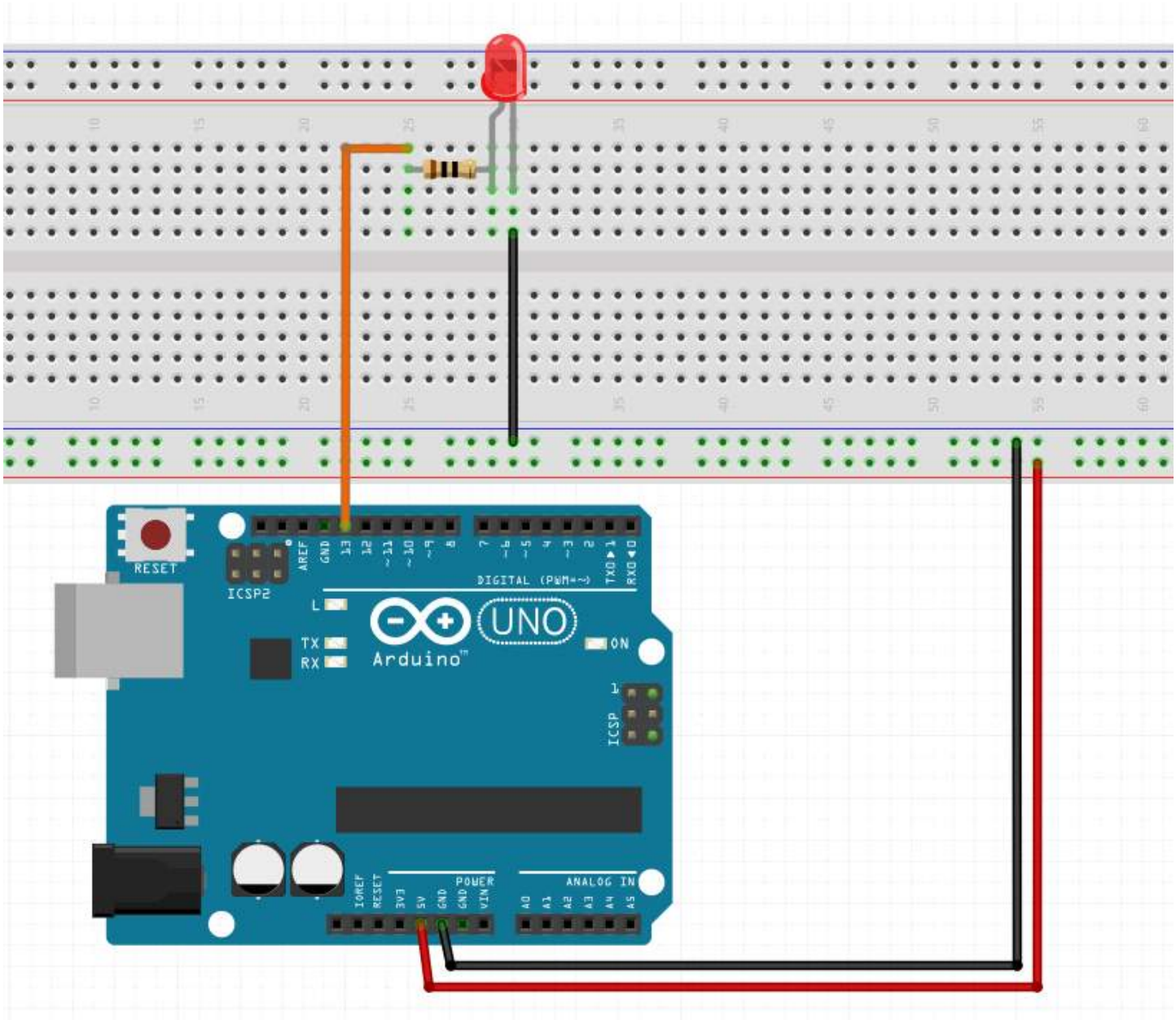
LED는 최소 구동전압과 최대 구동전압이 있다.
최소 구동 전압보다 낮으면 흐리게 빛이 나고 최대 구동 전압보다 높으면 LED가 파손된다.
정확한 스펙은 LED의 데이터 시트를 확인하면 좋지만 데이터 시트를 참고할 수 없는 경우 아래의 표를 참고한다.
LED를 작동할 때는 적당한 저항을 배치해서 LED에 가해지는 전압을 적절하게 조절해야 한다.
대게 LED는 최대 3.3V 전압을 허용하며 그 이상의 전압을 입력하게 되면 기능을 상실하게 된다.
그렇기 때문에 220옴의 저항을 추가적으로 연결하여 아두이노에서 출력되는 5V의 전압을 3V로 만들어준다.

(입력전압 - LED를 켜기 위한 최소전압) / 전류 = 저항값

예를들어서 적색 LED를 사용하고 입력전압이 5V라면
1000mA는 1A이다. 따라서 20mA는 20/1000A 즉 0.02A이다.
(5V(입력전압) - 1.8V(LED 최소전압))/0.02A(전류) = 160옴
즉 160옴 보다 큰 저항을 사용하면 된다.

색상	구 분	최소전압	최대전압	전류(일반)	전류(최대)
적●	Red	1.8V	2.3V	20 mA	50 mA
등●	Orange	2.0V	2.3V	30 mA	50 mA
황●	Real Yellow	2.0V	2.8V	20 mA	50 mA
초●	emerald Green	1.8V	2.3V	20 mA	50 mA
초●	Real Green	3.0V	3.6V	20 mA	50 mA
청●	sky Blue	3.4V	3.8V	20 mA	50 mA
청●	Real Blue	3.4V	3.8V	20 mA	50 mA
자●	Pink	3.4V	3.8V	20 mA	50 mA
백○	White	3.4V	4.0V	20 mA	50 mA

2. 아두이노 LED 제어하기



예제1) LED ON / OFF

LED ON	

void setup()
{
  pinMode(13, OUTPUT);
}
 	
void loop()
{
  digitalWrite(13,HIGH);
}
	void setup()
	{
	  pinMode(11,OUTPUT);
	  digitalWrite(11,HIGH);
	}
	void loop()
	{
	
	}

LED OFF	

void setup()
{
  pinMode(13, OUTPUT);
}
 	
void loop()
{
  digitalWrite(13,LOW);
}
	void setup()
	{
	  pinMode(11,OUTPUT);
	  digitalWrite(11,LOW);
	}
	void loop()
	{
	
	}

예제2) LED Blink

void setup()
{
  pinMode(11,OUTPUT);
  digitalWrite(11,HIGH);
  delay(1000);
  digitalWrite(11,LOW);
}
void loop()
{ }
	void setup()
	{
	  pinMode(13, OUTPUT);
	}
	void loop()
	{
	  digitalWrite(13,HIGH);
	  delay(1000);
	  digitalWrite(13,LOW);
	  delay(1000);
	}

아두이노가 LOW로 꺼지는 부분에서 delay가 없다면 어떻게 될까?
LED가 꺼지는 즉시 다시 켜지는데 그 찰나의 순간은 사람의 눈으로 포착하기 어렵기 때문에
우리 눈으로는 LED가 계속 켜져있는 것과 같이 보이게 된다.
아두이노 보드의 마이크로 컨트롤러 칩은 1초에 1600만 개의 명령을 수행한다. 이 말이 1600만 개의 명령을 처리한다는 것은 아니지만 그만큼 빠르다는 것을 말한다. delay(1000)이 없다면 수백만분의 1초동안 꺼졌다가 다시 켜진 것이다.

예제3) 시리얼 모니터로 LED 불켜고 끄기
외부에서 신호를 받았는지 확인하는 Serial.available()을 사용한다.
시리얼 포트에서 온 신호를 버퍼에 저장해두는데 이 값이 있는지 없는지를 호출해서 확인할 수 있다.
버퍼에 들어있는 값을 Serial.read()로 읽을 수 있다.

* available
  직렬 포트에서 읽을 수 있는 바이트 수를 가져온다.
  이 데이터는 이미 도착하여 직렬 수신 버퍼 (64바이트 보유)에 저장되어 있는 데이터이다.
  availabe()은 Stream 유틸리티 클래스에서 상속된다.

* Stream
  Stream은 문자 및 바이너리 기반 스트림의 기본 클래스이다.
  직접 호출되지는 않지만 의존하는 함수를 사용할 때마다 호출된다.
  Stream은 arduino에서 읽기 기능을 정의한다.

시리얼 모니터에 문자가 들어올때만 13번에 연결된 LED가 켜졌다가 2초뒤에 꺼진다.

```
void setup()
{
    pinMode(13, OUTPUT);
    Serial.begin(9600);
}

void loop()
{
    char ch = Serial.read();
    if(ch == 'h')
    {
       digitalWrite(13, HIGH);
       delay(2000);
       digitalWrite(13, LOW);
    }
}
```

```
void setup()
{
    pinMode(13, OUTPUT);
    Serial.begin(9600);
}
void loop()
{
    if(Serial.available())
    {
       int input = Serial.read();
       if (input == '1')
       {
          digitalWrite(13,HIGH);
       }
       else if (input == '0')
       {
          digitalWrite(13,LOW);
       }
    }
}
```

시리얼 포트에서 1을 입력하면 13번 핀에 연결된 LED가 들어오고 0이 들어오면 LED가 꺼지게 된다.

```
void setup()
{
   pinMode(6,OUTPUT);
   pinMode(5,OUTPUT);
}

void loop()
{
   digitalWrite(6,HIGH);
   digitalWrite(5,LOW );
   delay(1000);
   digitalWrite(6,LOW );
   digitalWrite(5,HIGH);
   delay(1000);
   digitalWrite(6,HIGH);
   digitalWrite(5,HIGH);
   delay(1000);
   digitalWrite(6,LOW );
   digitalWrite(5,LOW );
   delay(1000);
}
```

```
void setup()
{
  pinMode(11,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode( 9,OUTPUT);
}

void loop()
{
  digitalWrite(11,HIGH);
  digitalWrite(10,LOW );
  digitalWrite( 9,LOW );
  delay(1000);
  digitalWrite(11,LOW);
  digitalWrite(10,HIGH );
  digitalWrite( 9,LOW );
  delay(1000);
  digitalWrite(11,LOW);
  digitalWrite(10,LOW );
  digitalWrite( 9,HIGH );
  delay(1000);
  digitalWrite(11,HIGH);
  digitalWrite(10,HIGH );
  digitalWrite( 9,LOW );
  delay(1000);
  digitalWrite(11,HIGH);
  digitalWrite(10,LOW );
  digitalWrite( 9,HIGH );
  delay(1000);
  digitalWrite(11,LOW);
  digitalWrite(10,HIGH );
  digitalWrite( 9,HIGH );
  delay(1000);
  digitalWrite(11,HIGH);
  digitalWrite(10,HIGH );
  digitalWrite( 9,HIGH );
  delay(1000);
  digitalWrite(11,LOW);
  digitalWrite(10,LOW );
  digitalWrite( 9,LOW );
  delay(1000);
}
```

```
void setup()
{
  pinMode(11,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode( 9,OUTPUT);
}

void loop()
{
  digitalWrite(11,HIGH);
  digitalWrite(10,LOW );
  digitalWrite( 9,LOW );
  delay(1000);
  digitalWrite(10,HIGH );
  delay(1000);
  digitalWrite( 9,HIGH );
  delay(1000);
}
```

```
void setup()
{

}

void loop()
{
  analogWrite(11,255);
  analogWrite(10,0);
  analogWrite( 9,0);
  delay(1000);
  analogWrite(10,255);
  delay(1000);
  analogWrite( 9,0);
  delay(1000);
}
```

```
void setup()
{

}
void loop()
{
    analogWrite(7,0);
    delay(2000);
    analogWrite(7,255);
    delay(2000);
}
```

변수 사용 예제

```
int ledPin = 13;
int delayPeriod = 500;

void setup()
{
    pinMode(ledPin, OUTPUT):
}
void loop()
{
    digitalWrite(ledPin, HIGH);
    delay(delayPeriod);
    digitalWrite(ledPin, LOW);
    delay(delayPeriod);
}
```

```
int ledPin = 13;
int delayPeriod = 100;

void setup()
{
    pinMode(ledPin, OUTPUT):
}
void loop()
{
    digitalWrite(ledPin, HIGH);
    delay(delayPeriod);
    digitalWrite(ledPin, LOW);
    delay(delayPeriod);
    delayPeriod = delayPeriod + 100;
}
```

PWM 핀 예제

```
void setup()
{


}

void loop()
{
  analogWrite(11,0);
  delay(1000);
  analogWrite(11,50);
  delay(1000);
  analogWrite(11,100);
  delay(1000);
  analogWrite(11,150);
  delay(1000);
  analogWrite(11,200);
  delay(1000);
  analogWrite(11,255);
  delay(1000);
}
```

3. 기초 프로그래밍 : 변수

변수란 값을 임시로 저장할 수 있는 공간과 같다고 생각하면 된다.
그 값에는 숫자 뿐만 아니라 문자와 문자열까지 다양한 형태가 존재한다.
변수를 만드는 것을 변수의 선언이라 하며, 변수를 선언하면 메인 메모리에 공간이 만들어 진다.
변수의 이름을 통해 데이터 값의 저장 및 변경이 가능하며 데이터 값을 참조할 수도 있다.
변수를 사용하면 좋은 점은 setup()이나 loop()의 코드를 건들이지 않아도 쉽게 변경이 가능하다.
변수를 사용하지 않았다면 일일이 13을 11로 바꿔야 하지만 int led = 13 으로 선언했다면 13을 11로 바꾸면 된다.
변수를 쓸 때는 변수 이름과 함께 어떤 형태의 값을 저장할 것인지도 함께 선언해줘야 한다.

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

void loop()
{
    int led = 20;
    Serial.println(led);
    Serial.end();
}
```

int led = 20; // led를 선언하는 것과 동시에 20으로 초기화 한다.
이 부분을
int led;
led = 20;
이렇게 먼저 선언한 후에 변수를 20으로 초기화해도 결과는 똑같다.

자료형	변수	상수
int	led	20
저장공간크기	변하는 수	고정 수

led는 변수이고 20은 상수이다.
led라는 공간에 20이 저장될 수 있고 21이 저장될 수도 있기 때문에 변수라고 부른다.
하지만 20은 20이란 수로 고정되어 20을 21로 바꿀 수는 없기 때문에 상수라고 부른다.
여기서 변수란 프로그램에서는 저장공간이라는 개념을 추가해야 합니다.

프로그램에서는 어떤 값이 저장될 때 메모리주소지에 저장되고 그 주소지는 숫자형태로 되어 있다.
'0x7fffad92dd80'주소지가 있다면 이 주소지공간에 20이 저장되었다면 '0x7fffad92dd80=20' 실제로 들어갈 것이다.
하지만 주소지로 접근하는 건 힘들 기 때문에 변수라는 이름을 사용하여 그 주소지를 대신 접근할 수 있게 한다.

led의 주소지가 '0x7fffad92dd80'이라 했을 때 led=20; 하면 우리가 해독할 때는
"led 변수공간에 20이 저장된다" 라고 생각하지만 컴퓨터는 '399942732'주소지에 20이 기록된다고 생각하면 된다.

[변수 선언 시 주의사항]

1. 변수를 사용하기 위해선 반드시 선언을 해야한다.
2. 데이터 종류에 따라 변수의 자료형이 정의되어야 한다.
3. 변수의 이름은 영문자, 밑줄(_), 숫자로 구성되어야 한다.
4. 변수 이름의 첫 글자는 영문자 또는 밑줄(_)만 가능하다.
5. 예약어를 변수의 이름으로 사용할 수 없다.
   (예약어 : 이미 특정한 용도가 정해져 있는 이름. ex) if, for, int, double 등등...)
6. 변수 이름 사이에는 공백을 넣을 수 없다.
7. 변수의 이름에서 대문자와 소문자는 각각 다른 문자로 취급한다. (언어에 따라 다름)

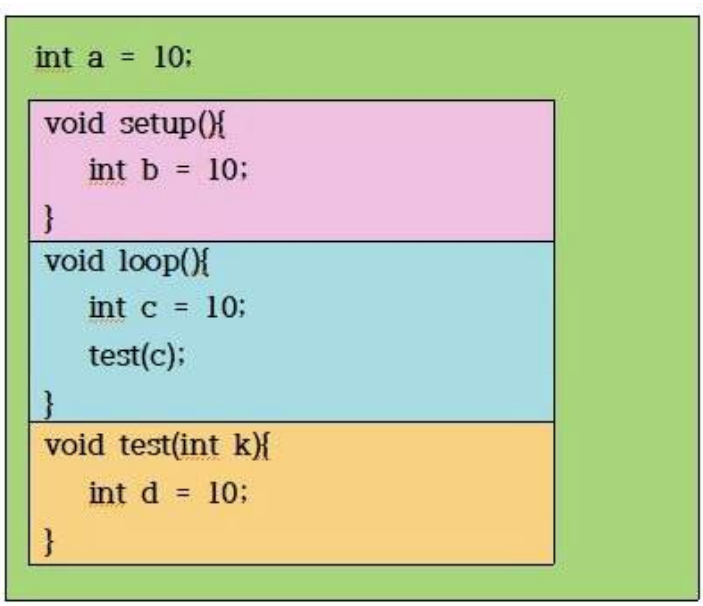
[데이터 타입 = 자료형]

void	- 함수 선언에 사용 - 이 함수는 자신을 호출한 함수에 반환할 정보가 없음을 나타낸다.
boolean	- true 또는 false 두 가지 값 중 하나 - boolean 변수는 메모리의 byte 차지
char	- 문자 값을 저장하는 메모리의 1byte를 차지하는 데이터 형식 - 'A'처럼 단일 문자는 작은 따옴표로 작성 - 여러 문자열은 "ABC" 처럼 큰 따옴표로 나타낸다.
unsigned char	- unsigned char는 메모리의 1byte를 차지하는 부호 없는 데이터 형식 - byte의 데이터 형식과 동일 - byte는 0에서 255까지의 8bit 부호 없는 수를 저장
int	- int는 정수 저장에 대한 기본 데이터 유형 - 이 유형은 MCU에 따라 저장 범위가 다르다. - 아두이노 우노 및 기타 ATmega 기반 MCU ▪ -2^{15}에서 $2^{15}-1$까지의 범위를 갖는 16비트 값으로 저장 - 아두이노 due ▪ -2^{31}에서 $2^{31}-1$까지의 범위를 갖는 32비트 값
unsigned int	- int형과 거의 같으며, 부호가 없는 수를 표시 - 아두이노 우노 및 기타 ATmega 기반 MCU ▪ 0에서 $2^{16}-1$까지의 범위를 갖는 2byte 값 - due에서는 0에서 $2^{32}-1$까지의 범위를 갖는 4byte 값
word	- 0에서 65535까지의 부호가 없는 16bit 값 저장
long	- -2^{31}에서 $2^{31}-1$까지의 범위를 갖는 32bit 값을 저장
unsigned long	- 0에서 $2^{32}-1$까지의 범위를 갖는 32bit 값
short	- 16bit 데이터 형으로 -2^{15}에서 $2^{15}-1$까지의 범위를 갖는 2byte 값으로 저장

[변수 영역]

- 변수의 접근 가능 범위
- 전역변수는 프로그램의 모든 함수에서 볼 수 있다
- 지역변수는 선언된 함수에서만 볼 수 있다.
  - 크고 복잡한 프로그램을 작성할 때 유용하다.

```
int gPWMval; //global variables
void setup(){
}
void loop(){
int i; // local variables i -> loop only
float f; // local variables f -> loop only
for(int j=0; j<100; j++){
 // local variables j -> for-loop only
  }
}
```



● static

- 함수 호출 및 종료 시에도 데이터는 보존되고 지속되는 변수
- static으로 선언된 변수는 함수가 처음 호출 될 때 초기화

● volatile

- 변수 정성자라는 키워드
- 일반적으로 변수의 데이터 형 이전에 붙여준다.
- 동시에 실행되는 스레드, 인터럽트가 사용하는 변수는 volatile로 선언

● const

- 변수를 읽기 전용으로 만든다, 상수
- const를 하면서 같이 초기화해서 값을 넣어주면 그 값이 고정된다.
- const 변수에 값을 할당하려고 하면 컴파일러 오류 발생

● sizeof( )

- 데이터형, 변수의 바이트 수 반환

[상수]

10
#define A 10
const int B = 10;

위의 3가지 모두 10을 상수로 쓰기위해 사용할 수 있는 표현들이다.
숫자 10은 상수로 쉽게 알고 있을 것이다.
두 번째 #define A 10은 A가 10으로 정의된 것이라서 10 대신에 A가 사용된다고 생각하면 된다.
A를 다시 정의하지 않는 이상 A가 10이 아닌 다른 수가 될 수 없다.
const가 붙으면 상수변수라고 생각하면 된다.
상수 변수를 사용하는 이유는 프로그램의 가독성을 높이기 위해서이다. 상수 그 자체로 쓰게되면
숫자가 가리키는 의미를 쉽게 알 수 없지만 상수로 사용하되 그 의미를 전달할 때 상수변수를 사용한다.
상수변수를 사용해서 좋은 점은 데이터 메모리 주소지를 사용하지 않고 프로그램 메모리에 기록된다.
컴파일 시 프로그램에 이식되기 때문이다.
컴파일은 기계가 이해할 수 있는 언어로 번역하는 과정이고 인간의 언어에서 기계어로 바뀌는 과정이다.
상수변수로 선언된 것들은 상수변수가 가리키는 숫자값이 그대로 상수변수가 사용된 위치에 숫자로 대입되어 컴파일을 거치게 된다. 그냥 int A = 10;을 선언했을 때는 A라는 변수의 주소를 불러서 그 안의 10을 데려오는데
const int A = 10;을 선언했을 때는 A위치에 10이 대입되어 컴파일이 되기 때문에 데이터 메모리를 차지하지 않는다.
#define과 const의 차이는 크기를 참고하냐 안하냐에 따라 다르다.
#define A 100은 A를 100으로 정의된 것이고 const int A = 100 이면 int형 A상수변수에 100이 저장되는 것이다.

[참고 : static]

static은 한번 선언되면 프로그램이 종료할 때까지는 소멸하지 않는 변수이다.

소스	결과

void loop(){
static int A = 10;
int B=10;
Serial.println(A);
Serial.println(B);
A=A+10;
B=B+10;
}
	10
	10
	20
	10
	30
	10
	.....

이러한 상황에서 static int A = 10을 했을 때 loop를 돌아도 초기화가 되지 않고 계속 값을 누적시킬 수 있다.
B는 loop를 돌때마다 10으로 초기화가 되는 것을 확인할 수 있다.

```
int in1Pin = 6;        // 뒷바퀴
int in2Pin = 7;        // 뒷바퀴
int in3Pin = 8;        // 앞바퀴
int in4Pin = 9;        // 앞바퀴
int SpeedPin_F = 2;    // 앞바퀴 속도
int SpeedPin_B = 3;    // 뒷바퀴 속도
int echoPin=4;          // 초음파
int trigPin=5;          // 초음파

void setup() {
        Serial.begin(9600);
        pinMode(in1Pin, OUTPUT);            // 제어 1번핀 출력모드 설정
        pinMode(in2Pin, OUTPUT);            // 제어 2번핀 출력모드 설정
        pinMode(SpeedPin_B, OUTPUT);            // PWM제어핀 출력모드 설정
        pinMode(in3Pin, OUTPUT);            // 제어 3번핀 출력모드 설정
        pinMode(in4Pin, OUTPUT);            // 제어 4번핀 출력모드 설정
        pinMode(SpeedPin_F, OUTPUT);            // PWM제어핀 출력모드 설정
        pinMode(trigPin,OUTPUT);
        pinMode(echoPin,INPUT);                                 // 초음파 들어와야함
}

void loop() {
        // float distance=ultrasonic();

        //   Serial.println(distance);
        // if(distance<30)
        //    StopM();
        //   else
        ForwardM();

        if(Serial.available() > 0){                 // 라즈베리에서 신호들어오면
                int turn = Serial.parseInt();
                Serial.println(turn);
                if(turn == 1)
                RightM();
                if(turn == 2)
                LeftM();
        }
        midM();

}

float ultrasonic(void)
{
        digitalWrite(trigPin,LOW);
        digitalWrite(echoPin,LOW);
        delayMicroseconds(2);
```

```
        digitalWrite(trigPin,HIGH);
        delayMicroseconds(10);
        digitalWrite(trigPin,LOW);

        unsigned long duration = pulseIn(echoPin,HIGH); // 에코핀이 high 유지한 시간(us) 리턴됨

// pulseIn(echoPin, High) pin은 입력모드, value는 포트의 레벨 지정
        float distance=((float)(340*duration)/10000)/2;    // high시간으로 거리 계산 (초음파가 나갔다가 다시 돌아오는 시간)

// 왕복거리이므로 2로 나눈다, 초음파 속도는 340m/s

// duration의 값(high 유지한 시간)은 us단위이므로 s단위로 바꿔준다. (us -> s) duration / 10^6

// 340m = 34000cm

// 거리(cm) = (34000(cm/s) * duration(s) / 10^6)/2

//              340*duration / 10000 / 2
        return distance;
}

void ForwardM()
{
        digitalWrite(in1Pin, LOW);                      //모터가 정방향으로 회전
        digitalWrite(in2Pin, HIGH);
        analogWrite(SpeedPin_B, 150);           //모터 속도를 최대로 설정
}

void StopM()
{
        digitalWrite(in1Pin, LOW);                      //모터가 정방향으로 회전
        digitalWrite(in2Pin, LOW);
        analogWrite(SpeedPin_B, 100);           //모터 속도를 최대로 설정
}

void LeftM(){
        digitalWrite(in1Pin, LOW);                      //모터가 정방향으로 회전
        digitalWrite(in2Pin, HIGH);
        analogWrite(SpeedPin_B, 150);
        digitalWrite(in3Pin, LOW);                      //모터가 정방향으로 회전
        digitalWrite(in4Pin, HIGH);
        analogWrite(SpeedPin_F, 250);           // 250 : 조향 각을 크게
        delay(1000);
}

void RightM(){
```

```
        digitalWrite(in1Pin, LOW);                  //모터가 정방향으로 회전
        digitalWrite(in2Pin, HIGH);
        analogWrite(SpeedPin_B, 150);          // 150 : 값이 크면 빠르게 달린다
        digitalWrite(in3Pin, HIGH);                 //모터가 역방향으로 회전
        digitalWrite(in4Pin, LOW);
        analogWrite(SpeedPin_F, 255);          // 255 : 조향 각을 크게
        delay(1000);
}

void midM(){
        digitalWrite(in3Pin, LOW);                  // 앞바퀴에 힘 풀려고 (축을 중앙으로 위치)
        digitalWrite(in4Pin, LOW);
        analogWrite(SpeedPin_F, 150);
}
```