LED 제어

OneKick

목차

Basic learning

기초 학습

구조(Structure)

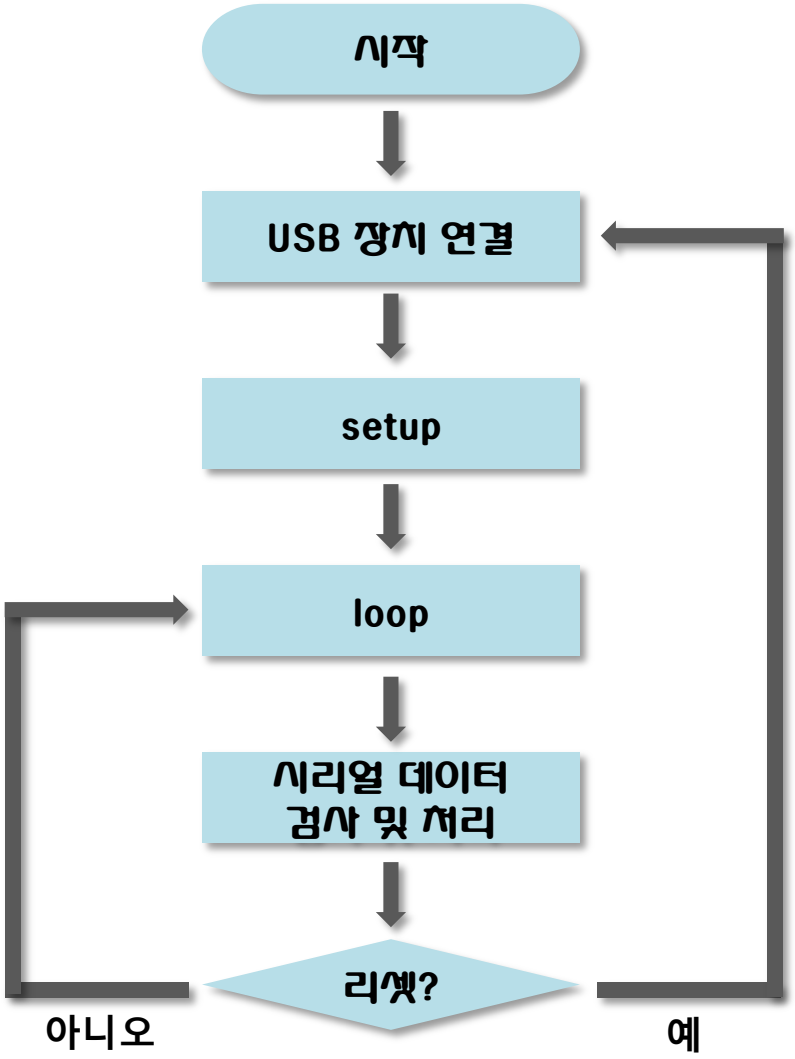
OneKick

- C++ 언어 사용
  - C/C++프로그램와 달리 setup()과 loop() 함수로 구성
  - 아두이노 소프트웨어는 자신이 main()함수를 정의하고 있음
  - arduino-1.x.x₩hardware₩arduino₩avr₩cores₩arduino₩main.cpp 참조

```
int main(void)
{
    init();                         // MCU 초기화

    initVariant();

#if defined(USBCON)                 // USB 장치 연결 설정
    USBDevice.attach();
#endif

    setup();                        // 사용자 초기화 함수

    for (;;) {
        loop();                     // 사용자 무한 반복 함수
        if (serialEventRun) serialEventRun();       // 시리얼 데이터 처리
    }

    return 0;
}
```

구조(Structure)

- 아두이노 스케치의 처리 흐름도



구조(Structure)

OneKick

- setup()
  - 스케치(아두이노 프로그램)가 시작 될 때 호출
  - 변수 초기화, 핀 모드 설정, 라이브러리 초기화 작업
  - 보드의 전원을 켜거나 Reset 되었을 때 한번 실행

- loop()
  - setup() 함수에서 초기화를 수행한 후에 loop() 함수 호출
  - 연속적으로 반복하여 실행
  - 아두이노 보드의 동작을 제어하는 프로그램 실행 함수

Digital I/O

OneKick

- pinMode( )
  - **특정 핀을 입력 또는 출력으로 설정**
  - 디지털 값을 읽거나(입력) 쓰도록(출력) 핀을 설정
  - 함수는 pinMode(pin, mode)의 형태로 사용
    - pin은 mode를 설정하고자 하는 핀의 번호
    - mode는 INPUT, OUTPUT, INPUT_PULLUP

- digitalWrite( )
  - **데이터를 출력하기 위한 함수**
  - 디지털 핀에 HIGH 또는 LOW를 출력
  - 함수는 digitalWrite(pin, value)의 형태로 사용
    - HIGH 일 경우에는 5V(VDD = 5V), LOW일 경우 0V

- digitalRead( )
  - **데이터 입력 값을 읽어오는 함수**
  - HIGH 또는 LOW 값
  - 함수는 digitalRead(pin)의 형태로 사용

Digital I/O

OneKick

- INPUT | OUTPUT | INPUT_PULLUP
  - **pin의 입출력 방향 설정**
  - ex) pinMode(pin, INPUT); , pinMode(pin, OUTPUT);
  - INPUT
    - pin은 입력으로 설정
    - high-impedance 상태 설정
    - 센서 값을 읽기에 유용하지만 LED 제어 불가
    - pull-up 및 pull-down 저항을 연결해야 할 경우도 있음
  - OUTPUT
    - pin은 출력으로 설정
    - low-impedance 상태 설정
    - LED 제어에 유용하나 센서 값을 읽기에 부적합
  - INPUT_PULLUP
    - pin을 입력으로 설정
    - MCU의 내부에서 pull-up 저항 연결

Digital I/O

OneKick

- HIGH | LOW
  - **pin이 취할 수 있는 유일한 두 가지 값**
  - ex) digitalWrite(pin, HIGH); , digitalWrite(pin, LOW);
  - HIGH
    - 값 1
    - pin의 입력 또는 출력 설정에 따라 차이
    - pin을 입력(INPUT) 설정한 경우
      - pin을 읽을 때, 3V(VDD=5V) 이상의 전압이 걸리면 HIGH로 인식
      - HIGH로 출력할 경우 내부 풀업저항을 설정
    - pin을 출력(OUTPUT) 설정한 경우
      - 출력 값을 HIGH로 설정할 경우 pin은 5V
  - LOW
    - 값 0
    - pin을 입력(INPUT) 설정한 경우
      - pin을 읽을 때, 1.5V(VDD=5V) 이하의 전압이 걸리면 LOW로 인식
    - pin을 출력(OUTPUT) 설정한 경우
      - 출력 값을 LOW로 설정할 경우 pin은 0V

전체 구조

OneKick

- 예시

```
int pinLed = 13;

void setup()                    // 초기화
{
  pinMode(pinLed, OUTPUT);
}

void loop()                     // 반복 실행
{
  digitalWrite(pinLed, HIGH);
  delay(1000);
  digitalWrite(pinLed, LOW);
  delay(1000);
}
```

기호

OneKick

- ; (semicolon)
  - 명령의 끝에 사용

```
int a = 13;
```

- {} (중괄호)
  - 열기 중괄호 "{"는 닫는 중괄호 "}"가 반드시 있어야 함
  - for, while, if문은 아래와 같이 중괄호 사용시에는 중괄호 안에 모든 내용을 실행
  - 중괄호 생략 시 ";"으로 구분되는 1개의 명령만 실행

```
if(x > 50)
{
  // 조건부가 참일 때 실행
}
else
  // 조건부가 거짓일 때 실행
```

기호

OneKick

- // 와 /* */
  - 주석은 프로그램에 대한 내용을 다른 사람에게 알리는데 사용
  - 컴파일러에 의해 제거
  - 프로그램의 공간을 차지하지 않음
  - 프로그램의 이해 또는 기억 및 작동 방법을 다른 사람에게 알릴 수 있도록 하기 위한 것
  - 주석을 표시하는 방법은 두 가지
    - 첫째 "//"는 한 라인, "/* */"는 여러 라인을 주석으로 사용
    - 여러 라인을 주석으로 처리하기 위해서 "/*"와 "*/"사이에 내용을 적어 사용

상수

- true | false
  - 아두이노 언어에는 참과 거짓을 표현하는데 사용
  - false
    - 값 0
  - true
    - 값 1
    - 0 이 아닌 모든 정수는 논리적 의미에서 true
  - HIGH, LOW, INPUT & OUTPUT 과 달리 소문자

상수

OneKick

- integer constants(정수형 상수)
  - 프로그램에서 직접 사용
  - 표현할 수 있는 범위는 단어의 길이에 의존
  - 일반적으로 10진수 정수로 처리
  - 정수형 상수는 4가지로 표현
    - 2진수(Binary), 8진수(Octal), 10진수(Decimal), 16진수(Hexa)
    - 2진수는 0과 1로 표현, 숫자 앞에 B 표시
    - 8진수는 0~7로 표현, 숫자 앞에 0 표시
    - 10진수는 0~9로 표현, 숫자 앞에 표시 없음
    - 16진수는 0~9 와 A~F까지 표현, 숫자 앞에 0x 표시

	2진수	8진수	10진수	16진수
123	B1111011	0173	123	0x7B
203	B11001011	0313	203	0xCB

상수

- floating point constants(부동 소수점 상수)
  - 코드를 더 알아보기 쉽도록 하는데 사용
  - 과학적 표기법으로 다양하게 표현
  - 'E'와 'e'는 모두 유효한 지수의 지표로 사용 가능

일반적 표현	또 다른 표현	Floating-point constant
10	10	10.0
234000	2.34 * 10^5	2.34E5
0.000067	6.7 * 10^-5	6.7E-5

데이터 타입

OneKick

- void
  - 함수 선언에 사용
  - 이 함수는 자신을 호출한 함수에 반환할 정보가 없음을 나타냄

```
void setup()
{
   //…
}
void loop()
{
   //…
}
```

데이터 타입

OneKick

- boolean
  - true 또는 false 두 가지 값 중 하나
  - boolean 변수는 메모리의 1byte 차지

- char
  - 문자 값을 저장하는 메모리의 1byte를 차지하는 데이터 형식
  - 'A' 처럼 문자형 리터럴은 작은 따옴표로 작성
  - 여러 문자열은 "ABC" 처럼 큰 따옴표로 작성

- unsigned char
  - 메모리의 1byte를 차지하는 부호 없는 데이터 형식
  - byte의 데이터 형식과 동일
  - 0에서 255까지의 8bit 부호 없는 수를 저장

데이터 타입

OneKick

- int
  - 정수 저장에 대한 기본 데이터 유형
  - **MCU에 따라 저장 범위**가 다름
  - 아두이노 Uno 및 기타 ATmega 기반 MCU
    - -2^{15}에서 $2^{15}-1$까지의 범위를 갖는 16bit(2byte) 값
  - 아두이노 due
    - -2^{31}에서 $2^{31}-1$까지의 범위를 갖는 32bit(4byte) 값

- unsigned int
  - int형과 거의 같으며, 부호가 없는 수를 표시
  - 아두이노 Uno 및 기타 ATmega 기반 MCU
    - 0에서 $2^{16}-1$까지의 범위를 갖는 16bit(2byte) 값
  - 아두이노 due
    - 0에서 $2^{32}-1$까지의 범위를 갖는 32bit(4byte) 값

- word
  - 0에서 65535까지의 부호가 없는 16bit(2byte) 값 저장

데이터 타입

- long
  - -2^{31}에서 $2^{31}-1$까지의 범위를 갖는 32bit(4byte) 값을 저장
- unsigned long
  - 0에서 $2^{32}-1$까지의 범위를 갖는 32bit(4byte) 값
- short
  - 16bit 데이터 형으로 -2^{15}에서 $2^{15}-1$까지의 범위를 갖는 2byte 값으로 저장

데이터 타입

OneKick

- float
  - 부동소수점 숫자 데이터 형식으로 소수점이 있는 숫자
  - 정수보다 큰 해상도를 가지고 있으므로, 아날로그 및 연속 값에 근접하는데 사용
  - 3.4028235e+38에서 3.4028235e+38까지의 범위를 갖는 4byte 값으로 저장
  - 단, 정확하지 않으며, 비교 시 올바르지 않은 결과값 도출
    - 예를 들어 "6.0 / 3.0" 은 2.0과 동일하지 않을 수 있음
  - integer 연산보다 훨씬 느리므로, 동작 시간이 중요한 경우에는 사용 자제

- double
  - 아두이노 Uno 및 기타 ATmega 기반 MCU에서는 float과 동일
  - 아두이노 due에서는 8byte로 저장

- **String**
  - 문자열, 문자만으로 이루어지는 배열
  - int나 char와 달리 클래스로 정의
  - char를 사용한 문자열은 번거로운 반면, String은 쉽게 제어
  - ex) String str = "abcd";

아두이노의 led 깜빡거리기

OneKick

- 파일 →예제 →basic→ blink예제

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

LED Control

LED 제어

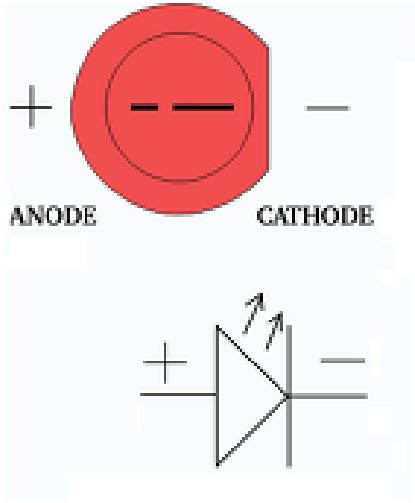
개요

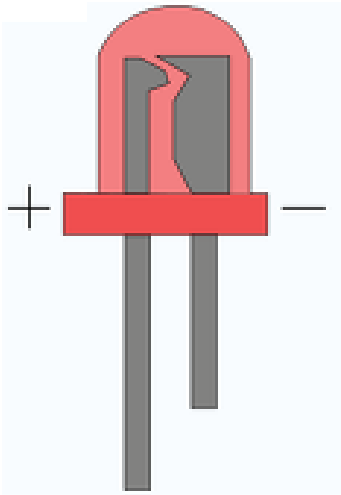
- 목적
  - 디지털 출력의 개념 이해
  - 디지털은 '0'과 '1'로 표현
  - Arduino Uno의 Atmega328 -> VCC 입력전압 5V 사용, High는 5V
  - Uno 보드 I/O 출력은 High -> 5V, Low -> 0V 출력

디지털 신호

개요

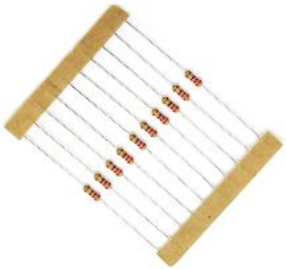
- 관련이론
  - LED는 Light Emitting Diode의 약자로 화합물 반도체의 특성을 이용해 전기에너지를 광(빛) 에너지로 변환시켜 주는 반도체 소자
  - LED 극성 주의

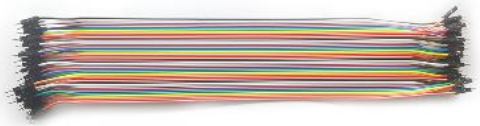




LED 깜빡이기

- 준비물
  - 아두이노 & USB 케이블
  - LED -1개
  - 220 저항 -1개
  - 브레드 보드 – 1개
  - M/M 점퍼 케이블 -2개

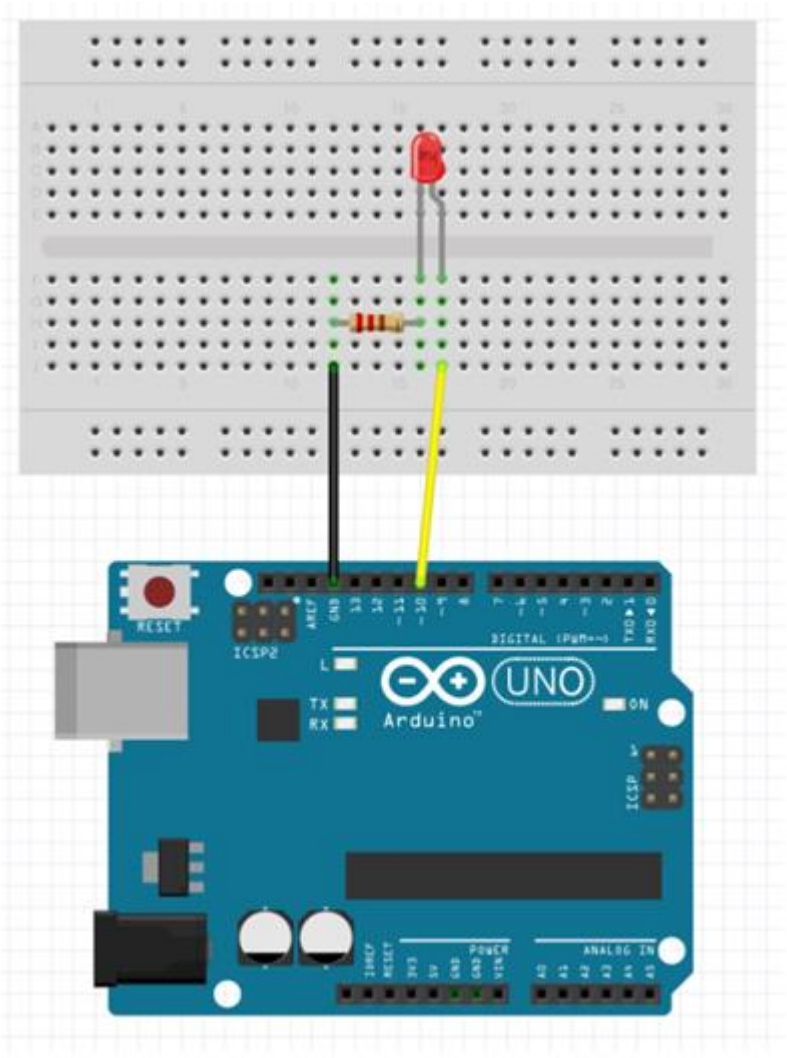




LED 깜빡이기(회로도)

- LED 점등 회로 구성
  - 디지털 10번 핀에 LED(+)연결(긴선)
  - LED(-)와 저항 220 연결
  - 저항 220옴과 GND핀에 연결

Name	LED Pin Number	Arduino Pin Number
LED0	①	10
GND	②	GND



LED 깜빡이기(코드설명)

- 프로그램 설명
  - 사용할 핀 정의

```
int pinLed = 13;
```

  - 초기화 구문

```
pinMode(pinLed, OUTPUT);      // LED pin Output 설정
```

  - loop 구문

LED 핀을 HIGH로 출력하고 1초 지연, 다시 LOW로 출력하고 1초 지연

```
digitalWrite(pinLed, HIGH);
delay(1000);
digitalWrite(pinLed, LOW);
delay(1000);
```

LED 깜빡이기(코드)

OneKick

```
int ledPin = 10; //LED가 연결된 아두이노의 디지털 10번(D10)은 "ledpin"으로 정의

void setup() {

        pinMode(ledPin, OUTPUT);  //ledpin(D10)은 출력
}

void loop() {
        digitalWrite(ledPin, HIGH); //ledpin(D10)에 HIGH의 디지털 출력
        delay(1000);

        digitalWrite(ledPin, LOW); //ledpin(D10)에 LOW의 디지털 출력
        delay(1000);

}
```

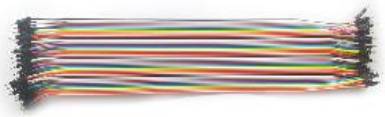
LED웨이브

- 6개의 LED와 반복문 for함수를 이용하여
  LED가 순차적으로 켜지고 꺼지는 웨이브 형식의 on/off 실험

- 준비물
  - 아두이노 & 케이블
  - LED – 6개
  - 220옴 저항 – 6개
  - 브레드보드 – 1개
  - M/M 점퍼케이블 – 7개





도전해보기

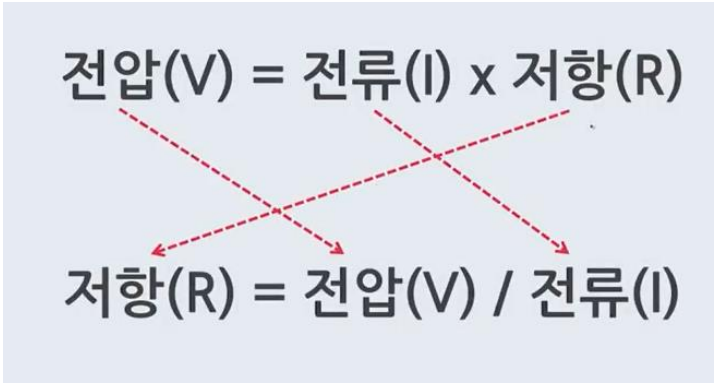
OneKick

- 멈추는 시간을 다르게 설정하기
- 다른 핀에 연결해보기
- LED 개수를 늘려서 실험하기

저항

- 저항 – LED와 같은 전자부품이 과전류로 인해 망가지는 것을 방지
  - 수도꼭지처럼 전기의 흐름을 제어
  - 5v가 흐르면 망가질 수 있음
  - 저항은 열에너지가 발생하므로 조심
  - 옴의 법칙 : V (전압) = I (전류) * R (저항)
    - 전압 : 얼마나 압력이 있는지
    - 전류 : 얼마나 흘러가는지
    - 저항 : 얼마나 안흘러가게 막는지

저항 계산하기



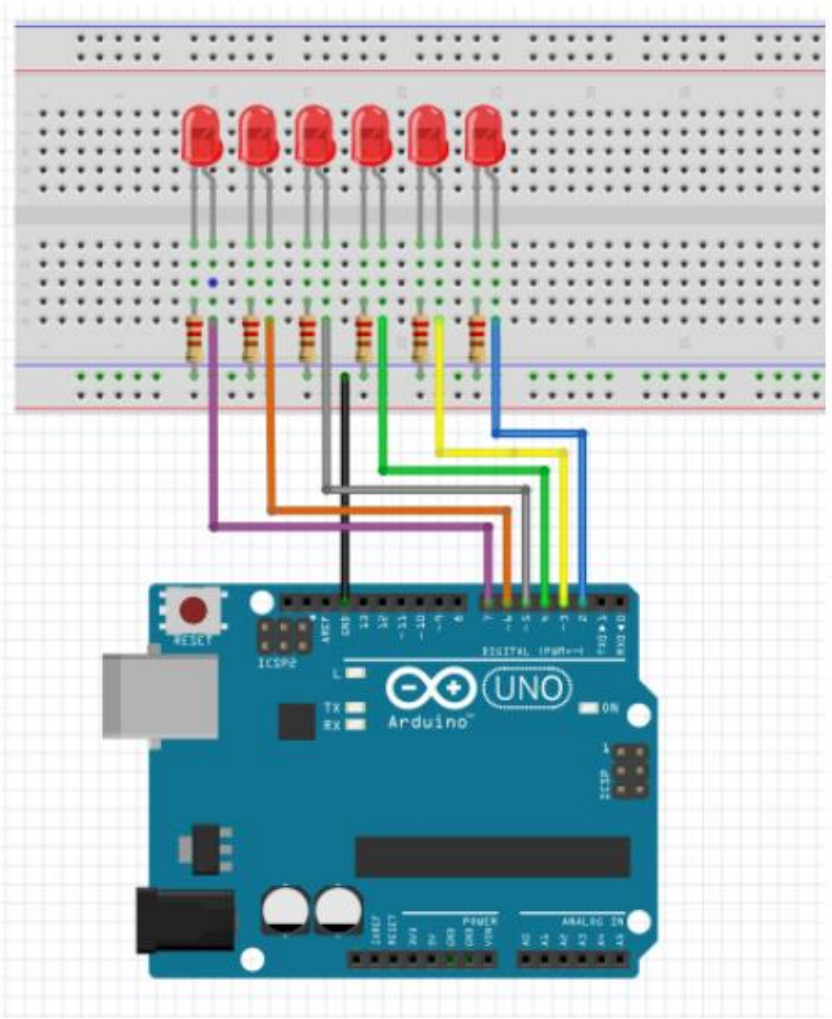
- 저항 R =
  전압(아두이노 전압 5V – LED 필요전압 1.7V) X 전류 (LED 필요전류 10mA)

- 저항 R = (5 – 1.7) X 0.01

- LED 앞에 연결해야하는 저항값 R = 330

LED웨이브(회로도)

- 회로도
  - 각 LED의 +쪽을 디지털 핀에 연결
  - 각 LED의 –에 220옴 저항연결
  - 아두이노의 GND에 연결



  - 저항을 이용하여 빵판연결

Name	LED Pin Number	Arduino Pin Number
LED0	①	2
LED1	②	3
LED2	③	4
LED3	④	5
LED4	⑤	6
LED5	⑥	7
GND	⑨	GN D

LED웨이브(코드)

OneKick

```
#define DELAY_TIME 100

void setup() {
      pinMode(2, OUTPUT);     // 디지털 핀 출력 설정
      pinMode(3, OUTPUT);
      pinMode(4, OUTPUT);
      pinMode(5, OUTPUT);
      pinMode(6, OUTPUT);
      pinMode(7, OUTPUT);
      }

void loop() {
      digitalWrite(2, HIGH);     //디지털 핀 출력은 LOW로 순차 꺼짐
      delay(DELAY_TIME);         //지연 0.1초
      digitalWrite(2, LOW);      //디지털 핀 출력은 LOW로 순차 꺼짐
      digitalWrite(3, HIGH);     //디지털 핀 출력은 LOW로 순차 꺼짐
      delay(DELAY_TIME);         //지연 0.1초
      digitalWrite(3, LOW);
```

LED웨이브(코드)

```
    digitalWrite(4, HIGH);      //디지털 핀 출력은 LOW로 순차 꺼짐
    delay(DELAY_TIME);          //지연 0.1초
    digitalWrite(4, LOW);
    digitalWrite(5, HIGH);      //디지털 핀 출력은 LOW로 순차 꺼짐
    delay(DELAY_TIME);          //지연 0.1초
    digitalWrite(5, LOW);
    digitalWrite(6, HIGH);      //디지털 핀 출력은 LOW로 순차 꺼짐
    delay(DELAY_TIME);          //지연 0.1초
    digitalWrite(6, LOW);
    digitalWrite(7, HIGH);      //디지털 핀 출력은 LOW로 순차 꺼짐
    delay(DELAY_TIME);          //지연 0.1초
    digitalWrite(7, LOW);

    }
```

응용실습

응용실습

- 홀수, 짝수 순서로 변경해보기
- 배열로 작성해보기

배열로 바꾸기

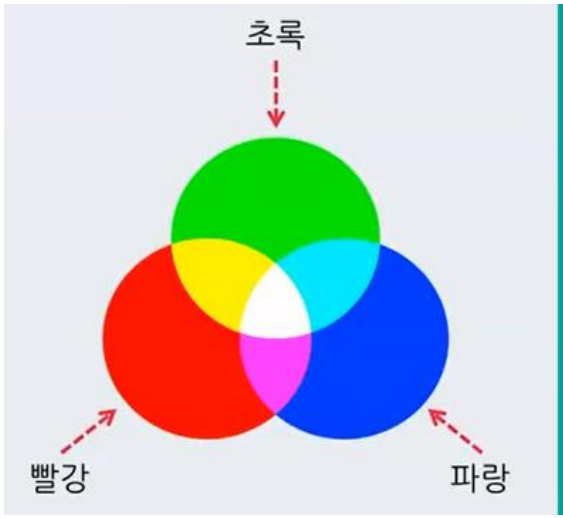
OneKick

- 전체 소스코드

```
1. int aPinLed[8] = {13, 12, 11, 10, 9, 8, 7, 6};
2.
3. void setup() {
4.   for(int i=0; i<8; i++) {
5.     pinMode(aPinLed[i], OUTPUT);
6.   }
7. }
8.
9. void loop() {
10.  int i;
11.  for(i=0; i<8; i++) {
12.    digitalWrite(aPinLed[i], HIGH);   // LED ON
13.  }
14.  delay(1000);
15.
16.  for(i=0; i<8; i++) {
17.    digitalWrite(aPinLed[i], LOW);    // LED OFF
18.  }
19.  delay(1000);
20.}
```

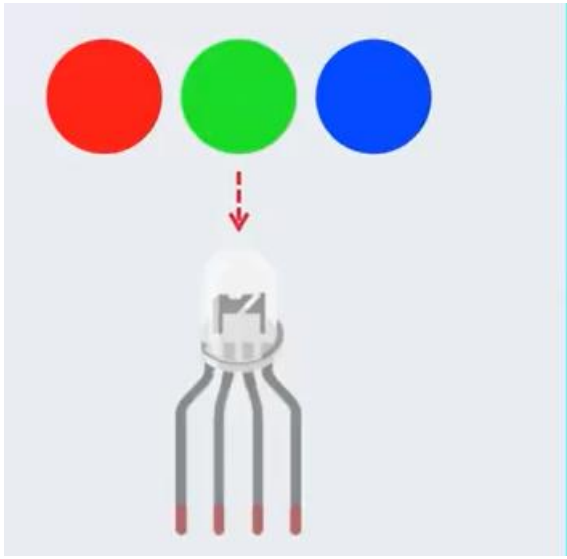
삼색 LED

삼색LED



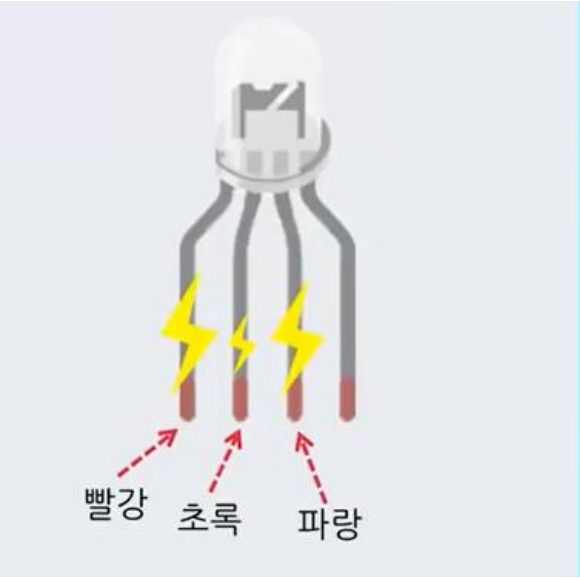
삼원색

삼원색은 빛을 구성할 때
핵심이 되는
세가지 색을 말한다.



삼색 LED는
삼원색의 양을 조절해
원하는 색을 만든다.

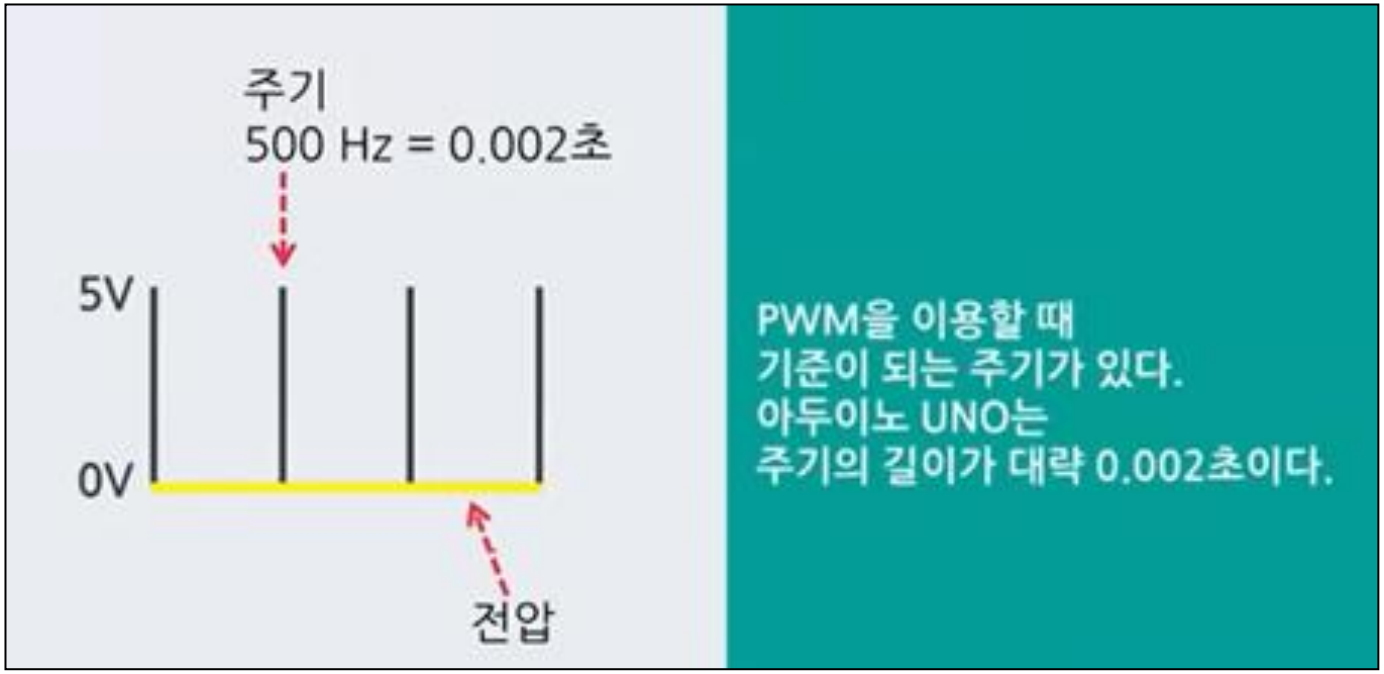
삼색LED



색의 양은 각 색에 해당되는
전압을 0~5V 사이의 값으로
조절해 변경한다.

- pwm : 0v와 5v사이의 주파수를 만들어주는데, 주파수폭의 변화를 줄 수 있는것
- 아두이노 하드웨어는 전압출력을 low또는 high로 설정하는 것 밖에 하지 못함

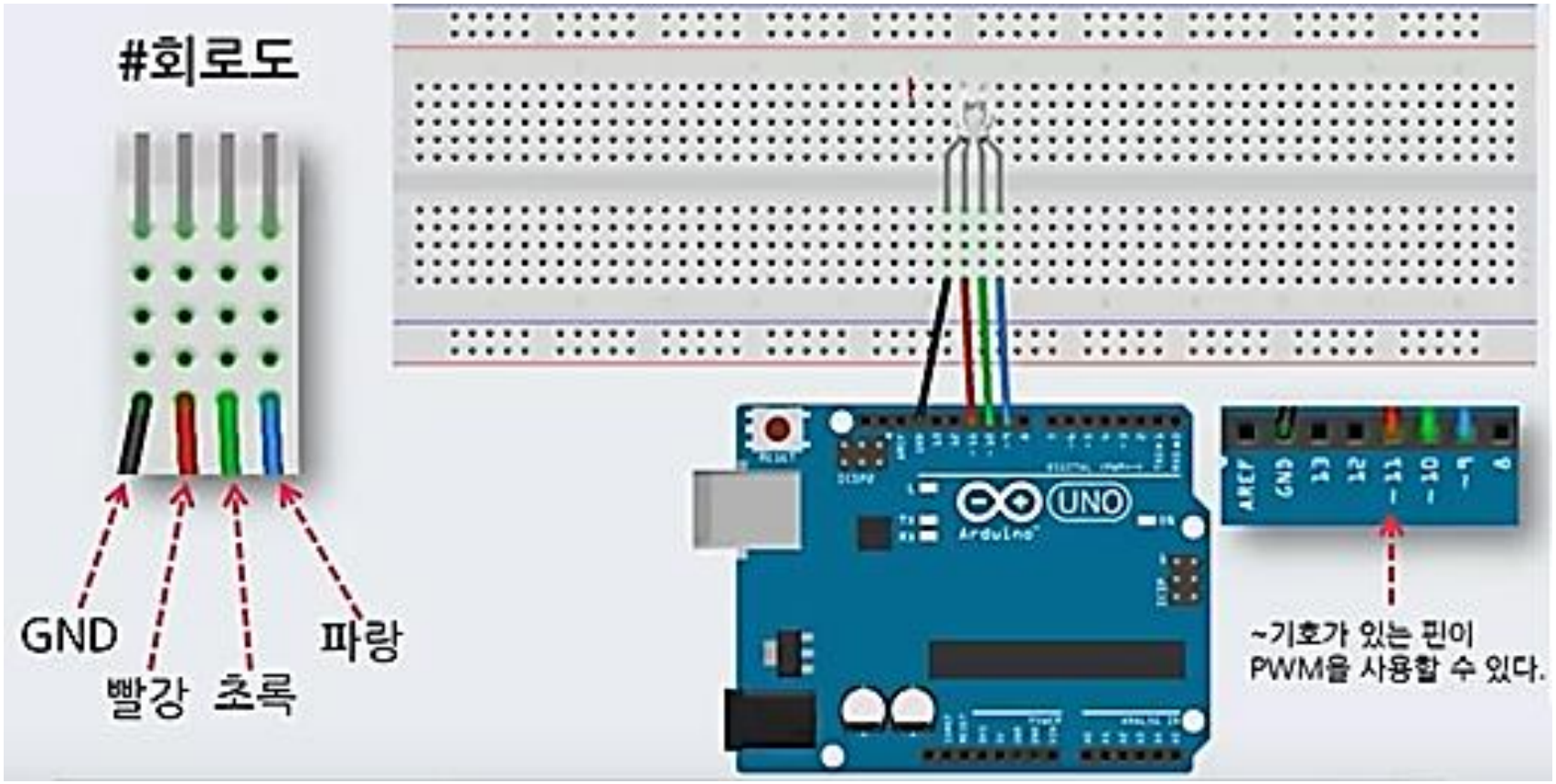
- Hz
  - 우노경우 :1초동안 500번을 껐다켰다하는 것
  - 이 주기동안 계속 전압이 0v 이거나 5v이면 출력전압은 0v이거나 5v가 됨



pwm

- pwm : 변화하는 주기를 바꿔서 표현해줄 수 있음
- 여기서는 5v의 25%인 1.25v가 됨. 이를 통해 전압출력을 0v에서 5v사이의 원하는 값으로 바꿀 수 있음

삼색LED 회로도



삼색LED 코드

OneKick

```
#define RED 11
#define GREEN 10
#define BLUE 9

void setup(){
	randomSeed( analogRead(0));
	//3원색 값을 랜던하게 발생시키기위한 난수발생장치 초기화
	//플로팅 상태로 인해 무작위값이 인식되는 아날로그 입력값을 키값으로 설정
	//계속 다른 값을 읽어오기 위해 사용
}
void loop(){
	analogWrite(RED, random(255);	//난수를 발생시켜 값 넣어줌
	analogWrite(GREEN, random(255); //0~255범위로 난수 생성
	analogWrite(BLUE, random(255);
	delay(1000)
}
```

가변저항

가변저항을 이용하여 LED 밝기 조절

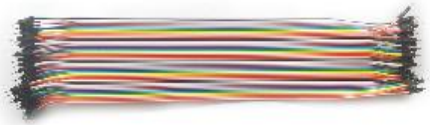
- 아날로그 인터페이스(가변저항)을 이용하여 LED 밝기 조절
- 가변저항의 입력 핀을 읽고 결과를 0~255까지의 범위로 매핑하여 그 값을 사용해 출력 핀의 펄스 폭 변조(PWM) 설정
- LED는 PWM의 펄스 값에 따라 밝기가 변하고 값은 시리얼 모니터에 출력

- 준비물
  - 아두이노 & 케이블
  - 가변저항
  - LED
  - 220옴 저항
  - 브레드보드
  - M/M 점퍼케이블



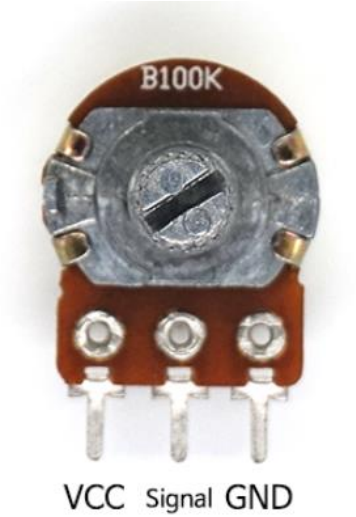


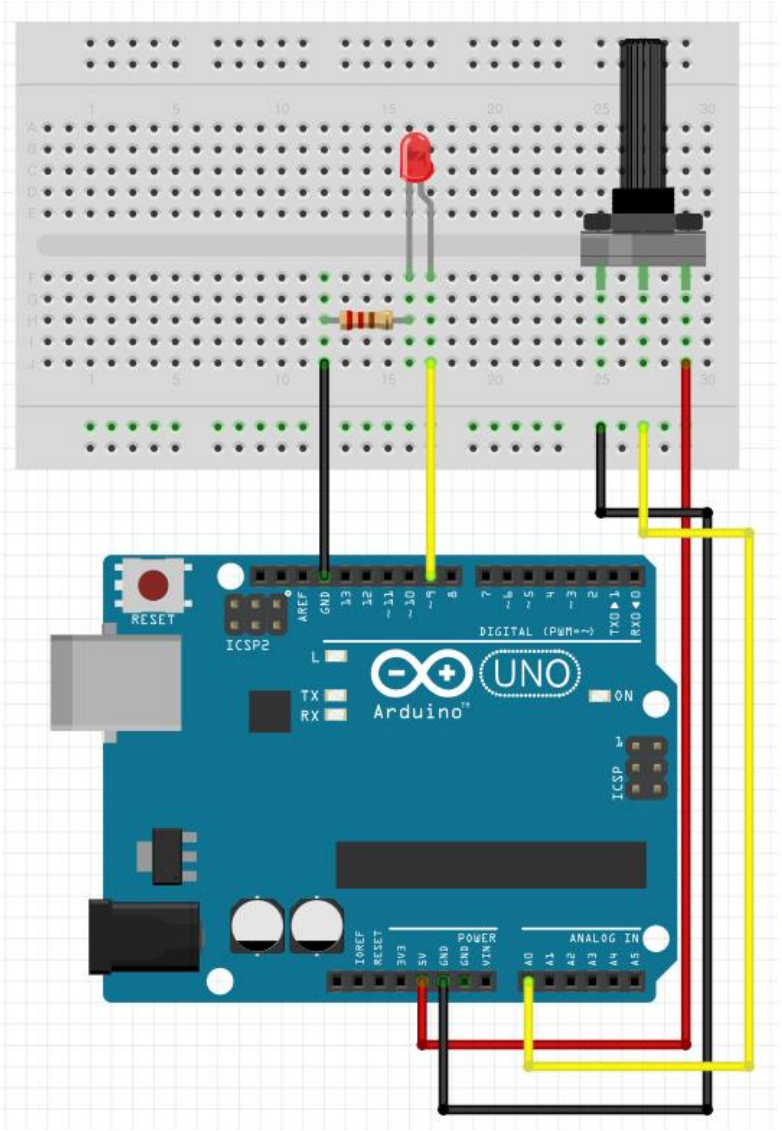




가변저항을 이용하여 LED 밝기 조절(회로도)

- 회로도
  - 디지털 9번 핀에 LED(+)연결
  - LED(-)와 저항 220 연결
  - 저항 220과 GND핀에 연결
  - 가변저항의 GND를 GND에 연결
  - 가변저항의 아날로그 입력핀을 A0에 연결
  - 가변저항의 VCC를 5V에 연결





LED웨이브(코드)

OneKick

- Analog-AnalogInOutSerial 시리얼 모니터 확인

```
const int analogInPin = A0;  // Analog input pin that the potentiometer is attached to
const int analogOutPin = 9; // Analog output pin that the LED is attached to

int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM (analog out)

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  // change the analog out value:
  analogWrite(analogOutPin, outputValue);

  // print the results to the Serial Monitor:
  Serial.print("sensor = ");
  Serial.print(sensorValue);
  Serial.print("₩t output = ");
  Serial.println(outputValue);

  // wait 2 milliseconds before the next loop for the analog-to-digital
  // converter to settle after the last reading:
  delay(2);
}
```