Project 1

LED 켜고 끄기

01

첫 프로젝트 시작하기

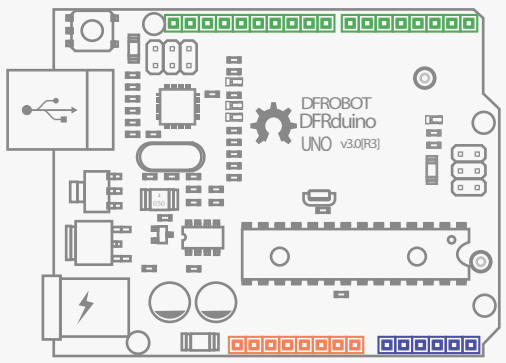
이번 프로젝트에서는 앞에서 사용한 Blink 예제를 약간 수정해서 다시 사용합니다. 매우 쉽고 간단한 내용이지만, 모든 프로그램에 널리 사용되는 매우 중요한 내용이므로 철저히 학습해야 합니다.

"가장 기본적인 것을 가장 철저히 학습" 하는 것이 아두이노 학습의 지름길입니다.

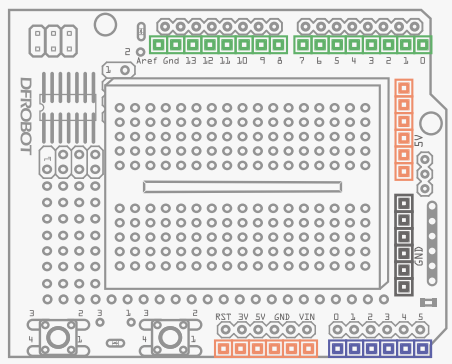
LED 켜고 끄기

이번 프로젝트에서는 앞에서 사용한 Blink 예제를 다시 사용하지만, D10 핀에 LED를 연결하고 사용하는 점이 다릅니다. 이 프로젝트를 익히면 LED의 동작원리와 사용법을 이해할 수 있게 됩니다.

필요한 부품



DFduino UNO R3 x1



Prototype Shield x1

Jumper Cables M/M x2

Resistor 220R x1

5MM LED x1

*사용하는 LED에 따라서 저항 값이 다른 저항을 사용해야 하는 경우도 있습니다. LED에 연결하는 저항 값을 정하는 방법은 뒤에서 자세히 설명합니다.

*DFRduino UNO는 DFRobot사의 엄격한 품질 관리 아래 제작된 아두이노 보드입니다. 아두이노 정품보드와 똑같은 회로를 사용하고 있으며, 기능과 성능도 모두 똑같습니다.

회로구성

(1)쉴드결합하기
먼저 아두이노 보드 위에 "UNO PCB 쉴드"를 포개지도록 겹쳐 꽂아야 합니다. "UNO PCB 쉴드" 아래 쪽에 있는 헤더핀(수)을 "Arduino UNO 보드" 위에 있는 헤더핀(암)에 꽂으면 되는데, 이때 위쪽과 아래쪽 헤더핀들의 위치를 잘 맞춰 꽂아야 하며, 헤더핀(수)이 휘어지지 않도록 주의해야 합니다.

(2)브레드보드 고정
브레드보드의 뒷면에는 접착면 보호종이가 붙어 있습니다. 이 종이를 벗긴 다음, 만능기판 쉴드위에 브레드 보드가 떨어지지 않도록 단단히 눌러서 붙입니다.

(3)전선의 색깔 구분
일반적으로 점퍼선을 사용할 때, +전원(VCC)에 빨간색 전선, –전원 (GND)에 검은색 전선, 디지털 핀에 녹색전선, 아날로그 핀에 파란색 전선을 사용하고, 그 밖의 곳에 흰색 전선을 사용합니다. 가능하면, 사용하는 점퍼선의 색깔은 이 교재와 똑같이 사용하면 좋지만, 다른 색을 사용해도 괜찮습니다.

(4)LED 연결
LED는 다리의 방향이 바뀌지 않도록 주의해야 합니다. LED의 긴 전극을 애노드(양극, Anode)라고 하고 짧은 전극을 캐소드(음극, Cathode)라고 부릅니다.

(5)USB 케이블 연결
아래 그림처럼 부품들을 연결하여 회로구성을 하고, 아두이노 보드에 USB케이블 꽂아 PC와 연결합니다.

DFRuino

칼라전선 사용기준

녹색 : 디지털 핀

청색 : 아날로그 핀

적색 : VCC(+5V)

흑색 : GND(-)

백색 : 기타

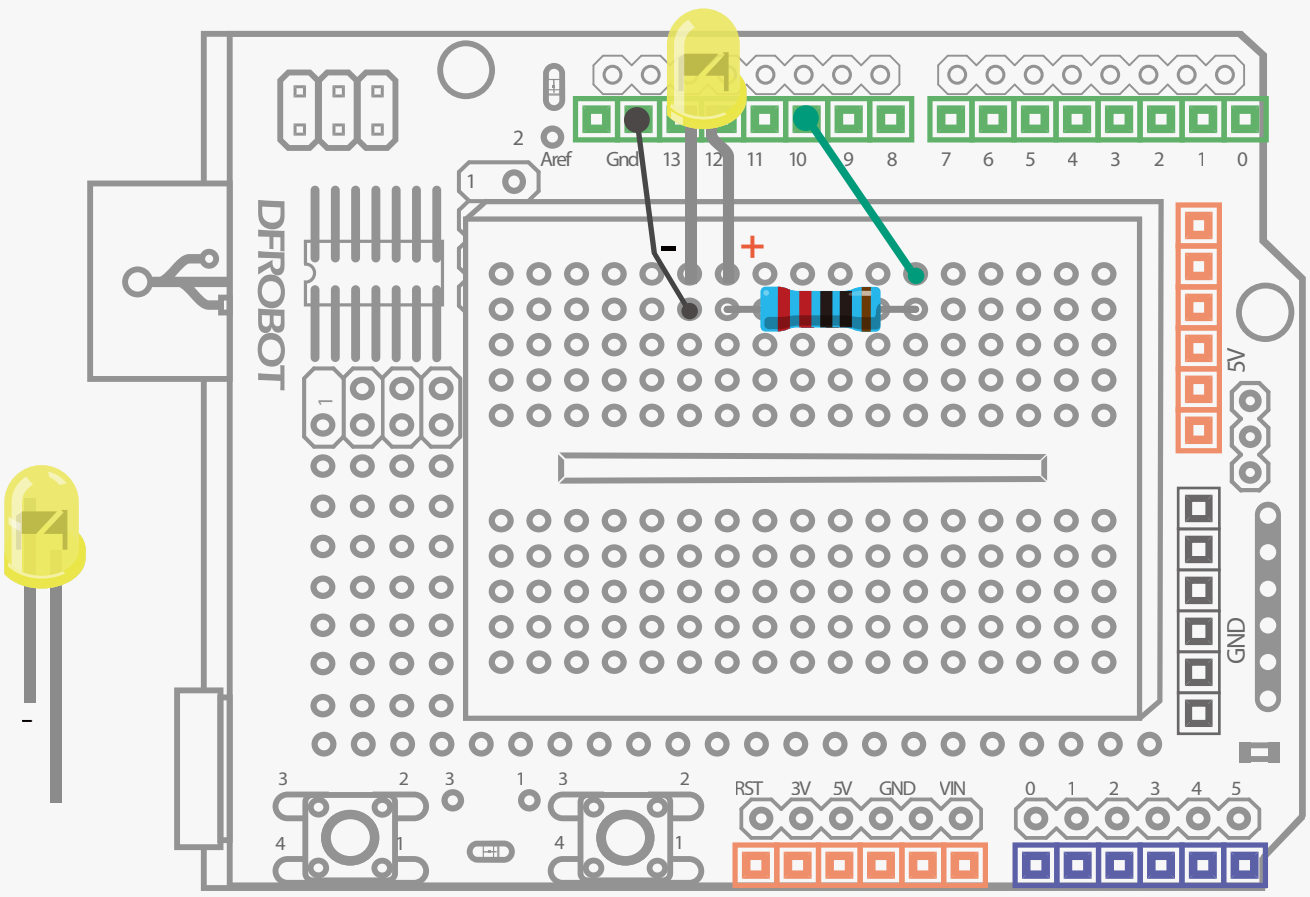


그림 1-1 LED 켜고 끄기 회로구성도

Arduino Sketch

아두이노 IDE를 실행한 뒤 프로그램 예제 1-1을 입력합니다.
프로그램은 복사해서 붙여 넣는 방법보다는 키보드로 직접 입력하는 것이 아두이노 IDE 사용법을 훨씬 빨리 익힐 수 있습니다.

프로그램 예제 1-1:

```
//예제-1 LED 켜고 끄기
/*
     1초 간격으로 LED를 켜고 끄는 프로그램
*/
int ledPin = 10;
void setup() {
      pinMode(ledPin, OUTPUT);
}
void loop() {
      digitalWrite(ledPin,HIGH);
      delay(1000);
      digitalWrite(ledPin,LOW);
      delay(1000);
}
```

프로그램 입력을 마쳤으면, 확인 아이콘을 눌러서 컴파일 합니다
만일 코드에 오류가 있다면, 에러 메시지가 출력되면서 컴파일작업이 중지됩니다.
이 경우, 프로그램의 오류를 찾아 수정한 뒤 다시 컴파일 해야 합니다.

업로드 아이콘을 클릭해서 아두이노 보드에 컴파일한 코드를 업로드 합니다.

업로드가 완료되면, 아두이노 보드는 프로그램을 자동실행 합니다
즉, D10핀에 연결된 LED가 1초 간격으로 계속 깜빡입니다.

CODE 해설

주석 (Comments)

//Project--blink a LED

프로그램 안에 "//" 표시가 있으면, 그 줄의 뒷부분 색깔이 자동으로 회색으로 바뀌게 되는데, 이 부분을 주석 (Comments)이라고 합니다. Arduino IDE는 주석 부분은 컴파일 하지 않습니다.

주석문은 프로그램에 적는 일종의 메모입니다. 보통 프로그램에 대한 간단한 설명을 적는 경우가 많은 데, 나중에 프로그램 작성자는 물론, 다른 사람이 프로그램을 쉽게 이해하는데 도움이 됩니다.

주석문은 이미 작성된 프로그램 중 일부를 실행되지 않도록 할 때도 사용하는데, "//" 를 붙여 프로그램의 일부를 실행되지 않도록 하는 것을 "주석 처리" 라고 부릅니다.

여러 줄 주석 (Multi-Line Comments)

/* 이 두 기호 사이에 있는 문장은 "여러 줄 주석" 즉, 프로그램에 기록하는 일종의 긴 메모 입니다. 이 부분은 실행 되지 않으며, 회색으로 표시 됩니다 */

여러 줄을 주석처리 할 때는 주석처리 할 부분의 처음과 끝에 /* 와 */를 붙여 줍니다.

한 줄 주석문과 마찬가지로, 주석 처리된 문장 전체가 회색으로 바뀌게 되며, Arduino IDE는 이 부분을 제외하고 컴파일합니다.

여러 줄 주석문은 프로그램에 대한 자세한 설명을 표기하는데 사용됩니다.

변수 선언

int ledPin = 10;

변수형태 변수명

프로그램에서 변수를 사용하려면 아래 처럼 미리 선언해야 합니다.

```
int ledPin = 10;
```

위 변수선언문은 "ledPin 이란 이름의 정수형(int) 변수에 숫자 데이터 10을 저장" 하라는 의미입니다.
변수의 이름은 지을 때는, 변수의 기능을 짐작할 수 있는 이름을 사용하는 것이 좋습니다. ledPin 이란 변수명은 "LED를 연결할 디지털 핀을 저장" 하는데 사용되는 변수임을 쉽게 짐작할 수 있습니다.
변수선언문의 끝에 있는 세미콜론(;)은 생략하면 안됩니다. 세미콜론이 없으면, 이 명령문을 인식하지 못하므로 컴파일할 때 에러 메시지가 출력됩니다. 따라서, 모든 명령 줄마다 ";" 이 빠지지 않았는지 늘 확인해야 합니다.

변수란 무엇인가?

변수는 데이터를 저장하는 기억장소에 이름을 붙인 겁니다. 변수의 뜻을 이해하려면, 변수의 종류(Type), 명칭(name), 저장값(Value)의 의미를 확실히 알아야 합니다.
int ledPin = 10; 에서 int는 변수의 종류가 int형(정수형)임을 의미하고, 변수의 이름은 ledPin이며, 이 변수에 임시로 저장된 값은 10 임을 알 수 있습니다. 아두이노 IDE에서 프로그램에 있는 모든 변수는 오렌지색으로 표시됩니다.
프로그램을 작성할 때 변수를 사용하면 매우 편리합니다. 수시로 변하는 데이터를 프로그램 작성자가 일일이 숫자로 입력하여 프로그램을 작성하는 것보다, 변수명만 입력하고 수시로 바뀌는 데이터는 아두이노가 알아서 처리하도록 하는 것이 훨씬 더 간편합니다.
변수의 이름은 첫 글자는 반드시 영문자이어야 하고, 나머지 글자들은 영문자, 숫자, 밑줄 등을 섞어 사용할 수 있습니다. 변수 이름에는 밑줄 이외의 특수문자를 사용할 수 없으며 대문자와 소문자는 서로 다른 문자로 인식됩니다. 아두이노는 C언어를 기반으로 합니다. 따라서, "main", "if", "while" 처럼 C언어에서 사용하는 명령어와 아두이노 IDE에 의해 미리 예약된 명령어나 함수들은 변수이름으로 사용할 수 없습니다. "예약어는 변수명으로 사용할 수 없다" 는 점을 꼭 기억합시다.

setup( ) 함수

```
void setup() {}
```

반환 값이 없음을 표시

빈 괄호

setup()함수는 아두이노보드에서 프로그램이 시작될 때, 한 번만 실행되는 함수로 아두이노 프로그램에 반드시 있어야 하는 필수함수 입니다. setup() 함수가 없으면 컴파일 작업시 에러메시지가 표시되며 컴파일이 중단됩니다.
setup() 함수에서 실행할 코드는 { }안에 적으면 되는데, 일반적으로 변수의 초기 값 설정, pinMode 설정, 기타 프로그램 실행에 필요한 초기값을 설정하는 코드를 적습니다 .
setup() 함수는 아두이노에 전원을 처음 연결하거나 리셋키를 눌렀을 때 한 차례만 실행 됩니다.

pinMode( ) 함수

```
pinMode(pin, mode);
```

사용할 디지털핀

사용방식

pinMode() 함수는 아두이노 보드의 디지털 핀을 입력(INPUT)으로 사용할지 아니면 출력 (OUTPUT)으로 사용할 지를 설정할 때 사용 합니다. pinMode() 함수를 사용할 때는 2개의 파라미터를 지정해야 합니다. 즉, 위의 명령문 예시처럼 괄호 속에 사용할 입출력 핀(Pin)과 사용모드 (OUTPUT)를 지정해 줘야 합니다.
pinMode를 INPUT으로 지정하면 디지털핀은 하이임피던스 상태가 됩니다. 하이임피던스 상태는 핀의 상태가 HIGH상태도 LOW상태도 아닌 상태로 디지털 핀에 연결된 장치를 오동작시키는 원인이 되기도 합니다. 따라서, INPUT으로 지정할 때는 풀업 또는 풀다운 저항을 연결해야 합니다. 외부 풀업 저항을 사용하지 않고 프로세서 내부 풀업 저항을 사용할 경우는 INPUT_PULLUP 으로 지정하면 됩니다.

"pinMode()함수" : 아두이노의 지정된 핀을 입력과 출력 중 어떤 방식으로 사용할 지를 설정하는 함수로 아래와 같이 2개의 파라미터를 사용합니다.
"pin" : 사용하고자하는 아두이노의 디지털 핀을 지정합니다. 숫자, 변수 또는 수식을 사용할 수 있습니다.
"mode" : 지정한 디지털핀의 사용 방식을 지정 합니다. INPUT, OUTPUT 또는 INPUT_PULLUP중 한 가지를 지정할 수 있습니다 .

연습문제 :
만일 2번 디지털 핀을 입력으로 사용하려고 한다면 pinMode() 함수로 코드를 어떻게 작성해야 할까요?

정 답 : pinMode (2, INPUT);

함 수

프로그램에서 정해진 동작만을 수행하도록 블록화된 코드를 함수라고 부릅니다. 프로그램을 작성할 때, "복잡한 절차" 나 "반복적으로 사용되는 특정 기능" 을 미리 함수로 정의하고, 필요할 때마다 호출하여 사용하면 편리합니다.
아두이노 스케치에는 setup()함수와 loop()함수가 반드시 존재해야 합니다. 함수로 정의하는 부분은 함수명 뒤에 있는 { }안에 위치해야 합니다. 사용자 정의 함수는 반드시 setup() 함수와 loop()함수의 { } 밖에서 정의해야 합니다.

INPUT과 OUTPUT의 차이

INPUT은 누름 스위치를 사용할 때처럼, 어떤 정보가 외부에서 아두이노 보드 쪽으로 전달되는 동작을 의미합니다. 반대로 OUTPUT은 신호가 아두이노 보드에서 LED나 스피커 같은 외부 장치로 전달되는 동작을 의미합니다. 아두이노 보드와 외부 장치 사이에 정보교환은 디지털 핀이나 아날로그 핀을 통해서만 이루어 집니다.

메인 프로그램. (루프함수)

```
void loop() {
        digitalWrite(led-
Pin,HIGH);
        delay(1000);
        digitalWrite(led-
Pin,LOW);
        delay(1000);
```

delay( ); 함수

```
        delay (1000);
```

delay() 함수는 ( ) 속에 적은 시간 동안 프로그램의 실행을 멈추게 합니다. (단위 mS)

digitalWrite( ); 함수

```
digitalWrite(pin,value);
```

사용할 디지털핀

출력 값

아두이노 프로그램에는 반드시 setup ()함수와 loop()함수가 포함되어 있어야 합니다. 그렇지 않으면 프로그램이 바르게 실행되지 않습니다.

setup()함수는 변수 값을 초기화 하고 프로그램 실행에 필요한 각종 기능을 설정하는데 사용되는데, 단 한 차례만 실행됩니다.

loop()함수는 그 이름이 말 하듯이 프로그램에서 지시한 연산처리나 , 입출력 작업등을 계속해서 반복하는 메인 프로그램 입니다.

loop () 함수의 첫 문장을 보면, 아래와 같이 digitalWrite() 함수가 사용된 것을 알 수 있습니다.

```
digitalWrite(ledPin,HIGH);
```

digitalWrite() 함수는 디지털 핀에 HIGH(on) 또는 LOW(off)신호를 출력하도록 지시하는 함수입니다.

HIGH 신호가 출력되면 D10 핀에 연결된 LED가 켜지고, LOW신호가 출력되면 꺼집니다.

digitalWrite()함수를 사용하려면, 먼저, pinMode() 함수로 디지털 핀을 OUTPUT 모드로 설정해야 합니다. 이 경우, HIGH 신호를 출력하면 디지털 핀에 5V가 출력되고, LOW 신호를 출력하면 0V 가 출력 됩니다.

만일, 아두이노 보드의 디지털 핀을 OUTPUT으로 설정하지 않고 digitalWrite()함수를 사용 하면 어떻게 될까요?
그렇게 사용하면 안되는 이유는 무엇일까요?
아래 설명을 읽어보세요.

pinMode()함수가 digitalWrite()와 digitalRead()에 미치는 영향

일 pinMode()함수를 사용해서 아두이노의 디지털 핀을 INPUT으로 설정했다면, digitalWrite()함수는 사용할 수 없고, digitalRead() 함수만 사용해야 합니다.
반대로, OUTPUT으로 설정했다면 digitalWrite()함수만 사용해야 합니다.

아두이노의 디지털 핀을 OUTPUT으로 설정하지 않고, 디지털 핀에 LED를 연결하고 digitalWrite()함수로 HIGH신호를 출력하면 LED는 밝게 켜지지 않고, 아주 어둡게 켜집니다. 왜냐하면 디지털 핀에 연결되어 있는 내부 저항이 LED로 흐르는 전류의 흐름을 방해하기 때문입니다.

LED나 부저 사용시 (출력)	누름 스위치 사용시 (입력)
pinMode(pin,OUTPUT)	pinMode(pin,INPUT)
digitalWrite(pin,HIGH/LOW)	digitalRead(pin)

Hardware 탐구

브레드 보드

브레드 보드는 전선이나 부품을 영구적으로 납땜하지 않고, 임시로 연결하여 실험할 회로를 구성할 때 사용 합니다.

브레드 보드에는 여러 개의 5핀 소켓들이 일정한 간격으로 배열되어 있습니다. 전자부품이나 전선을 5핀 소켓의 작은 구멍에 삽입하면, 브레드보드 위에 원하는 전기회로를 손 쉽게 구성할 수 있습니다. 브레드 보드의 구멍과 구멍사이의 간격은 0.1인치(약 2.5mm)입니다

브레드보드의 위쪽 블록과 아래쪽 블록에는 5핀□소켓이 17개씩 배치 되어 있습니다. 5핀 소켓은 세로 방향으로는 전기가 통할 수 있도록 서로 연결되어 있지만, 가로 방향 으로는 서로 전기가 통하지 않습니다.

브레드보드의 위쪽블록과 아래 쪽 블록들도 서로 전기가 통하지 않습니다.

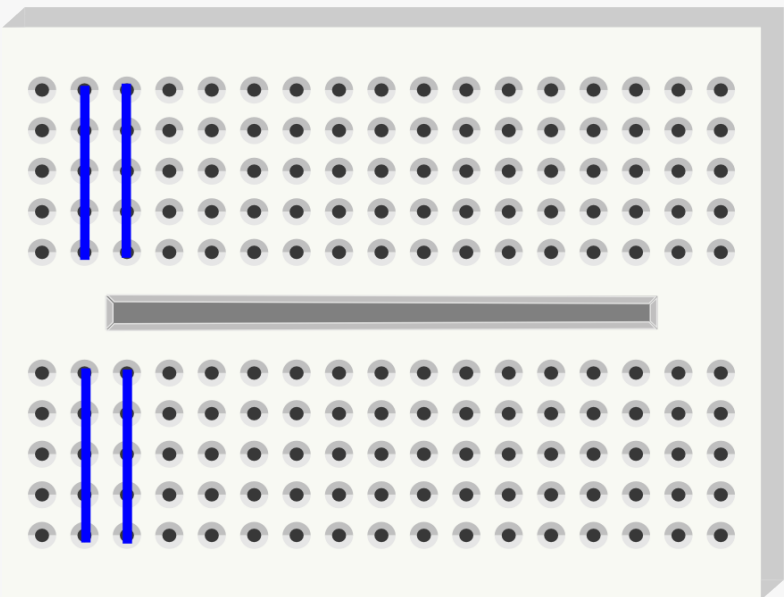


그림 1-2 브레드보드 연결방

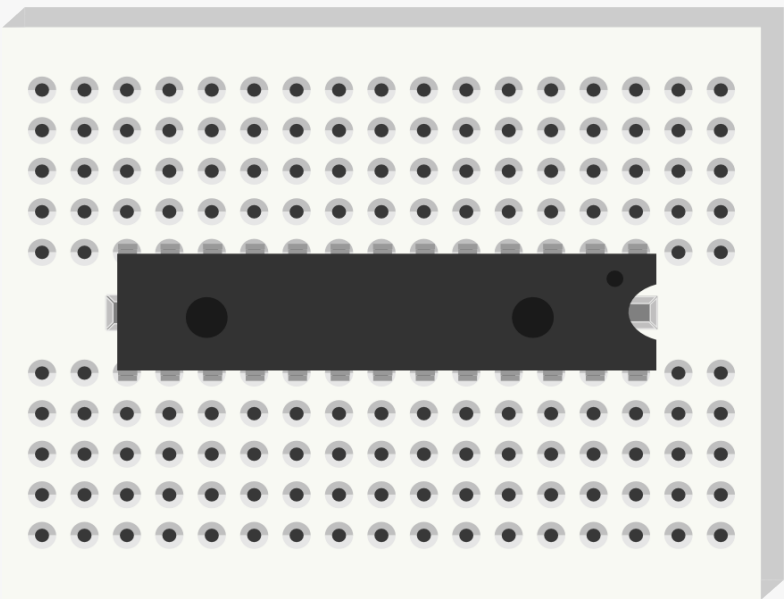


그림 1-3 브레드 보드와 DIP IC

저항

저항은 이름에서도 알 수 있듯이, 전기의 흐름을 억제하는 역할을 하는 부품입니다. 저항 값이 크면 클수록 전기회로에 전류가 더 적게 흐릅니다.

저항의 단위는 옴인데, 그리스 글자의 마지막 글자인 오메가(Ω)로 표시합니다.

저항은 LED와는 다르게 극성을 갖지 않는 전자부품입니다. 즉, +극과 -극이 따로 정해져 있지 않기 때문에 저항의 다리 방향을 바꿔서 사용해도 됩니다.

일반적으로 LED에 불을 켜려면 3V, 20mA의 전류를 흘려주어야 하는 데, 5V전원에 연결할 때는 100~220옴 정도의 전류제한용 저항을 직렬로 연결하고 사용합니다.

만일 100옴보다 낮은 저항을 사용하고, 5V전원에 연결할 경우, LED가 타버릴 수도 있습니다. 또한, 저항에서는 열이 발생할 수도 있으므로 조심해야 합니다.

저항의 색 코드 읽기

일반적으로 저항 값은 저항의 외부에 띠모양의 색으로 표시합니다.
본 교재에는 저항 값과 함께 색코드도 안내하기 때문에 어렵지 않게 저항을 사용할 수 있습니다.
하지만, 교재 없이도 저항 값을 알려면, 저항의 색깔 띠가 무엇을 의미하는 지를 알고 있아야 합니다.
저항의 색 코드를 보고 저항 값을 계산하는 방법은 아래 소개된 웹 사이트에서 확인할 수 있습니다.

http://www.21ic.com/tools/component/201003/54192.htm

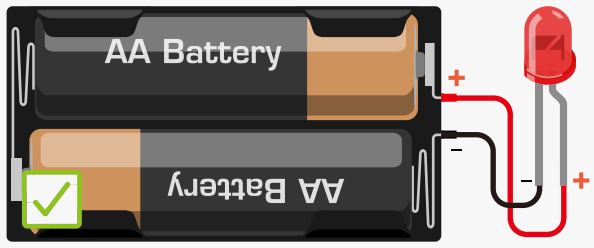
발광다이오드(LED)

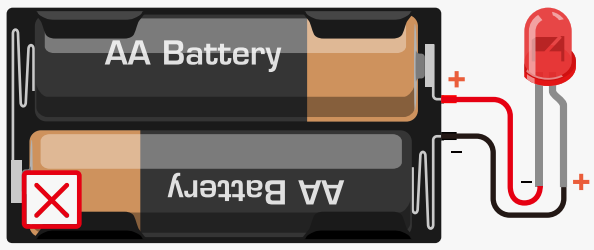
발광다이오드(LED)는 정해진 방향으로 전류가 흐르면 빛을 내는 전자부품입니다.

LED는 반도체로 만들어진 일종의 PN접합 다이오드입니다. LED의 + 극을 양극 (에노드, Anode), -극을 음극 (캐소드 , Cathode)라고 합니다.

LED의 전극은 다음 2가지 방법으로 확인할 수 있습니다.
첫째는, LED의 다리 길이를 비교하는 방법입니다. 긴 쪽이 + 극(Anode)이고 짧은 쪽이 -극 (Cathode) 입니다.
또 다른 방법은, 다리와 연결된 내부 전극의 모양을 살펴보는 방법입니다. -극 다리와 연결된 내부 전극의 면적이 훨씬 더 넓습니다.

LED는 극성을 바꿔서 연결하면 불이 켜지지 않습니다. LED의 양극(Anode)을 전원의 +극에 (VCC), 음극(Cathode)를 -극(GND)에 연결해야 불이 켜집니다. 반대 방향으로 연결하면 불이 켜지지 않습니다.





본 교재에는 다리가 4개인 LED가 들어 있습니다. 이를 RGB LED라고 부르는데, 적색, 녹색, 청색의 3 가지 색의 LED를 하나로 결합한 전자 부품입니다. 이에 대해서는 뒷 부분에서 좀 더 자세히 설명합니다.

Project 2

S.O.S 조난신호 발신기

02

Arduino Sketch

프로젝트-1에서 사용한 회로를 이용해서 모스부호 송신기를 만들어 봅시다.

모스코드는 소리나 빛 등의 길고 짧은 신호를 이용해서 문자정보를 송수신할 수 있도록 만든 통신부호 입니다.

LED의 느린 깜빡임으로 모스부호의 장점을 표시하고, 빠른 깜빡임으로 단점을 표시하면 LED로 모스부호 알파벳을 모두 표현할 수 있습니다.

예를 들면, 모스부호 S는 연속된 단점 3개로 표시하고, 모스부호 O는 연속된 장점 3개로 표시하는데, LED를 빠르게 3번 깜빡이면 S가되고, 느리게 3번 깜빡이면 O가 됩니다.

프로그램 예제2-1:

```
int ledPin = 10;
void setup() {
        pinMode(ledPin, OUTPUT);
}
void loop() {
```

```
// 3 단점 으로 "S" 표시
        digitalWrite(ledPin,HIGH);
        delay(150);
        digitalWrite(ledPin,LOW);
        delay(100);

        digitalWrite(ledPin,HIGH);
        delay(150);
        digitalWrite(ledPin,LOW);
        delay(100);

        digitalWrite(ledPin,HIGH);
        delay(150);
        digitalWrite(ledPin,LOW);
        delay(100);

        delay(100);    //문자마다 0.1초 간격
```

```
//3 장점으로 "O"를 표시
    digitalWrite(ledPin,HIGH);
    delay(400);
    digitalWrite(ledPin,LOW);
    delay(100);

    digitalWrite(ledPin,HIGH);
    delay(400);
    digitalWrite(ledPin,LOW);
    delay(100);

    digitalWrite(ledPin,HIGH);
    delay(400);
    digitalWrite(ledPin,LOW);
    delay(100);

    delay(100);    //문자마다 0.1초 간격
```

```
//3 3 단점을 으로 다시 "S" 표시
    digitalWrite(ledPin,HIGH);
    delay(150);
    digitalWrite(ledPin,LOW);
    delay(100);

    digitalWrite(ledPin,HIGH);
    delay(150);
    digitalWrite(ledPin,LOW);
    delay(100);

    digitalWrite(ledPin,HIGH);
    delay(150);
    digitalWrite(ledPin,LOW);
    delay(100);

    delay(5000);    // 5초후 다시 반복
}
```

Arduino Sketch

똑같은 동작이 여러 번 반복되는 프로그램을 작성하면, 프로그램의 길이가 길어집니다. 프로그램을 짧게 하는 방법은 없을까요? 아래 예제 프로그램 2-2를 살펴보면서 함께 생각해 봅시다.

프로그램 예제2-2:

```
//예제2-2 S.O.S 조난신호 발신기
int ledPin = 10;
void setup() {
        pinMode(ledPin, OUTPUT);
}
void loop() {
         // 단점 3회 반복으로 "S" 표시
         for(int x=0;x<3;x++){
             digitalWrite(ledPin,HIGH);          //LED 켬
             delay(150);                         //0.15초 대기
             digitalWrite(ledPin,LOW);           //LED 끔
             delay(100);                         //0.1초 대기
         }

 //문자마다 0.1초 간격
 delay(100);

//장점 3회 반복으로 "O" 표시
for(int x=0;x<3;x++){
     digitalWrite(ledPin,HIGH);                  //LED 켬
     delay(400);                                 //0.4초 대기
     digitalWrite(ledPin,LOW);                   //LED 끔
     delay(100);                                 //0.1초 대기
}

//문자마다 0.1초 간격
delay(100);
```

```
        // 다시 단점 3회 반복으로 "S" 표시
        for(int x=0;x<3;x++){
                digitalWrite(ledPin,HIGH);          //LED 켬
                delay(150);                         //0.15초 대기
                digitalWrite(ledPin,LOW);           //LED 끔
                delay(100);                         //0.1초 대기
         }

        // 5 초 대기후 처음부터 다시 시작
        delay(5000);
}
```

위 예제프로그램을 업로드하면, LED가 깜빡이면서 SOS 조난 신호를 5초마다 반복 하는 것을 확인할 수 있습니다. 이 프로그램을 잘 응용하면, 비상등은 물론, 방향 표시등 또는 각종 신호등을 만들 수도 있습니다.

CODE 해설

예제 2-1과 2-2의 첫 부분은 동일합니다. 이 부분은 변수의 초기값을 설정하고, 디지털 10번핀을 출력모드로 설정하는 부분입니다. loop()함수를 살펴보면 지난 Project에서 LED를 켜고 끄는 프로그램과 거의 비슷한 것을 알 수 있습니다. 다른 점이 있다면, 이번 예제는 loop()함수가 독립적인 3개의 블록으로 구성되어 있다는 점입니다.

첫 번째 블록은
3 단점을 송신하는 부분입니다.

```
for(int x=0;x<3;x++){
        digitalWrite(led-            //LED 켬
Pin,HIGH);                           //0.15초 대기
        delay(150);                  //LED 끔
        digitalWrite(ledPin,LOW);    //0.1초 대기
        delay(100);
```

for 반복문
"for 반복문" 은 ( ) 속에 정의한 조건에 따라서, { } 속에 정의된 동작을 반복할 때 사용하는 명령문입니다. ( ) 속에 반복 실행할 조건을 설정할 때는 보통 숫자를 증감시키는 변수를 사용하는데, 이 변수 값의 증가폭과 반복을 멈추는 조건을 함께 정의 합니다.

"for 반복문" 의 형식은 다음과 같습니다.

```
for(변수 초기값; 조건; 변수증감값)
{   조건이 참일 때 반복 실행할 내용
}
```

for 반복문은 프로그램에서 일정 동작을 반복하도록 하는 곳이라면 어디든 사용이 가능하지만, 특히, 배열변수와 함께 사용 하면서 연속된 데이터를 차례대로 읽거나 쓸 때 또는 여러 개의 디지털 핀을 차례로 동작시키는 동작을 반복할 때 사용하면 편리합니다.

아래 for반복문이 어떤 과정을 거쳐서
실행되는지 자세히 살펴 봅시다.

```
for(int x=0;x<3;x++){
        ......
}
```

Step 1: 카운터용 변수 x의 초기값을 0으로 설정한다.
Step 2: x가 3보다 작은 지를 판단한다. x=0이므로 참
Step 3: Step2가 참이므로{ }안에 정의된 명령을 실행한다.
Step 4: 변수 x의 값을 1증가시켜 1이 된다.
Step 5: Step2 부터 다시 반복한다, x=1 이므로 3보다 작으므로 참
Step 6: { }안에 정의된 명령을 실행한다.
Step 7: 변수 x의 값을 1증가 시켜 2가 된다.
Step 8: x가 3보다 작은 지를 판단한다.x=2 이므로 참
Step 9: { }안에 정의된 명령을 실행한다.
Step10: 변수 x의 값을 1증가시켜 3이 된다.
Step11: Step2 부터 다시 반복한다, x=3 이므로 3보다 작지 않으므로 거짓 조건이 참이 아니므로 for문을 벗어난다.
Step10: for 반복문 다음 부분을 실행한다.

x<3 으로 조건을 설정했기 때문에 반복횟수는 3 회입니다. 즉, x값이 0부터 2일때까지 3번 반복합니다.
아래 for반목문은 { }을 몇 실행할까요?
for(int x=0;x<100;x++){ ..... }

조건문에 사용되는 비교 연산자

"<" 처럼 2개의 값을 비교하는 부호를 비교연산자 또는 논리 연산자라고 합니다.
아래는 프로그램을 작성할 때 자주 사용하는 비교연산자들 입니다.

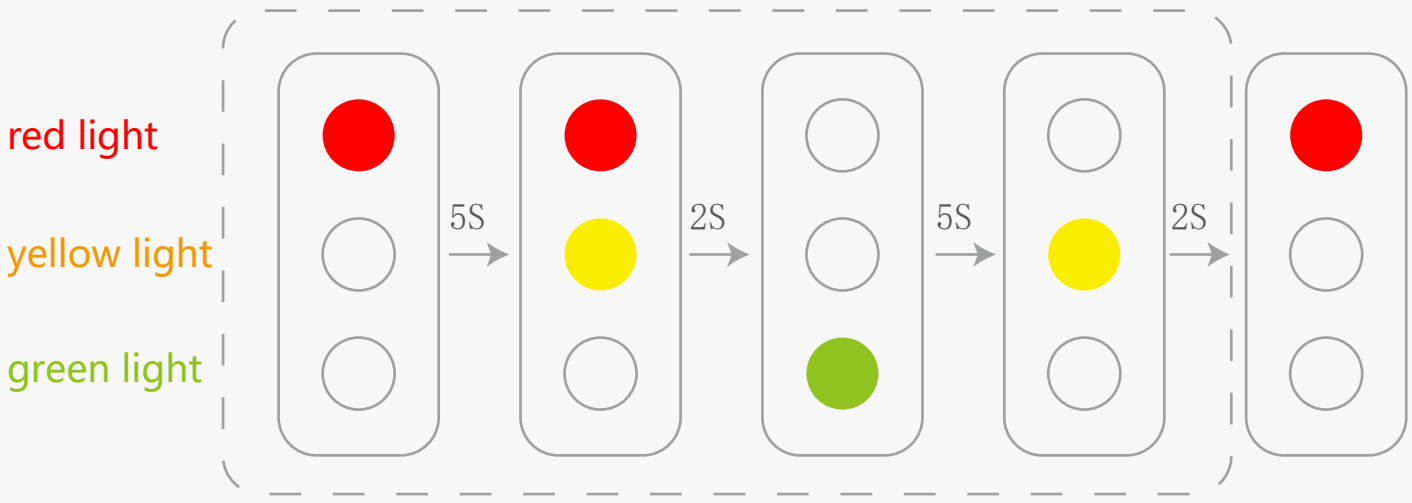
- == (같음)
- != (같지 않음)
- < (보다 작음)
- > (보다 작음)
- <= (보다 작거나 같음)
- >= (보다 크거나 같음)

특히, 비교연산자에 "==" 를 "=" 과 혼동하지 않도록 주의해야 합니다. (예를들면, if (x = 10) 처럼 사용하면 안 됩니다.).
x = 10; 에 사용된 "=" 는 10을 변수 x에 저장하라는 뜻의 예약어 입니다.
If (x == 10) 에서 사용된 "==" 는 x가 10과 같은 지를 비교하는 논리 연산자 입니다.
또한 <= 나 >= 에서 두 문자사이에 공백이 있으면 안된다는 점도 주의해야 합니다 .

예제 프로그램에는 3개의 for 반복문이 있습니다.
첫 번째 for 반복문은 LED를 3번 짧게 깜빡여서(단점) S를 표시합니다.
두 번째 for 반복문은 LED를 3번 길게 켜는 것(장점)을 반복해서 O를 표시합니다.
세 번째 for 반복 문은 첫 번째와 동일합니다.
지역변수와 전역변수
또 한가지 분명히 알아야 할 것은 조건문에 사용한 변수 x특성에 주의해야 합니다. 여기에 사용한 변수 x는 제한된 범위에서만 할 수 있습니다. 변수 x는 for 반복문 안에서 따로 선언한 변수이기 때문에 for 반복문 안에서만 사용할 수 있는 지역변수 입니다.
만일, 변수 x를 전역변수로 사용하려면, 프로그램의 첫 부분에서 선언 해야 합니다.
마찬가지로,setup()과 loop()함수 또는 다른 사용자 함수 안에서 선언한 함수는 그 함수안에서만 사용할 수 있는 지역변수 입니다.

연습문제

3개의 디지털 핀에 LED 3개를 연결하고 아래 그림과 같이 작동하는 교통신호등 프로그램을 작성해 봅시다.



Project 3

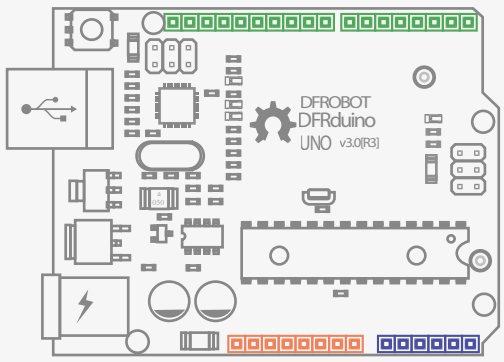
반응형 신호등 만들기

03

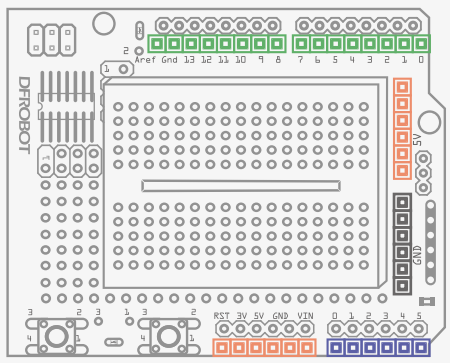
반응형 신호등 만들기

이번 프로젝트에서는 처음으로 상호작용이 가능한 작품을 만듭니다.
즉, 버튼을 누르면, 보행자가 길을 건널 수 있게 바뀌는 반응형 신호등을 제작합니다.

필요한 부품



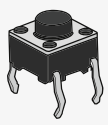
DFduino UNO R3 x1



Prototype Shield x1

Jumper Cables M/M x13

Resistor 220R x6



Pushbutton x1

5MM LED x2

5MM LED x2

5MM LED x1

*왜, LED는 5개인데 저항은 6개일까요?
나머지 저항 1개는 누름 스위치에 연결하는 풀다운 저항입니다.

회로구성

아래 그림처럼 브레드보드 쉴드 위에 실험할 회로를 꾸며보세요.
점퍼선의 색깔은 아래 그림과 다른 것을 사용해도 상관 없습니다.
회로구성을 마쳤으면, 키트에 함께 공급된 USB케이블을 아두이노 보드와 PC를 연결합니다. USB케이블은 5V의 직류전기를 아두이노에 안정적으로 공급해 줍니다.

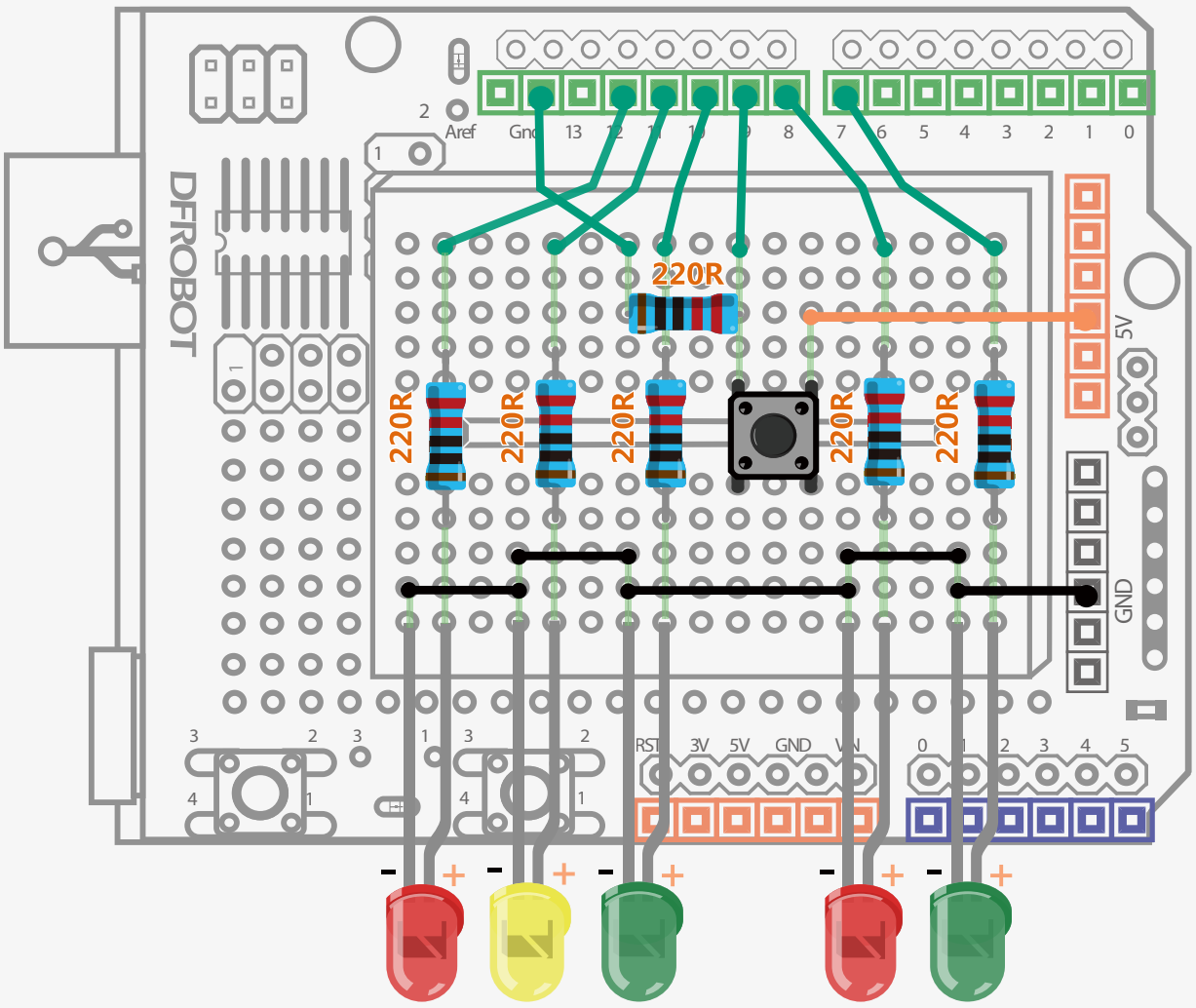


그림 3-1 반응형 신호등 회로도

Arduino Sketch

프로그램 예제 3-1

```
//예제 3-1 반응형 신호등 만들기
int carRed = 12;          //차량 적색
int carYellow = 11;  //차량 황색
int carGreen = 10;  //차량 녹색
int button = 9;            //누름 스위치
int pedRed = 8;          //보행자 적색
int pedGreen = 7;
int crossTime = 5000;                        //도로 횡단시간
unsigned long changeTime;                //신호등 함수 종료 시각

void setup() {
          //LED핀 출력 설정
           pinMode(carRed, OUTPUT);
           pinMode(carYellow, OUTPUT);
           pinMode(carGreen, OUTPUT);
           pinMode(pedRed, OUTPUT);
           pinMode(pedGreen, OUTPUT);
           pinMode(button, INPUT);        //스위치핀 입력 설정
           digitalWrite(carGreen, HIGH);   //차량 녹색 켬
           digitalWrite(pedRed, LOW);    //보행자 적색 켬
}

void loop() {
           int state = digitalRead(button);
           // 신호등이 변경되고 5초 이상 경과후 스위치를 눌렀는지 테스트
           if(state == HIGH && (millis() - changeTime)> 5000){
          //참이면 신호등 변경
          changeLights();
          }
}
```

```
void changeLights() {
        digitalWrite(carGreen, LOW);        //차량 녹색 끔
        digitalWrite(carYellow, HIGH);      //차량 황색 켬

        delay(2000);                                //2초 대기

        digitalWrite(carYellow, LOW);       // 차량 황색 끔
        digitalWrite(carRed, HIGH);          //차량 적색 켬
        delay(1000);                                // 1초 대기

        digitalWrite(pedRed, LOW);         //보행자 적색 끔
        digitalWrite(pedGreen, HIGH);     //보행자 녹색 켬

        delay(crossTime);                        // 횡단시간 대기

       //보행자 녹색등 10회 점멸
        for (int x=0; x<10; x++) {
                digitalWrite(pedGreen, HIGH);
                delay(250);
                digitalWrite(pedGreen, LOW);
                delay(250);
        }
        digitalWrite(pedRed, HIGH);          //보행자 적색 켬

        delay(500);

        digitalWrite(carRed, LOW);           //차량 적색 끔
        digitalWrite(carYellow, HIGH);       //차량 황색 켬
        delay(1000);
        digitalWrite(carYellow, LOW);       //차량 황색 끔
        digitalWrite(carGreen, HIGH);       //차량 녹색 켬

        changeTime = millis();                  // 실행 완료시각 저장
        //메인 프로그램 즉, loop()함수로 돌아 감
}
```

프로그램을 실행하고 LED가 어떻게 동작하는지 살펴봅시다.
먼저, 차량용 녹색등이 켜지고, 보행자용 적색등이 켜집니다. 즉, 자동차가 출발할 수 있는 상태가 됩니다.

이 상태에서, 누름 스위치를 누르면, 자동차용 신호등이 녹색에서 적색으로 바뀌고, 보행자용 신호등은 적색에서 녹색으로 바뀌어, 보행자가 길을 건너는 동안 그 상태가 계속 유지 됩니다.

횡단시간이 지나면, 보행자용 녹색등이 깜빡거리면서 보행자에게 주의 신호를 보냅니다. 깜빡임이 끝나면 신호등은 다시 처음 상태로 되돌아가서 차량 신호등은 녹색, 보행자 신호등은 적색이 켜집니다.

신호등 조작이 종료된 지 5초이상이 지나야 다시 스위치를 눌러 새롭게 신호등을 조작할 수 있습니다.

위 프로그램이 좀 복잡해 보이지만, 사실은 이해하기 어렵지 않습니다.

만일,이 프로그램이 어렵다고 느껴진다면 프로젝트2의 끝부분의 과제에 있는 그림을 다시 살펴보면 쉽게 이해할 수 있을 겁니다.

Code 해설 (함수 위치 잘 확인할 것)

양수만 사용할 경우의 변수선언

```
unsigned long changeTime;
```

앞에서 사용한 int형변수는 -32,768 부터 32,767 까지의 정수를 저장할 수 있지만, long형 변수는 -2,147,483,648 부터 2,147,483,647 까지의 정수를 저장 할 수 있습니다. 변수에 음수를 저장할 필요가 없을 때는 변수선언문의 맨 앞부분에 unsigned 라고 선언합니다. 즉, unsigned long 으로 선언하면 0부터 4,294,967,295까지의 큰 수를 저장할 수 있습니다.

만일, int형변수를 이용해서 ms단위로 시간을 측정한다면, 32초(32767ms)를 초과하는 시간은 측정할 수 없습니다. unsigned long변수를 사용할 경우 4,294,967,295ms(4,294,967초)까지도 측정할 수 있는데, 이는 약 49일에 해당하는 긴 시간입니다.

예제 프로그램의 setup( )함수의 일부인 아래 코드를 좀 더 살펴봅시다.

```
pinMode(button, INPUT);
```

Project 1에서 pinMode() 함수를 사용해 봤기 때문에 이미 익숙해 졌을 겁니다. 하지만, 버튼 스위치를 연결한 디지털 핀을 INPUT 모드로 사용하는 점이 이전 프로젝트와 다른 점 입니다.
꼭 기억합시다 버튼스위치는 INPUT으로만 사용합니다.
setup() 함수에서는 아래와 같이 신호등의 초기동작을 설정합니다.

```
digitalWrite(carGreen, HIGH); //차량 녹색등 켬
digitalWrite(pedRed, HIGH); //보행자 적색등 켬
```

변수에는 얼마나 큰 수를 저장할 수 있을까요

변수의 종류마다 저장할 수 있는 수의 범위가 다릅니다. 왜, 변수의 종류에 따라서 저장할 수 있는 데이터의 크기가 달라질까요? 그 이유는 아래표에 정리 되어 있는 것처럼, 변수마다 데이터를 저장하는 메모리의 크기가 다르기 때문입니다. 예를 들면, 2바이트의 저장공간을 사용하는 int형 변수보다는 4바이트의 저장공간을 사용하는 long형 변수가 훨씬 더 큰 수를 저장할 수 있습니다.

최대로 선언할 수 있는 변수의 개수는 아두이노 보드의 메모리 크기에 따라 달라집니다. Arduino UNO의 메인칩인 Atmega328의 메모리 용량은 32kb 입니다. 따라서, 변수로 사용되는 공간과 프로그램이 사용하는 메모리 공간의 합이 32kb를 초과할 수 없습니다.
자주 사용되는 변수:

Data Type	RAM	Range
boolean	1 byte	0 ~ 1 (True or False)
char	1 byte	-128 ~ 127
unsigned char	1 byte	0~255
int	2 byte	-32768 ~ 32768
unsigned int	2 byte	0 ~ 65535
long	4 byte	-2147483648 ~ 2147483647
unsigned long	4 byte	0 ~ 4294967295
float	4 byte	-3.4028235E38 ~ 3.4028235E38
double	4 byte	-3.4028235E38 ~ 3.4028235E38

digitalRead(pin) 함수

이 함수는 디지털 핀의 상태가 HIGH인지 LOW인지를 확인할 때 사용합니다. 파라미터 pin은 상태를 읽을 디지털 핀의 번호를 지정하면 되는데, 숫자나 변수를 사용할 수 있습니다. 그럼, 아래 코드의 의미를 살펴봅시다.

```
int state=
digitalRead(button);
```

이 코드는 디지털핀(button)의 상태를 int형 변수 state에 저장하라는 명령문입니다.
이 프로젝트에서 사용하는 누름스위치는 스위치를 누르지 않았을 때는 LOW(0V), 눌렀을 때는 HIGH(5V)가 됩니다. 따라서, 변수 state 가 HIGH인 경우 누름 스위치가 눌러진 상태이고, LOW 이면 누름 스위치가 눌러지지 않았음을 의미합니다.
누름 스위치가 누름 상태인지 아닌지를 확인하려 면 if 문을 사용해서 변수 state 에 저장된 값을 확인하면 됩니다.

if ( ) 명령문의 조건문
아래 코드의 의미를 생각해 봅시다.

```
if(state == HIGH) { 실행 }
```

If 명령은 ( )속에 설정된 조건을 만족할 경우는 명령을 수행하고, 만족하지 않으면 if문 밖으로 건너뛰도록 합니다.
바꿔 말하면, if문의 조건이 참이면 명령을 실행하고, 거짓이면 그 명령을 실행하지 않고 건너 뜁니다.
따라서, 위 코드는 스위치가 눌러진 상태일 경우만 { }안의 코드를 실행하고, 스위치가 눌러지지 않았을 때는 { }안을 실행하지 말고, 다음으로 건너뛰라는 의미입니다.

계속해서 다음 if 조건문을 살펴봅시다.

```
if ( state == HIGH &&
( millis() - changeTime ) >
5000 ) { changeLights(); }
```

위 명령문은 동시에 2가지 조건을 만족하는지를 판단합니다.
즉, 첫번째 조건은 변수 state가 HIGH인지 (버튼스위치가 눌러졌는지) 를 확인하고, 두 번째 조건은 시간이 5초이상 경과했는지를 확인합니다.
if문에 사용된 "&&" 는 두개의 조건을 연결하는 논리연산자 인데, 두 가지 조건을 모두 만족할 경우만 조건문이 참이 됩니다.
millis()함수는 아두이노 보드가 스위치를 켠 순간부터 시간이 얼마나 경과 했는지를 숫자로 Return 해주는 함수 입니다. Millis()함수 리턴 값의 단위는 ms 이며, long형 변수에 저장되므로 약 49일이 경과되면, millis() 함수의 반환 값이 다시 0부터 시작됩니다.

if ( ) 명령의 실행문

changeLights();함수

이 함수는 loop() 함수 밖에서 정의된 겁니다. 이 함수를 실행하려면 함수 이름만 써주면 됩니다. 이 함수는 파라미터를 설정할 필요도 없고, 리턴 값을 받지도 않습니다. 이런 함수를 void 함수라고 하는데, 함수를 정의할 때는 아래처럼 맨 앞에 void라고 표시합니다.

함수 선언방법
void changeLight() { }

프로그램에서 함수이름을 써주면 함수가 실행되며, 함수실행이 끝나면 다시 함수를 호출한 이전 프로그램의 다음 위치로 돌아가서 계속 프로그램을 수행하게 됩니다.
함수를 호출할 때는 함수 이름 뒤에 괄호도 꼭 함께 써야 합니다.
괄호가 없으면 아두이노는 함수를 인식하지 못합니다.

함수 호출법(실행방법)
changeLight();

논리 연사자

기본 논리 연산자

- && ——논리곱 (두 조건을 모두 만족할 때만 참)
- || —— 논리합(두 조건중 적어도 하나를 만족하면 참))
- ! —— 부정(조건을 만족하지 않을 때 참)

하드웨어 탐구

누름 스위치

우리가 사용하는 누름스위치는 핀이 4개입니다. 하지만 4개중 2개의 핀은 내부적으로 서로 연결되어 있습니다. 즉, 1번핀과 4번핀이 서로 연결되어 있고, 2번핀과 3번 핀도 마찬 가지 입니다. 따라서, 스위치를 누르면, 실제로는 2개의 핀을 연결시키는 것과 같은 동작을 합니다. 이번 프로젝트에서는 4핀인 스위치를 사용하지만, 만일 2핀 스위치를 사용해도 똑 같은 동작을 합니다.

누름스위치는 전기회로를 열었다 닫았다 하는 역할을 합니다. 이 프로젝트에서 누름 스위치를 누르면, D9핀이 HIGH(켜짐) 상태가 되고, 누르지 않았을 때는 LOW(꺼짐) 상태가 됩니다.

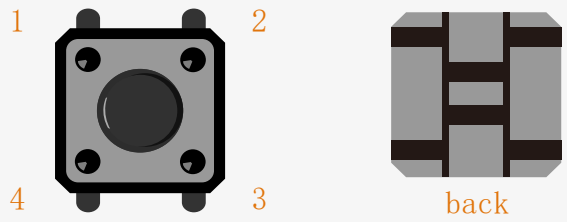


Fig 3-2 누름스위치의 구조 (앞,뒤)

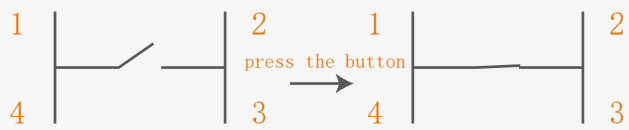


그림 3-3 누름스위치의 배선도

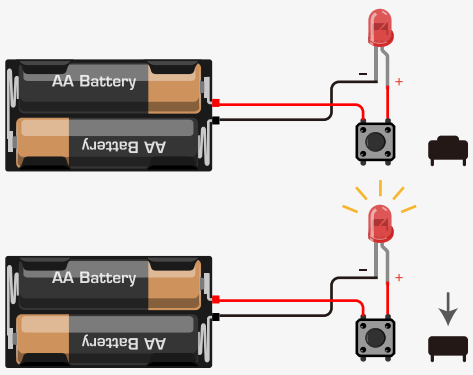
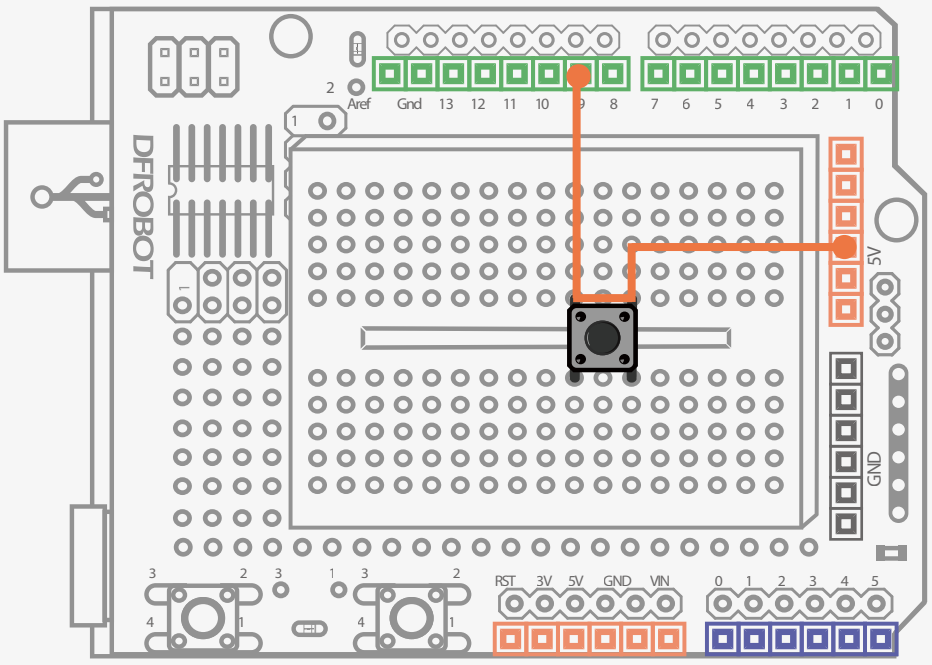


그림 3-4 누름스위치의 동작

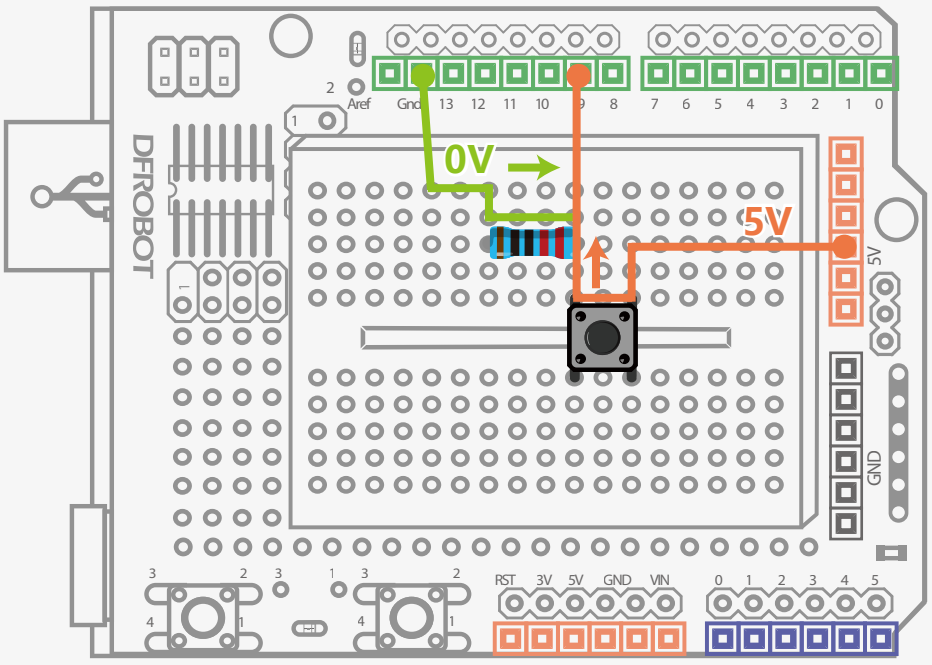
풀다운 저항이란 ?

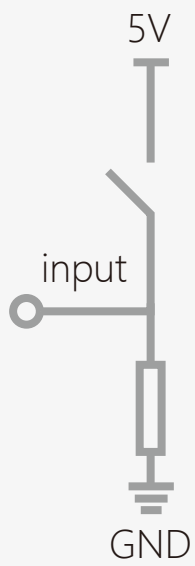
풀다운 저항은 아두이노 보드의 입력핀에 아무것도 연결하지 않았을 때도 입력 핀이 항상 LOW 상태가 되도록 만드는 저항입니다. 풀 다운이라는 명칭이 의미하듯이 입력 핀에 연결된 부품이 아무 동작을 하지 않을 때, 입력 핀의 전압을 GND와 동일하게, 즉 LOW(0V)로 만드는 역할을 합니다.

디지털 핀의 상태가 LOW상태도 HIGH상태도 아닌 상태를 하이 임피던스 상태라고 합니다. 디지털 핀의 상태는 항상 HIGH 아니면 LOW상태를 유지해야 프로그램이 논리적인 동작을 확실히 수행할 수 있습니다. 디지털핀이 하이 임피던스 상태일 경우에는 연결되지 않은 상태와 동일하므로, 프로그램이 논리적 동작을 제대로 수행하지 못하는 경우도 생길 수 있습니다. 이를 피하기 위해서 풀다운 저항을 사용합니다.



풀다운저항 미사용
(하이임피던스)

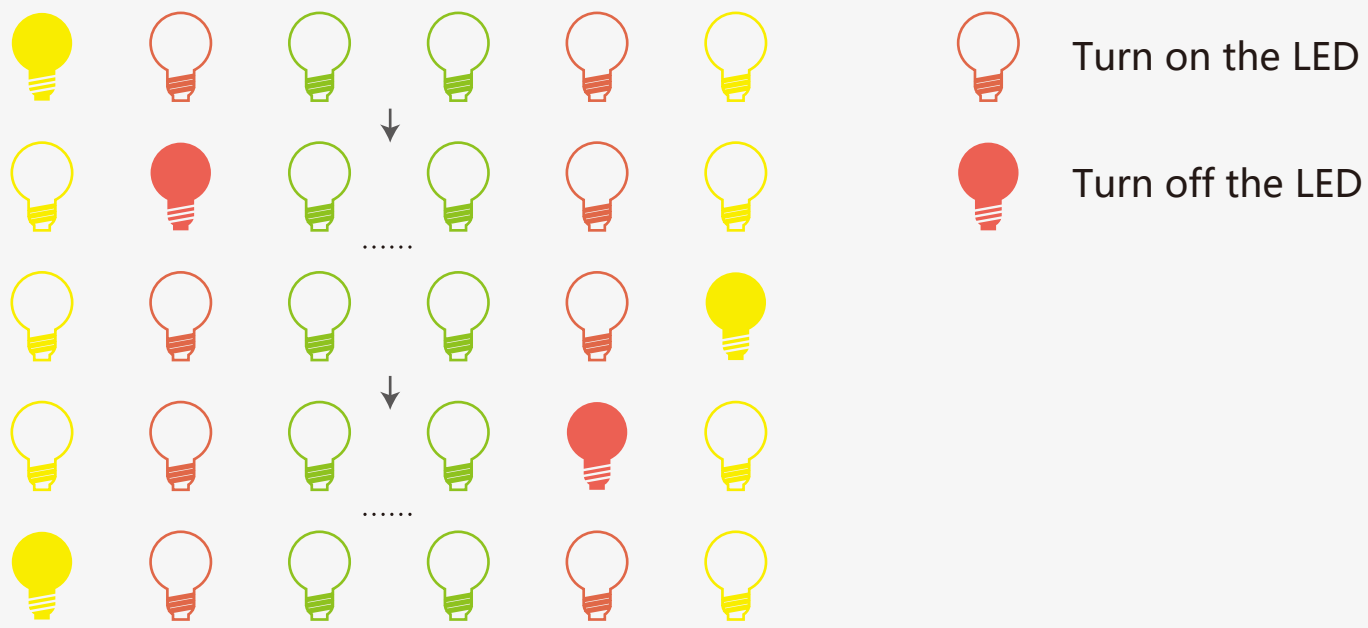




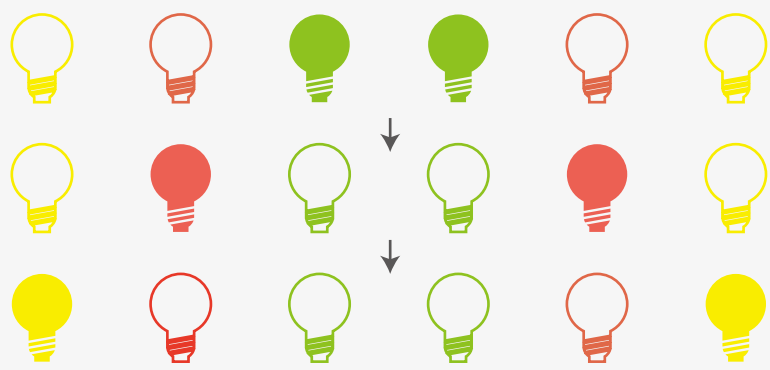
풀다운 저항 사용시
(LOW상태)

Homework After Class

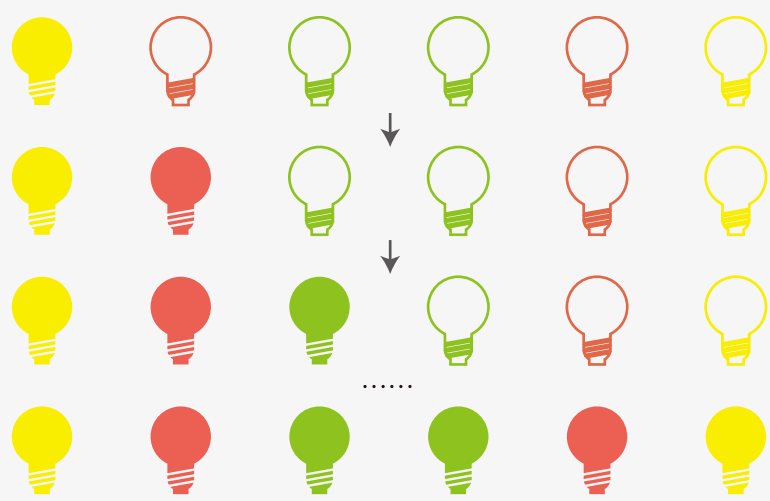
1.6개의 LED를 아두이노에 연결하고, 임의에 LED를 색깔과 순서와 관계없이 마음대로 동작시켜 보시오.



2.가운데 2개의 LED를 먼저 점등하고, 바깥쪽 이동하면서 켜지도록 프로그램을 작성하시오.



3.LED가 왼쪽부터 오른쪽으로 차례대로 켜지도록 프로그램을 작성해 보시오.



Project 4

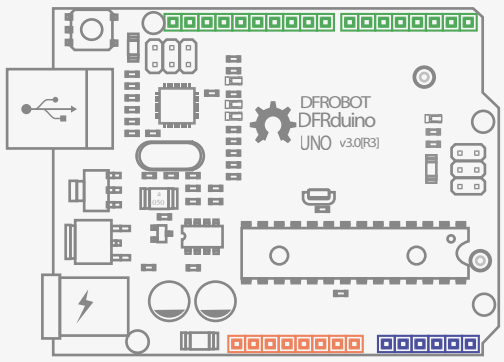
숨쉬는 LED

04

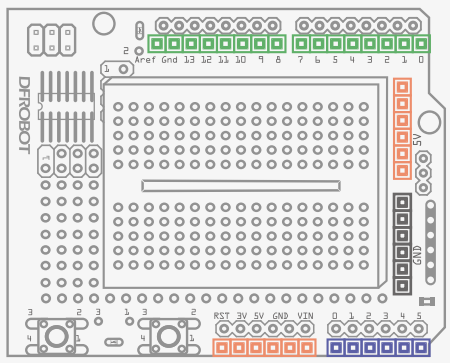
숨쉬는 LED

앞부분에서, LED를 켜고 끄는 방법을 공부했습니다. 이번 프로젝트에서는 LED의 밝기를 조절하는 방법을 알아봅니다. 아두이노 보드 위에 표시된 디지털 핀 번호를 살펴보면, 어떤 핀들은 번호 뒤에 "~" 표시가 있는 것을 볼 수 있습니다. 이 표시가 있는 디지털 핀들은 PWM 신호를 사용할 수 있는 핀들 입니다. 이번 프로젝트에서는 PWM핀을 이용해서 RGB LED의 밝기가 서서히 밝아지고 서서히 어두워지는 프로그램을 학습합니다.

필요한 부품



DFduino UNO R3 x1



Prototype Shield x1

Jumper Cables M/M x2

Resistor 220R x1

5MM LED x1

회로구성

회로구성은 아래 그림과 같습니다. 프로젝트1의 회로구성과 동일합니다.

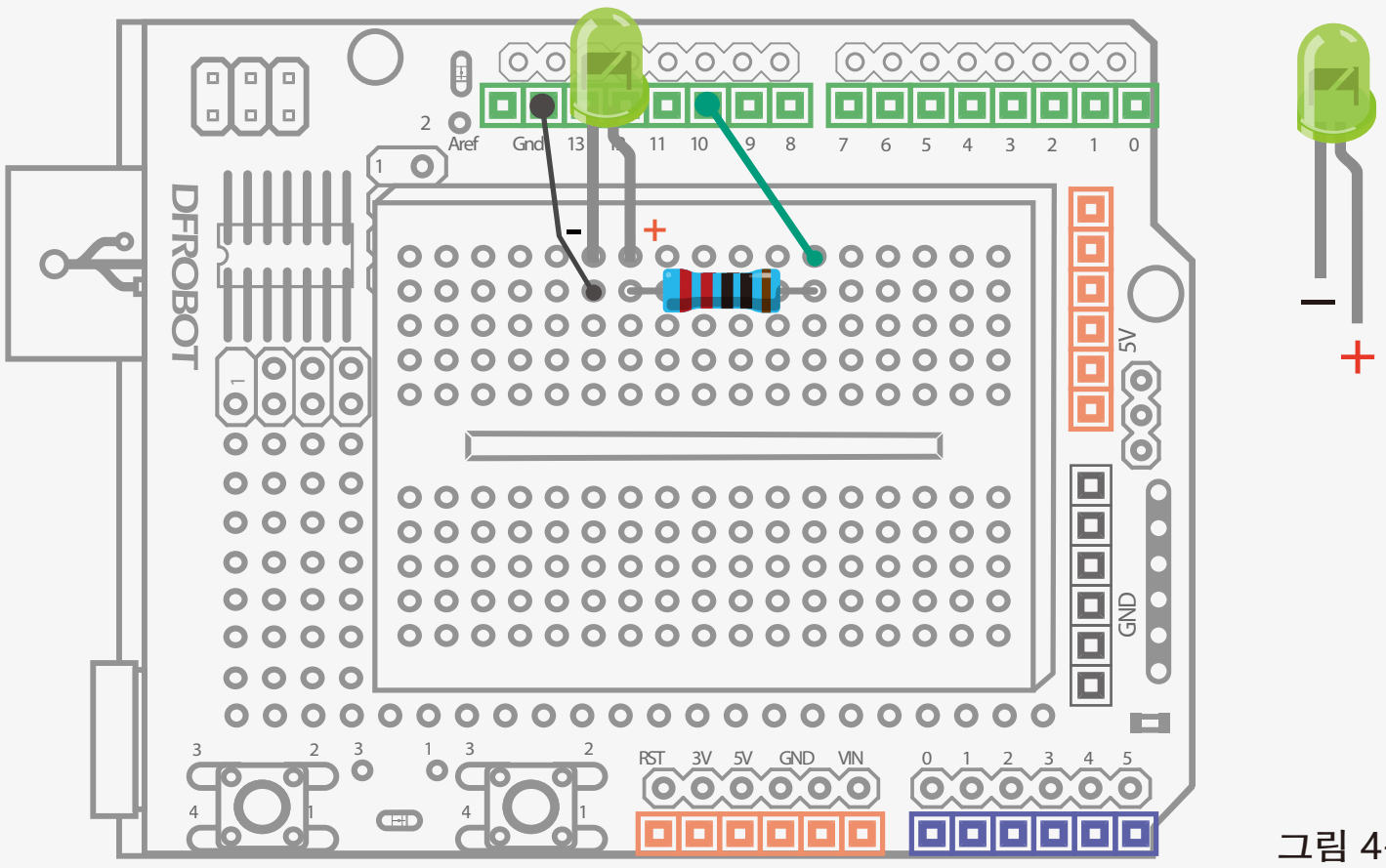


그림 4-1 숨쉬는 LED 회로도

Arduino Sketch

프로그램 예제 4-1 :

```
// 예제 4 숨쉬는 LED
int ledPin = 10;

void setup() {
      pinMode(ledPin,OUTPUT);
}

void loop(){
      fadeOn(1000,5);
      fadeOff(1000,5);
}

void fadeOn(unsigned int time,int increament){
     for (byte value = 0 ; value < 255; value+=increament){
           analogWrite(ledPin, value);
           delay(time/(255/5));
           }
}

void fadeOff(unsigned int time,int decreament){
     for (byte value = 255; value >0; value-=decreament){
           analogWrite(ledPin, value);
           delay(time/(255/5));
           }
}
```

디지털 핀에 연결된 LED가
반복적으로밝아졌다 어두워 졌다
하는 것을 관찰할 수 있습니다.

Code 해설

이전의 프로젝트 학습을 통해서, 변수를 선언하고 초기값을 저장하는 방법, 입출력 핀을 설정하는 방법, 함수를 정의하고 호출하여 사용하는 방법은 물론, loop()함수를 작성하는 방법에 이미 익숙해 졌을 겁니다.

이번 예제 프로그램의 loop()함수는 다음과 같이 아주 간단한데, 사용자 정의 함수 2개만 함수만 사용하고 있습니다.

```
void loop(){
fadeOn(1000,5);
fadeOff(1000,5);
}
```

그 중 하나인 fadeOn() 함수를 자세히 살펴보면서 이 함수에 사용된 핵심 아이디어를 익혀봅시다.

```
void fadeOn(unsigned int time, int increment)
{
for (byte value = 0 ; value < 255;
value+=increment)

{ analogWrite(ledPin, value);
delay(time/(255/5)); }

}
```

fadeOn() 함수는 2개의 파라미터를 사용합니다. int형 변수 time으로 시간 간격을 설정하며, int형 변수 increment를 사용해서 값이 증가되는 폭을 설정합니다. for 반복문을 사용해서 프로그램을 반복 실행하는데, 반복 실행 조건은 "value < 255" 이고, 변수 value의 값은 매번 반복 될 때마다 변수 increment에 저장된 값 만큼 증가합니다.

함수의 실행문에는 아래와 같은 새 명령어가 사용 되었습니다.

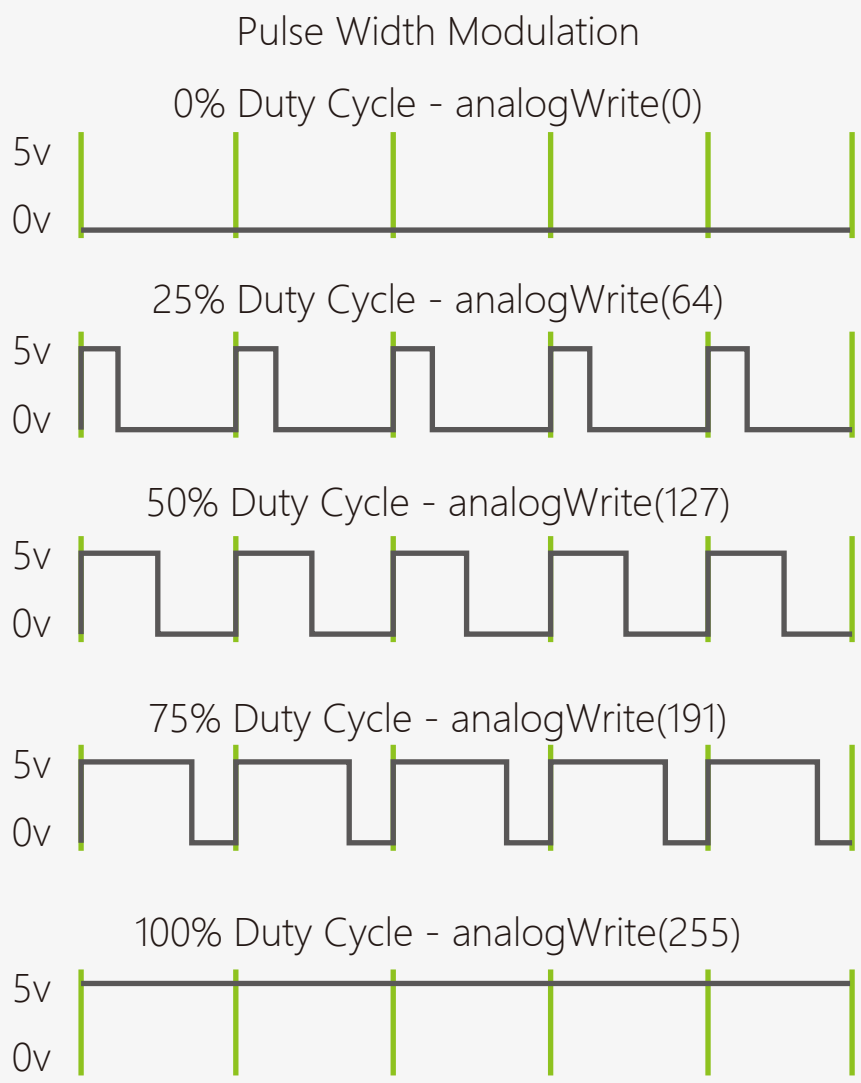
```
analogWrite(ledPin, value)
```

여기서, 어떻게 디지털핀에 아날로그 값을 출력할 수 있는가? 하는 의문이 생길 겁니다. 디지털핀 중에 D3~, D5~, D6~, D9~, D10~, D11~ 처럼 이름 뒤에 "~" 표시를 한 핀을 사용하면 LED에 보내는 전류를 조절할 수 있습니다. 이 기법이 바로 펄스폭 조절법(PWM)입니다.

analogWrite 명령문은 두 개의 파라미터를 사용합니다.
파라미터 ledPin은 PWM신호(아날로그 출력신호)를 출력할 핀입니다. 숫자와 변수, 수식 등을 사용할 수 있습니다.
파라미터 value는 0 부터 255의 값을 지정하면 됩니다. 역시, 숫자, 변수, 수식 등을 사용할 수 있습니다.

PWM 이란 무엇인가?
analogWrite() 함수는 지정된 PWM 핀에 0~255 사이의 아날로그 값을 출력하는 함수입니다 대략 1/500 초를 주기로 PWM pin핀들은 디지털 펄스신호를 출력합니다.
그런데, 이 펄스신호가 on상태를 유지하는 시간과 off 상태를 유지하는 시간의 비율을 조절하면, 디지털 핀으로 0~5V사이의 전압을 출력하는 것과 동일한 효과를 갖게 할 수 있습니다. 펄스신호의 길이를 이용해서 출력을 조절하는 방법이기 때문에 이 기법을 펄스 폭 제어 (PWM, Pulse Width Modulation)라고 부릅니다.

아래 그림을 보면서 PWM을 좀더 자세히 알아봅시다.



녹색선은 펄스신호가 반복되는 주기를 표시한 겁니다. analogWrite() 함수의 파라미터 값에 따라서 펄스신호가 High상태를 유지하는 시간과 Low상태를 유지하는 시간이 달라집니다.
펄스신호의 주기에 대한 High상태의 유지시간의 비율을 듀티 싸이클(duty cycle)이라고 부릅니다. 첫 번째 그림의 듀티 싸이클은 0입니다. 따라서, LED의 밝기가 0이 되고 LED를 끈 상태와 같게 됩니다.
HIGH상태인 시간이 긴 펄스신호(즉 Duty Cycle이 클수록) 일수록 LED의 밝기가 밝아집니다. 듀티 사이클의 최대값은 100%인데 PWM값이 255일때 LED가 제일 밝은 상태가 됩니다. 마찬가지로 PWM값이 0 ~ 255사이로 변하면 LED의 밝기가 점점 밝아지고, 255 ~0 사이값으로 변하면 LED의 밝기가 점점 어두워지게 됩니다.

PWM은 전등의 밝기를 조절하는데 널리 사용 될 뿐만 아니라 DC 모터의 회전속도를 조절하는데도 사용됩니다.
이 프로젝트는 프로젝트1과 동일한 회로를 사용하지만 아두이노 보드를 동작시키는 방법은 전혀 다릅니다. 이처럼, 같은 회로도 프로그램에 따라서 전혀 다른 결과를 가져오게 할 수도 있다는 점이 아두이노의 흥미와 관심을 갖게 하는 요인이며, 우리가 아두이노를 학습하는 이유이기도 합니다.

연습 문제

1. 아두이노 보드에 연결한 LED들의 PWM값을 임의로 조절함으로 마치 불꽃이 활활 타오르는 듯한 효과를 만들어 보세요. 밤에 LED위에 종이를 덮고 사용하면 멋있는 램프가 됩니다.
준비물: RED LED 1개, YELLOW LED 1개, 220Ω 저항 2개
임의의 난수를 발생시키는 함수가 “random()” 함수입니다. 이 함수를 사용하면 LED의 밝기를 임의로 조절할 수 있습니다.
프로그램 작성방법은,
먼저 LED의 밝기를 초기값(임의 값)으로 설정한다음 random(120)+135 같은 함수를 사용해서 무작위로 PWM 값이 바뀌도록 합니다. 이 방법을 사용하면 LED를 작은 불꽃이 움직이는 것처럼 만들 수 있습니다.

Random() 함수에 대한 자세한 설명은 아래 링크의 설명을 참고하세요.
https://www.arduino.cc/en/Reference/HomePage

2. 버튼 스위치 2개를 사용해서 LED 의 밝기를 조절하는 과제에 도전해 봅시다. 스위치를 누르면 LED가 점점 밝아지고, 다른 스위치를 누르면 점점 어두워지도록 하는 프로그램을 작성해보세요. 아래 소개된 웹페이지를 참고하세요.

Reference: http://www.geek-workshop.com/thread-1054-1-1.html

Project 5

칼라 RGB LED

05

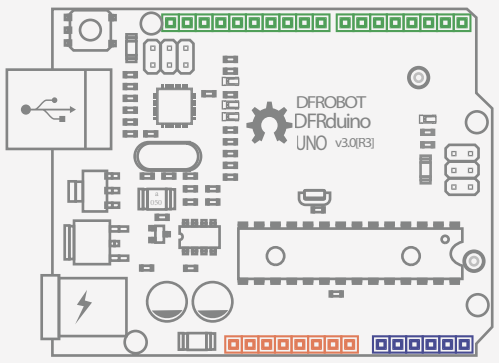
DFROBOT
DRIVE THE FUTURE

칼라 RGB LED

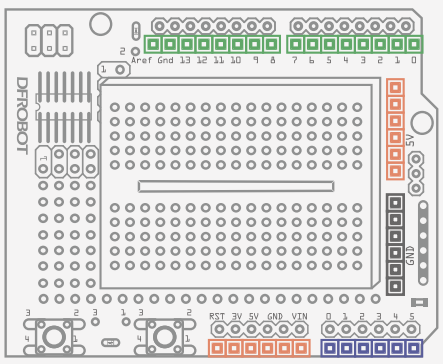
RBG LED라고 불리는 새로운 부품을 사용해서 프로젝트를 진행해 봅시다. 이 부품은 3원색 (빨강,파랑,녹색)의 LED 3개를 결합하여 만든 전자부품인데 각 LED의 밝기를 조절하면 여러가지 색깔을 표시할 수 있습니다. 사실, 컴퓨터 모니터도 수많은 RGB LED를 이용해서 이미지를 표시하는 겁니다.

RGB LED를 이용해서 여러가지 색깔을 만들어 봅시다.

필요한 부품



DFduino UNO R3 x1



Prototype Shield x1

Jumper Cables M/M x4

Resistor 220R x3

5mm RGB LED x1

회로 구성

회로 구성을 시작하기 전에, 먼저 여러분이 갖고 있는 RGB LED가 커먼 캐소드형 인지 커먼애노드형 인지 확인해 보세요. 만일, 확인하는 방법을 모른다면 먼저 이 프로젝트의 마지막 부분을 읽어 보세요. 이 프로젝트에서는 커먼 캐소드 방식의 RGB LED를 사용합니다.

아래 그림을 참고하여 실험할 회로를 구성 하시오.

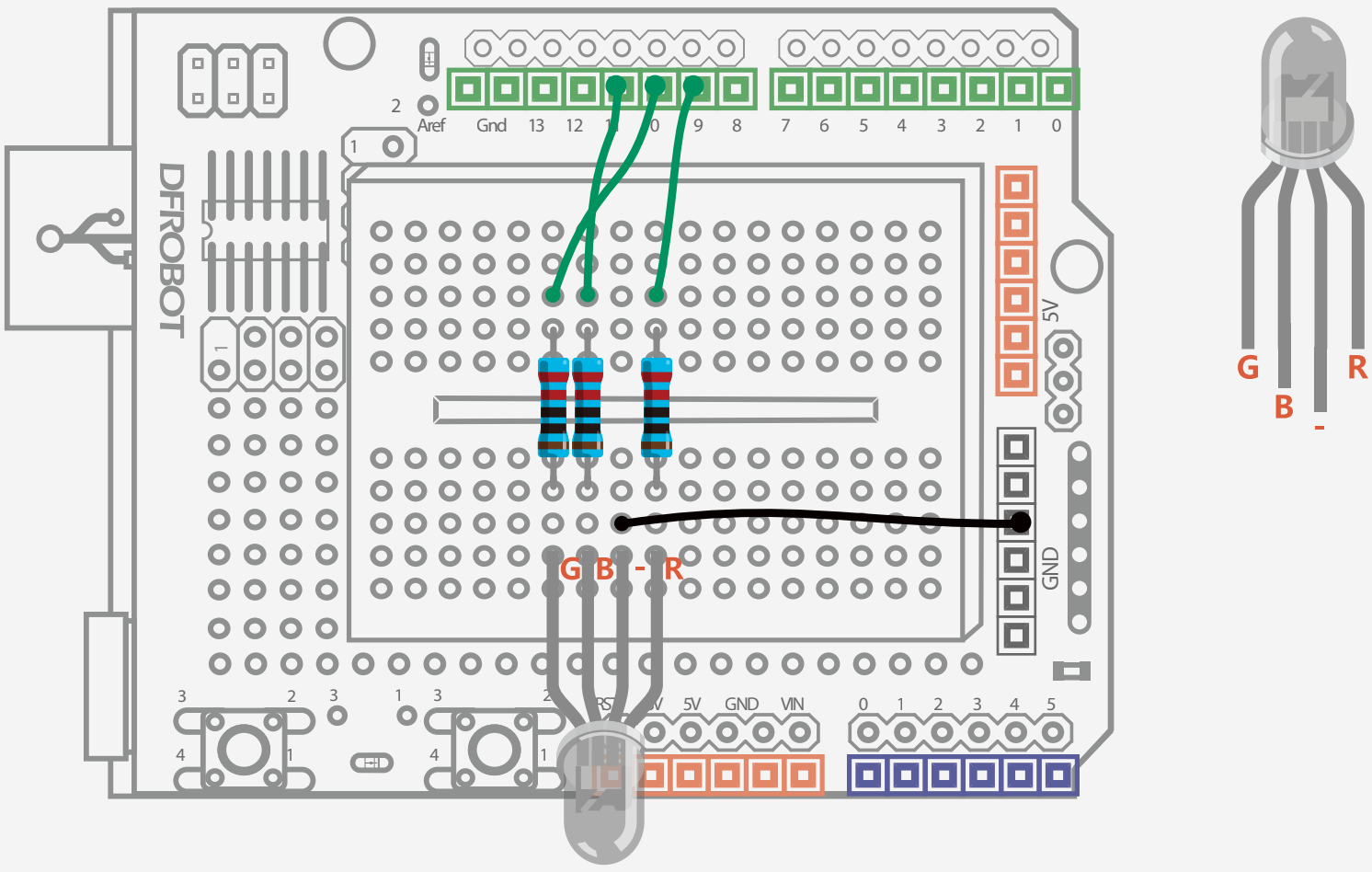


그림 5-1 칼라 RGB LED 회로도

Arduino Sketch

프로그램 예제 5-1:

```
//예제5 칼라 RGB LED
int redPin = 9;
int greenPin = 10;
int bluePin = 11;

void setup(){
    pinMode(redPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(bluePin, OUTPUT);
}

void loop(){
     //R:0-255 G:0-255 B:0-255
     colorRGB(random(0,255),random(0,255),random(0,255));
    delay(1000);
}

void colorRGB(int red, int green, int blue){
        analogWrite(redPin,constrain(red,0,255));
        analogWrite(greenPin,constrain(green,0,255));
        analogWrite(bluePin,constrain(blue,0,255));
}
```

이 프로그램을 실행하면, RGB LED가 임의의 색깔을 내며 1초 간격으로 깜빡이게 됩니다.

Code 해설

먼저 RGB LED를 연결할 3개의 PWM 디지털 핀을 OUTPUT 모드로 설정합니다.

이 프로그램의 핵심은 colorRGB()라는 사용자 함수를 만드는 겁니다. 이 함수는 0~255 사이의 값을 갖는 3개의 파라미터를 사용하여 RGB LED의 색깔을 마음대로 바꿀 수 있습니다.

예를 들면, colorRGB(255,255,255) 는 흰색을, colorRGB(0,0,0) 는 검정색을, colorRGB(255,0,0) 는 빨간색을 표시할 수 있습니다.

먼저 constrain()함수와 random()함수를 알아 봅시다. 지난 프로젝트에서 소개한 사이트를 방문해서 이 함수에 대한 설명을 혼자의 힘으로 읽고 이해 할 수 있는지 알아보세요.

https://www.arduino.cc/en/Reference/HomePage

constrain() 함수의 형식은 아래와 같습니다.

constrain() 함수는 3개의 x, a, b 파라미터가 필요합니다.

x 는 범위를 제한하려는 수이고, a는 제한 범위의 최소값, b는 제한범위의 최대값입니다.

만일 파라미터 x의 값이 최소값 a보다 작으면 constrain()함수는 a를 리턴하고, 최대값 b보다 크면, b를 리턴 합니다.

RGB LED의 색깔을 조절하려면, 빨간, 녹색, 파랑의 PWM 값을 조절하면 되는데, 이 값은 0~255범위를 넘지 말아야 합니다. constrain() 함수를 사용하면 이 값이 지정범위를 초과해서 프로그램의 오류가 발생하는 것을 방지할 수 있습니다.

random()함수는 무작위의 난수(특정한 순서나 규칙을 가지지 않는 연속적인 임의의 수)를 발생하는 함수입니다. 2개의 파라미터를 갖는데, 난수가 발생하는 범위의 최소값과 최대값을 아래와 같이 지정하여 사용합니다.

이 함수의 첫 번째 파라미터는 최소값이고, 두 번째는 최대값입니다. 따라서 0부터 255사이의 난수를 발생시키려면 프로그램에 random(0,255)와 같이 쓰면 됩니다.

하드웨어 탐구

RGB LED

RGB LED는 오른쪽 그림처럼, 3개의 LED를 한 데 모아 만든 전자부품입니다. RGB LED는 핀이 4개인데, 그 중 하나는 공동 전극입니다.

3개의 LED의 캐소드를 공동전극으로 하는 LED를 커먼 캐소드형 LED라고 하고, 공동전극이 애노드인 것을 커먼 애노드 형 LED라고 합니다.

이 프로젝트에서는 커먼캐소드 방식의 RGB LED를 사용합니다. 따라서, 공통 음극은 GND와 연결하고, 다른 3핀은 PWM제어가 가능한 디지털 핀과 연결해야 합니다.

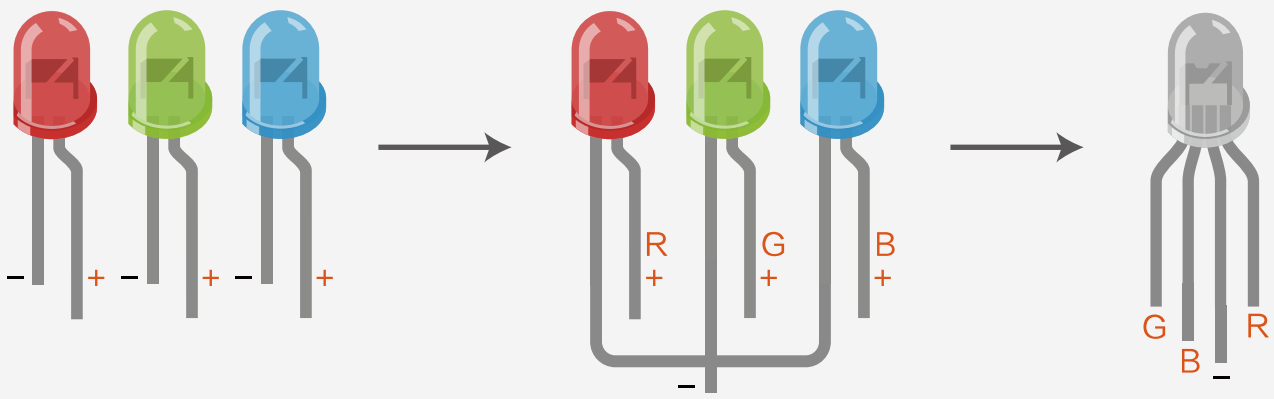


그림 5-2 RGB LED의 구성 (커먼 캐소드 형)

그림 5-3 3원색과 혼합색

RGB LED를 사용하면 16,777,216 가지의 색깔을 만들 수 있습니다. 왜냐하면, RGB LED의 각 LED를 PWM신호로 255단계로 조절할 수 있어서 표현 가능한 최대 색상의 수는 255x255x255가 되기 때문입니다.

Red	Green	Blue	Colour
255	0	0	Red
0	255	0	Green
0	0	255	Blue
255	255	0	Yellow
0	255	255	Blueish Green
255	0	255	Purplish Red
255	255	255	White

표 5-1 3원색 LED의 PWM값과 표현색상 예시

커먼 애노드형 RGB LED와 커먼 캐소드형 RGB LED의 차이점

커먼 애노드 형과 커먼 캐소드 형 RGB LED는 공통 전극의 극이 서로 다르 기 때문에 다음 2가지 점을 주의해서 사용해야 합니다.

(1) 전원 연결방법 : 커먼 애노드형 RGB LED는 공통전극을 5V와 연결해야 하고, 커먼 캐소드형은 GND와 연결해야 합니다. 만일 방향이 바뀌면 LED가 켜지지 않습니다.

(2) PWM 신호의 듀티 사이클이 같아도, 커먼 애노드형과 커먼 캐소드형의 밝기가 다르게 됩니다. 예를들면 커먼 캐소드 형 RGB LED는 듀티 사이클이 클수록 밝은 빛을 내지만, 커먼 애노드 형은 듀티 사이클이 클수록 어두운 빛을 냅니다. 따라서, RGB LED로 여러가지 색상을 만들 때 주의해야 합니다.

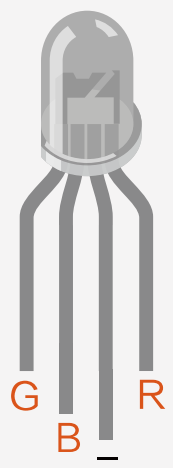


그림 5-4 커먼 캐소드 RGB LED

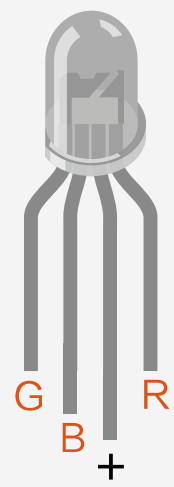


그림 5-5 커먼 애노드 RGB LED

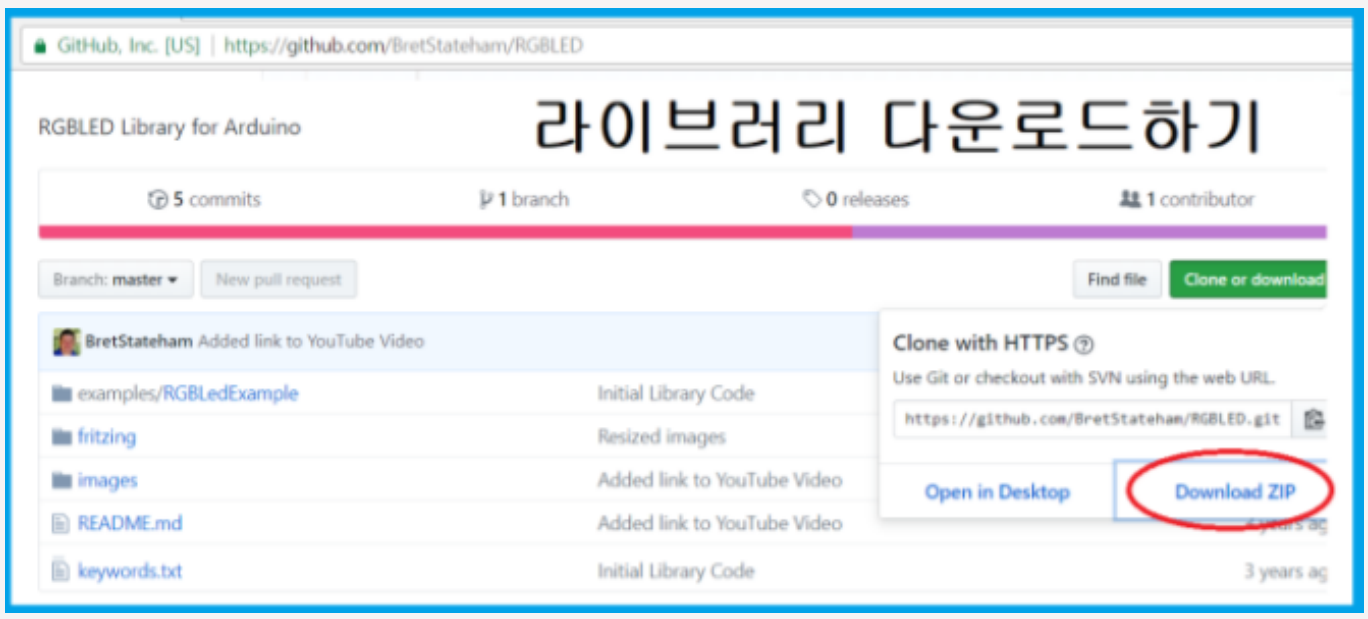
연습 문제

1. 이 프로젝트에서 사용한 프로그램을 응용하여 무지개색의 순서대로 RGB LED의 색깔이 변하는 프로그램을 작성해 보시오.

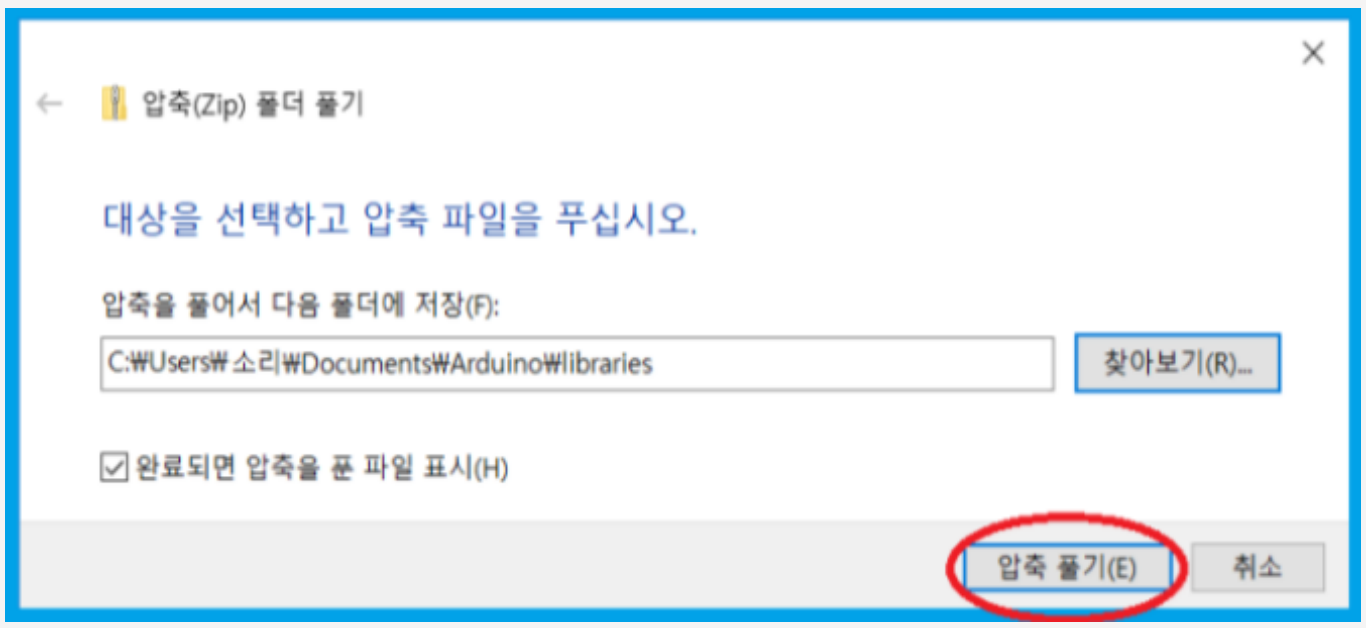
2. 무지개색 순서로 현재 표현한 색깔이 다음 색깔로 바뀔 때, 먼저 표현한 색깔이 서서히 사라진 다음 새로운 색깔이 표현되도록 프로그램을 고쳐 보시오.

라이브러리 설치하기

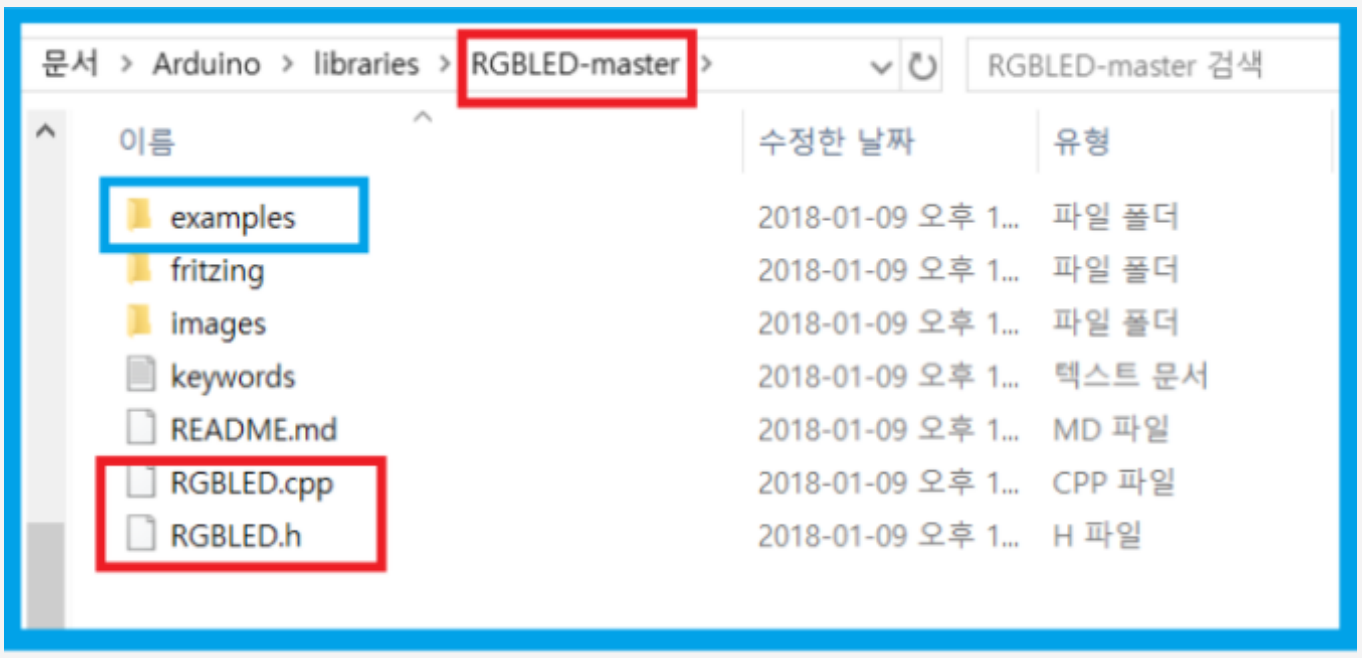
먼저, 인터넷에서 설치하고자 하는 라이브러리 파일을 다운로드 합니다.



다운로드한 파일을 아두이노 스케치 디렉토리 안에 있는 libraries 폴더안에 압축을 풀면 라이브러리가 설치됩니다.



라이브러리의 디렉토리명과 파일이름은 같습니다. 예를 들면, RGBLED.h 라는 라이브러리를 설치하면, RGBLED-master 라는 폴더가 생성되고, 그 폴더 안에 RGBLED.cpp, RGBLED.h, 그리고 keywords.txt파일이 포함되어 있습니다. 이 세 가지 파일이 동일한 디렉토리 안에 있지 않을 경우, Arduino IDE는 라이브러리를 인식하지 못합니다.
또한, 대부분의 라이브러리 폴더안에는 examples라는 하위 폴더가 있고, 그 안에 즉시 사용할 수 있는 예제 프로그램이 들어 있어서 라이브러리 사용법을 쉽게 익힐 수 있습니다.



Project 8

기울기 센서

08

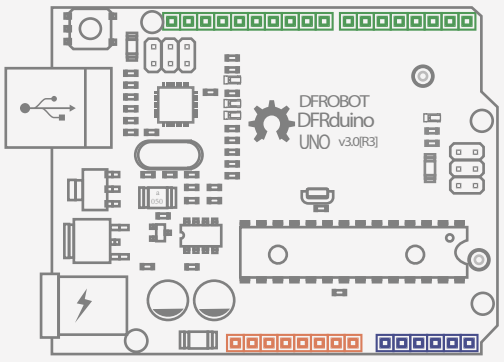
DFROBOT
DRIVE THE FUTURE

기울기 센서

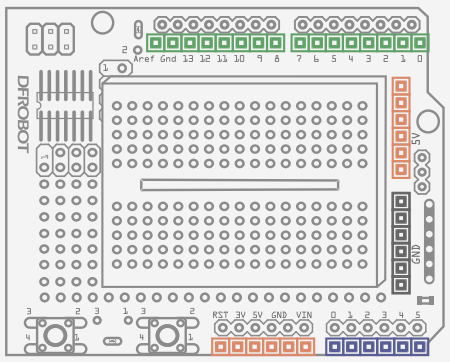
기울기 센서는 물체의 움직임과 방향을 감지하는 센서인데 2개의 조그만 금속구슬과 접점으로 구성되어 있습니다. 센서가 정해진 방향으로 기울어지면 작은 금속구슬 2개와 접점이 접촉되어 회로를 구성합니다.

이 프로젝트에서는 기울기센서가 움직임을 감지하면 LED를 켜는 프로그램을 작성합니다.

필요한 부품



DFduino UNO R3 x1

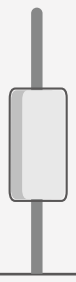


Prototype Shield x1

Jumper Cables M/M x5

5MM LED x1

Resistor 220R x2



Tilt Switch Sensor x1

회로 구성

기울기 센서는 누름스위치와 비슷하게 동작합니다. 센서가 연결된 디지털핀이 하이임피던스 상태가 되어 오작동하는 것을 방지하기 위하여 풀다운 저항을 함께 사용해야 합니다.

물론, LED에는 전류제한용 직렬저항도 연결해 줘야 합니다.

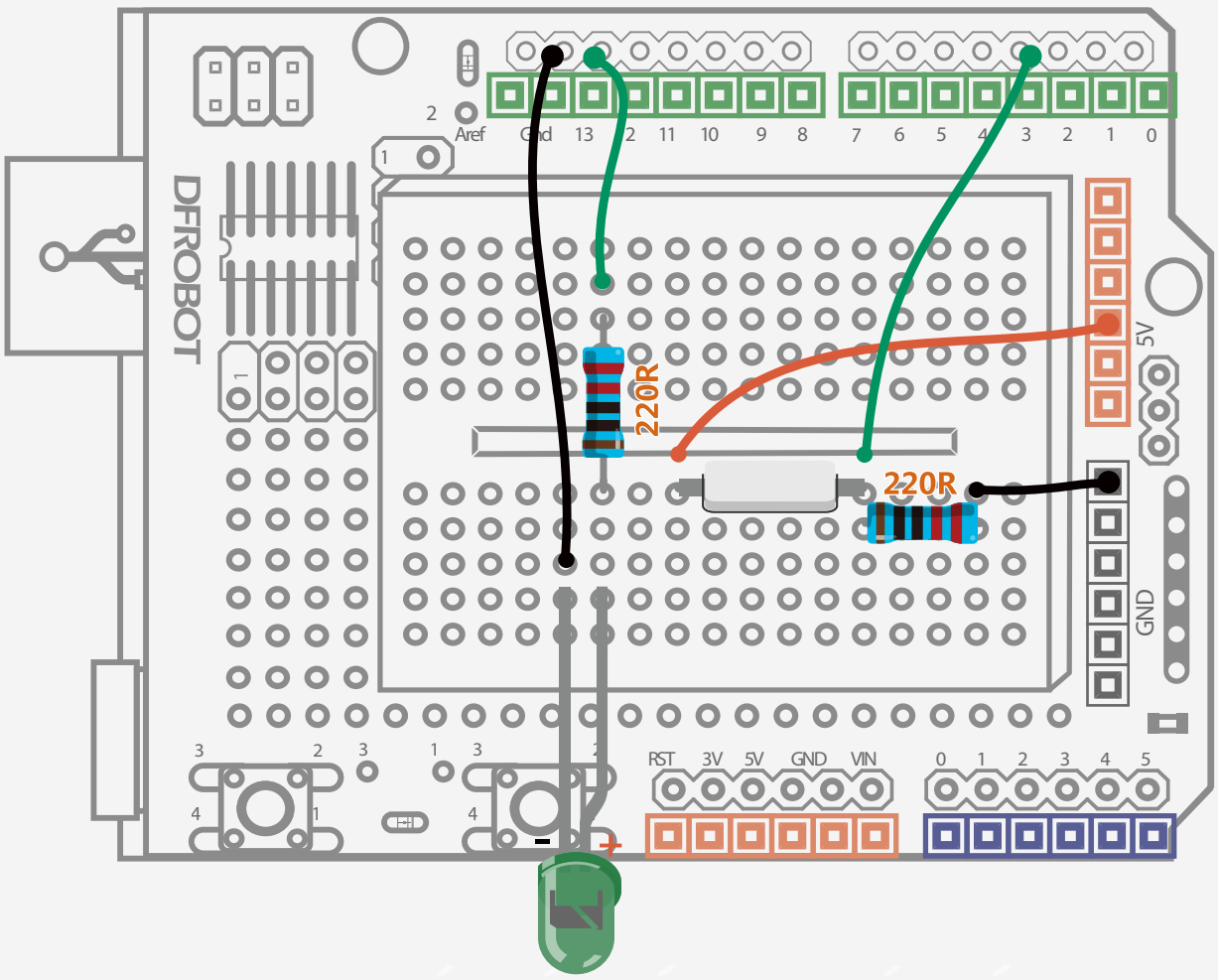


그림8-1 기울기 센서 회로도

Code 해설

프로그램 예제 8-1

```
//예제8 – 기울기 센서

int SensorLED = 13;                    //LED 는 D13로 정의
int SensorINPUT = 3;                   //기울기센서는 D3로 정의
unsigned char state = 0;

void setup() {
  pinMode(SensorLED, OUTPUT);          //LED를 출력으로 정의
  pinMode(SensorINPUT, INPUT);         //기울기 센서를 입력으로 정의

//D4핀이 0V(낮은 전력)에서 5V(높은 전력)로 상승할 때 인터럽트1 발생, 이때
blink함수 실행

attachInterrupt(1, blink, RISING);
 }

void loop(){
      if(state!=0){                    // state가 0 이 아니면
        state = 0;                     // state에 0 저장
        digitalWrite(SensorLED,HIGH);  // LED켜짐
        delay(500);                    // 0.5초 대기
      }
      else{
        digitalWrite(SensorLED,LOW);   // state가 1이면, LED꺼짐
     }
}

void blink(){                          //인터럽트시 실행할 함수
state++;                               //인터럽트시 state값 1 증가
}
```

프로그램 실행결과

아두이노 보드를 흔들어, 기울기센서가 움직임을 감지하면 LED가 켜지고, 움직임을 멈추면 LED가 꺼지는 동작을 반복하게 됩니다.

Code 해설

이 프로젝트에서는 처음으로 인터럽트 기능을 사용합니다. 이 인터럽트 기능의 실행과정은 다음과 같이 동작합니다.

스케치를 실행하면,

(1) 기울기 센서가 설치된 아두이노보드가 움직이지 않으면 인터럽트가 발생하지않습니다. 즉, 기울기 센서에는 움직임이 감지되지 않으며, LED는 꺼진 상태를 유지하고 있습니다.

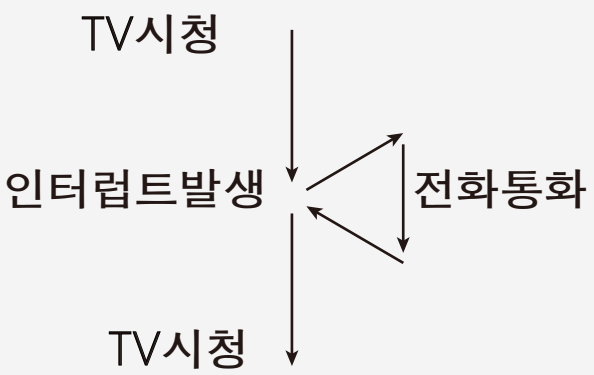
(2) 아두이노 보드를 흔들거나 움직이면, 기울기 센서가 움직임을 감지하고 인터럽트가 발생되어 메인프로그램이 일시 중단되고 blink()함수가 실행됩니다. blink()합수는 변수 state의 값을 1 증가시키는 함수인데, 이 함수의 실행이 끝나면, 다시 메인프로그램의 이전 실행위치 다음 부분으로 돌아갑니다.

(3) if문을 사용하여 state 값이 0이 아니면 LED를 켜고, state값이 0이면 LED가 꺼진 상태를 유지하면서 위 과정을 반복합니다.

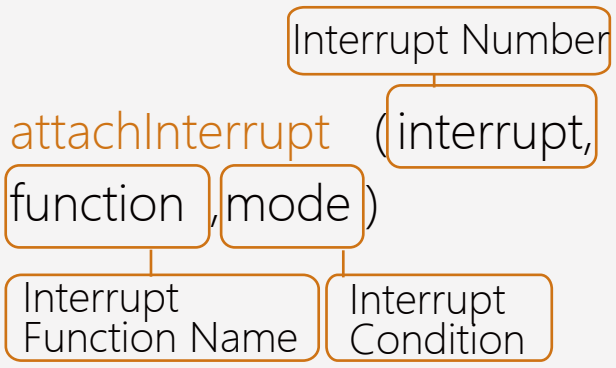
여러분이 집에서 TV를 시청하는 중에 전화벨이 울렸다고 가정해 봅시다. 이 경우, TV시청을 중단하고 전화를 받게 되겠죠. 물론, 전화통화를 마치면 TV 시청을 계속할 겁니다. 여기서, 전화벨은 인터럽트 발생조건이고, 전화통화를 인터럽트 실행이라고 합니다. 여기서, 인터럽트가 발생했을 때 실행할 함수 (ISR,Interrupt Service Routine)는 "전화통화"인데 이를 인터럽트라고 부르기도 합니다.

인터럽트를 발생할 수 있는 디지털 핀은 따로 정해져 있는데, 아두이노 보드의 종류에 따라 다릅니다. 아두이노 UNO보드는 2개의 인터럽트 핀이 있습니다. 즉, 인터럽트1은 D2핀, 인터럽트2는 D3핀을 사용해야 합니다. 아래 링크를 클릭하면 attchInterrupt() 함수에 대한 자세한 설명을 읽을 수 있습니다.

http://arduino.cc/en/Reference/Attachment



Code 해설



attachInterrup() 함수는 3개의 파라미터를 사용합니다.

interrupt:

사용하는 인터럽트 핀을 지정하는 파라미터입니다. 0 또는 1을 지정해야 합니다. 인터럽트0은 디지털 2번핀, 인터럽트1은 디지털3번핀을 사용합니다.

function:

인터럽트가 발생했을 때 호출하여 사용하는 함수를 의미합니다. 이 함수는 파라미터를 사용하지 않으며, 리턴 값도 없습니다. 인터럽트 함수가 실행되는 동안은 delay()함수와 millis()함수등 일부 함수를 사용할 수 없습니다. 또한, 인터럽트 함수를 사용하는 동안은 시리얼포트를 통해 데이터를 읽을 수 없게 되며, 시리얼포트에서 사용 중이던 일부 데이타를 잃을 수도 있습니다.

mode:
인터럽트가 발생되는 시점을 지정하는 파라미터이며 LOW, CHANGE, RISING, FALLING 등 4가지를 설정할 수 있습니다.

LOW 는 인터럽트 핀이 Low로 바뀔 때 인터럽트가 발생하게 됩니다.CHANGE 는 인터럽트 핀의 상태가 바뀔 때 인터럽트가 발생하게 됩니다.RISING은 인터럽트 핀이 LOW에서 HIGH로 바뀌는 순간 인터럽트가 발생하게 됩니다. FALLING은 인터럽트 핀이 HIGH에서 LOW로 바뀌는 순간 인터럽트가 발생하게 됩니다.
그럼, 프로그램을 다시 살펴봅시다.

attachInterrup(1, blink, RISING)

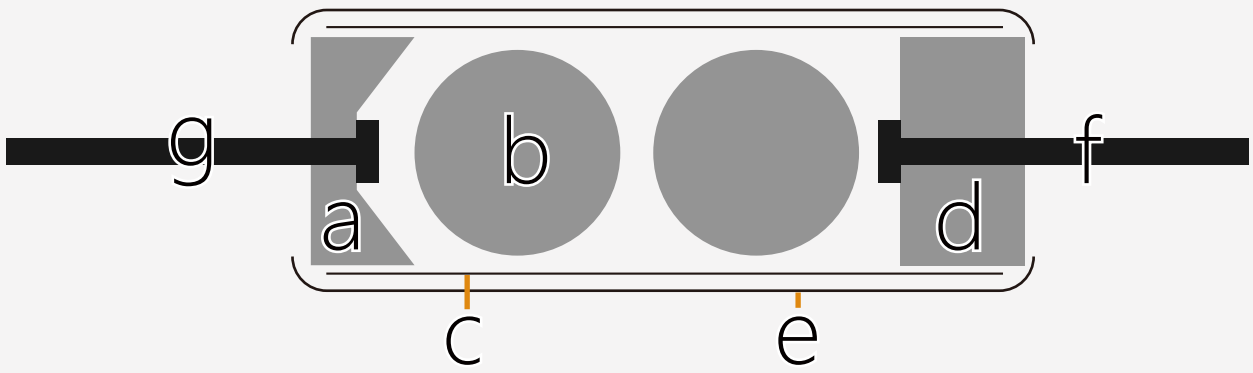
위 코드에서, 인터럽트1(즉 D3핀)을 사용하기 때문에 첫 번째 파라미터는 1입니다.
Blink는 두 번째 파라미터로, 인터럽트가 발생할 때 실행할 함수로 사용자가 따로 정의한 겁니다.
RISING은 인터럽트핀이 LOW 상태에서 HIGH상태로 바뀌는 순간 인터럽트가 발생하도록 설정합니다.
그러면, 이 코드에서는 왜 RISING모드를 사용했을까요?
기울기 센서의 특성 때문입니다. 즉, 센서가 움직임을 감지했을 센서가 0V에서 5V로 상태가 바뀌는 데, Rising 모드가 아닌 다른 모드를 사용하면, 센서의 변화를 감지할 수 없기 때문입니다.

Hardware 탐구

기울기 센서

기울기 센서는 "볼 스위치", "비드 스위치", "진동 스위치"라고도 불립니다. 이름은 다르지만, 동작원리와 기능은 모두 같습니다.

기울기 센서는 조그만 원통안에 구슬을 넣어 스위치 역할을 하도록 만든 부품입니다. 센서가 기울어지면, 내부의 구슬이 움직여 접점에 닿으면서 스위치를 켜는 역할을 하게 됩니다. 오른쪽의 그림을 살펴보면 동작원리를 이해할 수 있습니다.



a. 금속 덮개 b. 금속 구슬 c. 금속관
d. 핀고정 절연체 e. 열 수축튜브
f. 금속 핀
g. 금속 핀

그림 8-2 진동센서 구조

Project 6

경보음 발생기

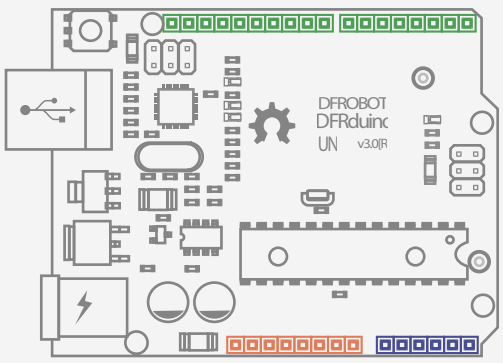
06

DFROBOT
DRIVE THE FUTURE

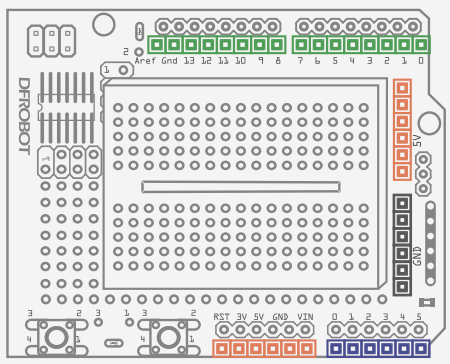
경보음 발생기

이번 프로젝트에서는 처음 사용하는 부품은 부저 입니다. 부저를 사용하면, 여러 가지 음색의 소리를 발생시킬 수 있습니다. 만일, LED와 부저를 함께 사용한다면 색 다른 경보 장치도 만들 수 있습니다.

필요한 부품

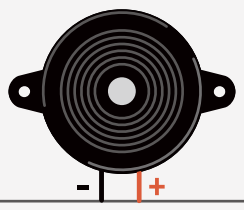


DFduino UNO R3 x1



Prototype Shield x1

Jumper Cables M/M x2



Buzzer x1

회로 구성

아래 그림처럼 실험할 회로를 구성하시오. 부저의 긴 다리가 +극이고 짧은 다리가 –극입니다. 부저의 -극을 GND와 연결하고, +극은 8번 디지털 핀과 연결합니다.

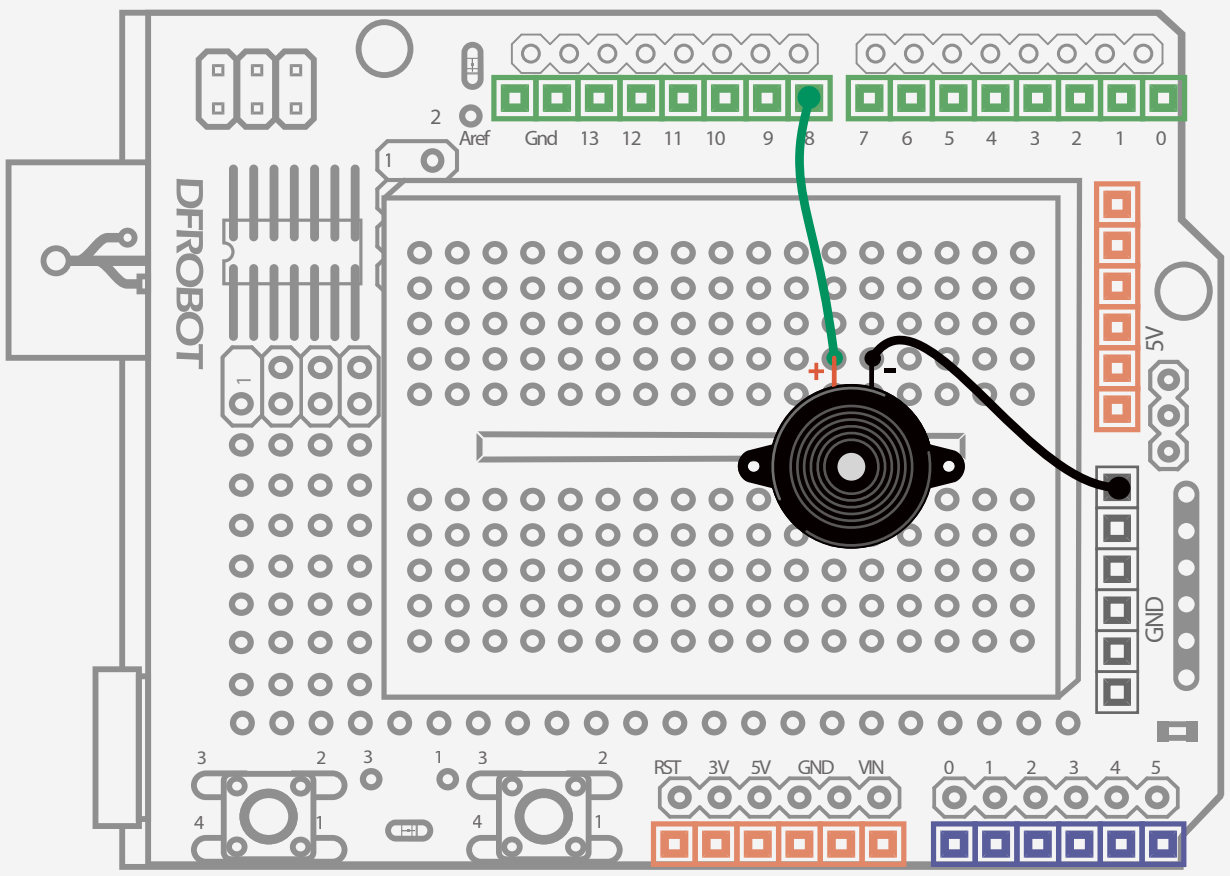


그림 6-1 경보음 발생기

Arduino Sketch

프로그램 예제 6-1

```
//예제-6 경보음 발생기
float sinVal;
int toneVal;

void setup(){
    pinMode(8, OUTPUT);
}

void loop(){
    for(int x=0; x<180; x++){
        //각도를 라디안으로 변환
        sinVal = (sin(x*(3.1412/180)));
        //Sin()값의 변화에 따라 달라지는 소리발생
        toneVal = 2000+(int(sinVal*1000));
        //D8번 핀으로 경보음 발생
        tone(8, toneVal);
        delay(2);
    }
}
```

이 프로그램을 실행하면, 높은 음과 낮은 음의 소리가 번갈아가며 발생합니다.

Code 해설

먼저, 변수 2개를 선언합니다

```
float sinVal;
int toneVal;
```

float형 데이타는 실수(소숫점이 포함된 수)형 데이터를 의미합니다.

정수형 데이터보다 실수형 데이터가 더 큰 수를 취급할 수 있기 때문에 실수형 데이타를 사용하면 더 정밀한 계산 결과를 얻을 수 있습니다. 이 프로그램에서는 시간에 따라 변하는 사인 함수값을 실수형 변수에 저장합니다.

사인파는 파형의 변화가 일정하기 때문에 주파수로 변환하기가 쉽습니다. 따라서, 이 프로그램에서는 변수 sinVal에 저장된 사인값을 주파수로 바꿔 변수 toneVal에 저장합니다.

sin() 함수의 파라미터는 각도인데 단위가 라디안입니다. 따라서 sin() 함수 값을 구하려면

계산:

라디안 = **각도** * 3.1412/180

을 사용해서 각도 x의 단위를 라디안으로 바꿔주어야 합니다.식

```
sinVal=(sin(x*(3.1412/180)));
```

그러면, 이 값을 알람음의 주파수로 바꿔줍니다:

```
toneVal = 2000+(int(sin-
Val*1000));
```

"sinVal"은 소수점을 사용할 수 있는 실수형 변수입니다. 주파수 값이 소수점이 있으면 불편합니다. 따라서, 아래 코드를 사용해서 소수점이 없는 정수형 데이터로 바꿔 줍니다.

```
int(sinVal*1000)
```

사람은 20hz ~ 20000HZ 사이의 소리만 들을 수 있습니다. 따라서, 구한 값에 1000을 곱하고 2000을 더해줘서, toneVal의 값이 2000 ~ 3000의 범위가 되도록 합니다.

발생할 소리의 주파수는 toneVal에 저장되어 있고, 부저는 DP8에 연결되어 있으므로 소리를 내려면 아래와 같이 코드를 작성하면 됩니다.

```
tone(8, toneVal);
```

소리 발생과 관련된 함수에 대하여 좀 더 자세히 알아 봅시다

1、tone(pin,frequency)

"pin"은 부저가 연결된 디지털 핀이고, frequency는 발생될 소리의 주파수(단위 Hz)입니다. 즉 tone() 함수는 지정된 디지털 핀으로 지정된 주파수의 소리를 출력하는 기능을 합니다.

2、tone(pin,frequency, duration)

여기서 pin 과 frequency는 위의 설명과 동일합니다. duration은 발생되는 소리가 지속되는 시간을 의미합니다.

만일 duration을 지정하지 않으면, 부저에서 소리가 계속 납니다.

3、noTone(pin)

noTone()함수는지정된 디지털핀에 발생되는 소리를 멈추는 기능을 합니다.

Hardware 탐구

부 저

부저는 소리를 발생시킬 때 사용하는 전자부품입니다. 일반적으로 피에조 부저와 마그네틱 부저의 2종류가 있습니다.

능동형 부저와 수동형 부저의 차이점

피에조 부저와 마그네틱 부저는 입력신호를 처리하는 작동방식에 따라서 능동형 부저와 수동형 부저로 더 세분할 수 있습니다. 두 방식의 차이는 전원 공급방식과 발진 회로가 내장되어 있는지 여부에 있습니다.

능동형 부저는 내부에 스스로 소리를 내는 부품이 있기 때문에 전원 스위치를 켜기만 하면 소리가 발생합니다. 즉, 능동형 부저는 직류 전원을 연결하기만 하면 일정한 주파수의 단일음을 발생하는 부저 입니다. 능동형 부저는 내부에 몰리브데넘으로 만들어진 특수 필름이 들어 있습니다. 외부에서 전원을 연결하면, 이 특수 필름이 진동을 하면서 소리를 발생시킵니다. 능동형 부저는 극성을 갖지 않기 때문에 어느 방향으로 연결해도 됩니다. 교재에는 능동형이 아닌 수동형 마그네틱 부저가 들어있습니다.

수동형부저는 내부에 스스로 진동음을 발생하는 부품(또는 회로)이 들어있지 않습니다. 따라서 소리를 발생시키려면 직류 전류 대신 2khz ~ 5khz 정도의 구형파 신호가 필요합니다. 수동형 부저는 극성이 있습니다. 따라서, 극성이 바뀌지 않도록 주의해서 사용해야 합니다. LED와 마찬가지로 일반적으로 긴 다리는 +극(애노드,anode)이고 짧은 다리는 –극(캐소드,cathode)입니다. 일반적으로 능동형 부저보다는 수동형 부저가 더 사용하기 쉽습니다.

부저에 대하여 더 깊이 알고 싶다면, 다음 아이디어를 참고하세요

수동형부저는 여러 가지 음악효과를 만드는데 사용하면 좋습니다.

또한, 부저는 다양한 분야에 응용할 수 있습니다. 예를 들면, 적외선 또는 초음파 센서를 이용한 감시장치, 물체접근 경보장치, 온도 경보장치, 가스누출 경보장치 등 여러가지 장치에 부저를 응용한 회로를 부가하여 사용할 수 있습니다. 경보장치 이외에도, 여러 가지 음악을 연주하는 장치로도 사용할 수 있습니다.

연습문제

1. 빨간색 LED를 사용해 경보장치를 만들어 보시오

이 프로젝트에서는 sin 함수를 이용해서 소리를 출력하고 있는데, 부저를 LED로 바꿔 보고 어떤 결과가 나타나는지 관찰해 보세요.

2. Project 3에서 학습한 내용을 응용하여 초인종을 만들어 보시오. 버튼스위치를 누르면 부저 소리가 울리도록 회로를 꾸미고, 프로그램도 작성해 보세요.

Project 7

온도 조절기

07

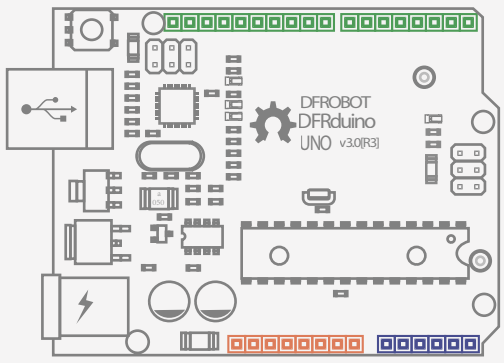
DFROBOT
DRIVE THE FUTURE

온도 조절기

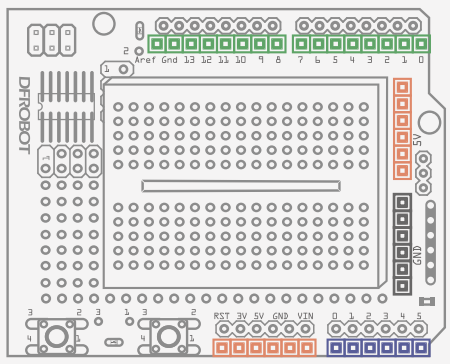
이전 프로젝트에서 완성한 "경보음 발생기"에 온도센서를 추가해서, 온도가 정해진 범위에 이르면, 소리가 발생하는 프로그램을 작성해 봅시다.

이번 프로젝트는 센서에 따라 어떤 반응을 하도록 하는 간단한 프로그램이지만, 응용범위가 아주 넓습니다. 이 프로젝트를 조금만 응용하면, 매우 유용한 온도조절기가 됩니다.

필요한 부품

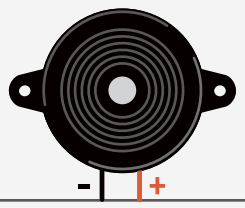


DFduino UNO R3 x1



Prototype Shield x1

Jumper Cables M/M x5



Buzzer x1

Tem. Sensor x1

회로구성

프로젝트6에서 꾸민 회로에 아래 그림처럼 LM35 온도센서를 추가하여 회로를 완성해 봅시다. 센서의 핀이 서로 바뀌지 않도록 주의해야 합니다. LM35는 5V, A0, GND핀과 연결되어야 합니다.

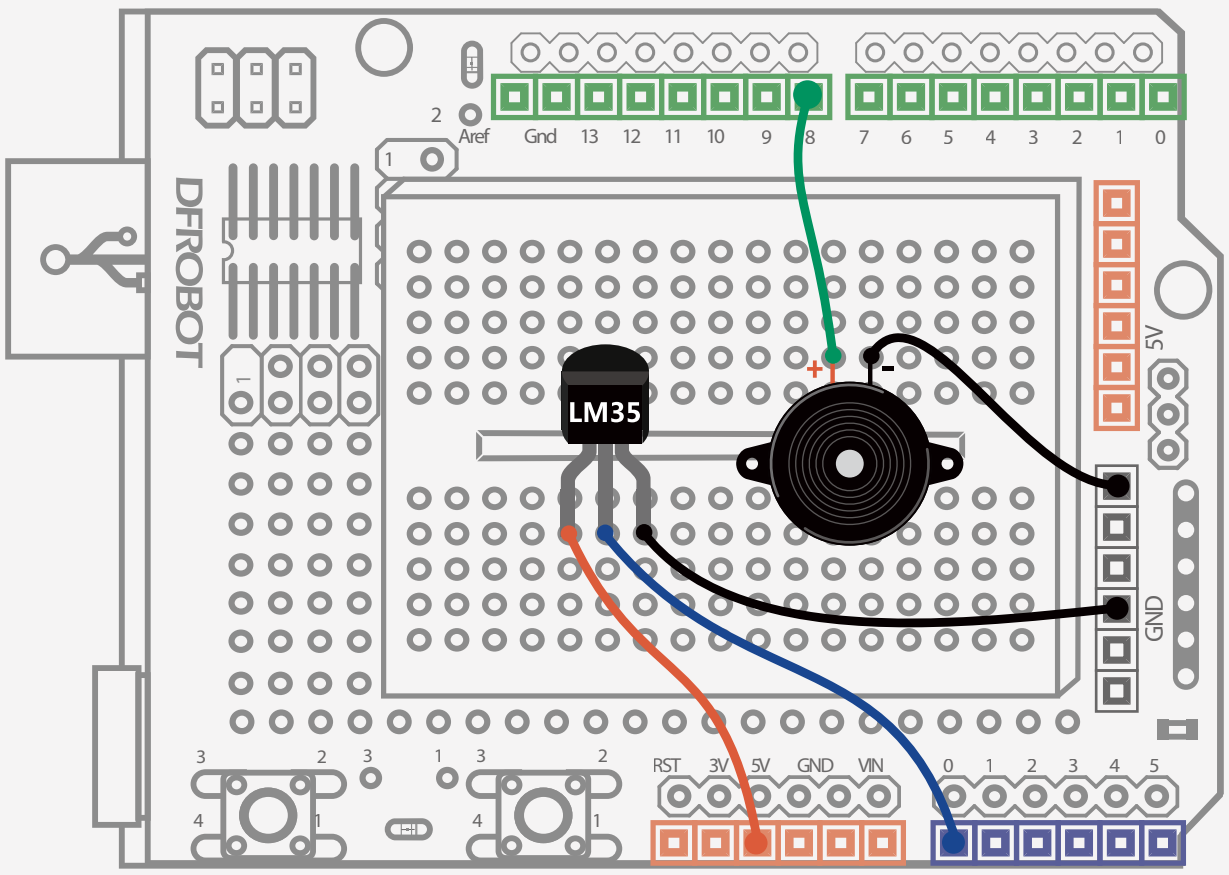


그림 7-1 온도 조절기 회로도

프로그램 예제 7-1:

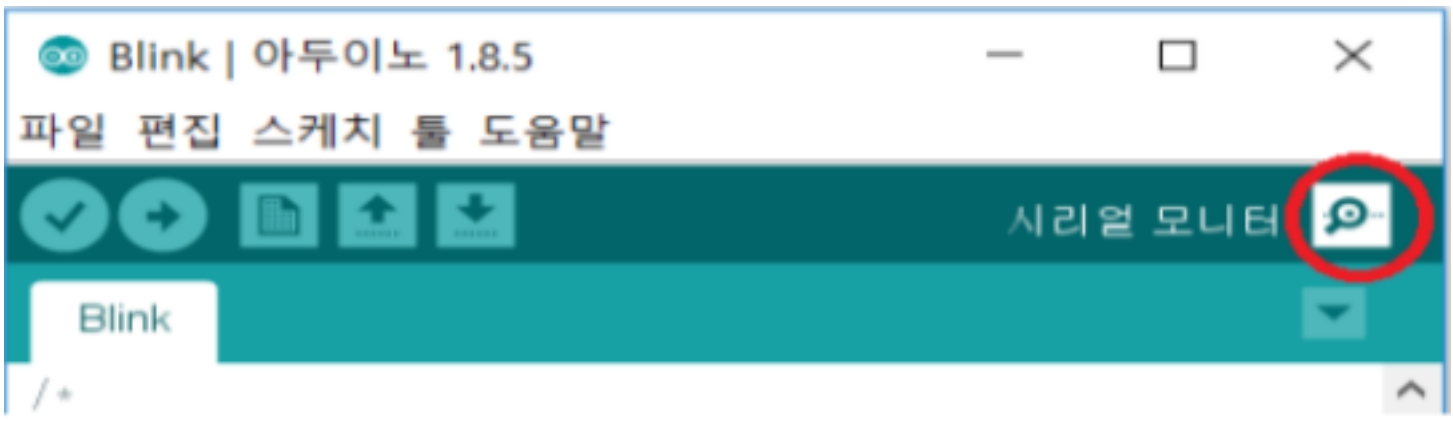
```
//예제 7 온도 조절기
float sinVal;
int toneVal;
unsigned long tepTimer ;

void setup(){
    pinMode(8, OUTPUT);                 // 부저 핀
    Serial.begin(9600);                 //9600 bps
}
void loop(){
    int val;                            //LM35 센서값 저장
    double data;                        //온도값 저장
    val=analogRead(0);                  //LM35를 A0핀에 연결, 센서값은 val에저장
    data = (double) val * (5/10.24);    // 측정한 전압을 온도로 변환

    if(data>27){   //27도 보다 높으면 소리발생
        for(int x=0; x<180; x++){
          //각도를 라디언으로 변환후 Sin값 저장
          sinVal = (sin(x*(3.1412/180)));
          //Sin값을 이용해 발생음 주파수 산출
          toneVal = 2000+(int(sinVal*1000));
          //D8핀으로 소리발생
          tone(8, toneVal);
          delay(2);
        }
    } else {                            // 27도 이하일 경우, 부저가 꺼짐
        noTone(8);                      //부저가 꺼짐
    }
    if(millis() - tepTimer > 50   // 50mS마다 온도 출력
          tepTimer = millis();
          Serial.print("temperature: "); // “temperature: “ 표시
          Serial.print(data);           // 온도값 표시
          Serial.println("C");          //온도 단위 “C”를 표시
      }
}
```

시리얼 모니터 열기

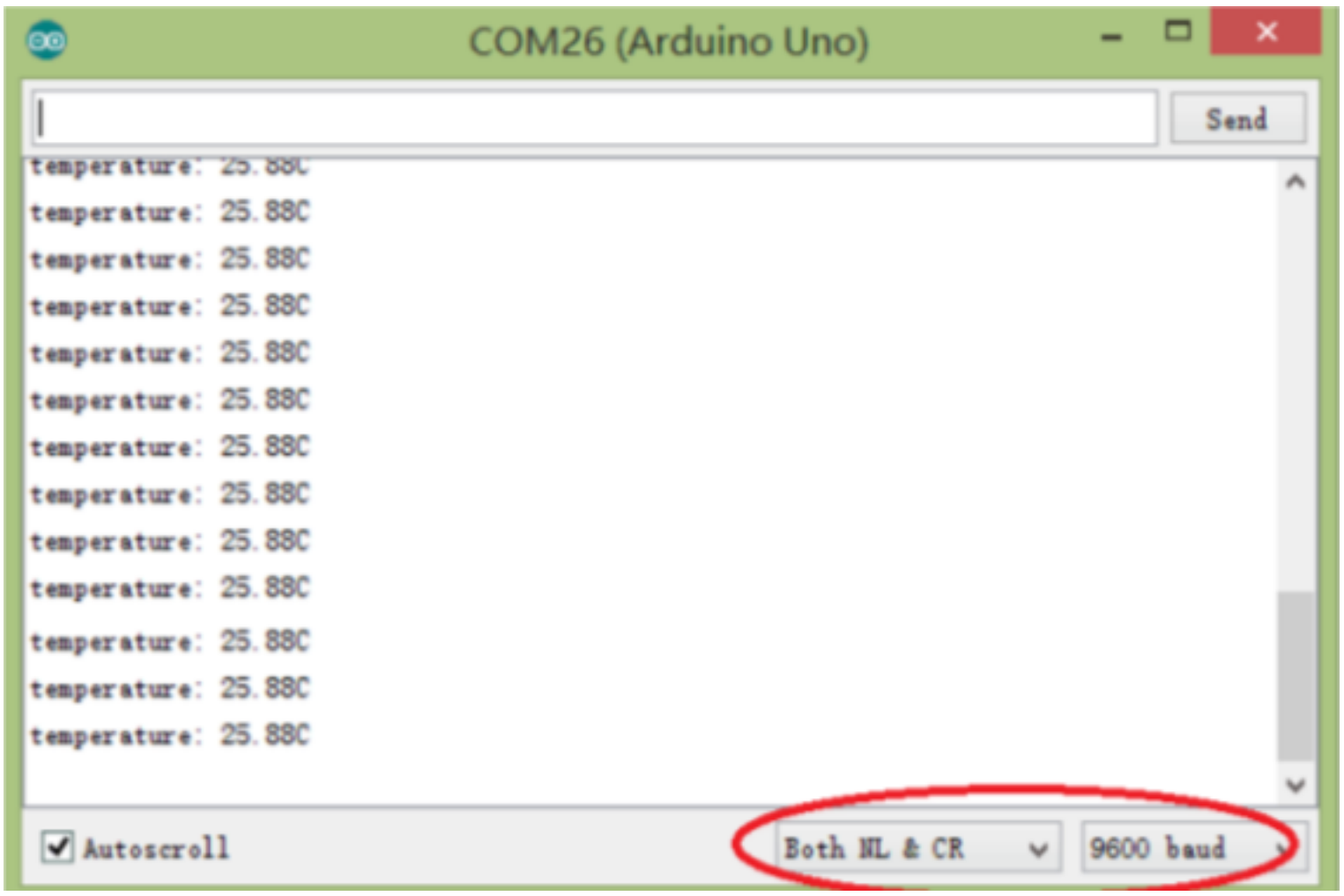
아두이노 스케치를 업로드하여 실행하고, 시리얼 모니터를 열고 표시되는 온도를 읽어 보시요

이 프로그램은 온도센서가 측정한 온도를 시리얼모니터로 표시하는 동시에, 온도가 27℃ 이상이면, 부저에서 소리가 나기 시작하고, 27℃ 아래로 내려가면 소리가 멈춥니다.

먼저, 시리얼 모니터 아이콘을 클릭합니다.



시리얼 모니터 창이 열리면 온도가 나타납니다. 글자가 깨져서 나타나면, 시리얼 통신속도가 프로그램과 일치하는지 확인합니다. 만일, LM35 온도센서에 손을 대고 있으면, 체온이 센서로 전달되어 표시된 온도가 빠르게 변하는 것을 알 수 있습니다.



대부분의 프로그램은 프로젝트6 과 동일합니다. 이 프로젝트를 통해 변수와 함수의 사용에 대하여 좀 더 분명하게 이해하기를 바랍니다.

프로그램의 첫머리에서 다음과 같이 3개의 변수를 선언 합니다.

```
float sinVal;
int toneVal;
unsigned long tepTimer ;
```

세 번째 변수 tepTimer 는 unsigned long형 변수인데 mS단위의 시간데이타를 저장하는 변수 입니다. 왜 "unsigned ling"형 변수를 사용했을까요?

그 이유는 오랜 시간동안 경보장치를 사용할 수 있도록 하려고 큰 수를 저장할 수 있는 long 형 변수를 사용하였으며, 음수 데이타를 사용하지 않기 때문에 unsigned 형으로 선언한 겁니다.

setup()함수의 첫 줄을 살펴보면 pinMode 명령어을 사용하고 있는데, 부저를 연결하는 디지털 핀만 OUTPUT으로 설정했고, LM35 온도센서를 연결한 핀은 아무런 설정도 하지 않고 있습니다. 이유가 뭘까요?

LM35 온도센서는 아날로그 핀을 사용하기 때문에 pinMode설정이 필요 없습니다. 디지털 핀만 pinMode 설정이 필요합니다.

다음은 시리얼 통신을 위한 코드입니다.

```
Serial.begin(9600);
```

위 코드에서 숫자 9600은 시리얼 통신속도를 의미합니다. 단위는 baud 입니다. 숫자가 클수록 빠른 속도로 데이터를 주고 받을 수 있습니다. 일반적으로 아두이노 프로그램에서 9600을 기본값으로 사용합니다. 하지만, 다른 속도로 변경하여 사용하는 경우도 많습니다.

loop()함수의 첫머리에서 변수 val 과 변수 data를 선언하고 있습니다. 이 변수는 loop() 함수 내부에서만 사용할 수 있는 지역변수 입니다.

```
val=analogRead(0);
```

위 코드는 아날로그 0번핀 (A0핀)의 전압을 측정한 값을0~1023 사이의 숫자로 변환하여 변수 val 에 저장하라는 뜻입니다.

"analogRead (pin)"는 지정된 아날로그핀의 전압을 읽는 함수입니다.

```
analogRead(pin)
```

아두이노보드의 아날로그핀은 10bit 의 ADC (analot to digital converter)와 연결되어 있습니다. ADC는 기본적으로 전압 측정기입니다. 아날로그핀이 연결된 0~5V 사이의 아날로그 전압을 0~1023사이의 디지털 값으로 바꾸는 기능을 합니다. 즉, 0은 0V, 1023은 5V입니다. LM35는 온도가 1℃ 변할 때마다 전압이 10mV씩 변합니다.

```
data = (double) val * (5/10.24);
```

아두이노가 LM35센서로부터 읽은 값은 0 ~ 1023 숫자이며, 실제 전압 값이 아닙니다. 이를 실제 전압으로 바꾸려면 단계값 (5/1023)을 곱해줘야 하고, 다시 온도로 바꾸려면 0.01로 나눠주어야 합니다. 온도값은 소수점이 포함된 수이므로 double를 표시해 double 형 데이터로 바꾼다음, double형 변수 data에 저장합니다.

The Serial Port

시리얼포트는 아두이노가 외부장치와 연결하여 데이터를 송수신하는 용도로 정해진 디지털핀 입니다. 아두이노 보드는 적어도 1개이상의 시리얼 포트를 갖고 있는데, 디지털핀 0 (RX/데이터 수신)과 디지털 1 번핀(TX/데이터 송신)이 바로 시리얼포트 입니다. 따라서, 시리얼통신을 사용할 경우 디지털핀 1번핀과 2 번핀을 사용합니다. 아두이노는 시리얼포트를 사용해서 프로그램을 다운로드합니다. 다운로드중에는 USB가 아두이노 포트를 사용하기 때문에 다른 외부장치와 시리얼 통신을 할 수 없습니다. 만일 다운로드 중 외부장치로 시리얼 통신을 시도하면, 다운로드를 제대로 하지 못하게 됩니다.

if else 조건문

다음은, if else 조건문을 알아보겠습니다.

if else문의 형식은 아래와 같습니다.

```
if (expression) {
  Statement 1;
} else{
   Statement 2;
}
```

if else문은 다음과 같이 실행됩니다.

먼저, 판단의 기준이 되는 조건이 정해져야 합니다.

만일, 이 조건이 참일 경우 명령문 1을 실행하고, 참이 아닐 경우 명령문 2로 건너뜁니다.

즉, if else 문에서는 실행문1 또는 실행문2 중 하나만 선택하여 실행하게되며, 실행문1,2가 동시에 실행될 수 없습니다.

```
if(data>27){
 for(int x=0; x<180; x++){
      ......
  }
} else {
      ......
}
```

위 코드에서 온도가 27도보다 높을 경우 실행문1 부분(부저를 울림)을 실행하고, 그렇지 않을 경우는, 실행문2 부분(부저를 멈춤)을 실행하게 됩니다.

시리얼 모니터

온도에 따라 부저를 울리게하는 기능 말고도, 측정 온도를 알고 싶을 때가 있을 겁니다. 이 경우는 측정온도를 시리얼포트로 출력하고 PC의 모니터로 읽으면 됩니다.

측정온도를 500mS 간격으로 시리얼포트로 출력하려면 "millis()"함수를 사용할 필요가 있습니다.(자세한 것은 프로젝트 3을 참고하세요)

```
Serial.print(val);
Serial.println(val);
```

Serial.print(val)함수와 Serial.println(val)함수는 1바이트형 데이터 val을 ASCII 문자로 바꿔 시리얼모니터에 출력하라는 명령인데, 아래와 같은 규칙에 따라서 데이터를 출력합니다.

1. (숫자) 숫자는 그대로 숫자로 출력합니다.

2. (소수) 소수점이 포함된 숫자는 소수 두 자리 까지만 출력됩니다.

3. ('문자열') 따옴표속의 문자는 문자열로 출력됩니다.

4. (문자) 따옴표 없는 첫문자에 해당하는 ASCII코드가 출력됩니다.

그러면, 실제 사용 예를 살펴봅시다.

Serial.print('A');

문자"A"가 출력됨.

Serial.print(A);

숫자65 ("A"의 ascii코드)가 출력됨.

Serial.print("Hello world."); "Hello world."가 출력됨

Serial.print(123);

숫자 123이 출력됨

Serial.print(123.4567);

숫자 123.45가 출력됨.

Serial.print(); 줄 바뀜 없음

Serial.println(); 줄 바꿈이 생김

Serial.wirte(); 와 Serial.print();의 차이점

Serial.write() 라는 명령은 ASCII형의 문자가 아닌 byte형의 데이타를 출력할 때 사용합니다. 아두이노의 공식 홈페이지를 방문해서 Serial.print()와 Serial.write()의 차이점을 알아보세요.

```
Serial.print(data);
```

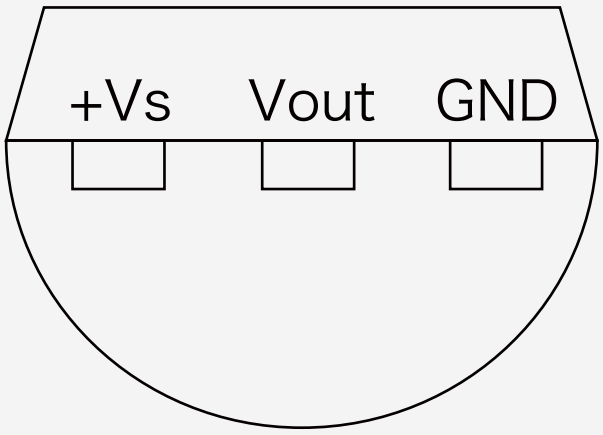
여기서 data는 왜 문자열이 아닌 숫자가 출력될까요?

정답은 이 프로그램의 setup() 함수에서 변수 data를 숫자를 저장하는 정수형 변수로 선언했기

Hardware 탐구

LM35 온도센서

LM35온도센서는 널리 사용되는 온도센서로 허용오차는 +/- 0.25℃ 입니다.



이 센서는 3개의 핀이 있습니다.

+Vs 는 +전원

Vout 은 전압출력

GND 는 - 전원입니다.

이 센서에서 출력되는 전압과 온도의 관계는 다음 식과 같습니다.

Vout = 10mV/C * T C

(단, 온도가 2~40°C 범위일 때)

센서에 대하여 좀 더 자세히 알고 싶다면, 이 센서의 데이터 쉬트를 참고하면 됩니다. 데이터 쉬트에는 온도센서가 어떤 원리로 동작하는지 자세하게 설명되어 있습니다.

여러 가지 전자부품들의 데이터 쉬트는 아래 사이트에서 쉽게 다운로드할 수 있습니다.

http://www.alldatasheet.com/.

연습 문제

LED를 추가하여, 온도가 설정된 범위에 있을 경우 LED가 켜지고 동시에 부저가 울리는 프로그램을 작성해 보세요.

좀 더 응용한다면, 온도설정 범위를 여러 구간으로 잘게 나눈 뒤, 각 구간마다 다른 색깔의 LED와 다른 음색의 부저 소리가 나도록 프로그램을 작성할 수도 있습니다.

예

- 온도가 10℃ 이하이거나 35℃ 이상일 경우 : 빨간색 LED 켜지고, 빠른 간격의 경보음 발생

- 온도가 25℃이상 35℃ 이하일 경우 : 노란색 LED가 켜지고, 느린 간격의 경보음 발생

- 온도가 10℃이상 25℃ 이하일 경우 : 녹색 LED가 켜지고, 경보음이 발생하지 않음

Project 9

빛 감지 LED

09

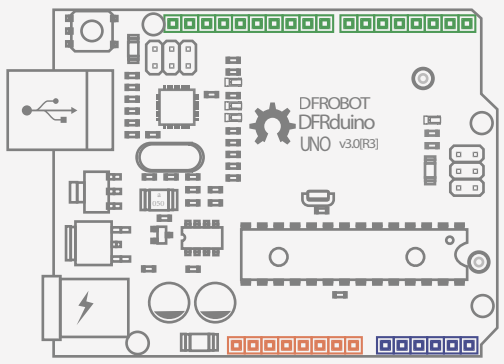
DFROBOT
DRIVE THE FUTURE

빛 감지 LED

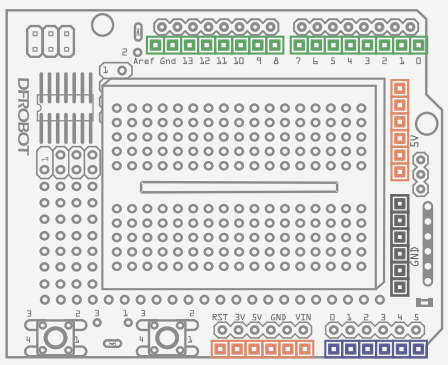
감광센서는 빛이 감지되면 저항 값이 변하는 센서입니다. 즉, 주위에 있는 빛이 강할수록 저항 값이 작아지고, 어두울수록 저항 값이 커집니다. 포토다이오드의 저항값을 읽음으로써 주위 환경에 따라 LED 밝기를 조절할 수 있습니다. 본 키트에 제공되는 포토다이오드는 전형적인 빛 감지 센서입니다.

이번 프로젝트에서는 감광센서의 특성과 사용법을 익혀서, 주위가 밝기에 따라서 LED의 밝기가 자동으로 조절되는 장치를 만들어 봅시다.

필요한 부품



DFduino UNO R3 x1



Prototype Shield x1

Jumper Cables M/M x5

5MM LED x1

Ambient Light Sensor x1

Resistor 220R x1

Resistor 10K x1

회로 구성

아래 그림처럼 실험 회로를 구성하세요.

LED와 마찬가지로 포토다이오드도 극성을 갖고 있다는 점에 주의해야 합니다. 포토다이오드의 극성이 바뀌면 회로가 바르게 동작하지 않습니다. 220Ω 저항은 LED와 10KΩ 저항은 포토다이오드와 연결 해야 합니니.

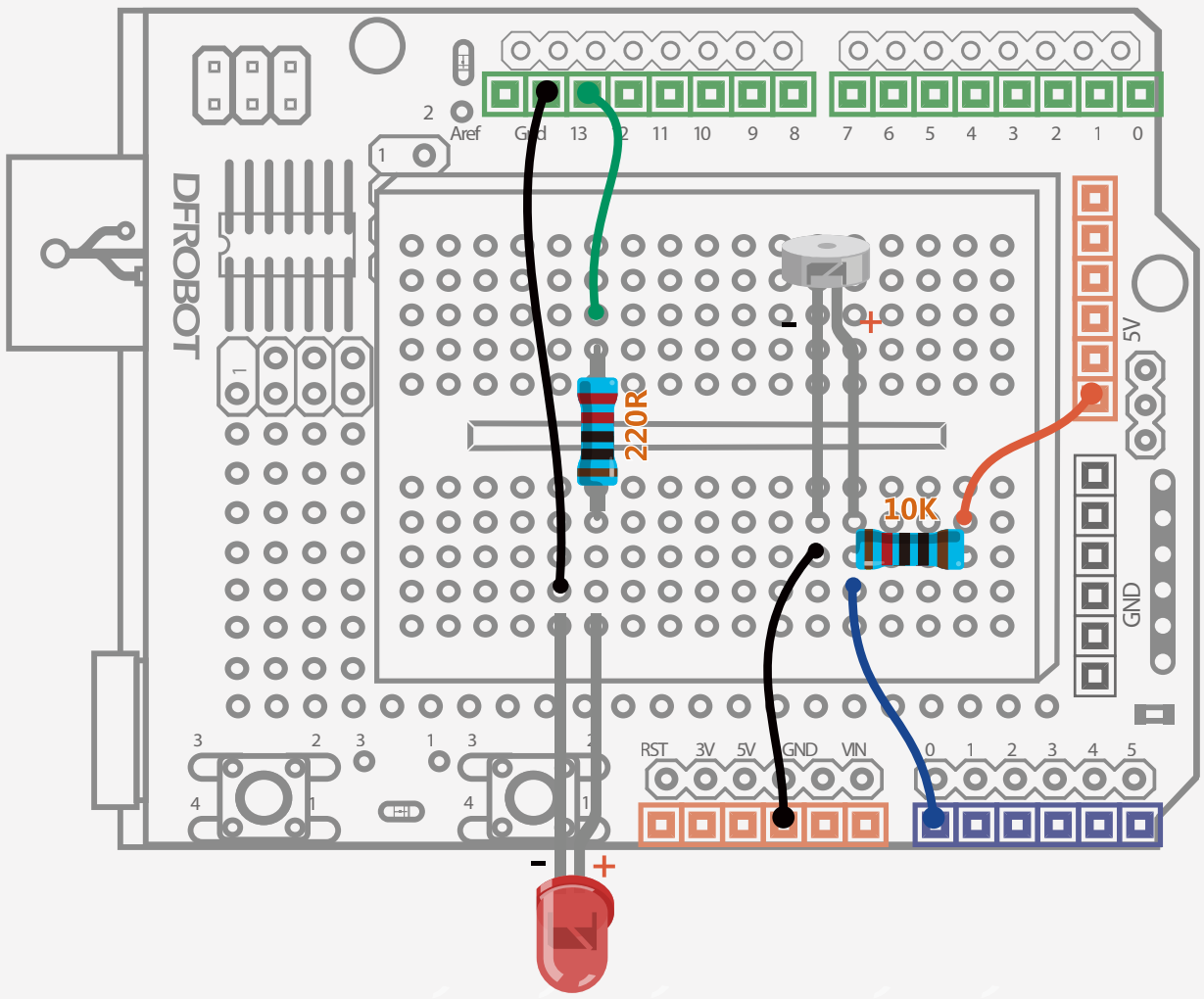


그림 9-1 빛 감지 LED 회로도

Arduino Sketch

프로그램 예제 9-1:

```
//예제9 빛감지 LED
int LED = 13;                    //LED를 D13으로 정의
int val = 0;                     //포토다이오드 전압 값을 D 0로 정의
void setup(){
    pinMode(LED,OUTPUT);         // LED(D13)를 출력으로 설정
    Serial.begin(9600);          // 9600 보율(baud rate)을
}

void loop(){
    val = analogRead(0);   // A0 측정값을 val에 저장 (범위: 0-1023)
    Serial.println(val);   // val을 시리얼모니터로 출력
    if(val<1000){          // 1000보다 작으면 LED 끔
        digitalWrite(LED,LOW);
    }else{                 // 1000보다 크거나 같으면
        digitalWrite(LED,HIGH);
    }
    delay(10);             // 10ms 대기
}
```

이 프로그램은, 포토다이오드가 어두운 곳에 위치하면 LED가 켜지고, 밝은 곳에 놓으면 LED가 꺼지는 동작을 반복 합니다.

Code 해설

LM35 온도센서와 마찬가지로, 포토다이오드는 아날로그 핀을 사용하기 때문에 pinMode 를 설정할 필요가 없습니다. 포토다이오드에서 측정되는 아날로그 전류는 0~1023의 디지털 값으로 측정됩니다. 이 측정값을 시리얼 모니터로 출력하고, 빛의 양이 변함에 따라서 어떻게 변하는지 관찰해 봅시다. 포토다이오드의 감도를 바꾸려면 어떻게 해야 하는지 생각해 봅시다.

Hardware 탐구

포토 다이오드

포토다이오드는 빛을 전류로 바꾸는 반도체 부품입니다. 빛(광자)이 포토다이오드에 흡수되면 전류가 발생하는 데, 주위의 빛이 강할수록 포토다이오드의 저항 값이 낮아집니다.

아두이노는 아날로그 핀을 이용하면 0~5V의 아날로그 전압을 0~1023의 디지털 값으로 바꿔 측정할 수 있습니다. 포토다이오드는 오른쪽 그림처럼 저항 2개로 구성되는 분압회로를 만들어 전압을 측정합니다.

이 회로에서 R1은 10k 고정저항을 사용하고, R2는 포토다이오드입니다. 직렬로 연결된 두 저항 R1과 R2의 양쪽 끝, 즉 Vin에 5V 전압 공급하고, R2 (포토다이오드) 양쪽의 전압을 아날로그핀으로 측정하면 됩니다.

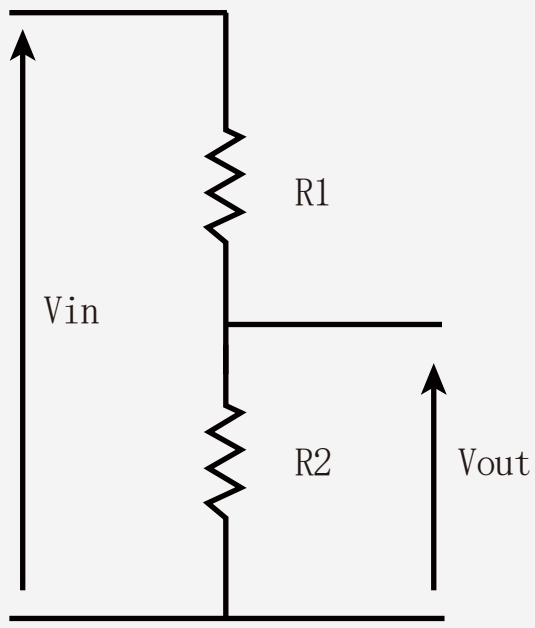


그림9-2 분압회로

오른쪽 아래의 관계식을 살펴보면, Vin과 R1은 고정값이기 때문에 R2의 저항값에 따라 측정값 Vout이 달라지는 것을 알 수 있습니다. 고정저항 R1은 1K부터 10K사이의 저항을 사용해야 합니다. 저항값이 이 범위를 벗어나면 포토다이오드의 변화를 측정하기가 어려워 집니다. R1으로 10K를 사용하는 것은 바로 이런 이유 때문입니다.

$$Vout = \frac{R2}{R1+R2} \times Vin$$

분압저항과 출력전압의 관계

Project 10

서보모터 제어법

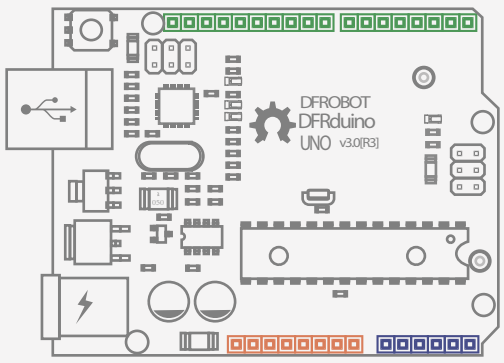
10

DFROBOT
DRIVE THE FUTURE

서보모터 제어법

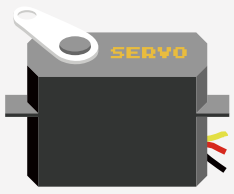
서보모터는 회전위치를 정확하게 제어할 수 있기 때문에 자동제어 장치에 사용하면 이상적입니다. 대부분의 서보모터는 최대 180도까지 회전시킬 수 있습니다만, 일부 대형 서보모터는 360도까지 회전 시킬 수 있는 것도 있습니다. 서보모터는 카메라의 초점 조절장치, 로봇 팔의 관절, 자동주행 자동차의 조향 장치 등에 폭넓게 사용할 수 있습니다.

필요한 부품



DFduino UNO R3 x1

Jumper Cables M/M x3



Servo x1

회로 구성

서보모터는 3개의 연결선이 있습니다. 일반적으로 빨간색 선은 +전원(5V)선이고, 검은색(또는 갈색)은 GND, 다른 한 선은 노란색 또는 주황색인데 콘트롤 신호선입니다. 아래 회로에서 콘트롤 신호선은 디지털 9번핀에 연결되어 있습니다.

아래 그림처럼 실험할 회로를 구성해 보시오.

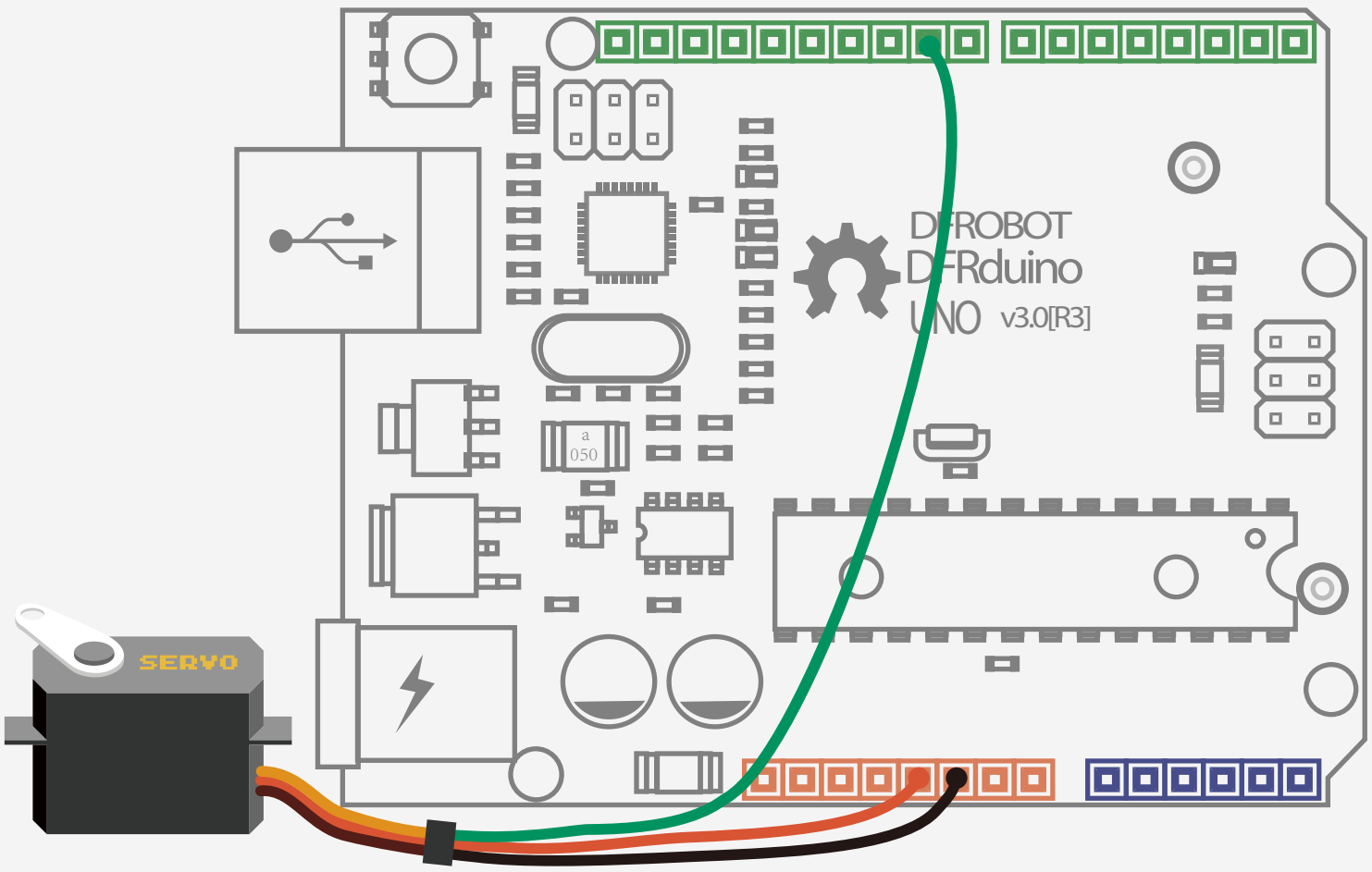


그림 10-1 서보모터 회로도

Arduino Sketch

프로그램 예제 10-1

```
//예제-10 서보모터 제어
#include <Servo.h>          //<Servo.h>라이브러리 포함
Servo myservo;              //오브젝트명 myservo 생성
int pos = 0;                //각도 저장
void setup() {
   myservo.attach(9);       //서보모터를 D9핀에 부착
}

void loop() {
    for(pos = 0; pos < 180; pos += 1){      //180 보다 작으면 1씩 커짐
        myservo.write(pos);  //각도를 서보로 전송
        delay(15);           //서보동작중 대기
}
   for(pos = 180; pos>=1; pos-=1) {         // 1보다 크면 1씩 작아 짐
        myservo.write(pos);  //각도를 서보로 전송
        delay(15);           //서보동작중 대기
  }
}
```

이 프로그램을 실행하면, 서보모터가 0도부터 180도까지 시계방향과 반시계방향으로 반복해서 움직입니다.

Code 해설

이 프로그램을 사용하려면 프로그램의 첫 머리에 아래와 같이 <Servo.h> 라이브러리를 추가하고, 이 라이브러리를

```
myservo.attach(9):
```

사용함을 표시하는 이름(object 명)을 만들어 줘야 합니다. 먼저, 다음과 같이 제어 대상인 서보모터의 이름을 붙여 줍니다.

```
#include <Servo.h>
//Servo.h 라이브러리 포함
Servo myservo;
//오브젝트명은 myservo
```

라이브러리는 아두이노에 새로운 기능을 추가하기 쉽도록 만들어진 일종의 블록형 프로그램입니다. 아두이노 IDE에는 Servo 라이브러리 외에도 많은 라이브러리가 마련되어 있습니다. 이들 라이브러리를 사용하면 아두이노 보드의 기능을 향상시켜 더 넓은 분야에 활용할 수 있게 됩니다.

Setup()함수를 살펴보면 아래와 같이 또 다른 코드가 있습니다.

attach()함수는 Servo.h 라이브러리에서 제공하는 함수입니다. 따라서, 이 함수를 사용하려면, attach()라는 함수명 앞에 오브젝트명(myservo)을 반드시 붙여주어야 합니다. 그렇지 않으면, Arduino IDE에서 이 함수를 인식하지 못하기 때문에 컴파일 되지않습니다. 오젝트명을 붙일 때는 "오브젝트명"과 "attach"사이에 "."을 꼭 붙여줘야 합니다.

```
Attach(pin);
```

attach(); 함수는 서보에 사용할 디지털 핀을 예약하는 기능을 합니다. 서보모터는 시리얼포트로 사용하는 D0,D1을 제외한 모든 디지털 핀과 연결해서 사용할 수 있습니다.

메인 프로그램인 loop()함수는 크게 2부분으로 나뉘어집니다.첫째 부분은 서보의 회전각을 0도부터 180도까지 1도 간격으로 증가시키는 명령문이고, 뒷 부분은 180도에서 0도까지 1도 간격으로 감소시는 명령문입니다.

```
myservo.write(pos);
```

서보모터를 회전시키려면 다음과 같이 사용해야 합니다.
여기서 myservo는 object명, pos는 각도입니다.

만일 servo 라이브러리안에 정의된 함수들에 대하여 더 자세히 알고 싶다면, 아두이노

http://arduino.cc/en/reference/servo

또는 DFRobot사의 웹사이트를 방문해 보세요.

www.dfrobot.com

Project 11

정밀제어 서보모터

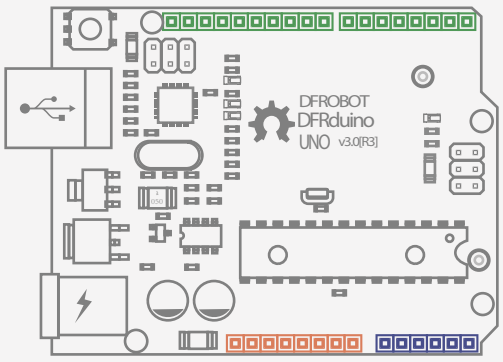
11

DFROBOT
DRIVE THE FUTURE

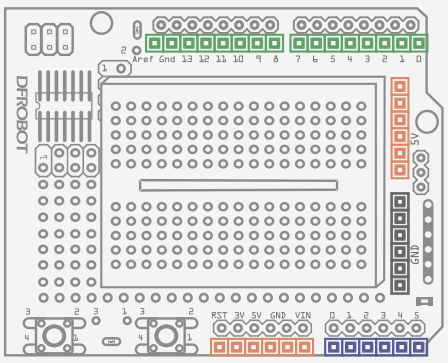
정밀제어 서보모터

이전 프로젝트를 통해서 서보모터를 제어하는 방법을 알아 봤습니다. 이번 프로젝트는 가변저항의 회전각도와 비례하여 서보 모터의 회전각도를 조절하는 방법을 학습합니다. 이 프로젝트는 창의적인 작품을 만드는데 널리 활용할 수 있습니다. 예를 들면, 가변저항으로 조절되는 서보 모터를 여러 개 사용하면 위치를 자유자재로 조절하는 로봇 팔을 만들 수도 있습니다.

필요한 부품



DFduino UNO R3 x1

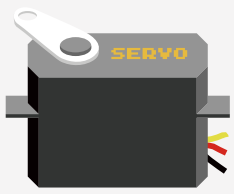


Prototype Shield x1

Jumper Cables M/M x3

Jumper Cables F/M x3

10K Potentiometer x1



Servo x1

회로 구성

아래 그림대로 회로를 구성해 봅시다.
이 프로젝트에서는 포텐셔미터라고도 불리는 가변저항을 처음 사용합니다. 가변저항은 3개의 연결핀이 있는데 손잡이를 돌리면 저항 값이 바뀌는 전자부품입니다. 가변저항의 양쪽 끝에 위치한 두 핀은 아두이노 보드의 5V와 GND에 연결하고, 가운데 있는 핀은 아날로그핀(A0)과 연결하여 사용합니다.

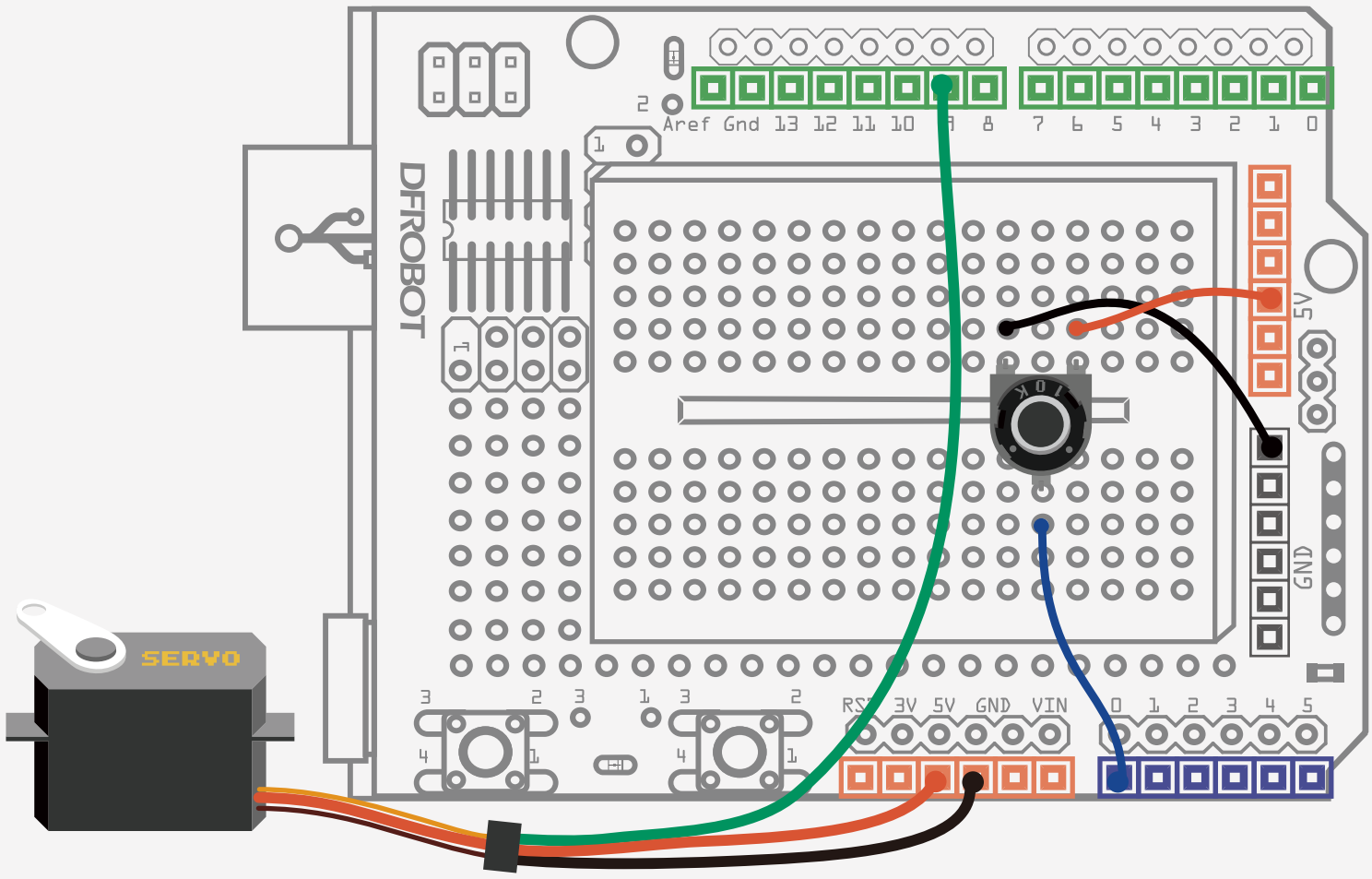


그림 11-1 정밀제어 서보모터 회로도

Arduino Sketch

프로그램 예제 11-1

```
//예제-11 정밀제어 서보

#include <Servo.h>        // 라이브러리 추가
Servo myservo;            // 오브젝트명은 myservo

int potpin = 0;           // 가변저항 가운데 핀은 A0핀
int val;                  //각도저장

void setup() {
  myservo.attach(9);      //서보모터 D9핀
}

void loop() {
  val = analogRead(potpin);          //가변저항의 전압 측정(0~1023)
  val = map(val, 0, 1023, 0, 179);   //전압을 각도(0~179)로 변환
  myservo.write(val);                // 각도를 서보로 전송
  delay(15);                         // 서보회전 중 대기

}
```

프로그램을 실행한 뒤, 가변저항을 돌리면, 움직인 각도에 비례해서 서모모터가 움직입니다.

앞의 프로젝트와 마찬가지로 프로그램의 첫 부분에서 <Servo.h> 라이브러리를 포함시키고, 변수를 선언한 후 가변저항을 연결할 아날로그핀 번호(A0)를 저장합니다.

이 프로그램은 "map" 함수를 사용합니다. 형식은 아래와 같습니다.

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

"map" 함수는 어느 특정범위에 있는 수를 다른 범위의 수로 변환하는 기능을 합니다. 이 함수는 5개의 파라미터를 정의하고 사용합니다.

Parameters
value: 변환할 수
fromLow: 변환전 범위(하한값)
fromHigh: 변환전 범위(상한값)
toLow: 변환후 범위(하한값)
toHigh: 변환후 범위 (상한값)

map()함수를 사용할 때 알아야 할 점은, 하한 값이 상한 값보다 항상 작아야 하는 것은 아닙니다. 즉, 상한 값보다 하한 값이 더 큰 경우에도 함수가 바르게 실행됩니다. 예를 들면, 아래와 같이 사용할 수도 있습니다.

```
y = map(x, 1, 50, 50, 1);
```

또한, map()함수에는 음수를 사용할 수도 있습니다. 따라서, 아래와 같이 사용해도 문제없이 잘 실행됩니다.

```
y = map(x, 1, 50, 50, -100);
```

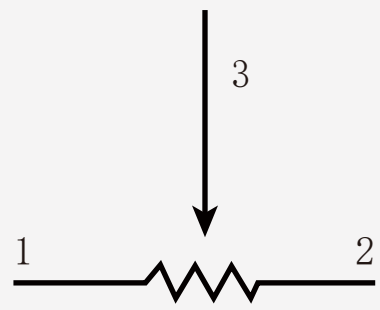
스케치에서는 아날로그핀의 0부터 1023의 값(val)을, 서보 모터의 0도부터 180도 사이의 값으로 바꾸기 위해서 아래와 같이 map()함수를 사용했습니다.

```
val = map(val, 0, 1023, 0, 179);
```

Hardware 탐구

가변 저항기

10k 가변저항은 0 ~ 10k범위 안에서 저항 값을 변경할 수 있는 전자부품입니다. 가변저항의 3개의 핀 중 양쪽 끝에 있는 두 핀 사이의 저항은 항상 10k 로 일정하지만, 가운데 핀과 양쪽 핀 사이의 저항 값은 손잡이의 움직임에 따라서 달라 집니다.



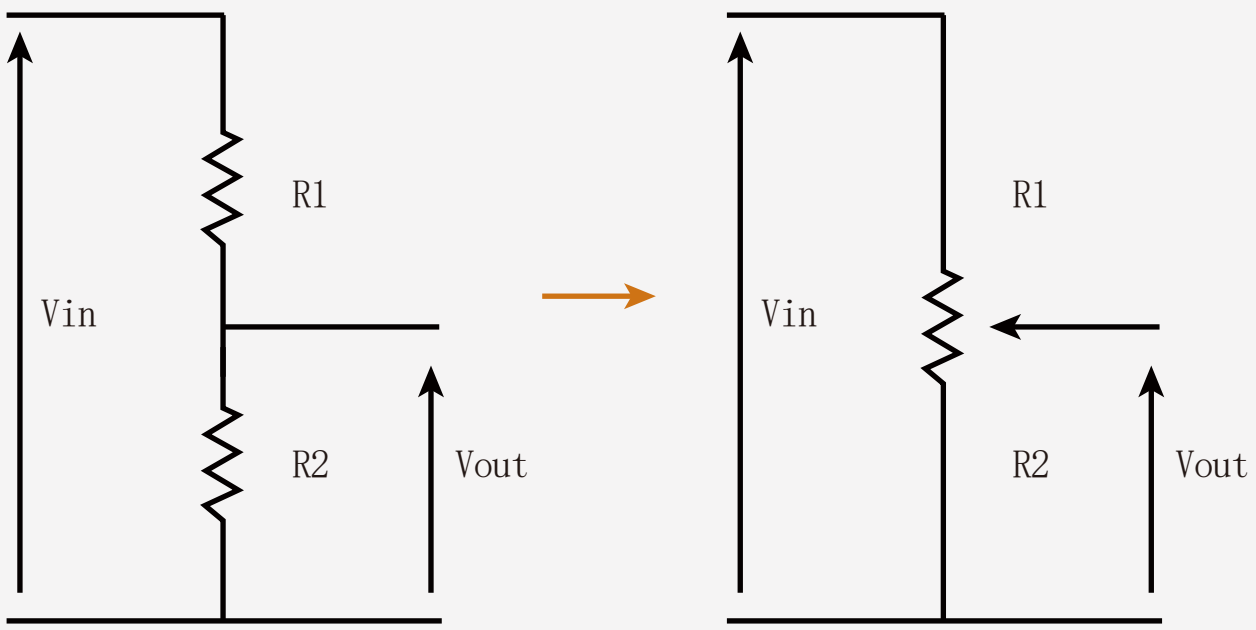
이 프로젝트에서는 가변저항의 양쪽 핀 중 하나를 5V에, 다른 하나는 GND에 연결합니다. 가변저항의 가운데 핀은 아두이노 보드의 아날로그핀 A0와 연결합니다.

가변저항은 프로젝트-9에서 사용한 분압 회로와 동일한 원리로 작동합니다.

즉, 가변저항의 손잡이를 돌리면, 손잡이의 회전 위치에 따라서 아날로그핀에 입력되는 전압이 0V ~ 5V 범위에서 바뀌게 됩니다.

만일, 가변저항기의 손잡이를 0V쪽으로 끝까지 돌리면 아날로그 입력전압은 0V가 되고, 반대로 5V쪽으로 끝까지 돌리면 5V가 됩니다.

바꿔 말하면, 손잡이를 GND쪽으로 끝까지 돌리면 analogRead() 함수로 읽은 값이 0이 되며, 손잡이를 5V쪽으로 끝까지 돌리면 analogRead()함수로 읽은 값이 1023이 됩니다. 물론, 손잡이를 중간위치에 놓으면, 0과 1023 사이의 중간 값이 됩니다.



Project 14

리모콘 제어 LED

14

DFROBOT
DRIVE THE FUTURE

리모콘 제어 LED

리모콘 송신기의 키를 누르면 고유한 16진수의 코드를 만들어 적외선 신호로 변환하여 송신합니다. 이 신호를 수신하는 장치가 리모콘 적외선 수신기 입니다.

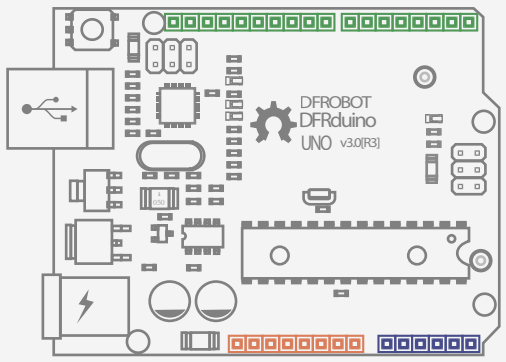
리모콘에 사용하는 적외선은 빛이기 때문에 원칙적으로 장애물이 없는 가까운 직선 거리에서만 사용할 수 있습니다. 하지만, 유리나 벽에 부딪히면 반사를 하기 때문에 직선거리가 아닌 곳에서도 리모콘 신호를 주고 받을 수 있는 경우도 있습니다.

적외선 리모콘 수신기의 위치가 좋지 않은 곳에 위치하면, 적외선 신호가 약해져서 리모콘 신호를 잘 수신할 수 없게 됩니다. 이런 현상을 터널비전이라고 부릅니다.

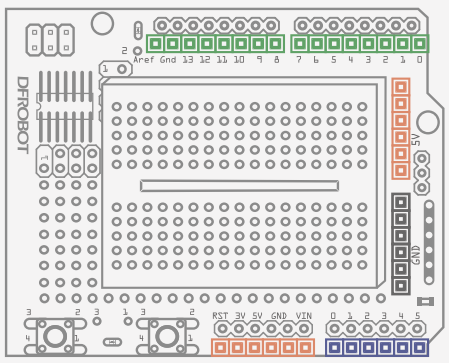
이 프로젝트는 2개의 실습과정으로 구성되어 있습니다.

실습1 : 리모컨 기본 동작실험

필요한 부품

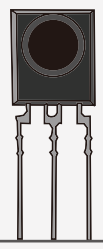


DFduino UNO R3 x1



Prototype Shield x1

Jumper Cables M/M x3



IR Receiver x1

IR Remote controller x1

회로 구성

아래 그림처럼 실험회로를 구성하시오.

리모콘 적외선 수신기의 V-out핀은 반드시 아두이노의 디지털 11번핀과 연결해야 합니다. GND 와 5V를 바꿔 연결하면 부품이 고장날 수도 있으므로 주의해야 합니다.

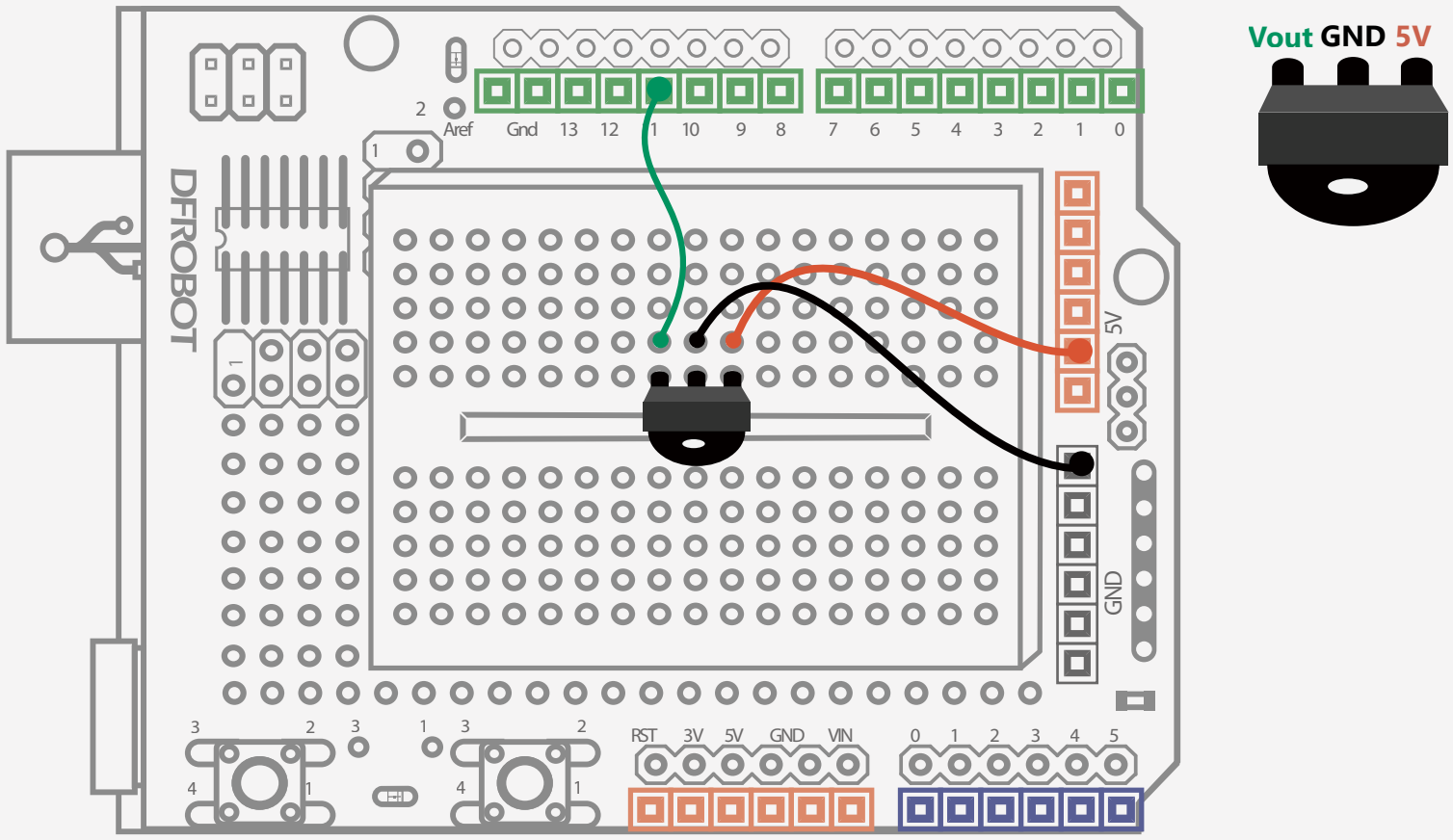


그림 14-1 리모콘 기본동작 실험회로

Arduino Sketch - 1

먼저, "IRremote" 라이브러리를 설치해야 합니다.

인터넷에서 라이브러리 파일을 다운로드 받은 다음, 아두이노 라이브러리 디렉토리 안에 압축을 풀어 저장하면 라이브러리 설치가 완료됩니다. IRremote 라이브러리 폴더안에 있는 "IRrecvDemo"란 예제 스케치를 Arduino IDE로 불러와 사용합니다.

라이브러리 설치방법을 잘 모르면 Project 5과에 있의 관련내용을 복습하세요.

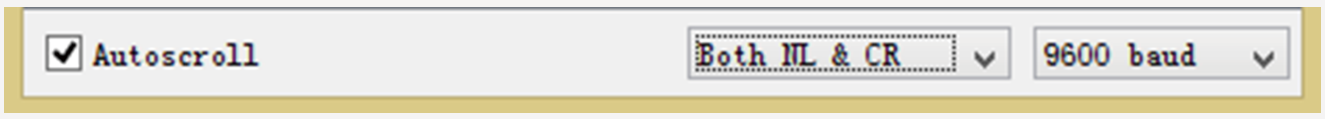
이 스케치는 IRremote 라이브러리 안에 포함된 예제파일인 "IRrecvDemo" 를 인용한 겁니다

프로그램 예제 14-1:

```
//프로젝트 14-1 리모콘 기본동작 실험
#include <IRremote.h>        //라이브러리 포함
int RECV_PIN = 11;           //리모콘 수신기 D11
IRrecv irrecv(RECV_PIN);     //D11을 리모콘 수신핀으로 정의
decode_results results;      //리모콘 수신코드 저장

void setup(){
  Serial.begin(9600);        //시리얼포트 사용
  irrecv.enableIRIn();       //리모콘 장치 초기화
}

void loop() {
    //리모콘 수신데이타가 있을시, 그 값을 변수에 저장
    if (irrecv.decode(&results)) {
        //수신한 데이터 시리얼모니터에 표시
        Serial.println(results.value, HEX);
        irrecv.resume();  // 처음부터 다시 시작
  }
}
```

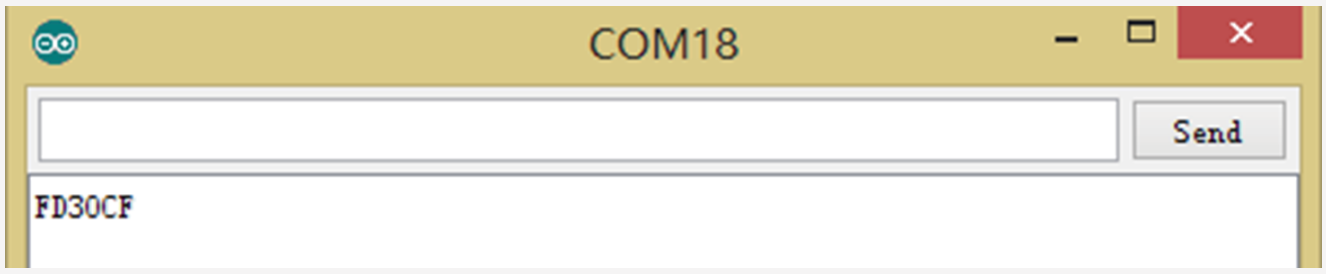
시리얼모니터 열고 수신데이타 확인

1. 시리얼모니터 열기
스케치를 업로드한 다음 아두이노 IDE의 시리얼모니터를 열고, baud rate를 9600으로 설정합니다. 예제 프로그램에서 Serial.begin(9600)함수를 사용하여 baud rate를 9600으로 설정했기 때문입니다.

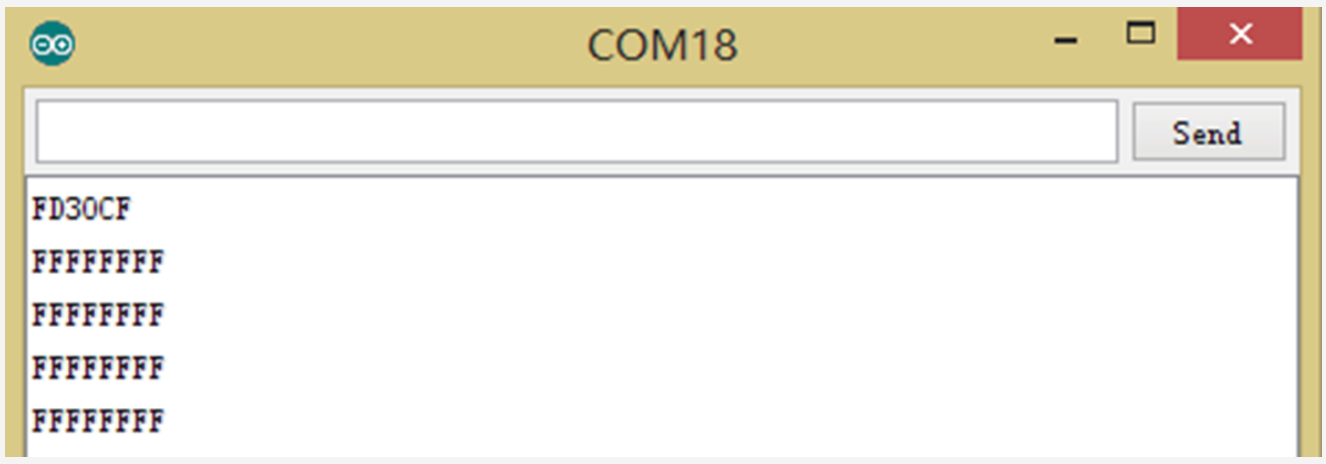


2. 리모콘 신호 송신
그 다음, 리모콘을 IR수신기쪽을 향하게 한 뒤, 리모콘의 스위치를 누릅니다. 스위치를 누를 때마다 그 스위치에 해당하는 16진수 코드가 IR수신기를 향해 송신 됩니다.

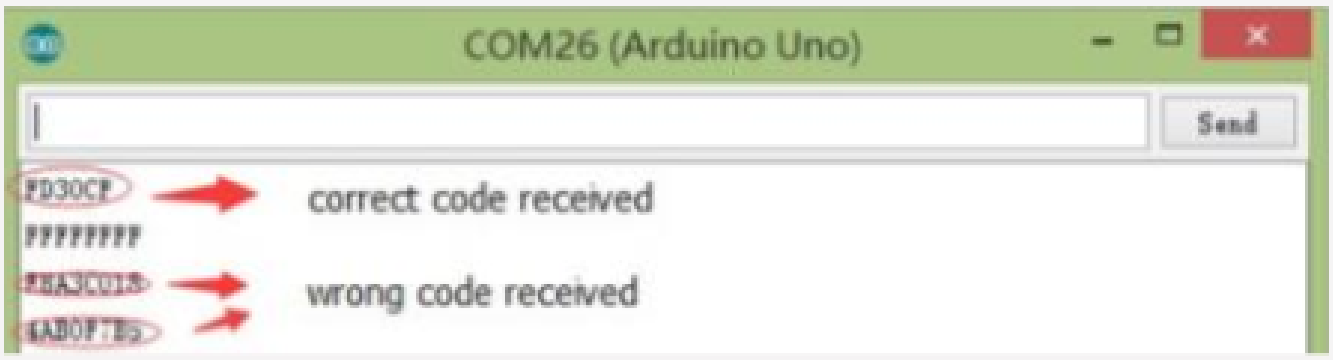
3. 리모콘 신호 수신 확인
아두이노는 이 신호를 해석하여 16진수 데이터로 바꿉니다. 예를 들어, 리모컨의 0번 키를 누르면 16진수 FD30CF가 시리얼모니터에 표시됩니다.



만일, 리모콘의 어떤 키를 계속누르고 있을 경우는, 시리얼 모니터에 "FFFFFFFF" 가 표시됩니다.



리모콘 코드가 바르게 수신될 경우 시리얼모니터에 FD로 시작하는 6자리의 16진수가 시리얼 모니터에 표시됩니다. 만일, 리모콘이 IR수신기를 향해 아무 신호도 송신하지 않으면 아래와 같은 오류코드가 표시됩니다.

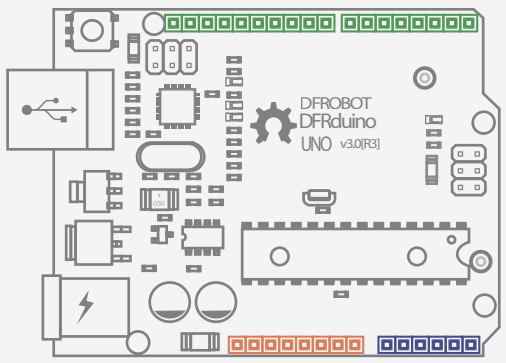


원래, 리모콘 신호를 송신하고 해석하는 프로그램은 매우 복잡하기 때문에 이 원리를 확실히 이해하는 사람이 작성 후 공유한 라이브러리를 사용하면, 프로그램 과정을 잘 알지 못하는 사람도 리모콘을 사용하는 프로그램을 작성할 수 있게 됩니다.

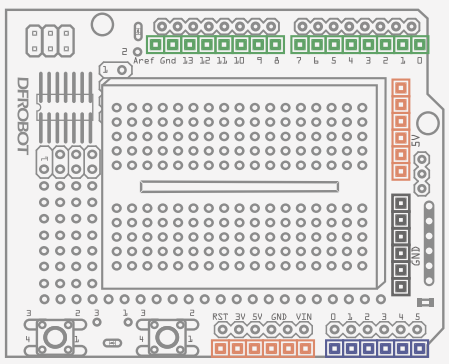
자, 이제 리모콘을 사용법을 알았으니, 이를 실제로 활용해 봅시다.

리모콘 제어 LED

필요한 부품

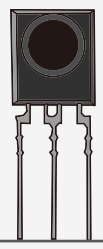


DFduino UNO R3 x1



Prototype Shield x1

Jumper Cables M/M x3



IR Receiver x1

IR Remote controller x1

5MM LED x1

Resistor 220R x1

회로 구성

리모콘 수신기가 잘 작동되는지 시리얼모니터로 확인되었으면, 아래 그림 처럼 저항과 LED를 10번 디지털 핀에 연결해 줍니다. 리모콘 수신기의 V-out핀이 D11 핀에 연결되어 있는지도 다시 한번 확인해 보세요.

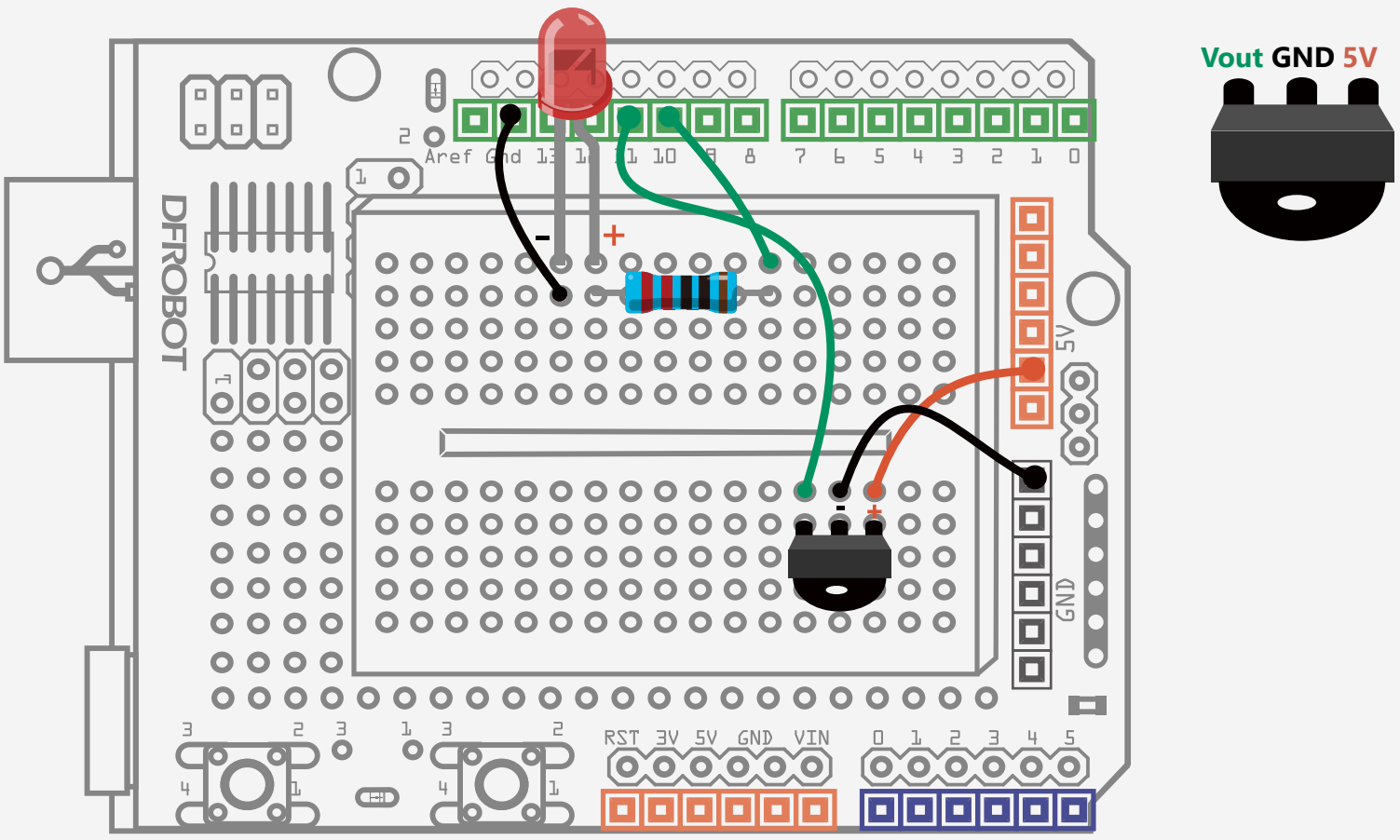


그림 14-2 리모콘 제어 LED 회로

프로그램 예제 14-2

```
#include <IRremote.h>
int RECV_PIN = 11;
int ledPin = 10;                    // LED D10핀
boolean ledState = LOW;             // LED상태저장
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup(){
  Serial.begin(9600);
  irrecv.enableIRIn();
  pinMode(ledPin,OUTPUT);           // LED핀 OUTPUT
}

void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);

    //리모콘 송신기의 전원키를 누르면 LED가 켜졌다 꺼졌다 반복
    if(results.value == 0xFD00FF){
        ledState = !ledState;           //상태 바꿈
        digitalWrite(ledPin,ledState);  //LED켜고 끄기
     }
    irrecv.resume();
  }
}
```

Code 해설

프로그램의 첫 부분은 실험1과 동일합니다.

```
#include <IRremote.h>        //라이브러리 추가
int RECV_PIN = 11;           //RECV_PIN을 11로 설정
IRrecv irrecv(RECV_PIN);     //RECV_PIN 를 IR수신핀으로 설정
decode_results results;      //해석된 코드를 저장할 변수 results 를 선언
int ledPin = 10;             //LED – digital 10
boolean ledState = LOW;      //LED의 on/off 상태를 저장할 변수 ledstate를 선언
```

setup() 함수에서는, 시리얼포트 설정, 리모콘 장치 초기화, 핀모드 설정등을 합니다.

Loop() 함수에서는, 수신된 리모콘 신호가 이미 저장된 값중 어느 것과 일치하는지를 판단하는데, 다음의 과정을 거칩니다.

먼저, 리모컨 수신기가 수신한 데이터가 있는지 다음과 같이 판단을 합니다.

```
if (irrecv.decode(&results))
```

만일, 아두이노가 리모콘 코드를 수신하였을 경우, 아래의 2가지 작업을 합니다.

첫째로, 아래와 같이 수신된 코드가 "전원"키를 눌러서 발생한 16진수 코드(FD00FF)인가를 확인합니다.

```
if(results.value == 0xFD00FF)
```

둘째로, 만일, 전원키를 눌러서 발생한 코드이면 LED의 상태를 바꿉니다.

```
ledState = !ledState;            //상태값 바꿈
digitalWrite(ledPin,ledState);   //싱태값에 따라 LED켜고 끔
```

위 코드에서 "!" 는 논리 부정을 의미합니다.
즉, !ledState는 ledState이 아닌 상태를 의미합니다. 여기서 ledState는 0과1의 2가지 논리 상태만 저장할 수 있는 boolean 형 변수입니다. 따라서 ledState = !ledState 를 계속 반복하면 ledState의 값이 0과 1로 번갈아 가며 바뀌게 됩니다.
위의 2가지 작업을 마치면, 아두이노는 다음 신호 리모콘 신호를 기다립니다.

```
irrecv.resume();
```

위 코드 실행후 loop()함수의 첫 부분으로 되 돌아갑니다.
즉, 리모콘 신호를 수신하면 다시 처음부터 프로그램을 반복 실행하게 됩니다.

연습 문제

1. 이번 프로젝트와 전동기 프로젝트를 합쳐보세요. LED와 전동기를 모두 제어하는 함수를 만들어 보세요.

2. 리모콘을 사용하는 여러분만의 작품을 만들어보세요. 예를 들면, 리모콘 신호로 움직이는 조그만 인형이나 장난감을 만들어 보세요.

Project 15

리모컨 숫자 표시모듈

15

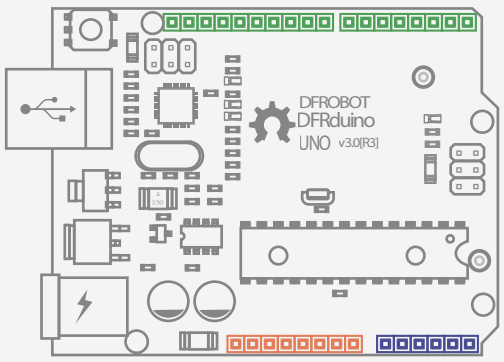
DFROBOT
DRIVE THE FUTURE

리모컨 숫자표시 모듈

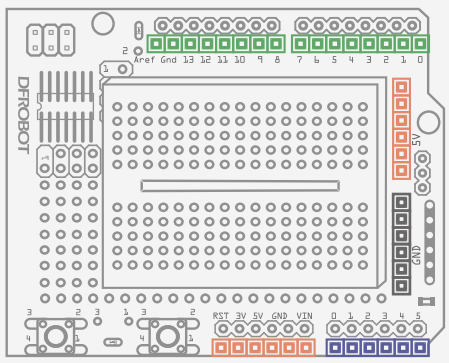
8개의 LED를 이용해서 한 자리 숫자(소수점 포함)를 표시하는 전자부품을 "8세그먼트 모듈"이라고 합니다. 이 부품의 8개의 조각LED를 아두이노의 디지털 핀으로 제어하면 원하는 숫자를 마음대로 표시할 수 있습니다.

이번 프로젝트에서는 "8세그먼트 모듈"을 사용하는 방법을 익힙니다.

필요한 부품



DFduino UNO R3 x1



Prototype Shield x1

Jumper Cables M/M x10

Resistor 220R x8

8-Segment LED x1

회로 구성

8세그멘트 LED에는 10개의 핀이 있습니다. 10개의 핀 중 2개는 공통 +극 (Anode)이고, 나머지 8핀은 8개의 조각LED의 -극(Cathode)입니다. 따라서, 공동전극은 VCC에 연결하고, 나머지 8개의 핀은 아두이노의 디지털핀 D2부터 D9까지에 연결하는데, 이 때 220Ω 저항을 직렬로 함께 연결하여 LED에 과도한 전류가 흐르지 않도록 해야 합니다.

디지털 핀에서 LOW신호를 출력하면 LED가 켜집니다.

아래 그림대로 실험회로를 완성 하시오.

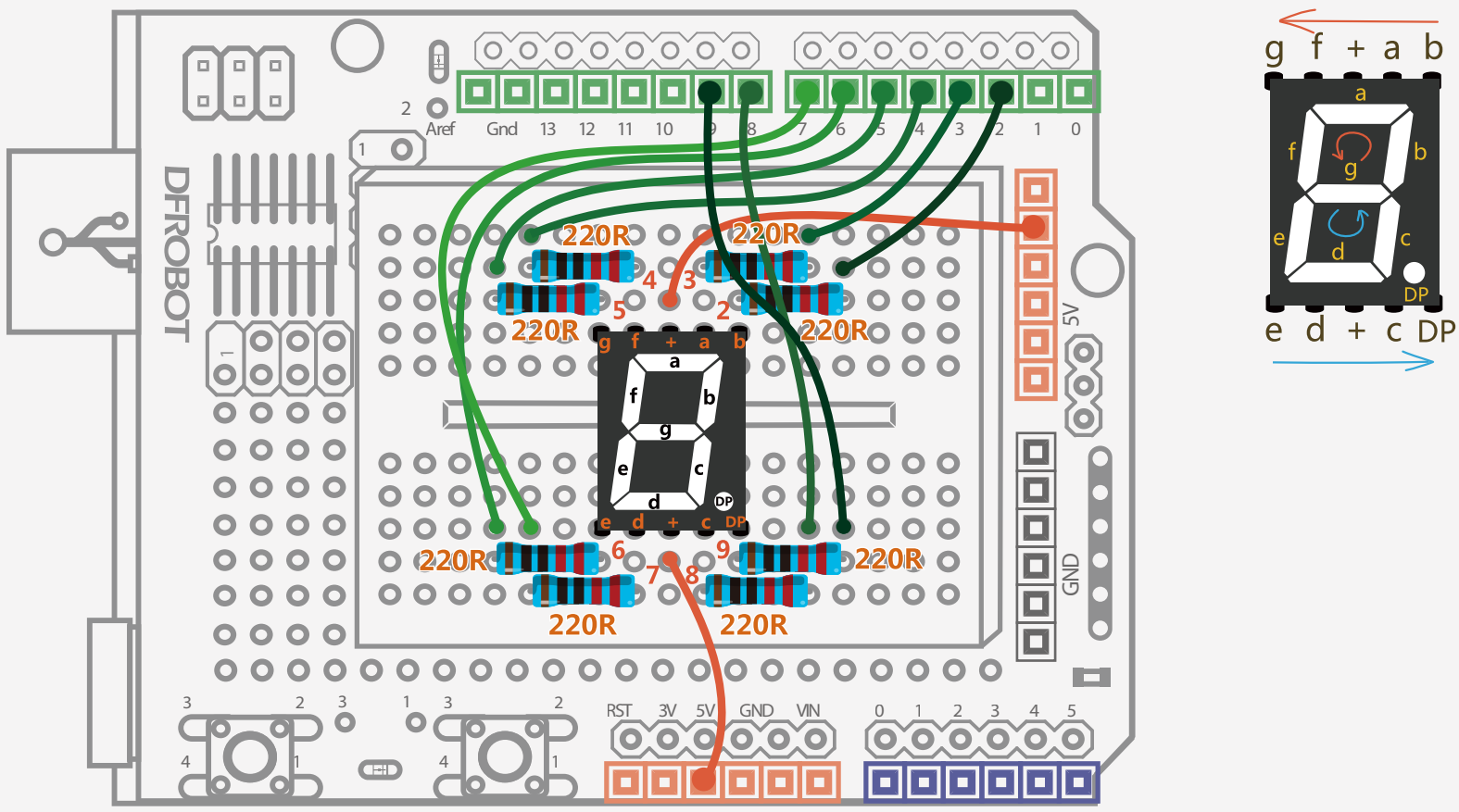


그림 15-1 8-세그먼트 LED모듈 회로도

Arduino Sketch

프로그램 예제 15-1:

```
//예제 15-1 8-세그먼트 LED 모듈
void setup(){
  for(int pin = 2 ; pin <= 9 ; pin++){       // D2 ~ D9 출력
      pinMode(pin, OUTPUT);
      digitalWrite(pin, HIGH);
  }
}

void loop() {
    // 숫자 0 데이터
    int n0[8]={0,0,0,1,0,0,0,1};
    //  숫자 0 출력
    for(int pin = 2; pin <= 9 ; pin++){
       digitalWrite(pin,n0[pin-2]);
    }
    delay(500);

    // 숫자 1 데이터
    int n1[8]={0,1,1,1,1,1,0,1};
    // 숫자 1 출력
    for(int pin = 2; pin <= 9 ; pin++){
       digitalWrite(pin,n1[pin-2]);
    }
    delay(500);

    // 숫자 2 데이터
    int n2[8]={0,0,1,0,0,0,1,1};
    // 숫자 2 출력
    for(int pin = 2; pin <= 9 ; pin++){
       digitalWrite(pin,n2[pin-2]);
    }
delay(500);

    // 숫자 3 데이터
    int n3[8]={0,0,1,0,1,0,0,1};
    // 숫자 3 출력
    for(int pin = 2; pin <= 9 ; pin++){
       digitalWrite(pin,n3[pin-2]);
    }
    delay(500);

    // 숫자 4 데이터
    int n4[8]={0,1,0,0,1,1,0,1};
    // 숫자 4 출력
```

```
  for(int pin = 2; pin <= 9 ; pin++){
    digitalWrite(pin,n4[pin-2]);
  }
  delay(500);

  // 숫자 5 데이터
  int n5[8]={1,0,0,0,1,0,0,1};
  // 숫자 5 출력
  for(int pin = 2; pin <= 9 ; pin++){
    digitalWrite(pin,n5[pin-2]);
  }
  delay(500);

  // 숫자 6 데이터
  int n6[8]={1,0,0,0,0,0,0,1};
  // 숫자 6 출력
  for(int pin = 2; pin <= 9 ; pin++){
    digitalWrite(pin,n6[pin-2]);
  }
  delay(500);

  //  숫자 7 데이터
  int n7[8]={0,0,1,1,1,1,0,1};
  // 숫자 7 출력
  for(int pin = 2; pin <= 9 ; pin++){
    digitalWrite(pin,n7[pin-2]);
  }
  delay(500);

  // 숫자 8 데이터
  int n8[8]={0,0,0,0,0,0,0,1};
  // 숫자 8 출력
  for(int pin = 2; pin <= 9 ; pin++){
    digitalWrite(pin,n8[pin-2]);
  }
  delay(500);

  // 숫자 9 데이터
  int n9[8]={0,0,0,0,1,1,0,1};
  // 숫자 9 출력
  for(int pin = 2; pin <= 9 ; pin++){
    digitalWrite(pin,n9[pin-2]);
  }
  delay(500);
}
```

이 프로그램을 실행하면 0부터 9까지의 숫자가 8세그먼트 표시 장치 위에 0.5초 간격으로 반복 표시됩니다.

회로 구성

8 세그먼트 모듈

8세그먼트 모듈은 a,b,c,d,e,f,g 핀과 DP핀, 그리고 2개의 +극(공통전극)으로 구성되어 있습니다. 공동전극은 VCC에 연결하고 나머지 핀은 전류제한 저항을 통해서 아두이노의 디지털 핀과 연결해야 합니다.

주의해야 할 점은, 디지털 핀을 연결하는 순서가 a,b,c,d,e,f,g,DP 가 아니라 b,a,f,g,e,d,c,DP순으로 연결 한다는 점입니다.

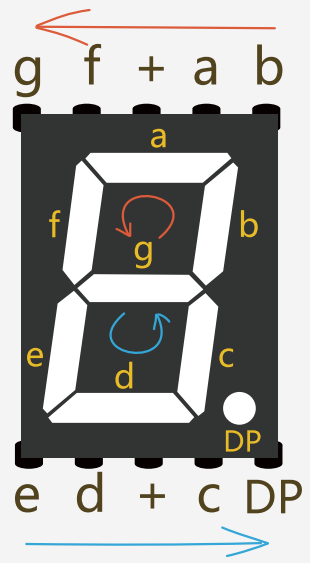
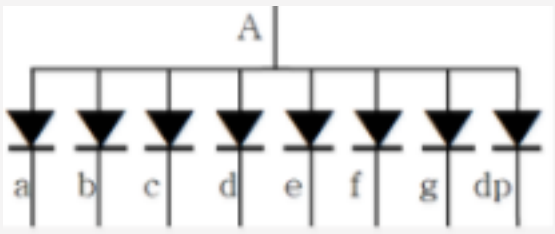
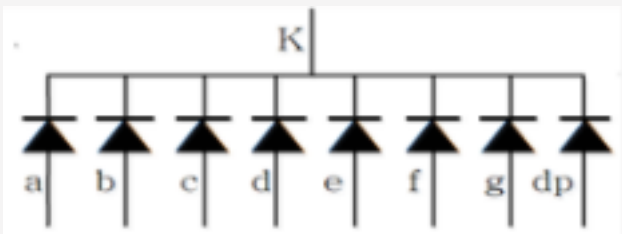


그림 15-2 8세그먼트 모듈의 핀배열

커먼 캐소드형과 커먼 애노드형의 비교

구 분	공통양극(커먼 애노드)	공통음극(커먼 캐소드)
공통전극	VCC에 연결	GND에 연결
a,b,c,d,e,f,g,DP 핀	전류제한용 직렬저항 연결 후 디지털 핀과 연결	
점 등	LOW신호	HIGH신호
회 로		

Code 해설

8세그먼트 모듈은 아두이노의 디지털핀 8개를 사용하는데, 이 핀들은 모두 OUTPUT으로 설정하고 사용해야 합니다.

아래 코드는 for반복문을 사용해서 8개의 디지털핀(d2~d9)을 output으로 설정하고, HIGH신호를 주어 LED를 모두 꺼주는 초기화 과정입니다.

```
for(int pin = 2 ; pin <= 9 ; pin++){
    pinMode(pin, OUTPUT);
    digitalWrite(pin, HIGH);
}
```

이 프로젝트에서는 컴먼 에노드 방식의 8세그먼트 모듈을 사용합니다. 따라서, 각 세그먼트 핀에 LOW 신호를 출력하면 불이 켜지고, HIGH신호를 출력하면 꺼집니다.

예를 들어, 숫자 0을 표시하려면 아래 그림과 같이 b,g,f,a,b,c,d,e,f 핀에는 LOW신호를 주어 켜고, g와 DP 핀에는 HIGH신호를 주어 꺼야 합니다 .

숫자0을 표시하는 프로그램은 다음과 같습니다.

```
digitalWrite(2,LOW);  //b핀 on
digitalWrite(3,LOW);  //a핀 on
digitalWrite(4,LOW);  //f핀on
digitalWrite(5,LOW);  //g핀 off
digitalWrite(6,LOW);  //e핀 on
digitalWrite(7,LOW);  //d핀 on
digitalWrite(8,LOW);  //c핀 on
digitalWrite(9,LOW);  //DP핀 off
```

그런데, 위와 같이 작성하면 코드가 길어진다는 단점이 있습니다. 좀 더 간단한 방법은 없을까요?

배열변수와 for 반복문을 사용하면 됩니다.

같은 이름을 사용하는 여러 변수들의 집합을 배열변수라고 하는데, 배열변수는 [ ]안에 숫자로 각각 변수를 구분합니다.

예제프로그램에는 아래와 같이 배열변수를 선언합니다.

```
int n0[8]={0,0,0,1,0,0,0,1};
```

위 코드에서 int n0[8] 은 n0라는 같은 이름을 사용하는 int형 변수 8개를 선언하라는 의미이고, ={0,0,0,1,0,0,0,1}; 은 8개의 배열변수에 저장할 데이터들입니다.

즉, 이 코드는 아래와 같은 의미입니다.

```
n0 [0] =0;
n0 [1] =0;
n0 [2] =0;
n0 [3] =1;
n0 [4] =0;
n0 [5] =0;
n0 [6] =0;
n0 [7] =1;
```

예제 코드를 살펴보면 아래와 비슷한 코드가 반복됩니다.

어떤 의미인지 알아봅시다.

```
int n0[8]={0,0,0,1,0,0,0,1};
for(int pin=2; pin<=9; pin++){
digitalWrite(pin,[pin-2]);
}
```

위 코드는 숫자 0을 표시하는 코드입니다.

그러면, 다음은 어떤 숫자를 표시하는 코드일까요?

```
int n0[8]={0,0,1,1,1,1,0,1};
for(int pin=2; pin<=9; pin++){
digitalWrite(pin,[pin-2]);
}
```

그럼, 예제 코드를 다시 자세히 살펴봅시다. 이젠 모두 이해할 수 있을 겁니다.

앞에서, 배열변수를 사용해서 8세그먼트 모듈에 숫자 0 ~ 9를 반복하여 표시하는 프로그램을 학습했습니다. 배열변수를 사용한다고는 하지만, 여전히 프로그램의 길이가 짧지 않습니다. 더 간단하게 하는 방법은 없을까요? 앞의 프로그램에서는 0~9 를 표시하기 위해서 1차원 배열변수를 10개를 사용했습니다. 2차원 배열변수를 사용할 경우 프로그램을 훨씬 더 간단하게 할 수 있습니다.

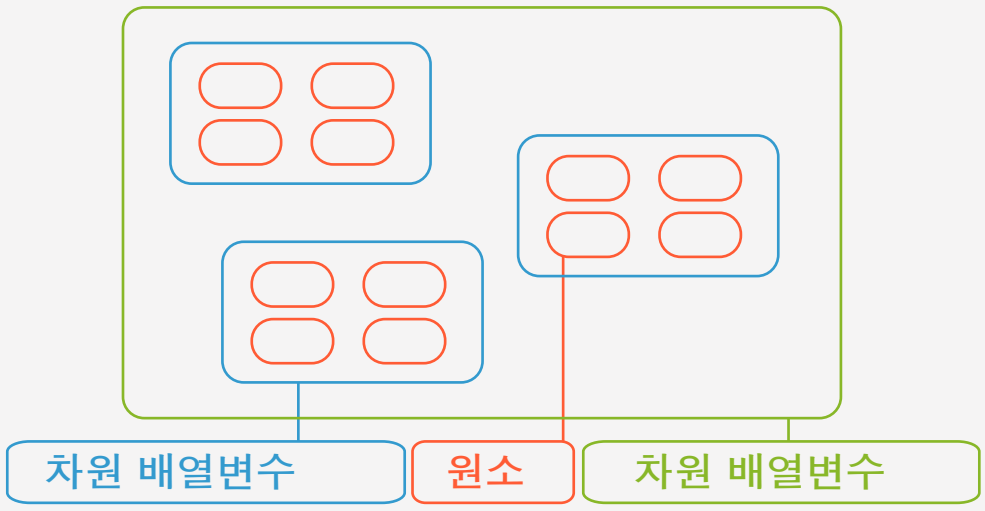
그럼, 아래 예제코들 통해 2차원 배열변수가 무엇이고 어떻게 사용하는지 살펴봅시다.

프로그램 예제 15-2:

```
//예제 15-2 8세그먼트 모듈
int number[10][8] =
{
  {0,0,0,1,0,0,0,1},     //숫자 0
  {0,1,1,1,1,1,0,1},     //숫자 1
  {0,0,1,0,0,0,1,1},     //숫자 2
  {0,0,1,0,1,0,0,1},     //숫자 3
  {0,1,0,0,1,1,0,1},     //숫자 4
  {1,0,0,0,1,0,0,1},     //숫자 5
  {1,0,0,0,0,0,0,1},     //숫자 6
  {0,0,1,1,1,1,0,1},     //숫자 7
  {0,0,0,0,0,0,0,1},     //숫자 8
  {0,0,0,0,1,1,0,1}      //숫자 9
};
void numberShow(int i){                    //숫자표시 함수
   for(int pin = 2; pin <= 9 ; pin++){
       digitalWrite(pin, number[i][pin - 2]);
    }
}
void setup(){
   for(int pin = 2 ; pin <= 9 ; pin++){    //  D2~D9 출력설정후 LED 모두 끔
       pinMode(pin, OUTPUT);
       digitalWrite(pin, HIGH);
  }
}

void loop() {
   for(int j = 0; j <= 9 ; j++){
      numberShow(j);                       //   변수 j 값을 모듈에 표시.
       delay(500);
    }
}
```

Code 해설

2차원 배열변수는 1차원 배열변수를 원소로 하는 배열변수 입니다. 아래 그림을 보고 1차원 배열변수와 2차원 배열변수의 관계를 살펴봅시다.



연습프로그램 1에서는 8개의 원소를 갖는 1차원 배열변수 10개를 사용했습니다. 즉, 80개의 원소를 저장하기 위해서, 번거롭게도 배열변수 선언을 10번이나 했습니다. 하지만, 2차원 배열변수를 사용하면, 아래와 같이 배열변수 선언을 1번만 해도 80개의 원소를 모두 저장할 수 있습니다.

number[0][0]

```
int number[10][8] =
{
  {0,0,0,1,0,0,0,1},    // display0
  {0,1,1,1,1,1,0,1},    // display1
  {0,0,1,0,0,0,1,1},    // display2
  {0,0,1,0,1,0,0,1},    // display3
  {0,1,0,0,1,1,0,1},    // display4
  {1,0,0,0,1,0,0,1},    // display5
  {1,0,0,0,0,0,0,1},    // display6
  {0,0,1,1,1,1,0,1},    // display7
  {0,0,0,0,0,0,0,1},    // display8
  {0,0,0,0,1,1,0,1}     // display9
};
```

number[9][7]

number[10][8]의 배열변수의 이름은 number이고, (원소가 8개인)1차원 배열변수 10개를 원소로 갖는 2차원 배열변수를 선언 한다는 의미입니다.

프로그램을 작성할 때, 숫자는 항상 0부터 시작합니다.

따라서, 2차원 배열변수의 첫 번째 원소는 number[0][0]의 저장 값인 0이고, 마직막 원소(80번째)는 number[9][7]의 데이터인 1입니다.

아래는 numberShow()함수를 선언하는 코드입니다. 이 함수는 아두이노의 8개의 디지털 핀에 배열변수에 저장된 테이타 들을 출력해서 8세그먼트 모듈에 숫자를 표시합니다.

```
void numberShow(int i){
  for(int pin = 2; pin <= 9 ; pin++)
      digitalWrite(pin, number[i][pin - 2]);
}
```

즉, numberShow(3); 은 숫자 3을 표시하고

numberShow(8); 은 숫자 8을 표시합니다.

위의 numberShow()함수를 for문을 사용해서 0부터 9까지 반복하면 숫자 0부터 9까지 0.5초 간격으로 순서대로 표시할 수 있게 됩니다. 아래의 loop()함수가 바로 그것입니다.

```
void loop() {
  for(int j = 0; j <= 9 ; j++){
      numberShow(j);
      delay(500);
  }
}
```

위의 loop()함수는 2차원 배열함수를 사용하기 때문에 전체 프로그램이 훨씬 짧아 졌습니다.

이 loop()함수에서, for 반복문의 지역변수 j는 0부터 9까지 1씩 증가하면서 변합니다.

따라서, numberShow()함수는 다음처럼 차례대로 실행됩니다.

즉, 처음에는 numberShow(0), numberShow(1), ... numberShow(5)... numberShow(8), numberShow(9)를 계속 반복하며 숫자0부터 9까지를 8-세그먼트 모듈위에 0.5초 간격으로 표시하게 됩니다.

2차원 배열을 사용하면 1차원 배열을 사용한 프로그램과 어떻게 다른지 살펴봅시다.

1차원 배열을 이용해서, 숫자 0을 표시하려면 아래 코드를 사용합니다.

만일, 숫자 0부터 9까지 표시하려면, 아래 프로그램을 10번 반복 작성해야 합니다.

```
int n0[8]={0,0,0,1,0,0,0,1};
// 숫자 0 데이타
for(int pin = 2; pin <= 9 ; pin++){
  digitalWrite(pin,n0[pin-2]);
 }
```

하지만, 2차원 배열변수를 사용할 경우는 아래 처럼 프로그램이 훨씬 간단해 집니다.

2차원 배열변수를 사용하는 함수 numberShow();를 정의하고, for 반복문 하나로 이 함수를 불러와서 숫자 0 부터 9까지를 표시할 수 있습니다.

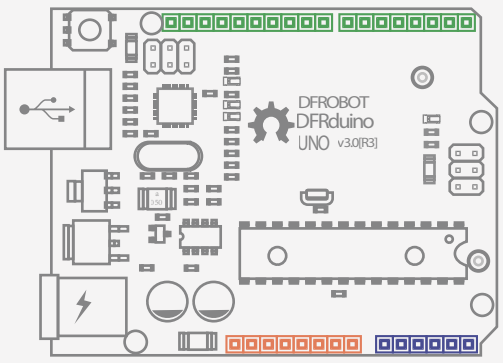
```
void loop(){
for (int j = 0; j <=9 ; j++){
      numberShow(j); delay(500);
      }
}

Void numberShow(int i){
    for(int pin = 2; pin <=9; pin++)
            digitalWrite(pin,number[i][pin-2]; )
}
```

리모콘 제어 숫자표시 모듈

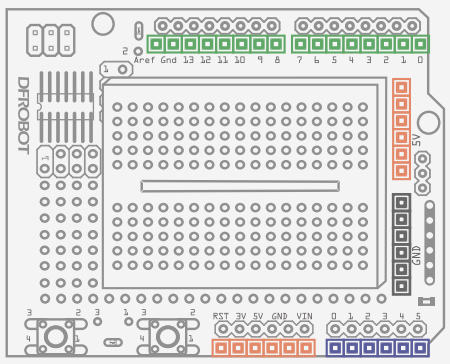
8세그먼트 모듈과 적외선 리모콘 수신기를 결합하여 응용하는 것은 작동원리를 익히면 누구든지 쉽게 달성할 수 있습니다. 아두이노 보드는 리모콘 송신기에서 보내는 신호를 처리하여 LED 모듈을 제어할 수 있습니다.

리모콘 송신기의 0부터 9까지의 숫자를 누르면, LED모듈에 누른 숫자가 표시되고, +키를 누르면 숫자가 증가하고, -키를 누르면 감소하도록 만드는 것도 그다지 어렵지 않습니다.

필요한 부품



DFduino UNO R3 x1

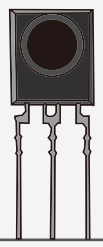


Prototype Shield x1

Jumper Cables M/M x13

Resistor 220R x8

8-Segment LED x1



IR Receiver x1

IR Remote controller x1

회로 구성

아래 그림과 같이 부품을 연결하여 회로를 구성하시요.

사용하는 회로는 프로젝트14와 프로젝트15를 합한 것이지만, 변경된 부분은 거의 없습니다. 회로구성이 이해가 안되면 앞에서 학습한 내용을 복습해 보십시오.

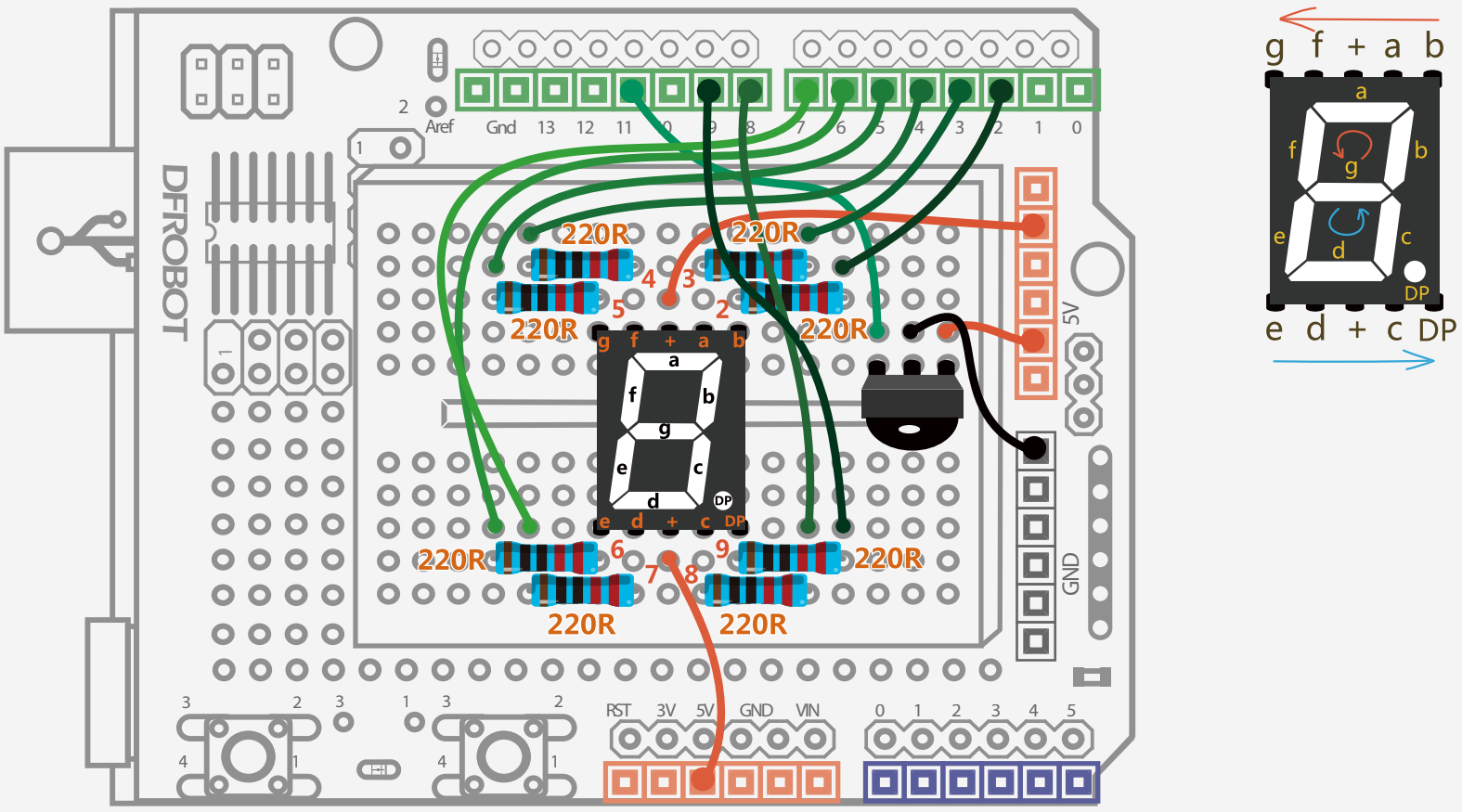


그림 15-3 리모콘 숫자표시 모듈 회로도

Arduino Sketch

프로그램예제 15-1

```
//예제 15-1 리모콘 숫자표시 모듈
#include <IRremote.h>      //라이브러리 추가
int RECV_PIN = 11;         //R수신기 D11
IRrecv irrecv(RECV_PIN);   //D11을 IR수신포트로 지정
decode_results results;    //리모콘 수신데이타 저장
int currentNumber = 0;     //현재 숫자 저장

long codes[12]=            //리모콘 16진 코드저장
{
  0xFD30CF,0xFD08F7,       // 0 ,1
  0xFD8877,0xFD48B7,       // 2 ,3
  0xFD28D7,0xFDA857,       // 4 ,5
  0xFD6897,0xFD18E7,       // 6 ,7
  0xFD9867,0xFD58A7,       // 8 ,9
  0xFD20DF,0xFD609F,       // + ,-
};

int number[10][8] =        //8세그먼트 모듈용 숫자데이타
{
  {0,0,0,1,0,0,0,1},//0
  {0,1,1,1,1,1,0,1},//1
  {0,0,1,0,0,0,1,1},//2
  {0,0,1,0,1,0,0,1},//3
  {0,1,0,0,1,1,0,1},//4
  {1,0,0,0,1,0,0,1},//5
  {1,0,0,0,0,0,0,1},//6
  {0,0,1,1,1,1,0,1},//7
  {0,0,0,0,0,0,0,1},//8
  {0,0,0,0,1,1,0,1},//9
};

void numberShow(int i) {   //숫자표시 함수
  for(int pin = 2; pin <= 9 ; pin++){
  digitalWrite(pin, number[i][pin - 2]);
  }
}
void setup(){
  Serial.begin(9600);      //9600 baud
  irrecv.enableIRIn();     //IR 수신기 초기화

for(int pin = 2 ; pin <= 9 ; pin++){    //숫자표시핀 OUTPUT
  pinMode(pin, OUTPUT);
  digitalWrite(pin, HIGH);
  }
}
```

```
void loop() {
  //리모컨 수신 데이타가 존재 하는가
  if (irrecv.decode(&results)) {
      for(int i = 0; i <= 11; i++){
        //리모콘 수신값이 숫자 인가
        if(results.value == codes[i]&& i <= 9){
            numberShow(i);                  //모듈에 숫자표시
            currentNumber = i;              //현재 숫자 변수에 저장
            Serial.println(i);
            break;
        }

        // 숫자 감소키 누름상태, 현재 숫자의 저장값이 0인가
        else if(results.value == codes[10]&& currentNumber != 0){
            currentNumber--;                //현재숫자 1감소
            numberShow(currentNumber);      //감소된 숫자표시
            Serial.println(currentNumber);  //시리얼 모니터에도 표시
            break;
        }

        //숫자 증가키 누름상태, 현재 숫자의 저장값이 9인가
        else if(results.value == codes[11]&& currentNumber != 9){
            currentNumber++;                //현재 숫자 1증가
            numberShow(currentNumber);      //증가된 숫자 표시
            Serial.println(currentNumber);  //시리얼모니터에도 표시
            break;
        }
    }

  Serial.println(results.value, HEX);       //리모콘 16진코드 표시
  irrecv.resume();                          //리모콘 수신대기
  }
}
```

그림 15-4 리모콘 송신기

이 코드를 실행하고, 리모컨의 버튼을 누르면 LED모듈이 정해진 순서대로 숫자를 표시합니다.

Code 해설

이 프로그램의 첫 부분은 리모콘 송수신기를 사용할 수 있도록 설정하는 부분인데, 프로젝트-14에서 사용했던 프로그램과 동일합니다. 아래 코드는 리모콘을 사용할 때, 거의 고정적으로 사용하는 정형화된 코드입니다. 아래 4줄의 코드만으로도 리모콘 사용을 위한 기본적인 준비가 모두 완료된 상태입니다.

리모콘 수신기는 D11핀에 연결하고, 리모콘 수신기로 수신한 코드는 변수 결과(result)에 저장하도록 코드가 작성되어 있습니다.

```
#include <IRremote.h>      //라이브러리 추가
int RECV_PIN = 11;         //RECV_PIN을 11로 설정
IRrecv irrecv(RECV_PIN);   //RECV_PIN 를 IR수신핀으로 설정
decode_results results;    //해석된 코드를 저장할 변수 results 를 선언
```

이 프로그램에서는, 리모콘 송신기의 12개 키만을 사용합니다. 즉, 숫자 0~9, 그리고 "증가" 키와 "감소"키가 그것입니다. 리모콘 송신기의 12개의 키를 누를 때마다 각각의 키에 정해진 16진수 코드가 송신됩니다. 따라서, 아두이노가 수신한 코드가 리모콘 송신기의 어느 키에 해당하는 지를 알려면, 미리 저장해 둔 코드와 비교하는 과정이 필요합니다.

이 프로그램에서는, 12개의 각각의 키에 해당하는 16진 코드를 아래와 같이 codes[ ] 라는 배열변수에 저장했습니다.

```
ong codes[12]= //리모콘 16진 코드저장
{
0xFD30CF,0xFD08F7, // 0 ,1
0xFD8877,0xFD48B7, // 2 ,3
0xFD28D7,0xFDA857, // 4 ,5
0xFD6897,0xFD18E7, // 6 ,7
0xFD9867,0xFD58A7, // 8 ,9
0xFD20DF,0xFD609F, // + ,-
};
```

만일, 아두이노가 수신한 리모콘의 16진 코드가 0xFD08F7일 경우, 숫자 1 번키를 눌렀음을 쉽게 알 수 있습니다.

If ~ else if 조건문

이 프로그램을 수행하려면, 아래의 3가지 조건의 확인 필요합니다.

첫째 조건은 8세그먼트 모듈에는 리모콘의 눌려진 숫자가 표시되어야 합니다.

둘째 조건은, 감소키를 누르면, 0이 될 때 까지 1씩 감소된 숫자가 표시되어야 합니다.

셋째 조건은 증가키를 누를 때, 9가 되기 전까지 1씩 증가된 숫자가 표시되어야 합니다.

지금까지는 프로그램을 작성할 때, 주어진 조건을 판단하려면 "if 조건문"을 사용해 왔습니다. 하지만, 이번 프로젝트에서는 "if 조건문"이 아닌 "if ~ else if 조건문"을 사용했습니다. "If 조건문"과 "if ~ else if 조건문"은 어떤 차이가 있을까요?

If 조건문의 뒷부분에는 일반적으로 else if 를 사용합니다. 다시 말하면, 앞부분에 if 조건문이 있어야만 else if 를 사용할 수 있습니다.

앞의 제시한 3가지 조건을 if 명령어를 사용해서 판단하는 프로그램을 작성한다면 아래와 같습니다. 천천히, 살펴봅시다.

```
if (results.value == codes[i]&& i <= 9)
if (results.value == codes[10]&& currentNumber != 0)
if (results.value == codes[11]&& currentNumber != 9)
```

첫 번째 if문은 수신된 리모콘 코드가 0~9사이의 숫자에 해당하는 코드인지를 판단하는데, 배열변수 codes[0]부터 codes[9]에 저장된 데이터와 수신된 리모콘 코드가 일치하는지를 일일이 판단하게 됩니다.

두 번째 if문은 수신된 리모콘 코드가 감소키를 눌렀을 때 발생하는 코드인지를 판단하는데, 역시 codes[10]에 저장된 데이터와 수신된 리모콘 코드가 일치하는 지를 판단하고, 동시에 현재 표시된 숫자가 0이 아닌지를 판단합니다.

세 번째 if문은 수신된 리모콘 코드가 증가키를 눌렀을 때 발생하는 코드인지를 판단하는데, 역시 codes[11]에 저장된 데이터와 수신된 리모콘 코드가 일치하는 지를 판단하고, 동시에 현재 표시된 숫자가 9가 아닌지를 판단합니다.

위의 세가지 if문을 수행하려면, 배열변수 codes[ ]에 저장된 값을 수시로 비교해야 합니다. 따라서, if문을 for 반복문과 함께 사용해야 합니다. 이 프로젝트에서 사용한 예제코드에서는 3개의 if문을 사용하므로, for 반복문을 3번 사용해야 하지만, for 반복문을 단 1번만 사용했습니다. 결과적으로 훨씬 더 효율적인 프로그램이 되었습니다. "if 조건문"과 "if ~ else if 조건문"의 차이점에 대한 해답은 여기서 찾을 수 있습니다.

이번 프로젝트는 본 교재에서 다루는 프로젝트 중 가장 이해하기 어려운 부분입니다. 하지만, 여러분의 꾸준한 노력이 더해진다면 어렵지 않게 이해할 수 있을 것으로 확신합니다.

연습 문제

이 프로젝트에서 익힌 리모콘 사용법을 응용하여 여러분 만의 멋진 작품을 만들어 보세요.

이 교재에서 익힌 여러가지 프로젝트는 아주 기본적인 것들이지만, 여러분의 창의력이 더해지면 얼마든지 멋진 작품으로 재탄생 할 수 있습니다. 예를 들면, 아두이노 보드에 여러 개의 LED와 서보 모터를 추가해서 만든 장난감 로봇을 리모콘 송수신 장치를 이용하여 마음대로 움직이게 할 수도 있습니다.