아두이노 실습 매뉴얼

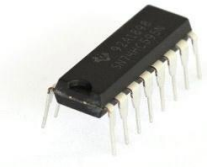
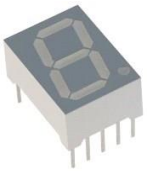
박 상 우, 정 동 화, 김 지 민

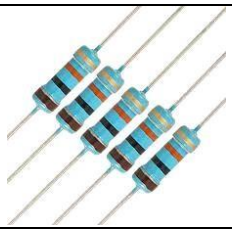
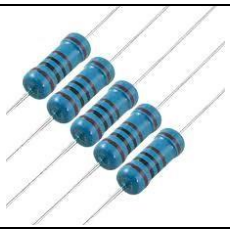
메카솔루션

머리말

여러분이 본 실습 매뉴얼을 읽고 계신다면, 아두이노에 대해서는 한번쯤 이야기를 들어보셨으리라 생각됩니다. 공학 혹은 디자인과 관련된 분야를 전공하는 대학(원)생, 제품 개발 및 연구하는 분, 교육하는 분, 그리고 창의 교육의 일환으로 배우는 초중고 학생들에 이르기까지 아두이노는 다양하게 사용되고 있습니다. 아두이노가 세상에 나오고 국내에 알려진 후로, 다양한 교재가 번역 및 집필되고 있습니다. 2009 년 가을, 미국에서 기계공학 박사과정을 밟으면서 처음 접한 아두이노라는 플랫폼은 가히 혁신적이었습니다. AVR, PIC, 그리고 ARM 등의 마이크로컨트롤러를 프로그래밍하기 위해 레지스터를 익히고 인터럽트를 배워야했던 것에 비해, 아두이노라는 프로세서는 쉽게 익힐 수 있었고, 특유의 오픈 하드웨어라는 장점으로 인해 그 파급력은 상상 이상이었습니다.

훌륭히 집필된 책들보다 나은 책을 쓰려는 과욕은 버리되, 책을 구입하기 전에 한번 읽어볼만한 자료로서 집필된 본 매뉴얼의 주된 목적은 여러분의 아이디어를 아두이노를 이용하여 만드는 것을 도와주는 것입니다. 특히, 아두이노를 처음 접하는 중/고등학교 학생들부터 전기전자/메카트로닉스/로보틱스 전공을 하는 학부생들의 실습 목적으로 구성되었으며, 쉽게 따라하는 방식으로 배울 수 있도록 하였습니다.

			
Uno R3	**USB cable**	**브레드보드**	**8x8 LED kit**
			
10 Red LEDs	**10 Green LEDs**	**10 Yellow LEDs**	**5 White LEDs**
			
RGB LED	**74HC595**	**Piezo Buzzer**	**1-digit display**
			
4-digit display	**Push button**	**Mini push button**	**조도센서**

			
10KΩ 저항(갈검주)	1KΩ 저항(갈검빨)	220Ω 저항(빨빨갈)	5K 가변저항
			
10K 가변저항	LM35	LCD display	조이스틱 모듈
			
미니 서보 모터	점퍼 케이블	암-암 케이블	헤더소켓(수)
			
헤더소켓(암)	기울기 센서	불꽃 센서	초음파 거리센서
			
인체감지 센서	QRD1114	9V battery holder	고급 케이스

기술지원 :

Email: roboholic84@gmail.com, techms5499@gmail.com

페이스북:

http://facebook.com/likemechasolution

목록 • 아두이노

시작하기

- 아두이노란?
- 소프트웨어 설치하기
- 통합개발환경에 대하여
- 첫번째 아두이노 프로젝트
- 구조, 변수, 함수
- LED 깜박이기
- “Hello Arduino”
- 키 입력으로 LED 켜기 • 브레드보드와 쉴드
- 브레드보드 사용 방법
- 아두이노 쉴드란?
- 저항 읽기
- 풀업 저항과 풀다운 저항
- 버튼입력
- 어떻게 저항값을 결정하는가?
- 신호등 만들기
- 아날로그 센서
- 조도센서
- 온도 측정하기
- 포텐시오미터로 LED 밝기 조절하기

- PS2 Joystick module
- 디지털 센서
- 기울기센서로 LED 켜기
- HC-SR04 초음파 거리센서
- PIR 모션센서
- QRD1114 짧은 거리센서
- 디스플레이
- 라이브러리 추가하기
- 16x2 LCD
- 74HC595 시프트 레지스터를 이용한 7 세그먼트
- 7 세그먼트 4 개 사용하기
- MAX7219 를 이용한 8x8 도트매트릭스
- LED 이동 디스플레이 만들기
- 디지털 온도계 만들기
- 소리
- 슈퍼마리오 재생하기
- 빛 감지 알람
- 모터
- Pulse Width Modulation 이란?
- 어떻게 서보모터가 회전하는가
- 포텐시오미터로 서보모터 조종하기

• 아두이노 시작하기

• 아두이노란?

아두이노를 키워드로 말한다면 '쉽다', '오픈 소스', '마이크로컨트롤러' 등으로 표현될 수 있을 것 같습니다. 이렇듯, 아두이노는 여러분의 새로운 아이디어를 쉽게 구현할 수 있도록 디자인된 마이크로 컨트롤러입니다. 이 마이크로컨트롤러와 센서, 모터, 디스플레이, 그리고 무선 통신 모듈과 같은 다양한 전자 소자들을 연결하여 여러분들께서 생각하고 계시는 '그것'을 구현할 수 있도록 도와주게 됩니다. 아두이노는 단지 하드웨어를 의미하는 것 뿐만 아니라 프로그래밍을 쉽게 할 수 있도록 돕는 통합개발환경도 일컫게 되는데 많은 사람들이 공유하는 소스코드를 하드웨어에 업로드하거나 변경하여 손쉽게 동작시킬 수 있는 장점을 가지고 있습니다.

오픈소스라는 장점으로 인해서 다양한 모양과 성능의 아두이노 혹은 아두이노 호환보드가 시중에 나와 있는데, 시스템의 사이즈, 요구되는 성능, 필요한 입출력 포트의 수 등, 목적에 맞게 사용하시면 되며 본 매뉴얼에서는 가장 많이 사용되고 있는 우노를 사용하도록 하겠습니다.

Fig. 1. UNO R3

• 소프트웨어 설치하기

아두이노 Uno R3(이하 보드)가 알 수 있도록 코드를 써주기 위해서는 소프트웨어가 필요합니다. 다행스럽게도, 이 소프트웨어는 무료입니다. 인터넷이 연결되어 있다면 다음의 순서대로 소프트웨어를 설치할 수 있습니다.

먼저 아래 사이트에 접속합니다. 혹은 구글에서 "Arduino install"로 검색하셔도 됩니다.

http://arduino.cc/en/Main/Software

Windows 사용자

1) 보드를 USB 로 연결하고 드라이버 설치과정을 시작할 때까지 기다립니다. 잠시 후 실패했다고 뜨게 되면 정상입니다.

2) 시작메뉴를 클릭하고 제어판을 클릭합니다.

3) 제어판에서 시스템 및 보안을 클릭합니다. 그 후 시스템을 클릭한 후 장치관리자를 엽니다.

4) 포트부분(COM & LPT)에서 “Arduino Uno(COMxx)”라는 부분을 찾습니다.

5) “Arduino Uno(COMxx)”를 우 클릭한 후 “드라이버 소프트웨어 업데이트”를 클릭합니다.

6). “컴퓨터에서 드라이버 소프트웨어 찾아보기”를 클릭합니다.

7) 최종적으로 아두이노 폴더 안의 “Drivers” - “ArduinoUno.inf”파일을 클릭합니다. (“FTDI USB Drivers”가 아닙니다.)

8) 드라이버 설치가 완료되었습니다.

9) 아두이노 실행파일을 실행시킵니다.

10) LED blink 예제를 불러옵니다. (파일 > 예제 > 1.Basics > Blink)

11) 도구 – 보드 메뉴에서 Arduino Uno 를 클릭합니다.

12) 시리얼 포트를 선택합니다. (만약 찾지 못했다면 보드를 연결 해제한 후 빠진 곳이 있는지 체크합니다.

13) 업로드 버튼을 클릭합니다.

14) “Done uploading”이라는 메시지가 나오면 보드의 USB 포트부분 위에 있는 L 부분이 반짝이는 것을 볼 수 있습니다.

MAC 사용자

1) 보드를 USB 로 연결합니다.

2) 하드 드라이브에 아두이노 실행파일을 드래그 합니다.

3) 네트워크 설정에서 적용 혹은 “Apply”를 클릭합니다. (/dev/tty/usb.) 4) 프로그램을 실행합니다.

5) LED blink 예제를 불러옵니다. (파일 > 예제 > 1.Basics > Blink)

6) 도구 – 보드 메뉴에서 Arduino Uno 를 클릭합니다.

7) 시리얼 포트를 선택합니다. (만약 찾지 못했다면 보드를 연결 해제한 후빠진 곳이 있는지 체크합니다.

8) 업로드 버튼을 클릭합니다.

9) “Done uploading”이라는 메시지가 나오면 보드의 USB 포트부분 위에 있는 L 부분이 반짝이는 것을 볼 수 있습니다.

• 통합개발환경에 대하여

아두이노 소프트웨어가 설치가 되면 다음의 스케치 창을 확인할 수 있습니다. 아두이노 보드를 선택하고, 연결된 시리얼포트를 설정하며, 아두이노를 통해서 어떤 일을 할지 프로그래밍을 쓰고, 보드에 업로드하는 기능을 수행합니다. 이렇듯, 다양한 일을 하기 때문에 통합 개발 환경 (IDE)이라고도 부릅니다. 다음은 메뉴 아이콘들에 대한 설명입니다.

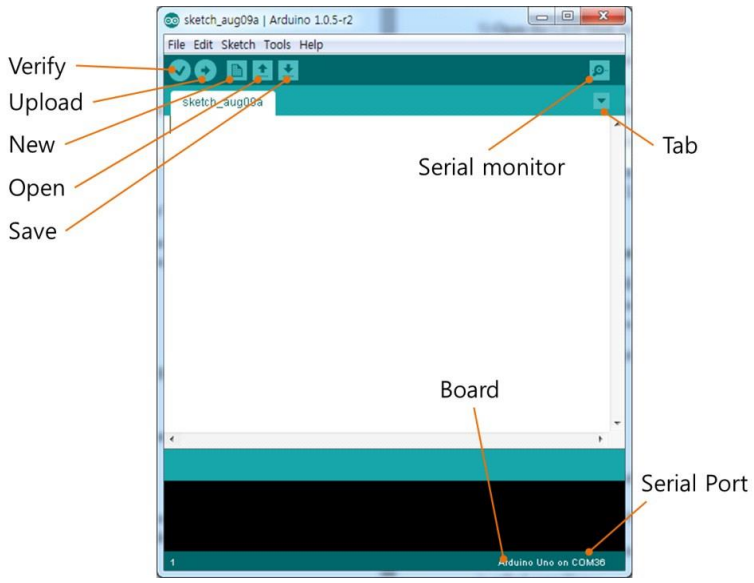


Fig. 2. Arduino IDE

• 확인(Verify) : 코드를 보드에 보내기 전에 보드가 이해할 수 있도록 명령어를 변경합니다.

• 업로드(Upload) : 확인, 번역한 후에 USB 케이블로 보드에 전송합니다.

• 새 파일(New) : 새 스케치를 만들기 위해 새 창을 띄웁니다.

• 열기(Open) : 컴퓨터에 있는 다른 스케치 파일을 엽니다.

• 저장(Save) : 현재 작업을 스케치로 변환하여 저장합니다.

• 시리얼 모니터(Serial monitor) : 아두이노에서 매우 유용한 기능입니다. USB 를 통해서 보드로 메시지를 보내거나, 받을 수 있습니다.

• 탭(Tab) : 여러 개의 스케치를 작업 할 수 있게 해줍니다. 이 수업에서 하는 것보다 더 고급 프로그래밍을 할 때 사용됩니다.

• 첫번째 아두이노 프로젝트

자 그럼, 소프트웨어가 설치되었으니, 첫번째 프로젝트를 수행해보도록 하겠습니다. 먼저, 기본 골격을 알아야겠죠? 가장 기본이 되는 것은 setup()과 loop()라는 함수입니다. Fig. 3 에 보이다시피, 이 두 함수는 꼭 써줘야 합니다. 그래서 라이브러리를 그대로 사용하는 것이 아닌 프로그램을 작성한다면 이 두 함수는 습관처럼 적어도 될 것입니다. (라이브러리 관련해서는 다음에 설명하도록 하겠습니다.)

• 구조, 변수, 함수

소위 구조, 변수, 함수가 프로그램을 구성하는 요소인데, 반드시 알았으면 하는 것들을 아래에 설명하였습니다. 모두 다 이해할 필요도 암기할 필요도 없지만, 적어도 setup()과 loop()에 대해서는 이해하도록 합시다.

- 구조
- setup();

  setup() 함수는 스케치를 시작할 때 불러옵니다. 초기 변수, 핀 상태, 사용 라이브러리 시작등에 사용합니다. setup 기능은 전원이 켜졌을 때 혹은 리셋버튼을 눌렀을 때 한번만 실행됩니다.
- loop():

  setup()을 생성한 후에 loop() 함수가 프로그램의 변화와 응답 내에서 연속적으로 작동합니다. 아두이노 보드를 제어하는데 적극적으로 사용하세요.

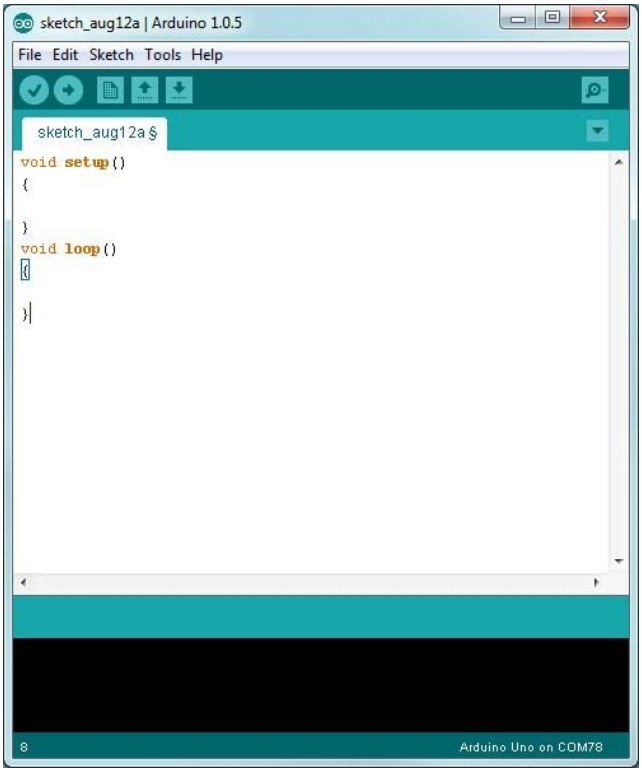


Fig. 3. 기본 구조 – setup and loop

위 Fig. 3 에 보이는 바와 같이 setup()과 loop()는 아두이노에서 컴파일과 업로드를 하기 위해 기본적으로 요구되는 부분입니다.

- if

if / else 문은 여러 개의 그룹으로 묶여있는 코드의 흐름을 더 부드럽게 합니다. 예를 들어 아날로그 입력 값이 500 미만일 때 A 번 행동(action A)이 실행된다면 다른 행동(action B)은 500 이상일 때 실행됩니다. 코드는 아래와 같습니다.

```
if (pinFiveInput < 500)
{
  // action A
} else
{
  // action B
}
```

else 는 if 문에서 언급한 조건이 맞지 않을 때 실행되며, 이도 저도 아닐 경우에 실행되는 구문입니다. 즉, 참이 되는 것을 찾을 때까지 진행되며, 그렇지 않으면 else 구문이 실행됩니다.

else if 문은 if 문과 접속하고 else 문과 접속하기도, 안하기도 합니다. 이 Else if 문은 무한하게 쓸 수 있습니다.

다음의 코드는 pinFiveInput 이라는 값이 500 보다 작으면 A 를 실행하고(action A), 1000 보다 크거나 같으면 B 를 실행합니다 (action B). 그리고 어느 쪽에도 해당되지 않으면 C 를 수행합니다 (action C).

```
if (pinFiveInput < 500)
{
  // action A
} else if (pinFiveInput >=
1000)
{
  // action B
} else
{
  // action C
}
```

예를 들어, 100 이라는 숫자가 pinFiveInput 에 할당되면, A

- for:

for 구문은 외부로부터 격리된 채로 반복하기위해 쓰입니다. 증가하는 카운터는 루프의 종료와 증가를 위해 쓰입니다. for 구문은 반복되는 동작에서 매우 유용하며 종종 배열화된 데이터나 핀의 무리에서 동작하기 위해 사용합니다.

```
for (initialization; condition; increment) {
//statement(s);
}
```

- 변수

- HIGH

HIGH 의 의미(핀에서 사용하는 부분)는 핀의 설정이 INPUT 모드냐 OUTPUT 모드냐에 따라 다릅니다. pinMode 가 INPUT 모드일 때, digitalRead 를 썼을 경우 3V 이상이 들어 올 경우입니다. 또 digitalWrite 와 함 께 쓸 수도 있습니다. 그러면 내부 20K 의 풀업저항이 설정됩니다. 그럼 이 핀은 외부 회로에 의해 LOW 가 되기 전까지 HIGH 가 됩니다. 이것은 INPUT_PULLUP 으로 불립니다.

pinMode 가 OUTPUT 으로 설정되었을 경우 digitalWrite 에서 HIGH 로 설정하면 5V 가 출력됩니다. 이 상태에선 전류를 흘려 보낼 수 있습니다. 예로 LED 등을 킬 수 있습니다. 이것은 저항을 통해 ground 로 연결될 때 LOW 상태가 됩니다.

- LOW

LOW 의 의미도 INPUT 과 OUTPUT 일 때 다른 의미입니다. pinMode 가 INPUT 일 경우 digitalRead 를 썼을 때, 2 볼트 이하인 상태입니다.

OUTPUT 상태일 때 digitalWrite 에서 LOW 로 설정하면 핀은 0 볼트가 됩니다. 전류를 받아 들일 수 있습니다. 예로 5V 와 저항에 연결된 LED 를 연결하게 되면 LED 에 불이 들어옵니다.

- true

true 는 종종 1 로 정의됩니다. 이것은 '참'이란 의미지만 실제 true 는 더 넓은 의미입니다. Boolean 에서 0 이 아닌 모든 정수는 '참'입니다. 그래서 -1 이나 2, -200 등은 모두 참으로 정의됩니다.

- false

false 는 0 으로 정의됩니다.

- Functions
- pinMode(pin, mode)

핀의 상태를 입력 혹은 출력으로 설정합니다. 핀의 목적을 보다 자세하게 묘사해 줍니다.

- digitalWrite(pin, value)

디지털 핀에 위에서 설명한 HIGH 혹은 LOW 를 입력합니다.

- digitalRead(pin)

디지털 핀의 구체적인 값을 읽습니다. HIGH 혹은 LOW 가 나옵니다.

- analogRead(pin)

구체적인 아날로그 핀의 값을 읽습니다. 보드에는 6 개의 채널이 있으며(나노와 미니에는 8 개) 10 비트의 ADC(아날로그 디지털 컨버터)가 있습니다. 이것은 0 ~ 5 볼트가 0 ~ 1023 의 값으로 변환되는걸 의미합니다.

- analogWrite(pin, value)

핀에 아날로그 값(PWM 형식)을 넣습니다.

- **LED 깜박이기**

준비물 : Uno R3, USB cable

배경 : Uno 에는 디지털 13 번 핀과 연결된 내장 LED 가 있습니다. USB 로 PC 와 보드를 연결하면 깜빡이는 LED 를 볼 수 있습니다.

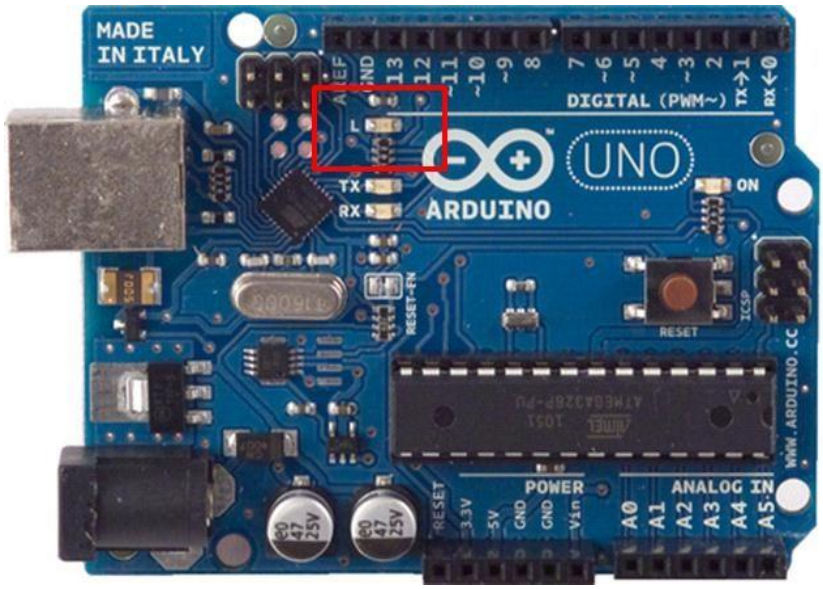


Fig. 4. Arduino UNO R3

스텝 1:

파일 – 예제 – Basics – Blink 에 들어가서 예제 파일을 불러옵니다. 그러면 아래와 같은 새 창이 뜨게 됩니다.

코드:

```
/*
  Blink   Turns on an LED on for one second, then off for one second,
repeatedly.

  This example code is in the public domain.
*/

// Pin 13 has an LED connected on most Arduino boards. //
give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

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

설명:

- /* comments */ 소위 주석이라 부르며, /*으로 시작해서 */로 끝납니다. 이 부분은 컴파일러가 무시하며 과정에 포함되지 않습니다. 그래서 이 부분은 Atmega 에서 아무런 공간을 차지하지 않습니다.
- 한 줄인 경우에는 //를 사용합니다.
- 보드에 있는 LED 의 경우 디지털 13 번 핀과 연결되어 있습니다. 그래서 코드에서는 정수로 핀 번호를 13 번으로 정의했습니다.
- 모든 맨 끝 문장에는 세미콜론이 붙어 있습니다.
- setup()안에서는 한번만 실행될 것을 넣어주세요. 만약 반복할게 있다면 loop() 안에 넣어주세요,
- pinMode(led,OUTPUT)은 pinMode(13,OUTPUT)과 같습니다.

- digitalWrite(led, HIGH)가 LED 를 켜고 뒤에 나오는 delay(1000) 함수가 1000 ms 만큼 프로그램을 일시 정지시킵니다.

스텝 2:

도구 – 보드 – Arduino Uno 를 클릭하여 보드를 선택합니다. 그 후 도구 – 시리얼 포트에 가면 여러 개의 포트가 있습니다. USB 를 뺐다 꽂으면 사라졌다가 나오는 포트가 있습니다.

스텝 3:

확인을 클릭한 후 업로드를 클릭합니다. 그리고 동작을 확인합니다.

- **"Hello Arduino"**

이 부분에서 우리는 아두이노 통합개발환경의 시리얼 인터페이스를 사용할 것입니다. 모니터로 원하는 것을 볼 수 있습니다.

스텝 1: 아래 코드를 따라 타이핑합니다.

코드(Hello Arduino):

```
void setup()
{
      Serial.begin(9600);
}
void loop()
{
  Serial.println("Hello Arduino");
  delay(1000);
}
```

설명:

- Serial.begin(baud)는 데이터를 전달할 때 1 초당 데이터 속도를 정해줍니다. 보통 무선통신은 XBee 나 블루투스일 경우 9600bps 가 디폴트로 설정되어 있습니다. 만약 보드 속도를 더 올린다면, 더 빨리 데이터를 보낼 수 있지만 통신중에 데이터를 잃어 버릴 수도 있다.

- Serial.println(val, format) 혹은 Serial.println(val)은 새로운 줄에 데이터를 출력할 수 있다.

스텝 2:

확인과 업로드를 누르면 컴파일 후 보드에 업로드를 합니다. 그러면 “컴파일 완료”와 “업로드 완료”라는 메시지가 출력된다. 아 꼭 세미콜론을 잊지 말도록!

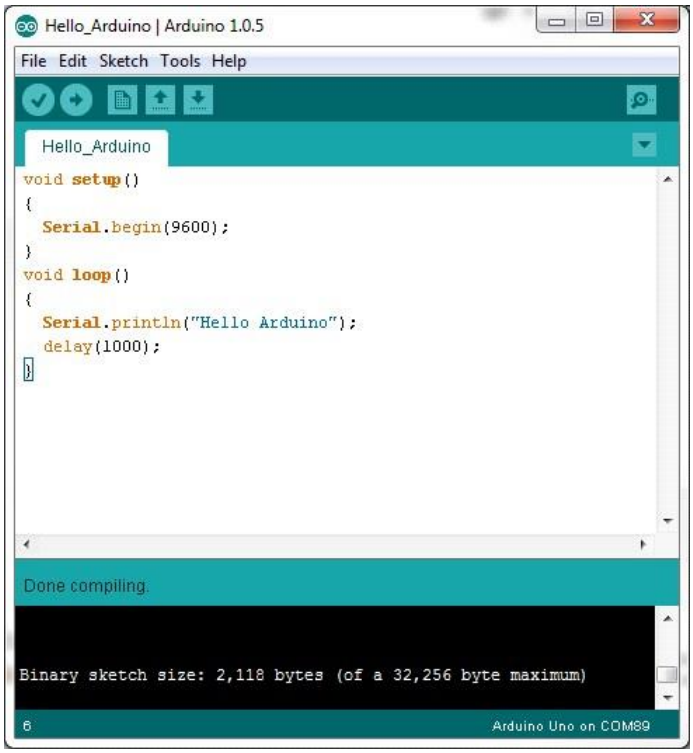


Fig. 5. 컴파일 완료

Fig. 6. 업로드 완료

스텝 3:

우측 상단에 “시리얼 모니터”를 클릭하면 새 창이 뜨고 매 1 초마다 “Hello Arduino”라는 메시지가 출력될 것입니다.

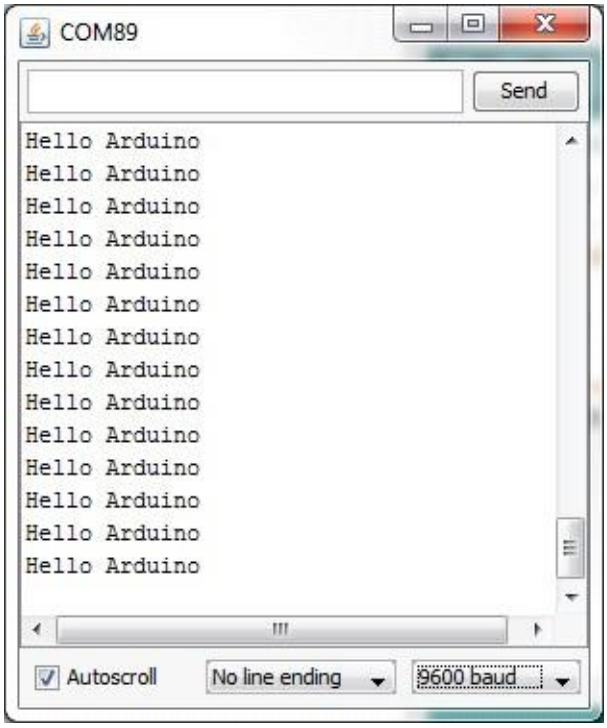


Fig. 7. 시리얼 모니터 창

• 키 입력으로 LED 켜기

위 부분에서 시리얼 모니터를 통해 출력하는 방법을 배웠습니다. 이 시리얼 모니터는 키 입력에도 사용할 수 있습니다. 여기서 우리는 시리얼 모니터창에서 값을 입력받아 내장 LED 를 키고 끄는 것을 해 보겠습니다.

스텝 1:

아래 코드를 따라 타이핑합니다.

코드(key_led):

```
int led = 13;

void setup() {
  // open the serial port:
  Serial.begin(9600);
  pinMode(led, OUTPUT);
}

void loop() {
  // check for incoming serial data:
  if (Serial.available() > 0) {     // read incoming serial data:
    char inChar = Serial.read();
    if (inChar == '1')
    {
      digitalWrite(led, HIGH);
    }
    else if (inChar == '2')
    {
      digitalWrite(led, LOW);
    }
  }
}
```

설명:

- Serial.available()는 읽기 가능한 바이트의 숫자를 반환합니다.
- Serial.read()를 이용하여 입력받는 데이터를 “inChar”에 저장합니다.
- 입력받은 글자가 “1”일 때 LED 가 켜지고(HIGH) 글자가 “2”일 때 LED 가 꺼집니다.

스텝 2:

파일을 업로드 한 후에 시리얼 모니터를 키고 “1”과 “2”를 입력해보고 보드의 LED 를 확인합니다.

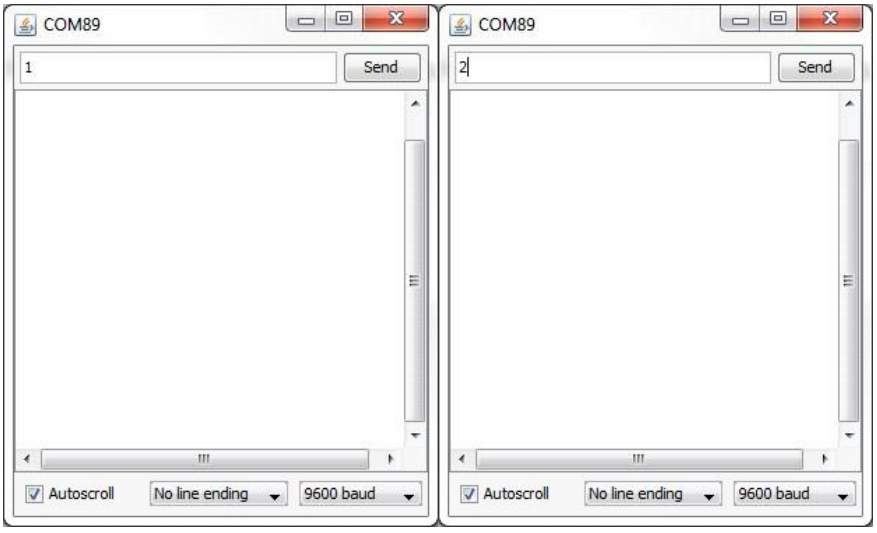


Fig. 8. “1”과 “2”를 입력하면 LED 가 바뀌는 것을 볼 수 있습니다.

- **브레드보드와 쉴드**

• 브레드보드 사용 방법

프로토타입 회로나 시험용 부품을 납땜없이 확인하기 위해 브레드보드를 사용할 수 있습니다. 일반적으로 길게 연결된 2 줄의 버스줄(빨간색과 파란색)이 있습니다. 그리고 대부분의 전자 부품은 5 구 터미널줄

(보드의 중앙)에 끼워집니다.

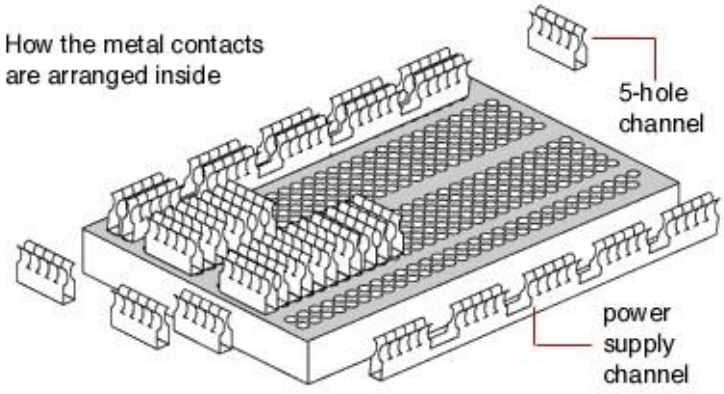


Fig. 9 브레드보드 조직도

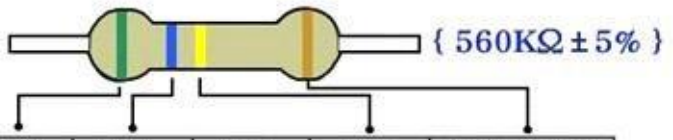
• 아두이노 쉴드란?

아두이노 쉴드는 보드에 연결하여 성능을 늘리기 위해 사용합니다. 현 시점에서 그 종류는 300 개가 넘게 있습니다. 대부분의 아두이노 쉴드는 Uno 에 맞게 제작되었으며 Mega 2560 보드와도 호환이 가능합니다. 자주 사용되는 것으로는 LCD 버튼 쉴드, 이더넷쉴드, 와이파이쉴드와 SD 카드 쉴드가 있습니다.

Fig. 10. 아두이노 쉴드

• 저항 읽기

저항을 읽는 방법은 멀티미터(테스터기)의 프로브를 통해서 디지털 혹은 아날로그 방식으로 읽을 수 있지만, 다음의 색상표를 통해서도 읽을 수 있습니다.



색상	1번띠	2번띠	3번띠	승 수	오차(등급코드)
검 정	0	0	0	1Ω	
갈 색	1	1	1	10Ω	± 1% (F)
빨 강	2	2	2	100Ω	± 2% (G)
주 황	3	3	3	1KΩ	
노 랑	4	4	4	10KΩ	
초 록	5	5	5	100KΩ	± 0.5% (D)
파 랑	6	6	6	1MΩ	± 0.25% (C)
보 라	7	7	7	10MΩ	± 0.10% (B)
회 색	8	8	8		± 0.05%
흰 색	9	9	9		
금 색				0.1	± 5% (J)
은 색				0.01	± 10% (K)

위 예시의 560K 옴 저항은 초록색, 파란색, 노란색으로 구성되어 있고, 마지막에는 금색으로 띠가 둘러져 있습니다. 첫번째와 두번째 색상띠인 5, 6 에 세번째색이 의미하는 승수 4 를 통해서 값을 결정하는데 56 X 10^4 승인 560000 옴이 되고, 이는 560K 옴이 됩니다. 종결키트에 있는 저항 220 옴, 1K 옴, 10K 옴의 색상은 따라서 다음과 같이 됩니다. 220 옴: 빨강-빨강-갈색

1K 옴: 갈색-검정-빨강

10K 옴: 갈색-검정-주황

• 풀업 저항과 풀다운 저항

풀업저항은 아두이노에 장착된 외부 장치가 끊어지거나 높은 저항을 가질때 전자회로에 입력을 보장해 주기 위한 장치로 사용됩니다. 입력 핀에 아무것도 연결되어 있지 않다면 논리적으로 1 인 상태입니다.

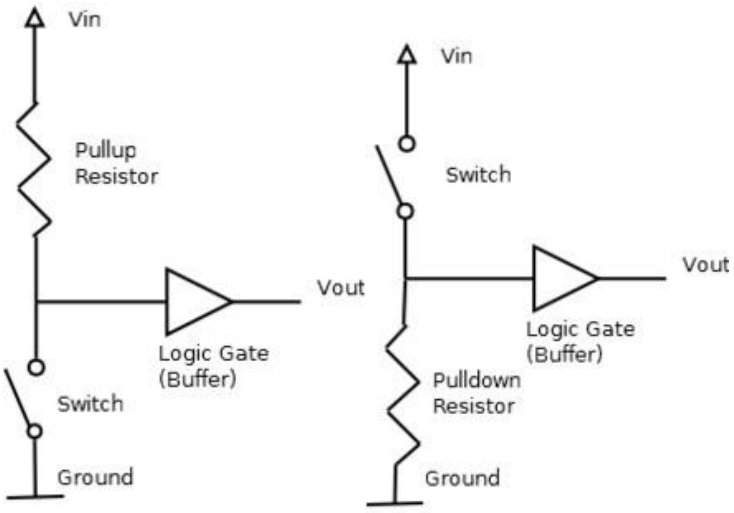


Fig. 11. 풀업저항과 풀다운 저항

풀업저항은 다른 부품이 동작하고 있지 않을 때 그 전원 공급선과 연결되어 있기 때문에 전압을 약하게 당깁니다. 스위치가 눌려지고 전류가 흐르면 그것은 매우 높은 저항이 되며, 끊어진 것처럼 동작합니다. 다른 회로가 끊어진 것처럼 동작한 후에는 회로가 끊어진 것처럼 동작합니다. 그리고 풀업저항이 높은 전압을 가져옵니다. 다른 회로가 동작할 때 그것은 풀업저항이 만들어 낸 높은 전압을 무시합니다. 풀업저항은 동작하지 않는 장치가 연결되어 있는지 까지 알아 낼 수 있습니다.

풀다운 저항은 똑같이 작동하지만 ground 와 연결되어 있습니다. 그것은 동작하지 않는 장치가 연결되어 있을 때 논리신호 0 근처에 있습니다. 풀다운과 풀업저항의 값은 연결된 장치에 매우 깊게 관여되어 있습니다.

• 버튼입력

준비물 : Uno R3, USB cable, push button, 10K 저항, red LED, 220Ohm 저항, 점퍼선

배경 : 푸쉬버튼이 열려 있을 때(안눌려진 상태) 버튼의 두 다리 사이는 연결되지 않은 상태입니다. 풀다운 저항을 쓰지 않으면, 핀의 논리적인 값은 매우 애매모호 합니다. 아두이노의 핀이 LOW 신호라는 확신을 주기 위해 10K 저항을 풀다운 저항으로 사용합니다. 풀업저항이 HIGH 상태를 유지 해 줄 것이고 버튼이 눌리게 되면 LOW 가 될 것입니다.

LED 는 Light Emitting Diode 의 약자입니다. 여기서 우리는 LED 가 Diode 의 일종인 것을 알 수 있습니다. 다

이오드는 오직 1 방향으로 전류가 흐르도록 합니다. 만약 반대로 연결하면 동작하지 않을 것입니다.

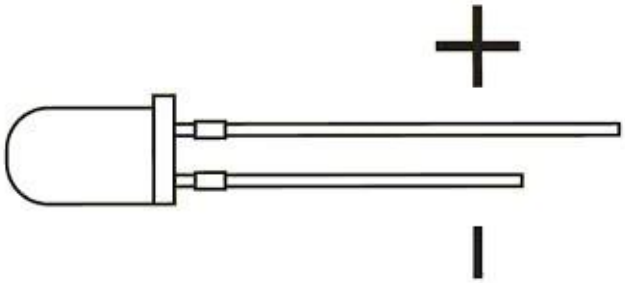
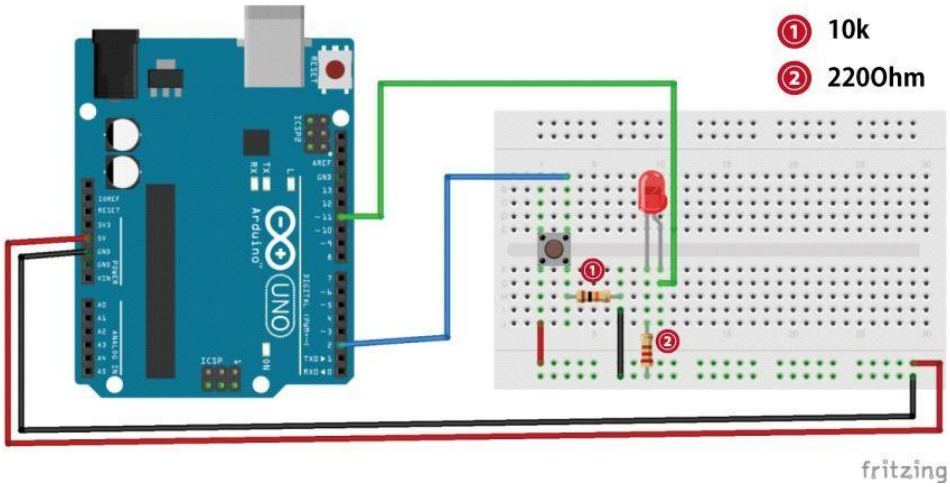


Fig. 12. LED 의 극성 : 긴 쪽은 anode(+) 짧은 쪽은 cathode(-)

LED 의 극성은 위와 같습니다. 긴쪽이 anode(+) 짧은쪽이 (-)입니다. 만약 LED 가 전원부와 그라운드로 바 로 연결되어 있다면 많은 양의 전류가 흐를 것이고, LED 는 손상을 입을 것입니다. 이걸 방지하기 위해서 전류를 제한하기 위한 저항이 필요합니다. 다음 챕터에서 어떻게 저항값을 결정하는지 배울 것입니다.

회로 :



코드(button_led):

```
int buttonpin = 2;
int LED = 11;

void setup()
{
  pinMode(buttonpin, INPUT);
  pinMode(LED, OUTPUT);
}
void loop()
{
  int buttoninput = digitalRead(buttonpin);
```

```
 if (buttoninput == 1)
 {
  digitalWrite(LED, HIGH);
 }
 else
 {
  digitalWrite(LED, LOW);
 }
}
```

결과 :

LED 는 푸쉬버튼이 눌려져 있을 때 켜져 있을 것이고, 아닐 경우에는 꺼질 것 입니다.

• 어떻게 저항값을 결정하는가?

LED 들을 사용 할 때, 저항값을 찾아 낼 필요가 있습니다. 간단하게 다음의 규칙을 보자면, 저항값을 계산할 수 있습니다. 그 전에 데이터시트나 설명서에서 오는 정보(공급 전압, LED 전압, LED 전류)를 알 필요가 있습니다. 아래는 스파크펀의 5mm 의 빨간 LED 의 데이터시트입니다.

Absolute Maximum Ratings: (Ta=25℃) .

ITEMS	Symbol	Absolute Maximum Rating	Unit
Forward Current	I_F	20	mA
Peak Forward Current	I_{FP}	30	mA
Suggestion Using Current	I_{su}	16-18	mA
Reverse Voltage (V_R=5V)	I_R	10	uA
Power Dissipation	P_D	105	mW
Operation Temperature	T_{OPR}	-40 ~ 85	℃
Storage Temperature	T_{STG}	-40 ~ 100	℃
Lead Soldering Temperature	T_{SOL}	Max. 260℃ for 3 Sec. Max. (3mm from the base of the expoxy bulb)	

Absolute Maximum Ratings: (Ta=25℃)

ITEMS	Symbol	Test condition	Min.	Typ.	Max.	Unit
Forward Voltage	V_F	I_F=20mA	1.8	---	2.2	V
Wavelenength (nm) or TC(k)	$\Delta\lambda$	I_F=20mA	620	---	625	nm
*Luminous intensity	I_V	I_F=20mA	150	---	200	mcd
50% Viewing Angle	2θ 1/2	I_F=20mA	40	---	60	deg

만약 우리가 Uno 를 사용한다면, 공급전압은 5V 이며 LED 는 약 2V 와 20mA 를 소비할 것 입니다. 옴의 법칙을 보면 $R = \frac{V}{I}$

그리고

$$R = \frac{(공급\ 전압 - LED\ 전압)}{전류} = \frac{5-2}{0.02} = 150ohm$$

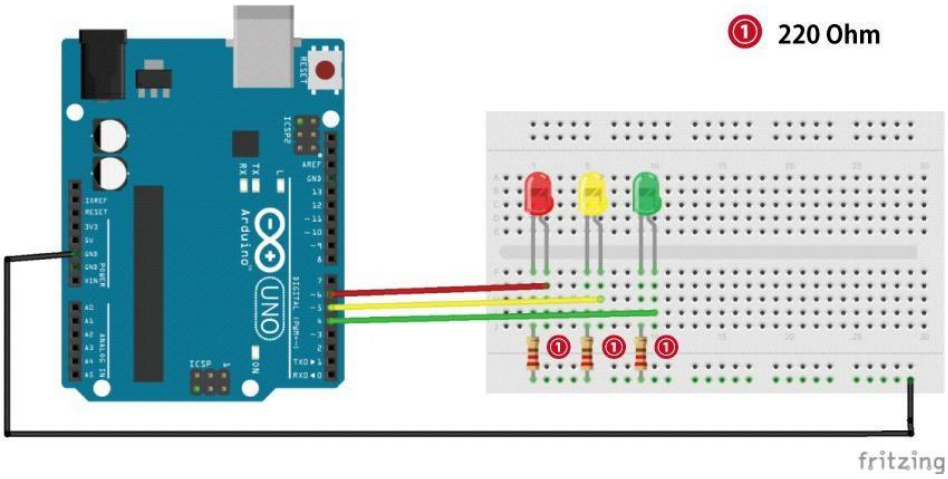
즉, 150 옴 저항을 사용하거나 그와 비슷한 저항을 사용하면 무리가 없을 것입니다.

• 신호등 만들기

준비물 : Uno R3, USB cable, 브레드보드, 빨강 노랑 초록 LED, 220Ohm 저항 (3 개), 점퍼선

3 개의 LED 와 저항을 이용하여 신호등을 만들 것 입니다. 빨강은 5 초, 노랑은 1 초, 초록은 5 초입니다.

회로:



코드(traffic_light):

```
int REDpin = 6;
int YELLOWpin = 5;
int GREENpin = 4;
void setup()
{
  pinMode(REDpin, OUTPUT);
  pinMode(YELLOWpin, OUTPUT);
  pinMode(GREENpin, OUTPUT);
}
void loop()
{
  digitalWrite(REDpin, HIGH);
  delay(5000);
  digitalWrite(REDpin, LOW);
  digitalWrite(YELLOWpin, HIGH);
  delay(1000);
  digitalWrite(YELLOWpin, LOW);
  digitalWrite(GREENpin, HIGH);
  delay(5000);
  digitalWrite(GREENpin, LOW);
}
```

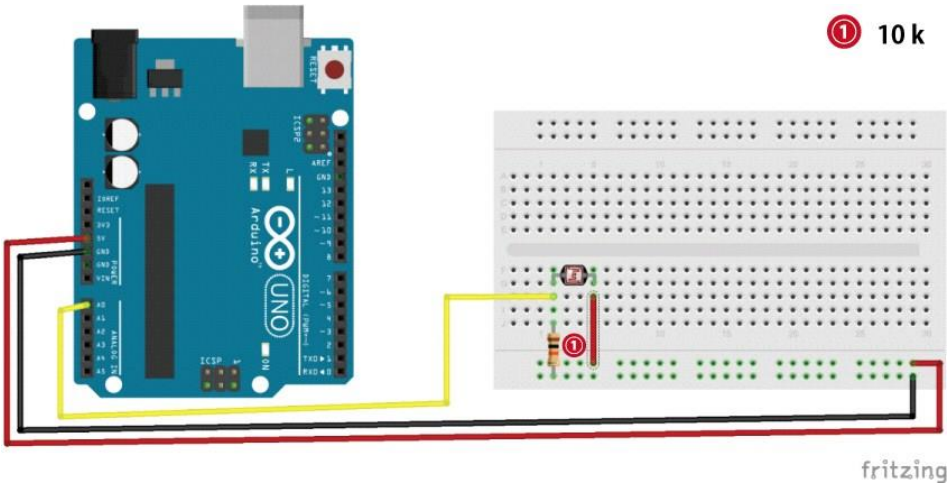
• 아날로그 센서

• 조도센서

준비물 : Uno R3, USB cable, 브레드보드, 황화 카드뮴 셀, 10K 저항, 점퍼선

배경 : 황화 카드뮴 셀(이하 CdS)은 저항의 일종이며 광전자를 이용한 반도체 효과를 이용하여 외부 빛의 조도에 의해 저항값이 결정됩니다. 빛이 강해지면 저항값이 약해집니다.

회로 : 밝기는 시리얼 모니터 기능에 의해 관찰될 것입니다. 아래와 같이 회로를 구성하고 코드를 업로드 합니다. 코드가 업로드 되면 시리얼 모니터를 엽니다.



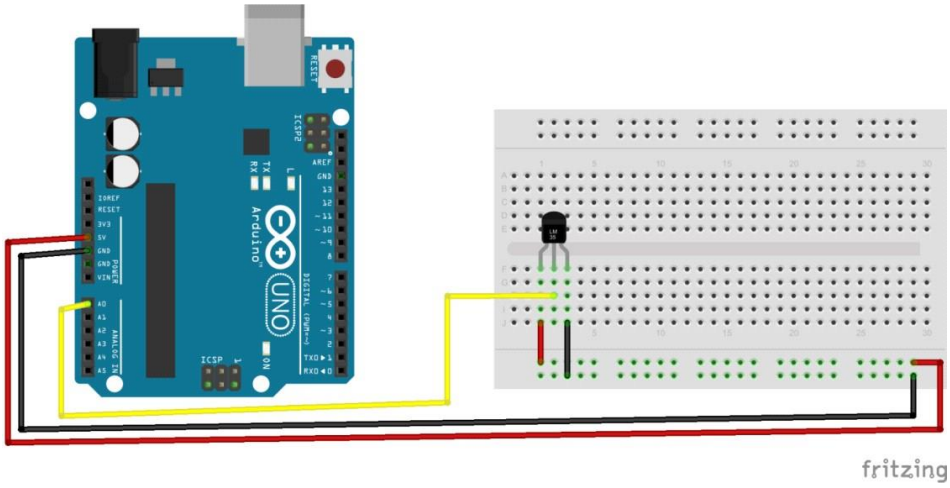
코드(CdS):

```
 int potpin = 0; //set analog interface #0 to connect photocell
int ledpin=13;
int val = 0; //define variable val
void setup()
{
  pinMode(ledpin, OUTPUT); //set digital interface #11 as output
  Serial.begin(9600);//set baud rate as 9600
}
void loop()
{
  val = analogRead(potpin); //read analog
  Serial.println(val);
  delay(10);//delay 0.01 s
}
```

• 온도 측정하기

준비물 : Uno R3, USB cable, 브레드보드, LM35 온도센서, 점퍼선

배경 : LM35 는 사용하기 매우 간편한 온도 센서입니다. -55 도에서 +150 도 사이로 0.5 도 간격으로 아날로 그 출력을 내보냅니다. **회로:**



코드(LM35):

```
int potPin = 0; //Set analog interface #0 accessed to LM35
void setup()
{
  Serial.begin(9600);//Set baud rate as 9600
}
void loop()
{
  int val;//
  int dat;//define variable
  val = analogRead(0); // read the analog value and assign to val
  dat = (125 * val) >> 8; //temperature calculation formula
  Serial.print("Tep:");
  Serial.print(dat);//output dat value
  Serial.println("C");//output character string C
  delay(500);//delay 0.5 s
}
```

• 포텐시오미터로 LED 밝기 조절하기

준비물 : Uno R3, USB cable, 브레드보드, 포텐시오미터, 220Ohm 저항,빨간 LED, 점퍼선

배경 : 포텐시오미터로부터 아날로그 값을 읽은 후 PWM(Pulse Width Modulation)을 이용하여 LED 의 밝기를 제어 할 것입니다. PWM 은 펄스의 폭을 컨트롤 하는 주기 제어방법입니다. “On”되는 시간에 따라 그 주기가 달라집니다. 주기가 낮다면 그에 따라 전압이 약해집니다. 왜냐하면 전압이 꺼지는 시간이 대부분이기 때문입니다. 다음 그림은 주기를 퍼센트로 나타 낸 것입니다.

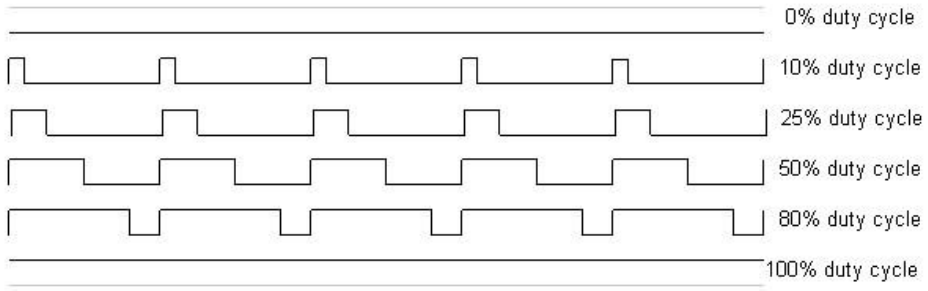
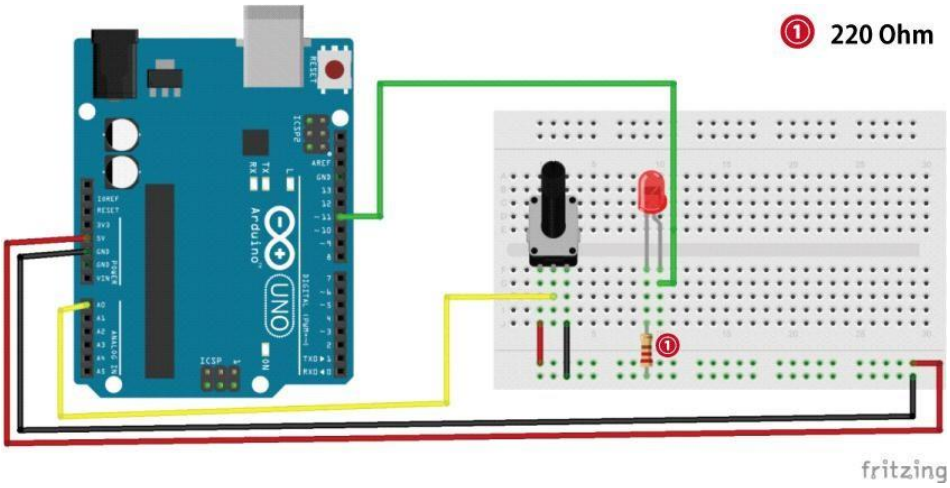


Fig. 13. PWM 주기 기간

PWM 의 사용 용도:

- LED 흐리게 하기
- 아날로그 출력하기
- 오디오 신호 만들기
- 모터 공급용 속도조절하기

회로: 포텐시오미터로 LED 의 밝기 제어하기



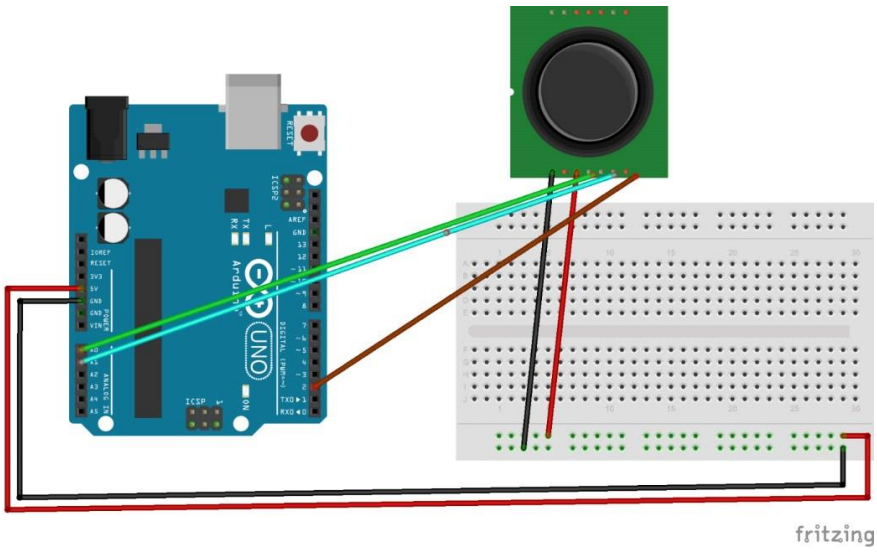
코드(pot_led):

```
int potpin = 0; //Define analogue interface.#0
int ledpin = 11; //Define digital interface #11.(PWM output)
int val = 0; //temporary storage the Variable value from the sensor
void setup()
{
  pinMode(ledpin, OUTPUT); // Define digital interface #11 as output
  Serial.begin(9600);//Setup Baud rate as 9600
}
void loop()
{
  val = analogRead(potpin); // Read analogue value from sensor and assign to val
  Serial.println(val);//Show val variable.
  analogWrite(ledpin, val / 4);
  delay(10);//Delay 0.01 sec.
}
```

- **PS2 Joystick module**

준비물 : Uno R3, USB cable, 브레드보드, PS2 joystick module, 점퍼선

배경 : PS2 joystick module 은 RC 자동차나 게임장치로 사용 할 수 있습니다. 이것은 x 와 y 방향으로 2 개의 아날로그 신호를 출력하며 1 개의 출력버튼이 있습니다. **회로 :** 5 개의 점퍼선이 조이스틱과 아두이노를 연결합니다. 각각 %v, GND, X, Y, 버튼을 의미합니다.



코드(joystick):

```
int xpin = A0;
int ypin = A1;
int buttonpin = 2;

void setup()
{
  Serial.begin(9600);
  pinMode(buttonpin, INPUT);
}
void loop()
{
  int xval = analogRead(xpin);
  int yval = analogRead(ypin);
  int buttonpressed = digitalRead(buttonpin);

  Serial.print(xval);
  Serial.print("   ");
  Serial.print(yval);
  Serial.print("   ");
  Serial.println(buttonpressed);
}
```

• 디지털 센서

• 기울기센서로 LED 켜기

준비물 : Uno R3, USB cable, 브레드보드, 기울기 센서, 빨간 LED, 220Ohm, 점퍼선

배경 : 내부에서 수은이 자유롭게 움직이며 2 개의 선을 건드립니다. 빽빽한수은방울은 진동하면서 저항을 전달합니다. 센서가 기울어지면 스위치가 열립니다.

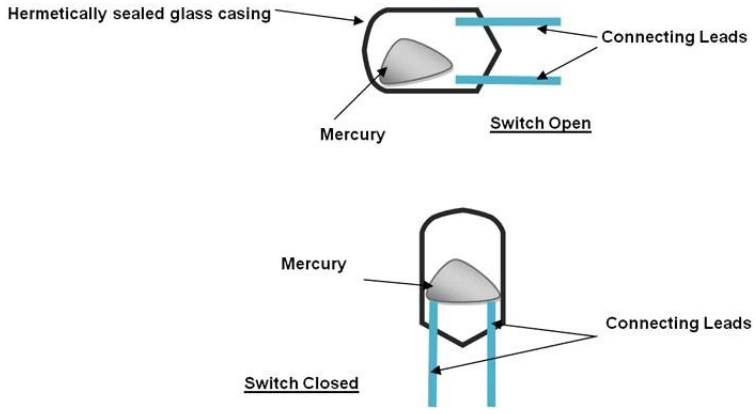
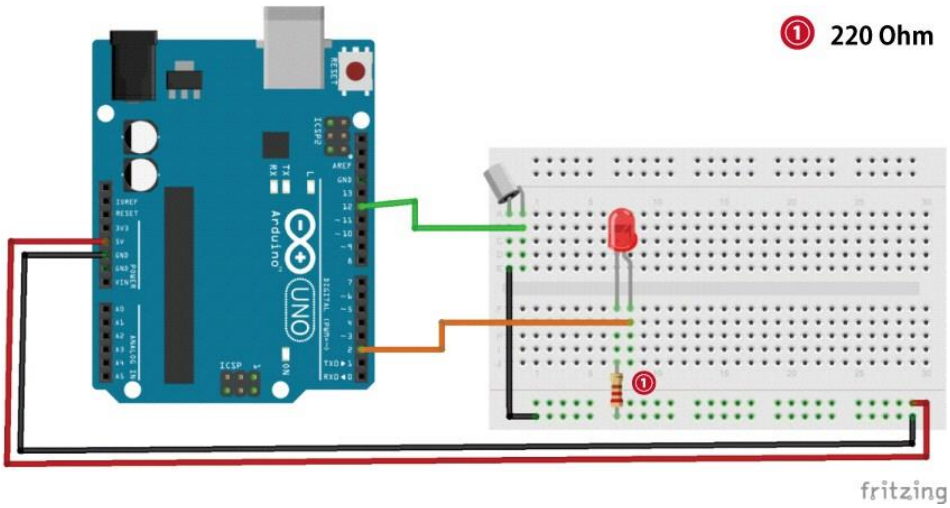


Fig. 14. 수은 기울기 센서 원리

회로: 센서가 기울어지면 LED 가 꺼집니다.



코드(tilt):

```
int LED = 2;
int tilt = 12;
void setup()
{
  pinMode(LED, OUTPUT); // set digital 2 for LED output
  pinMode(tilt, INPUT); // set digital 12 for tilt sensing
}
void loop()
{ if (digitalRead(tilt) == HIGH)
  {
    digitalWrite(LED, HIGH); //light up led
  }
  else//otherwise
  {
    digitalWrite(LED, LOW); //turn off led
  }
}
```

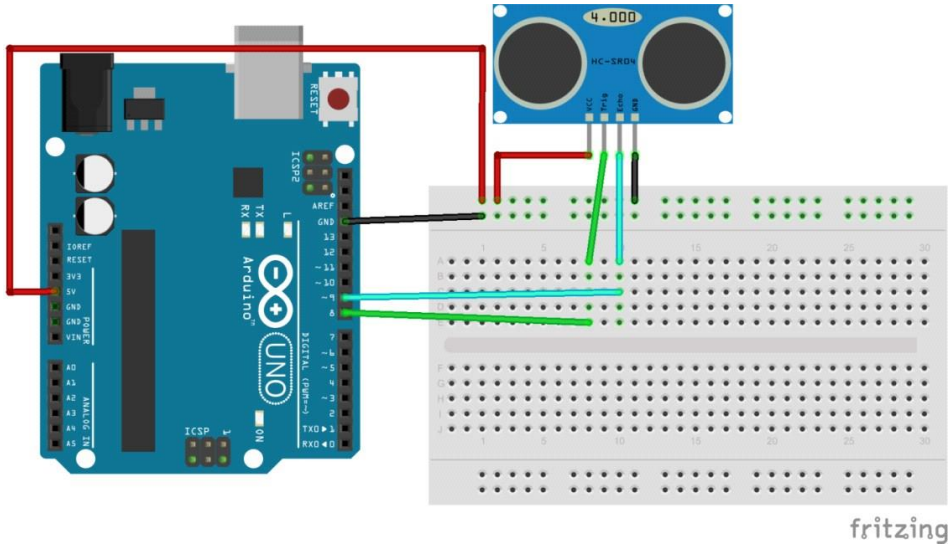
- **HC-SR04 초음파 거리센서**

준비물 : Uno R3, USB cable, 브레드보드, HC-SR04 초음파 거리센서, 점퍼선

배경 : 거리를 측정하는 것으로는 보통 적외선이나 초음파를 많이 사용합니다. 각각 센서들은 장점과 단점들이 있습니다. 그러나 이 HC-SR04 는 매우 싸며 유명한 거리센서입니다. 적외선과 초음파센서는 각각 빛과 소리를 보냅니다. 신호는 물체에 닿으면 반사되며 신호를 보내고 다시 받는 사이의 시간을 측정합

니다. 적외선 센서는 검은 물체에 작동하지 않으며 초음파센서는 흡수성의 물체(예로 스펀지)에 잘 작동하지 않습니다.

회로:



코드(ultra):

```
const int TriggerPin = 8;      //Trig pin const
int EchoPin = 9;         //Echo pin
long Duration = 0;

void setup() {
  pinMode(TriggerPin, OUTPUT); // Trigger is an output pin
  pinMode(EchoPin, INPUT);     // Echo is an input pin
  Serial.begin(9600);          // Serial Output
}

void loop() {
  digitalWrite(TriggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(TriggerPin, HIGH);          // Trigger pin to HIGH
  delayMicroseconds(10);                   // 10us high
  digitalWrite(TriggerPin, LOW);           // Trigger pin to HIGH

  Duration = pulseIn(EchoPin, HIGH);       // Waits for the echo pin to get high
  // returns the Duration in microseconds
  long Distance_mm = Distance(Duration);   // Use function to calculate the distance

  Serial.print("Distance = ");             // Output to serial
  Serial.print(Distance_mm);
  Serial.println(" mm");
  delay(1000);                             // Wait to do next measurement
}

long Distance(long time)
{
  // Calculates the Distance in mm
  // ((time)*(Speed of sound))/ toward and backward of object) * 10

  long DistanceCalc;                       // Calculation variable
  DistanceCalc = ((time / 2.9) / 2);    // Actual calculation in mm
  //DistanceCalc = time / 74 / 2;         // Actual calculation in inches
  return DistanceCalc;                     // return calculated value
}
```

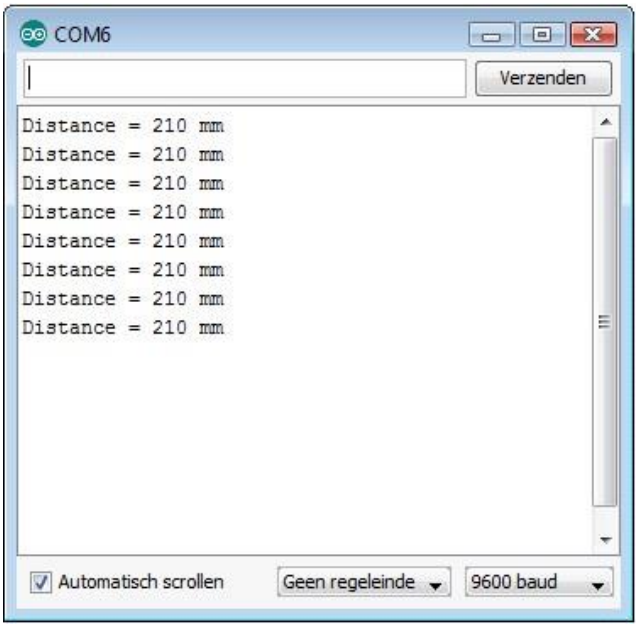


Fig. 15. 결과

- **PIR 모션 센서**

준비물 : Uno R3, USB cable, PIR motion sensor, M-F cable

배경 : PIR(수동 적외선 센서) 움직임 센서는 프리넬 렌즈를 이용하여 움직임을 찾아냅니다.

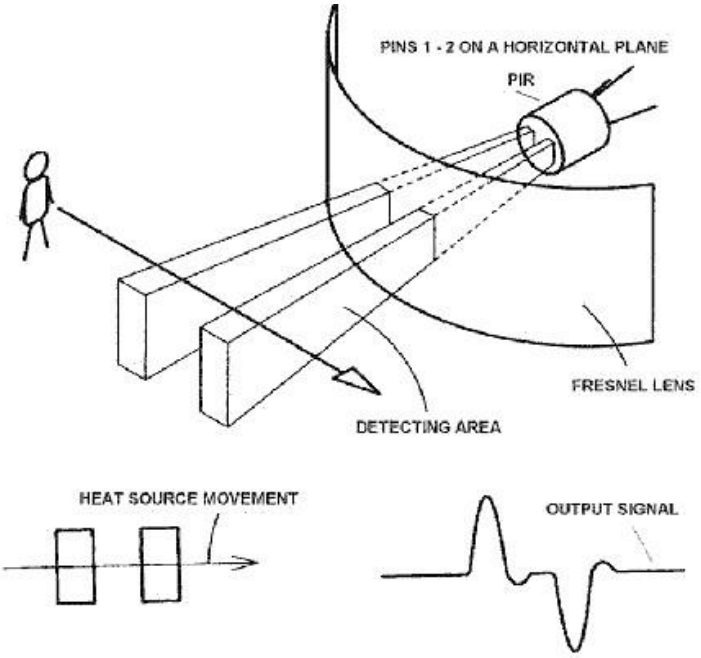


Fig. 16. PIR 움직임 센서의 원리

IR 센서는 노이즈, 온도, 습도에 대한 면역력을 키우기 위해 봉해진 금속으로 밀폐되어 있습니다. 적외선이 통과 가능한 물질(얻기 쉬워진 코팅된 실리콘)로 만들어진 창이 만들어 져 있습니다. 창 뒤에는 2 개의 안정된 센서가 있습니다.

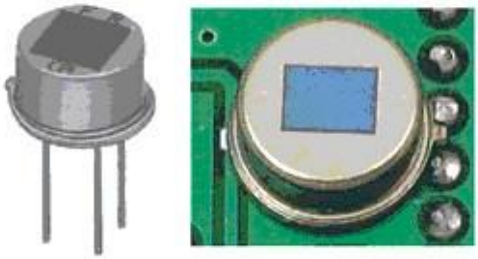


Fig. 17. PIR 센서 안의 적외선 센서

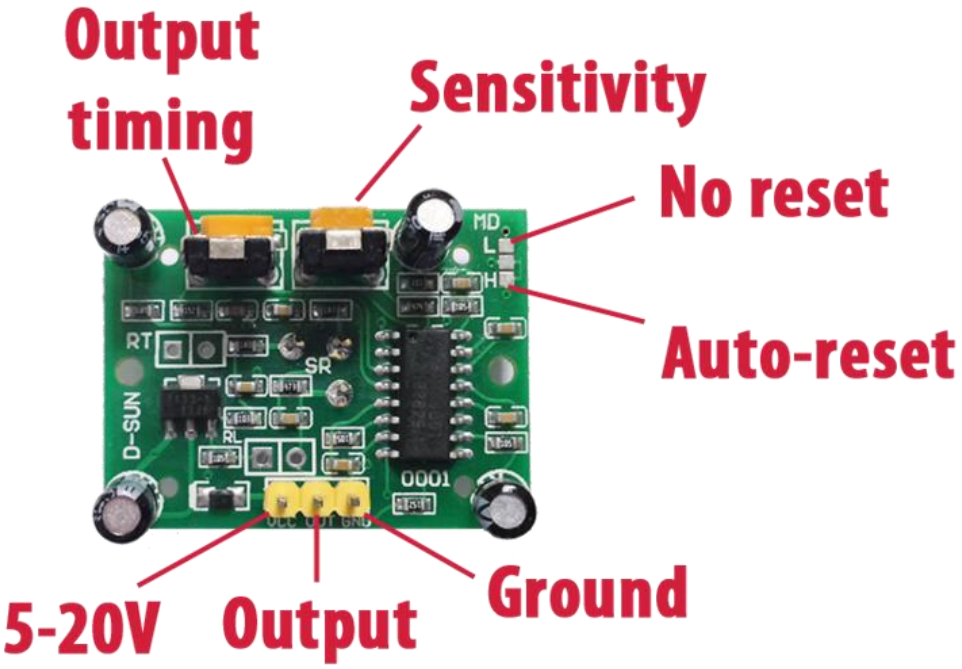


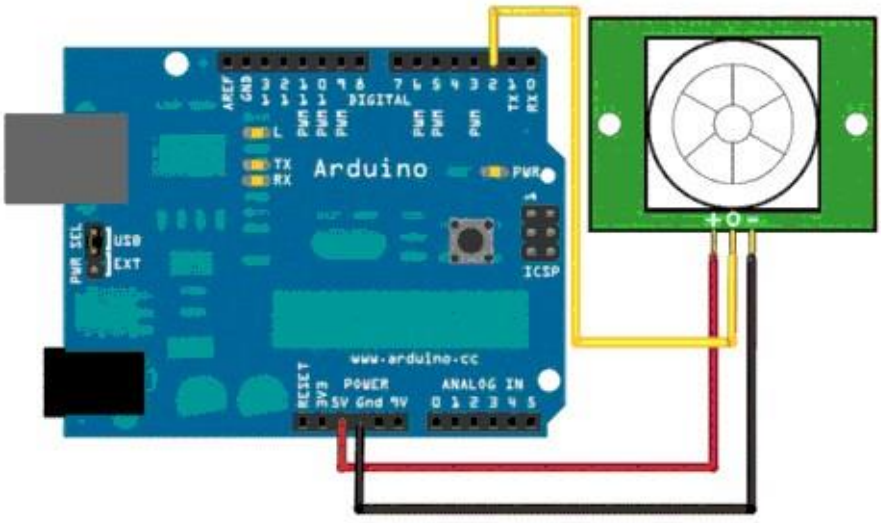
Fig. 18. PIR 센서의 뒷면

출력: 움직임이 감지되면 3V 의 HIGH 디지털 신호를 출력합니다. 감지가 되지 않으면 LOW 를 출력합니다. 신호출력 길이는 센서마다 다르며 PCB 의 저항과 콘덴서로 인해 결정됩니다.

감지 범위 : 20 feet(6 meter) 110° x 70°

공급 전원 : 3 – 9V 입력 전원, 5V 가 가장 이상적입니다.

회로:



코드(PIR):

```
/* PIR sensor tester*/

int ledPin = 13; // choose the pin for the LED
int inputPin = 2; // choose the input pin (for PIR sensor)
int pirState = LOW; // we start, assuming no motion detected
int val = 0; // variable for reading the pin status

void setup() {
  pinMode(ledPin, OUTPUT); // declare LED as output
  pinMode(inputPin, INPUT); // declare sensor as input

  Serial.begin(9600);
}

void loop() {
  val = digitalRead(inputPin); // read input value
  if (val == HIGH) { // check if the input is HIGH
    digitalWrite(ledPin, HIGH); // turn LED ON
    if (pirState == LOW) { // we have just turned on
      Serial.println("Motion detected!");
      // We only want to print on the output change, not state
      pirState = HIGH;
    }
  }
  else {
    digitalWrite(ledPin, LOW); // turn LED OFF
    if (pirState == HIGH) { // we have just turned of
      Serial.println("Motion ended!");
      // We only want to print on the output change, not state
      pirState = LOW;
    }
  }
}
```

- **<u>QRD1114 짧은 거리센서</u>**

준비물 : Uno R3, USB cable, 브레드보드, QRD1114, 220Ohm 저항, 10K Ohm 저항, 점퍼선

배경 : QRD1114 는 2 개의 부품으로 되어 있습니다. 하나는 IR 발신기이며 나머지는 포토 트렌지스터입니다. 두개가 1 세트에 들어가 있습니다. 이것은 GP2Y0A21YK 라는 센서와 비슷하지만 이것은 IR 빛이 빛나며 얼마나 튀었는지 알 수 있습니다. 물체가 가까울수록 많은 빛이 튀게 됩니다. QRD1114 는 오직 0 – 3 cm 안에서 감지가 가능하며 아두이노에서 값은 600 ~ 1024 의 값을 내보냅니다.

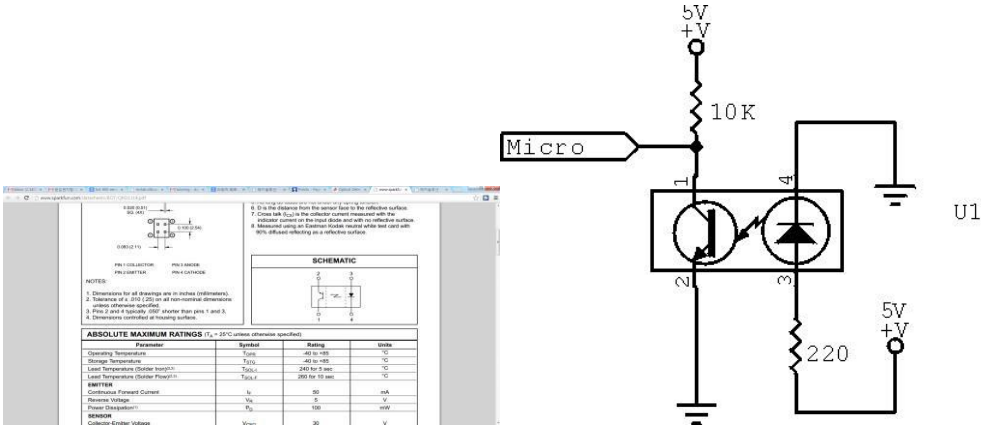
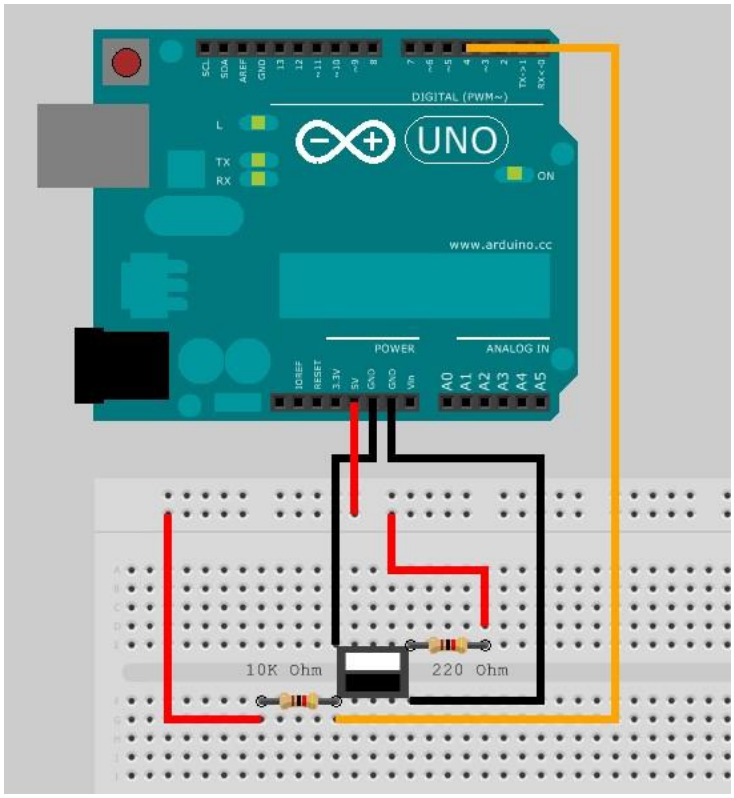


Fig. 19. QRD1114 의 내부 구성도와 간단한 회로

회로:



코드(qrd1114):

```
int signal = 4;
int onoff;

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

void loop()
{
  onoff = digitalRead(signal);
  Serial.println(onoff);
}
```

- **디스플레이**

  - **라이브러리 추가하기**

한번쯤은 아두이노 소프트웨어가 만들어진 함수를 사용하기에 편하다는 생각을 해 보았을 겁니다. 아마 추가 라이브러리로 당신의 아두이노의 능력을 더 확장시키고 싶을 것입니다.

라이브러리들은 센서에 연결을 쉽게하거나, 디스플레이, 모듈등을 사용하기 쉽게 코드의 집합을 해 둔 것입니다. 예를 들어 “LiquidCrystal” 라이브러리는 LCD 디스플레이를 사용하기 쉽게 만들어 둔 것입니다.

인터넷에서 다운로드 가능한 라이브러리만 수백개가 있습니다. 여기서 언급한 것은 외부 라이브러리를 어떻게 추가하는지 입니다. **어떻게 라이브러리를 추가하는가?**

라이브러리는 보통 “zip”형태로 배포됩니다. 예를 들어 나중에 필요할 NeoPixel 의 라이브러리입니다.

NeoPixel (https://learn.adafruit.com/adafruit-neopixel-uberguide/overview)

라이브러리는 밑의 웹사이트에 ZIP 으로 저장되어 있습니다.

https://github.com/adafruit/Adafruit_NeoPixel

ZIP 파일을 받을 수 있으며 압축 해제한 파일입니다. 그러면 아래의 파일이 나올 것입니다.

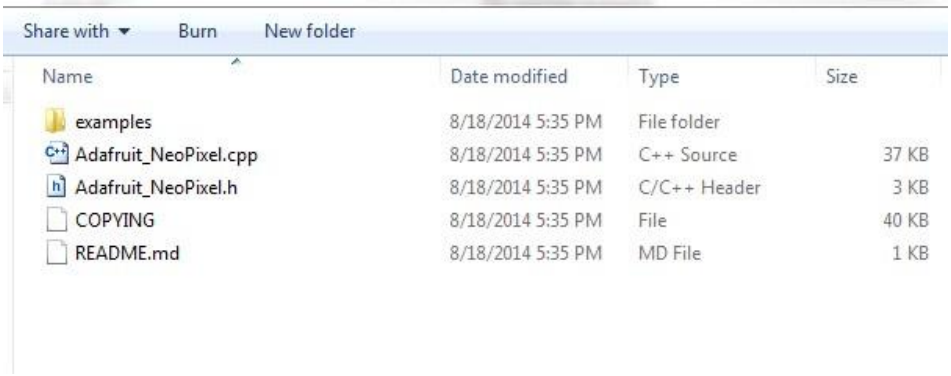


Fig. 20. 압축 해제된 라이브러리 폴더

라이브러리를 사용하기 전에 폴더 명이 기본 문자들과 숫자로 이루어 져 있는지 확인해야 합니다. Adafruit_NeoPixel-master 는 이름을 바꾸거나, _와 –를 제거(AdafruitNeoPixelmaster) 해 주어야 합니다. 그리고 Arduino – libraries 폴더에 폴더를 복사합니다.
폴더를 복사한 후 라이브러리 설치를 위해 아두이노 소프트웨어를 다시 시작합니다. 그러면 파일 – 예제에서 설치된 라이브러리와 예제를 볼 수 있습니다.

- **16x2 LCD**

준비물 : Uno R3, USB cable, 브레드보드, 16x2 LCD, 암 헤더핀, 포텐시오미터, 점퍼선

배경 : 16x2 LCD 디스플레이는 각각 줄에 16 개씩 32 개의 문자를 표현 할 수 있습니다. 그것은 0 ~ 255 사이의 ASCII 코드의 문자면 전부 표현이 가능합니다. 모듈의 뒷면에는 PCB 가 붙어 있으며 신호를 처리할 수 있게 회로들이 포함되어 있습니다. 회로의 주요 부품은 저장공간과 제어기구입니다.

Fig. 21. 16x2 LCD 의 뒷부분

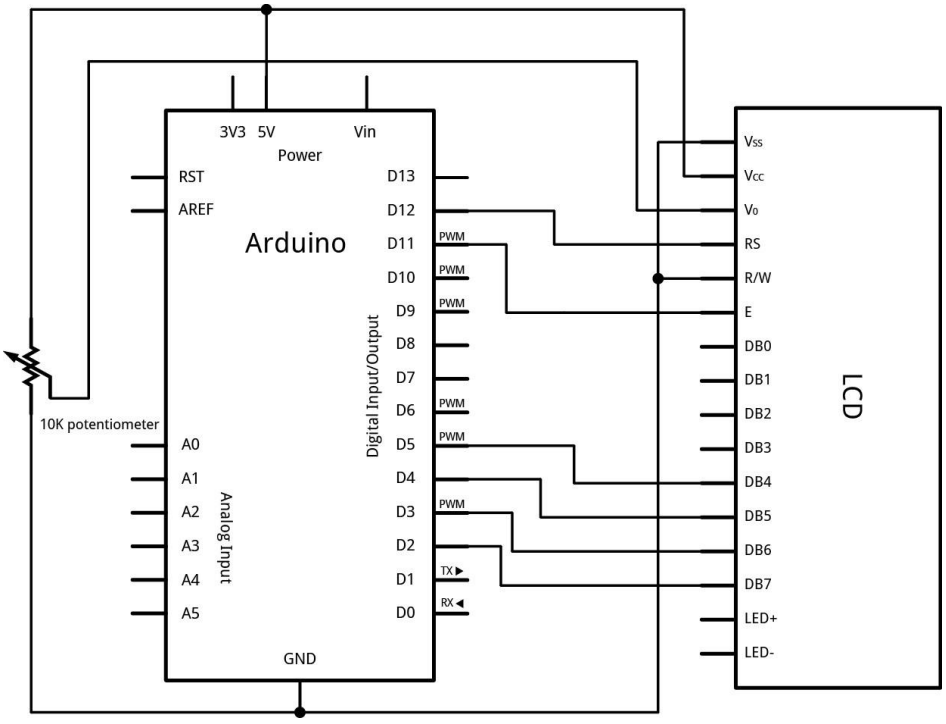
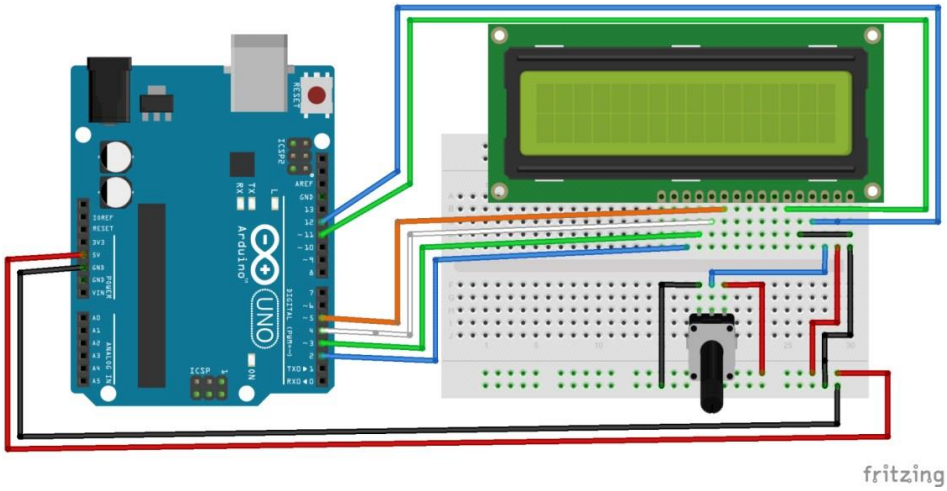


Fig. 22. 아두이노와 LCD 모듈의 배선도

회로:



코드(lcd):

```
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```

- **74HC595 시프트 레지스터를 이용한 7 세그먼트**

준비물 : Uno R3, USB cable, 브레드보드, 220Ohm 저항, 7-segment, 74HC595 시프트 레지스터, 점퍼선

배경 :

7-Segment 는 2 개의 종류가 있습니다. 1 개는 공통 캐소드 타입, 나머지는 공통 애노드 타입입니다.

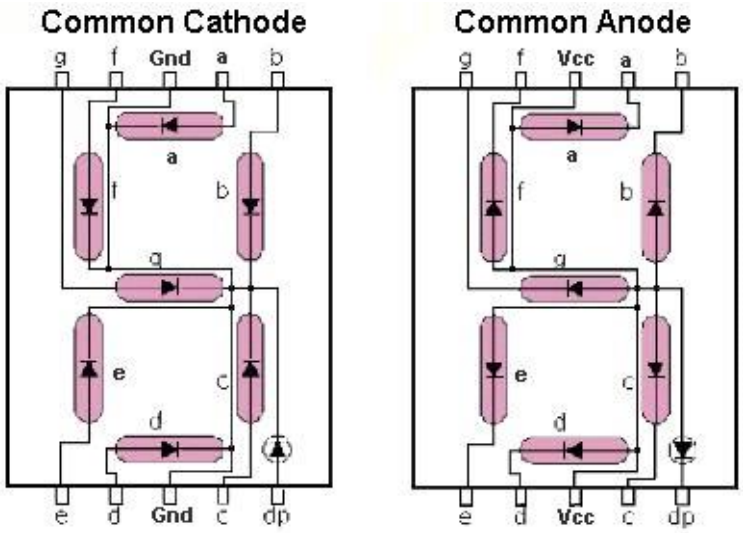


Fig. 23. 7-Segment 의 2 가지 종류

우리 키트에서는 공통 캐소드타입이 포함되어 있습니다. 내부는 아래의 그림과 같습니다.

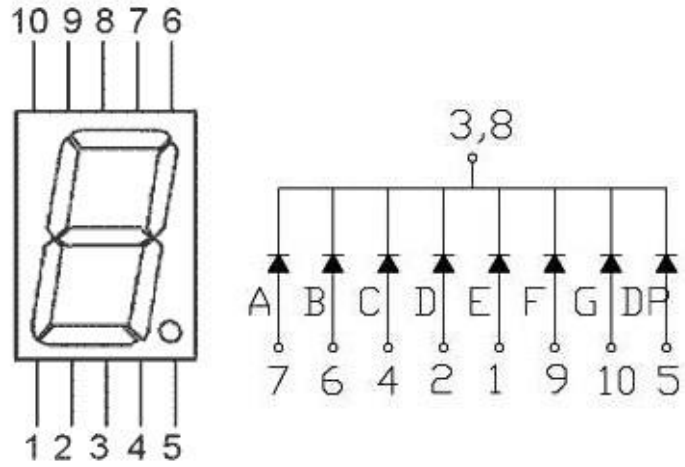


Fig. 24. 7-Segment 의 공통 캐소드타입

74HC595 는 8 비트의 시프트레지스터입니다. 7-Segment 나 8-Segment(점 포함)에 7 개 혹은 8 개의 디지털 포트가 필요하고 이것을 74HC959 가 3 개의 핀(datain, latch, clk)을 이용하여 8 비트로 만들어 줍니다.

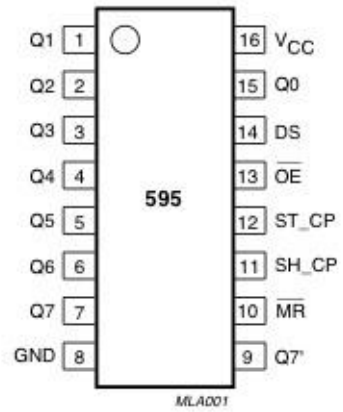
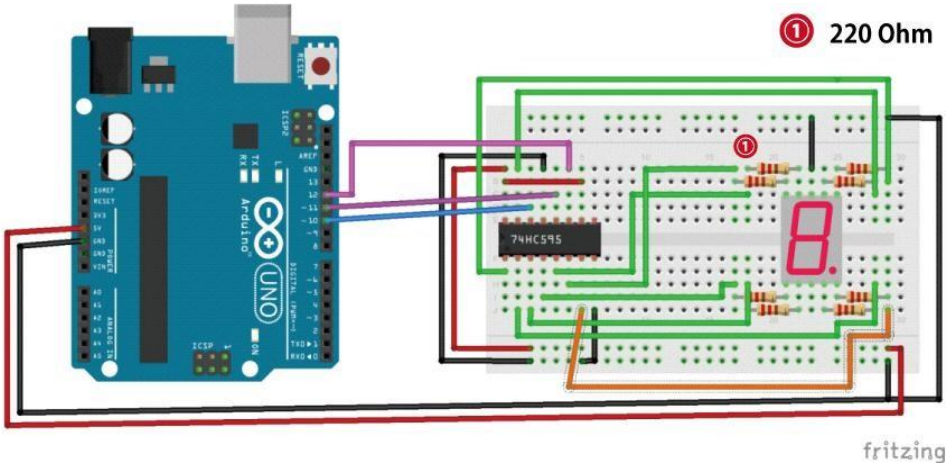


Fig. 25. 74HC595 시프트 레지스터의 핀맵

회로: 이 회로에서는 0 ~ 9 까지 출력합니다.



코드(_7-seg):

```
int dataPin = 10;
int latchPin = 11;
int clockPin = 12;

byte dec_digits[] = {
  0b00111111, 0b00000110, 0b01011011, 0b01001111, 0b01100110, 0b01101101, 0b01111100, 0b00000111,
0b01111111, 0b01100111
};

void setup() {
  //set pins to output so you can control the shift register
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
}

void loop() {
  for (int numberToDisplay = 0; numberToDisplay < 10; numberToDisplay++) {     // take the latchPin low so
    // the LEDs don't change while you're sending in bits:
    digitalWrite(latchPin, LOW);     // shift out the bits:
    shiftOut(dataPin, clockPin, MSBFIRST, dec_digits[numberToDisplay]);      //take the latch pin high so the
LEDs will light up:
    digitalWrite(latchPin, HIGH);     // pause before next value:
    delay(300);
  }
}
```

- **7 세그먼트 4 개 사용하기**

준비물 : Uno R3, USB cable, 브레드보드, 4-세그먼트, 220Ohm 저항 8 개

배경 : 4-세그먼트는 총 12 개의 핀을 가지고 있습니다.

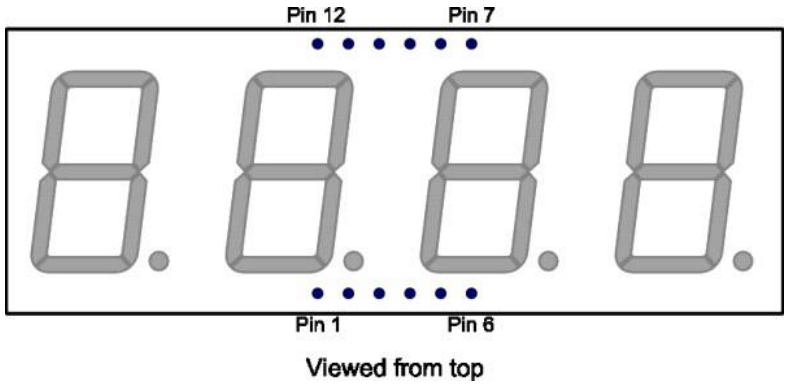
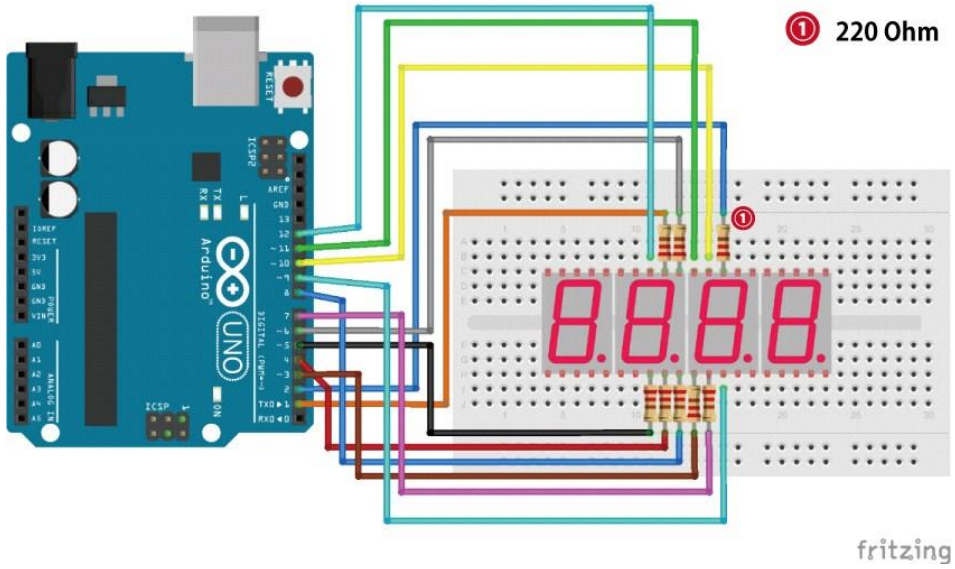


Fig. 26 4-세그먼트의 핀맵

회로:



코드(_4-seg):

```
//set the cathode interface
int a = 1;
int b = 2;
int c = 3;
int d = 4;
int e = 5;
int f = 6;
int g = 7;
int p = 8;
// set the anode interface
int d4 = 9;
int d3 = 10;
int d2 = 11;
int d1 = 12; //set variable
long n = 0;
int x = 100;
int del = 55; //fine-tuning value for
clock
void setup() {
  pinMode(d1, OUTPUT);
  pinMode(d2, OUTPUT);
  pinMode(d3, OUTPUT);
  pinMode(d4, OUTPUT);
  pinMode(a, OUTPUT);
  pinMode(b, OUTPUT);
  pinMode(c, OUTPUT);
  pinMode(d, OUTPUT);
  pinMode(e, OUTPUT);
  pinMode(f, OUTPUT);
  pinMode(g, OUTPUT);
  pinMode(p, OUTPUT);
}
void loop() {
  clearLEDs();
  pickDigit(1);
  pickNumber((n / x / 1000) % 10);
  delayMicroseconds(del);
  clearLEDs();
  pickDigit(2);
  dispDec();
  pickNumber((n / x / 100) % 10);
  delayMicroseconds(del);
  clearLEDs();
  pickDigit(3);
  pickNumber((n / x / 10) % 10);
  delayMicroseconds(del);
  clearLEDs();
  pickDigit(4);
  pickNumber(n / x % 10);
  delayMicroseconds(del);
  n++;
  if (digitalRead(13) == HIGH) {
    n = 0;
  }
```

```
}
void pickDigit(int x) //define
pickDigit(x),to open dx port
{
  digitalWrite(d1, HIGH);
  digitalWrite(d2, HIGH);
  digitalWrite(d3, HIGH);
  digitalWrite(d4, HIGH);
  switch (x) {

    case 1:
      digitalWrite(d1, LOW);
      break;
    case 2:
      digitalWrite(d2, LOW);
      break;
    case 3:
      digitalWrite(d3, LOW);
      break;
    default:
      digitalWrite(d4, LOW);
      break;
  }
}
void pickNumber(int x) //define
pickNumber(x)to display number x
{
  switch (x) {
    default:
      zero();
      break;
    case 1:
      one();
      break;
    case 2:
      two();
      break;
    case 3:
      three();
      break;
    case 4:
      four();
      break;
    case 5:
      five();
      break;
    case 6:
      six();
      break;
    case 7:
      seven();
      break;
    case 8:
      eight();
```

```
      break;
    case 9:
      nine();
      break;
  }
}
void dispDec() //set to start the
decimal point
{
  digitalWrite(p, HIGH);
}
void clearLEDs() //clear contents
{
  digitalWrite(a, LOW);
  digitalWrite(b, LOW);
  digitalWrite(c, LOW);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, LOW);
  digitalWrite(p, LOW);
}
void zero() //define 0 as cathode pin
switch
{
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, HIGH);
  digitalWrite(g, LOW);
}
void one() // define 1 as cathode pin
switch
{
  digitalWrite(a, LOW);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, LOW);
}
void two() // define 2 as cathode pin
switch
{
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, LOW);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, LOW);
  digitalWrite(g, HIGH);
```

```
}
void three() // define 3 as cathode
pin switch
{
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, HIGH);
}
void four() // define 4 as cathode pin
switch
{
  digitalWrite(a, LOW);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
}
void five() // define 5 as cathode pin
switch
{
  digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, LOW);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
}
void six() // define 6 as cathode pin
switch
{
  digitalWrite(a, HIGH);
  digitalWrite(b, LOW);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
}
void seven() // define 7 as cathode
pin switch
{
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
```

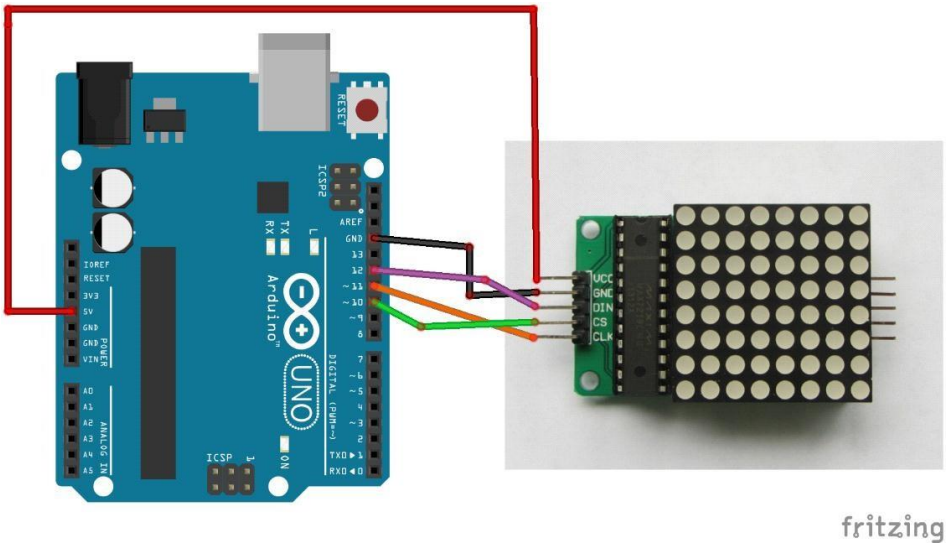
```
  digitalWrite(g, LOW);
}
void eight() // define 8 as cathode
pin switch
{
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
}
void nine() // define 9 as cathode pin
switch
{
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);
  digitalWrite(e, LOW);
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
}
```

- **MAX7219 를 이용한 8x8 도트매트릭스**

준비물 : Uno R3, USB cable, 8x8 도트매트릭스 키트, 암-암 케이블, 점퍼선

배경 : MAX7219 는 도트매트릭스를 제어하기에 정말 좋습니다. 3 개의 핀(DIN, CLK, LOAD 혹은 CS)를 사용하여 64 개의 LED 와 막대그래프, 8 개의 디지털 제어가 가능합니다. 이 튜토리얼에서는 8x8 도트매트릭스를 사용할 것이며 MAX7219, 저항, 콘덴서가 장착되어 있습니다.

회로 : 보드와 바로 연결해야 하기 때문에 암-암 케이블과 점퍼선을 사용해야 합니다. 우노 헤더와 매트릭스 키트를 바로 연결합니다.



배선도 ;

Uno R3	**MAX7219 LED matrix kit**
5V	VCC
GND	GND
D12	DIN
D10	CS
D11	CLK

코드(dotmatrix):

```
unsigned char i;
unsigned char j;
/*Port Definitions*/
int Max7219_pinCLK = 11;
int Max7219_pinCS = 10;
int Max7219_pinDIN = 12;

int vol;
int val;

unsigned char disp1[65][8] = { {
    0xFF,
    0xFF,
    0xFF,
    0xFF,
    0xFF,
    0xFF,
    0xFF,
    0xFF
  }, //0
  {
    0xFE,
    0xFF,
    0xFF,
    0xFF,
    0xFF,
    0xFF,
    0xFF,
    0xFF
  }, //0
  {
    0xFC,
    0xFF,
    0xFF,
    0xFF,
    0xFF,
    0xFF,
    0xFF,
    0xFF
  }, //0
  {
    0xF8,
    0xFF,
    0xFF,
    0xFF,
    0xFF,
    0xFF,
    0xFF,
    0xFF
  }, //0
  {
    0xF0,
    0xFF,
    0xFF,
```

```
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
 }, //0
 {
  0xE0,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
 }, //0
 {
  0xC0,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
 }, //0
 {
  0x80,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
 }, //0
 {
  0x00,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
 }, //0

 {
  0x00,
  0x7F,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
```

```
  0xFF,
  0xFF
}, //0
{
  0x00,
  0x3F,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
}, //0
{
  0x00,
  0x1F,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
}, //0
{
  0x00,
  0x0F,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
}, //0
{
  0x00,
  0x07,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
}, //0
{
  0x00,
  0x03,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
}, //0
{
```

```
  0x00,
  0x01,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
}, //0
{
  0x00,
  0x00,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
}, //0

{
  0x00,
  0x00,
  0xFE,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
}, //0
{
  0x00,
  0x00,
  0xFC,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
}, //0
{
  0x00,
  0x00,
  0xF8,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
}, //0
{
  0x00,
  0x00,
  0xF0,
```

```
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
 }, //0
 {
  0x00,
  0x00,
  0xE0,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
 }, //0
 {
  0x00,
  0x00,
  0xC0,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
 }, //0
 {
  0x00,
  0x00,
  0x80,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
 }, //0
 {
  0x00,
  0x00,
  0x00,
  0xFF,
  0xFF,
  0xFF,
  0xFF,
  0xFF
 }, //0

 {
  0x00,
  0x00,
  0x00,
  0x7F,
  0xFF,
  0xFF,
```

```
  0xFF,
  0xFF
 }, //0
 {
  0x00,
  0x00,
  0x00,
  0x3F,
  0xFF,
  0xFF,
  0xFF,
  0xFF
 }, //0
 {
  0x00,
  0x00,
  0x00,
  0x1F,
  0xFF,
  0xFF,
  0xFF,
  0xFF
 }, //0
 {
  0x00,
  0x00,
  0x00,
  0x0F,
  0xFF,
  0xFF,
  0xFF,
  0xFF
 }, //0

 {
  0x00,
  0x00,
  0x00,
  0x07,
  0xFF,
  0xFF,
  0xFF,
  0xFF
 }, //0
 {
  0x00,
  0x00,
  0x00,
  0x03,
  0xFF,
  0xFF,
  0xFF,
  0xFF
 }, //0
```

```
{
 0x00,
 0x00,
 0x00,
 0x01,
 0xFF,
 0xFF,
 0xFF,
 0xFF
}, //0
{
 0x00,
 0x00,
 0x00,
 0x00,
 0xFF,
 0xFF,
 0xFF,
 0xFF
}, //0

{
 0x00,
 0x00,
 0x00,
 0x00,
 0xFE,
 0xFF,
 0xFF,
 0xFF
}, //0
{
 0x00,
 0x00,
 0x00,
 0x00,
 0xFC,
 0xFF,
 0xFF,
 0xFF
}, //0
{
 0x00,
 0x00,
 0x00,
 0x00,
 0xF8,
 0xFF,
 0xFF,
 0xFF
}, //0
{
 0x00,
 0x00,
```

```
  0x00,
  0x00,
  0xF0,
  0xFF,
  0xFF,
  0xFF
 }, //0
 {
  0x00,
  0x00,
  0x00,
  0x00,
  0xE0,
  0xFF,
  0xFF,
  0xFF
 }, //0
 {
  0x00,
  0x00,
  0x00,
  0x00,
  0xC0,
  0xFF,
  0xFF,
  0xFF
 }, //0
 {
  0x00,
  0x00,
  0x00,
  0x00,
  0x80,
  0xFF,
  0xFF,
  0xFF
 }, //0
 {
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0xFF,
  0xFF,
  0xFF
 }, //0

 {
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
```

```
  0x7F,
  0xFF,
  0xFF
}, //0
{
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x3F,
  0xFF,
  0xFF
}, //0
{
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x1F,
  0xFF,
  0xFF
}, //0
{
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x0F,
  0xFF,
  0xFF
}, //0
{
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x07,
  0xFF,
  0xFF
}, //0
{
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x03,
  0xFF,
  0xFF
}, //0
```

```
  {
   0x00,
   0x00,
   0x00,
   0x00,
   0x00,
   0x01,
   0xFF,
   0xFF
  }, //0
  {
   0x00,
   0x00,
   0x00,
   0x00,
   0x00,
   0x00,
   0xFF,
   0xFF
  }, //0

  {
   0x00,
   0x00,
   0x00,
   0x00,
   0x00,
   0x00,
   0xFE,
   0xFF
  }, //0
  {
   0x00,
   0x00,
   0x00,
   0x00,
   0x00,
   0x00,
   0xFC,
   0xFF
  }, //0
  {
   0x00,
   0x00,
   0x00,
   0x00,
   0x00,
   0x00,
   0xF8,
   0xFF
  }, //0
  {
   0x00,
   0x00,
```

```
  0x00,
  0x00,
  0x00,
  0x00,
  0xF0,
  0xFF
}, //0
{
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0xE0,
  0xFF
}, //0
{
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0xC0,
  0xFF
}, //0
{
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x80,
  0xFF
}, //0
{
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0xFF
}, //0

{
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
```

```
  0x00,
  0x00,
  0x7F
}, //0
{
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x3F
}, //0
{
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x1F
}, //0
{
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x0F
}, //0
{
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x07
}, //0
{
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x00,
  0x03
}, //0
```

```
  {
    0x00,
    0x00,
    0x00,
    0x00,
    0x00,
    0x00,
    0x00,
    0x01
  }, //0
  {
    0x00,
    0x00,
    0x00,
    0x00,
    0x00,
    0x00,
    0x00,
    0x00
  }, //0
};

void Write_Max7219_byte(unsigned char DATA) {
  unsigned char i;
  digitalWrite(Max7219_pinCS, LOW);
  for (i = 8; i >= 1; i--) {
    digitalWrite(Max7219_pinCLK, LOW);
    digitalWrite(Max7219_pinDIN, DATA & 0x80); // Extracting a bit data              DATA =
DATA<<1;
    digitalWrite(Max7219_pinCLK, HIGH);
  }
}

void Write_Max7219(unsigned char address, unsigned char dat) {
  digitalWrite(Max7219_pinCS, LOW);
  Write_Max7219_byte(address); //address,   code of LED
  Write_Max7219_byte(dat); //data,   figure on LED
digitalWrite(Max7219_pinCS,HIGH);
}

void Init_MAX7219(void) {
  Write_Max7219(0x09, 0x00); //decoding ：BCD
  Write_Max7219(0x0a, 0x03); //brightness
  Write_Max7219(0x0b, 0x07); //scanlimit；8 LEDs
  Write_Max7219(0x0c, 0x01); //power-down mode：0,   normal mode：1
  Write_Max7219(0x0f, 0x00); //test display：1；EOT,   display：0
}

void setup() {
  Serial.begin(9600);
  pinMode(Max7219_pinCLK, OUTPUT);
  pinMode(Max7219_pinCS, OUTPUT);
  pinMode(Max7219_pinDIN, OUTPUT);
```

```
  delay(50);
  for (i = 1; i < 9; i++) {
    Write_Max7219(i, disp1[0][i - 1]);
  }
  Init_MAX7219();
}

void loop() {
  for (j = 0; j < 64; j++) {
    for (i = 1; i < 9; i++) {
      Write_Max7219(i, disp1[64 - j][i - 1]);
      delay(10);
    }
  }
}
```

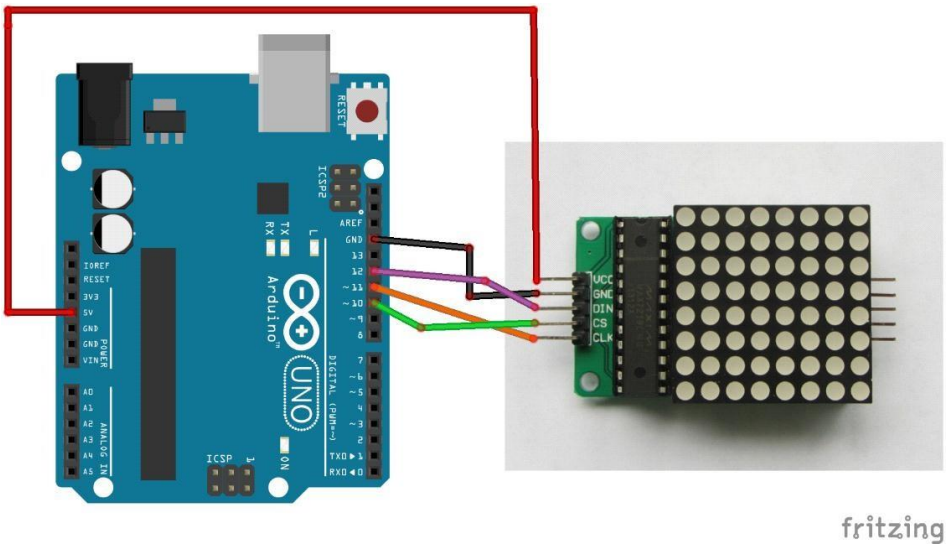
- **LED 이동 디스플레이 만들기**

준비물 : Uno R3, USB cable, 8x8 도트 매트릭스 키트, 암-암 케이블, 점퍼선

배경 : 이 프로젝트는 같은 매트릭스 키트를 사용한다는 것을 보면 위와 비슷합니다. LedControl.h. 라는 라이브러리를 추가해 주셔야 합니다. 라이브러리는 아래 링크에서 다운받으실 수 있습니다.

https://github.com/wayoda/LedControl

회로:



배선도 :

Uno R3	**MAX7219 LED matrix kit**
5V	VCC
GND	GND
D12	DIN
D10	CS
D11	CLK

코드(dotmatrix_library):

```
#include <LedControl.h>
const int numDevices = 1; // number of MAX7219s used
const long scrollDelay = 150; // adjust scrolling speed
unsigned long bufferLong[14] = {
  0
};
```

```
LedControl lc = LedControl(12, 11, 10, numDevices);
const unsigned char scrollText[]PROGMEM = {
  " MECHASOLUTION,LLC \0"
};
void setup() {
  for (int x = 0; x < numDevices; x++) {
    lc.shutdown(x, false); //The MAX72XX is in power-saving mode on startup
    lc.setIntensity(x, 8); // Set the brightness to default value
    lc.clearDisplay(x); // and clear the display
  }
}
void loop() {
  scrollMessage(scrollText);
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////
const unsigned char font5x7[]PROGMEM = { //Numeric Font Matrix (Arranged as 7x font data + 1x kerning data)
  B00000000, //Space (Char 0x20)
  B00000000,
  B00000000,
  B00000000,
  B00000000,
  B00000000,
  B00000000,
  6,
  B10000000, //!
  B10000000,
  B10000000,
  B10000000,
  B00000000,
  B00000000,
  B10000000,
  2,
  B10100000, //"
  B10100000,
  B10100000,
  B00000000,
  B00000000,
  B00000000,
  B00000000,
  4,
  B01010000, //#
  B01010000,
  B11111000,
  B01010000,
  B11111000,
  B01010000,
  B01010000,
  6,
  B00100000, //$
  B01111000,
```

```
B10100000,
B01110000,
B00101000,
B11110000,
B00100000,
6,
B11000000, //%
B11001000,
B00010000,
B00100000,
B01000000,
B10011000,
B00011000,
6,
B01100000, //&
B10010000,
B10100000,
B01000000,
B10101000,
B10010000,
B01101000,
6,
B11000000, //'
B01000000,
B10000000,
B00000000,
B00000000,
B00000000,
B00000000,
3,
B00100000, //(
B01000000,
B10000000,
B10000000,
B10000000,
B01000000,
B00100000,
4,
B10000000, //)
B01000000,
B00100000,
B00100000,
B00100000,
B01000000,
B10000000,
4,
B00000000, //*
B00100000,
B10101000,
B01110000,
B10101000,
```

```
B00100000,
B00000000,
6,
B00000000, //+
B00100000,
B00100000,
B11111000,
B00100000,
B00100000,
B00000000,
6,
B00000000, //,
B00000000,
B00000000,
B00000000,
B11000000,
B01000000,
B10000000,
3,
B00000000, //-
B00000000,
B11111000,
B00000000,
B00000000,
B00000000,
B00000000,
6,
B00000000, //.
B00000000,
B00000000,
B00000000,
B00000000,
B11000000,
B11000000,
3,
B00000000, ///
B00001000,
B00010000,
B00100000,
B01000000,
B10000000,
B00000000,
6,
B01110000, //0
B10001000,
B10011000,
B10101000,
B11001000,
B10001000,
B01110000,
6,
```

```
B01000000, //1
B11000000,
B01000000,
B01000000,
B01000000,
B01000000,
B11100000,
4,
B01110000, //2
B10001000,
B00001000,
B00010000,
B00100000,
B01000000,
B11111000,
6,
B11111000, //3
B00010000,
B00100000,
B00010000,
B00001000,
B10001000,
B01110000,
6,
B00010000, //4
B00110000,
B01010000,
B10010000,
B11111000,
B00010000,
B00010000,
6,
B11111000, //5
B10000000,
B11110000,
B00001000,
B00001000,
B10001000,
B01110000,
6,
B00110000, //6
B01000000,
B10000000,
B11110000,
B10001000,
B10001000,
B01110000,
6,
B11111000, //7
B10001000,
B00001000,
```

```
B00010000,
B00100000,
B00100000,
B00100000,
6,
B01110000, //8
B10001000,
B10001000,
B01110000,
B10001000,
B10001000,
B01110000,
6,
B01110000, //9
B10001000,
B10001000,
B01111000,
B00001000,
B00010000,
B01100000,
6,
B00000000, //:
B11000000,
B11000000,
B00000000,
B11000000,
B11000000,
B00000000,
3,
B00000000, //;
B11000000,
B11000000,
B00000000,
B11000000,
B01000000,
B10000000,
3,
B00010000, //<
B00100000,
B01000000,
B10000000,
B01000000,
B00100000,
B00010000,
5,
B00000000, //=
B00000000,
B11111000,
B00000000,
B11111000,
B00000000,
```

```
B00000000,
6,
B10000000, //>
B01000000,
B00100000,
B00010000,
B00100000,
B01000000,
B10000000,
5,
B01110000, //?
B10001000,
B00001000,
B00010000,
B00100000,
B00000000,
B00100000,
6,
B01110000, //@
B10001000,
B00001000,
B01101000,
B10101000,
B10101000,
B01110000,
6,
B01110000, //A
B10001000,
B10001000,
B10001000,
B11111000,
B10001000,
B10001000,
6,
B11110000, //B
B10001000,
B10001000,
B11110000,
B10001000,
B10001000,
B11110000,
6,
B01110000, //C
B10001000,
B10000000,
B10000000,
B10000000,
B10001000,
B01110000,
6,
B11100000, //D
```

```
B10010000,
B10001000,
B10001000,
B10001000,
B10010000,
B11100000,
6,
B11111000, //E
B10000000,
B10000000,
B11110000,
B10000000,
B10000000,
B11111000,
6,
B11111000, //F
B10000000,
B10000000,
B11110000,
B10000000,
B10000000,
B10000000,
6,
B01110000, //G
B10001000,
B10000000,
B10111000,
B10001000,
B10001000,
B01111000,
6,
B10001000, //H
B10001000,
B10001000,
B11111000,
B10001000,
B10001000,
B10001000,
6,
B11100000, //I
B01000000,
B01000000,
B01000000,
B01000000,
B01000000,
B11100000,
4,
B00111000, //J
B00010000,
B00010000,
B00010000,
```

```
B00010000,
B10010000,
B01100000,
6,
B10001000, //K
B10010000,
B10100000,
B11000000,
B10100000,
B10010000,
B10001000,
6,
B10000000, //L
B10000000,
B10000000,
B10000000,
B10000000,
B10000000,
B11111000,
6,
B10001000, //M
B11011000,
B10101000,
B10101000,
B10001000,
B10001000,
B10001000,
6,
B10001000, //N
B10001000,
B11001000,
B10101000,
B10011000,
B10001000,
B10001000,
6,
B01110000, //O
B10001000,
B10001000,
B10001000,
B10001000,
B10001000,
B01110000,
6,
B11110000, //P
B10001000,
B10001000,
B11110000,
B10000000,
B10000000,
B10000000,
```

```
6,
B01110000, //Q
B10001000,
B10001000,
B10001000,
B10101000,
B10010000,
B01101000,
6,
B11110000, //R
B10001000,
B10001000,
B11110000,
B10100000,
B10010000,
B10001000,
6,
B01111000, //S
B10000000,
B10000000,
B01110000,
B00001000,
B00001000,
B11110000,
6,
B11111000, //T
B00100000,
B00100000,
B00100000,
B00100000,
B00100000,
B00100000,
6,
B10001000, //U
B10001000,
B10001000,
B10001000,
B10001000,
B10001000,
B01110000,
6,
B10001000, //V
B10001000,
B10001000,
B10001000,
B10001000,
B01010000,
B00100000,
6,
B10001000, //W
B10001000,
```

```
B10001000,
B10101000,
B10101000,
B10101000,
B01010000,
6,
B10001000, //X
B10001000,
B01010000,
B00100000,
B01010000,
B10001000,
B10001000,
6,
B10001000, //Y
B10001000,
B10001000,
B01010000,
B00100000,
B00100000,
B00100000,
6,
B11111000, //Z
B00001000,
B00010000,
B00100000,
B01000000,
B10000000,
B11111000,
6,
B11100000, //[
B10000000,
B10000000,
B10000000,
B10000000,
B10000000,
B11100000,
4,
B00000000, //(Backward Slash)
B10000000,
B01000000,
B00100000,
B00010000,
B00001000,
B00000000,
6,
B11100000, //]
B00100000,
B00100000,
B00100000,
B00100000,
```

```
B00100000,
B11100000,
4,
B00100000, //^
B01010000,
B10001000,
B00000000,
B00000000,
B00000000,
B00000000,
6,
B00000000, //_
B00000000,
B00000000,
B00000000,
B00000000,
B00000000,
B11111000,
6,
B10000000, //`
B01000000,
B00100000,
B00000000,
B00000000,
B00000000,
B00000000,
4,
B00000000, //a
B00000000,
B01110000,
B00001000,
B01111000,
B10001000,
B01111000,
6,
B10000000, //b
B10000000,
B10110000,
B11001000,
B10001000,
B10001000,
B11110000,
6,
B00000000, //c
B00000000,
B01110000,
B10001000,
B10000000,
B10001000,
B01110000,
6,
```

```
B00001000, //d
B00001000,
B01101000,
B10011000,
B10001000,
B10001000,
B01111000,
6,
B00000000, //e
B00000000,
B01110000,
B10001000,
B11111000,
B10000000,
B01110000,
6,
B00110000, //f
B01001000,
B01000000,
B11100000,
B01000000,
B01000000,
B01000000,
6,
B00000000, //g
B01111000,
B10001000,
B10001000,
B01111000,
B00001000,
B01110000,
6,
B10000000, //h
B10000000,
B10110000,
B11001000,
B10001000,
B10001000,
B10001000,
6,
B01000000, //i
B00000000,
B11000000,
B01000000,
B01000000,
B01000000,
B11100000,
4,
B00010000, //j
B00000000,
B00110000,
```

```
B00010000,
B00010000,
B10010000,
B01100000,
5,
B10000000, //k
B10000000,
B10010000,
B10100000,
B11000000,
B10100000,
B10010000,
5,
B11000000, //l
B01000000,
B01000000,
B01000000,
B01000000,
B01000000,
B11100000,
4,
B00000000, //m
B00000000,
B11010000,
B10101000,
B10101000,
B10001000,
B10001000,
6,
B00000000, //n
B00000000,
B10110000,
B11001000,
B10001000,
B10001000,
B10001000,
6,
B00000000, //o
B00000000,
B01110000,
B10001000,
B10001000,
B10001000,
B01110000,
6,
B00000000, //p
B00000000,
B11110000,
B10001000,
B11110000,
B10000000,
```

```
B10000000,
6,
B00000000, //q
B00000000,
B01101000,
B10011000,
B01111000,
B00001000,
B00001000,
6,
B00000000, //r
B00000000,
B10110000,
B11001000,
B10000000,
B10000000,
B10000000,
6,
B00000000, //s
B00000000,
B01110000,
B10000000,
B01110000,
B00001000,
B11110000,
6,
B01000000, //t
B01000000,
B11100000,
B01000000,
B01000000,
B01001000,
B00110000,
6,
B00000000, //u
B00000000,
B10001000,
B10001000,
B10001000,
B10011000,
B01101000,
6,
B00000000, //v
B00000000,
B10001000,
B10001000,
B10001000,
B01010000,
B00100000,
6,
B00000000, //w
```

```
B00000000,
B10001000,
B10101000,
B10101000,
B10101000,
B01010000,
6,
B00000000, //x
B00000000,
B10001000,
B01010000,
B00100000,
B01010000,
B10001000,
6,
B00000000, //y
B00000000,
B10001000,
B10001000,
B01111000,
B00001000,
B01110000,
6,
B00000000, //z
B00000000,
B11111000,
B00010000,
B00100000,
B01000000,
B11111000,
6,
B00100000, //{
B01000000,
B01000000,
B10000000,
B01000000,
B01000000,
B00100000,
4,
B10000000, //|
B10000000,
B10000000,
B10000000,
B10000000,
B10000000,
B10000000,
2,
B10000000, //}
B01000000,
B01000000,
B00100000,
```

```
  B01000000,
  B01000000,
  B10000000,
  4,
  B00000000, //~
  B00000000,
  B00000000,
  B01101000,
  B10010000,
  B00000000,
  B00000000,
  6,
  B01100000, // (Char 0x7F)
  B10010000,
  B10010000,
  B01100000,
  B00000000,
  B00000000,
  B00000000,
  5
};
int scrollFont() {
  for (int counter = 0x20; counter < 0x80; counter++) {
    loadBufferLong(counter);
  }
}
// Scroll Message
int scrollMessage(const unsigned char * messageString) {
  int counter = 0;
  int myChar = 0;
  do {
    // read back a char
    myChar = pgm_read_byte_near(messageString + counter);
    if (myChar != 0) {
      loadBufferLong(myChar);
    }
    counter++;
  } while (myChar != 0);
}
// Load character into scroll buffer
int loadBufferLong(int ascii) {
  if (ascii >= 0x20 && ascii <= 0x7f) {
    for (int a = 0; a < 7; a++) { // Loop 7 times for a 5x7 font
      unsigned long c = pgm_read_byte_near(font5x7 + ((ascii - 0x20) * 8) + a); // Index      into character table
to get row data
      unsigned long x = bufferLong[a * 2]; // Load current scroll buffer
      x = x | c; // OR the new character onto end of current
      bufferLong[a * 2] = x; // Store in buffer
    }
    byte count = pgm_read_byte_near(font5x7 + ((ascii - 0x20) * 8) + 7); // Index into     character table for
kerning data
```

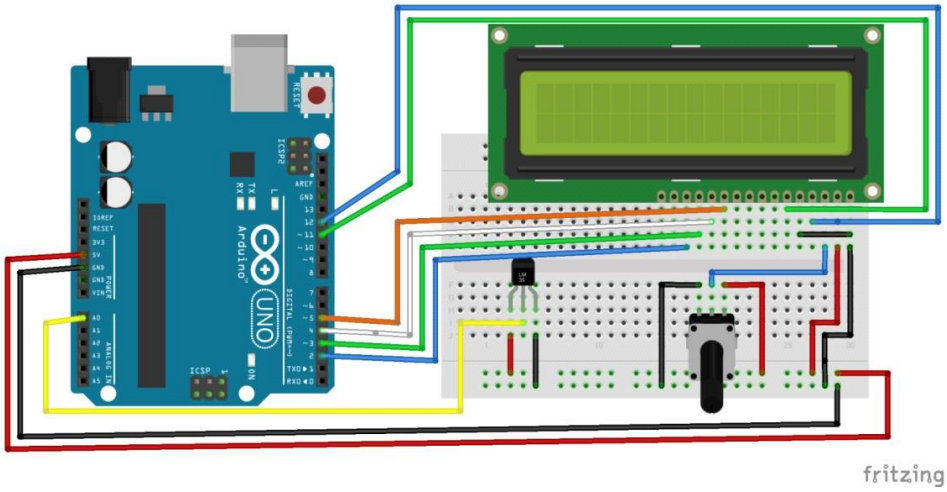
```
    for (byte x = 0; x < count; x++) {
      rotateBufferLong();
      printBufferLong();
      delay(scrollDelay);
    }
  }
}
// Rotate the buffer
int rotateBufferLong() {
  for (int a = 0; a < 7; a++) { // Loop 7 times for a 5x7 font
    unsigned long x = bufferLong[a * 2]; // Get low buffer entry
    byte b = bitRead(x, 31); // Copy high order bit that gets lost in rotation
    x = x << 1; // Rotate left one bit
    bufferLong[a * 2] = x; // Store new low buffer
    x = bufferLong[a * 2 + 1]; // Get high buffer entry
    x = x << 1; // Rotate left one bit
    bitWrite(x, 0, b); // Store saved bit
    bufferLong[a * 2 + 1] = x; // Store new high buffer
  }
}
// Display Buffer on LED matrix
int printBufferLong() {

  for (int a = 0; a < 7; a++) { // Loop 7 times for a 5x7 font
    unsigned long x = bufferLong[a * 2 + 1]; // Get high buffer entry
    byte y = x; // Mask off first character
    lc.setRow(3, a, y); // Send row to relevent MAX7219 chip
    x = bufferLong[a * 2]; // Get low buffer entry
    y = (x >> 24); // Mask off second character
    lc.setRow(2, a, y); // Send row to relevent MAX7219 chip
    y = (x >> 16); // Mask off third character
    lc.setRow(1, a, y); // Send row to relevent MAX7219 chip
    y = (x >> 8); // Mask off forth character
    lc.setRow(0, a, y); // Send row to relevent MAX7219 chip
  }
}
```

• 디지털 온도계 만들기

준비물 : Uno R3, USB cable, 브레드보드, 포텐시오미터, LM35, 점퍼선

배경 : LM35 는 사용하기 매우 간편한 온도 센서입니다. -55 도에서 +150 도 사이로 0.5 도 간격으로 아날로그 출력을 내보냅니다. 16x2 LCD 를 사용하여 온도를 표현 해 보겠습니다. 배면광은 포텐시오미터로 제어 할 것입니다.

회로:



코드(thermometer):

```
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int lm35 = A0; void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 0);
  int val = analogRead(lm35);
  int dat = (125 * val) >> 8; //temperature calculation formula
  lcd.print("Temperature");
  lcd.setCursor(0, 1);
  lcd.print(dat);
}
```

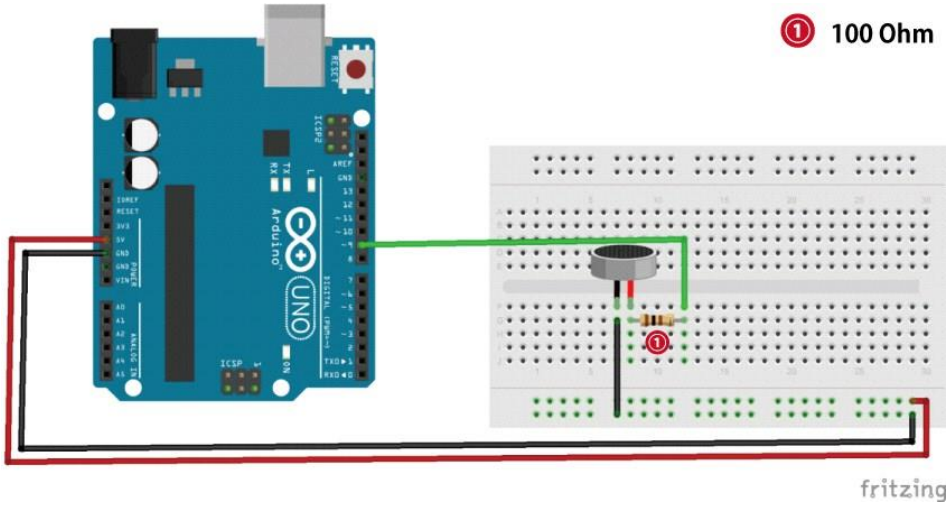
- **소리**

  - **슈퍼마리오 재생하기**

준비물 : Uno R3, 브레드보드, 버저 , 100 Ohm 저항, 점퍼선

배경 : 만들어 져 있는 함수를 이용 할 것입니다. 아두이노는 핀에 주파수 파형을 내보낼 수 있습니다. tone() 주파수의 헤르츠를 지정해 줄 수 있으며 부가적으로 시간도 지정해 줄 수 있습니다.

회로:



코드(supermario):

```
void setup() {}

// funcion = tone(pin, frequency, duration)

void loop() {
  // CANCION DE MARIO BROSS
  //NOTA 01 noTone(9); tone(9, 660, 100); delay(75);

  //NOTA 02 noTone(9); tone(9, 660, 100); delay(75);


  //NOTA 03 noTone(9); tone(9, 660, 100); delay(150);

  //NOTA 04 noTone(9); tone(9, 660, 100); delay(150);

  //NOTA 05 noTone(9); tone(9, 660, 100); delay(50);

  //NOTA 06 noTone(9); tone(9, 770, 100); delay(150);

  //NOTA 07 noTone(9); tone(9, 380, 100); delay(275);

  //NOTA 08 noTone(9); tone(9, 510, 100); delay(287);

  //NOTA 09 noTone(9); tone(9, 380, 100); delay(225);

  //NOTA 10 noTone(9); tone(9, 320, 100); delay(200);
```

```
//NOTA 11 noTone(9); tone(9, 440, 100); delay(250);

//NOTA 12 noTone(9); tone(9, 480, 80); delay(150);

//NOTA 13 noTone(9); tone(9, 450, 100); delay(165);

//NOTA 14 noTone(9); tone(9, 430, 100); delay(75); //NOTA 15 noTone(9); tone(9, 380, 100); delay(150);

//NOTA 16 noTone(9); tone(9, 660, 80); delay(100); //’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’
//NOTA 17 noTone(9); tone(9, 760, 50); delay(100); //’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’
//NOTA 18 noTone(9); tone(9, 860, 100); delay(75);

//NOTA 19 noTone(9); tone(9, 700, 80); delay(150);

//NOTA 20 noTone(9); tone(9, 760, 50); delay(75);

//NOTA 21 noTone(9); tone(9, 660, 80); delay(175);

//NOTA 22 noTone(9); tone(9, 520, 80); delay(150);

//NOTA 23 noTone(9); tone(9, 580, 80); delay(75);

//NOTA 24 noTone(9); tone(9, 480, 80); delay(75);

//NOTA 25 noTone(9); tone(9, 510, 100); delay(175);

//NOTA 26 noTone(9); tone(9, 380, 100); delay(275);

//NOTA 27 noTone(9); tone(9, 320, 100); delay(200);

//NOTA 28 noTone(9); tone(9, 440, 100); delay(250);

//NOTA 29 noTone(9); tone(9, 480, 80); delay(150);

//NOTA 30 noTone(9); tone(9, 450, 100); delay(165);

//NOTA 31 noTone(9); tone(9, 430, 100); delay(75);

//NOTA 32 noTone(9); tone(9, 380, 100); delay(150);

//NOTA 33 noTone(9); tone(9, 660, 80); delay(100);

//NOTA 34 noTone(9); tone(9, 760, 50); delay(100);

noTone(9);
tone(9, 860, 100);
delay(75);

noTone(9);
tone(9, 700, 80);
```

```
delay(150);

tone(9, 760, 50);
delay(75);
noTone(9);

tone(9, 660, 80);
delay(175);
noTone(9);
tone(9, 520, 80);
delay(150);
noTone(9);

tone(9, 580, 80);
delay(75);
noTone(9);

tone(9, 480, 80);
delay(75);
noTone(9);

tone(9, 500, 100);
delay(250);
noTone(9);

tone(9, 760, 100);
delay(150);
noTone(9);

tone(9, 720, 100);
delay(50);
noTone(9);

tone(9, 680, 100);
delay(75);
noTone(9);

tone(9, 620, 150);
delay(75);
noTone(9);

tone(9, 650, 150);
delay(150);
noTone(9);

tone(9, 380, 100);
delay(150);
noTone(9);

tone(9, 430, 100);
delay(75);
```

```
noTone(9);

tone(9, 500, 100);
delay(75);
noTone(9);

tone(9, 430, 100);
delay(150);
noTone(9);

tone(9, 500, 100);
delay(75);
noTone(9);

delay(50);
tone(9, 570, 100);
delay(110);
tone(9, 500, 100);
delay(150);
tone(9, 760, 100);
delay(50);
tone(9, 720, 100);
delay(75);
tone(9, 680, 100);
delay(75);
tone(9, 620, 150);
delay(150);
tone(9, 650, 200);
delay(150);
tone(9, 1020, 80);
delay(150);
tone(9, 1020, 80);
delay(75);
tone(9, 1020, 80);
delay(150);
tone(9, 380, 100);
delay(150);
tone(9, 500, 100);
delay(150);
tone(9, 760, 100);
delay(50);
tone(9, 720, 100);
delay(75);
tone(9, 680, 100);
delay(75);
tone(9, 620, 150);
delay(150);
tone(9, 650, 150);
delay(150);
tone(9, 380, 100);
delay(75);
```

```
tone(9, 430, 100);
delay(75);
tone(9, 500, 100);
delay(150);
tone(9, 430, 100);
delay(75);
tone(9, 500, 100);
delay(50);
tone(9, 570, 100);
delay(110);
tone(9, 500, 100);
delay(150);
tone(9, 760, 100);
delay(50);
tone(9, 720, 100);
delay(75);
tone(9, 680, 100);
delay(75);
tone(9, 620, 150);
delay(150);
tone(9, 650, 200);
delay(150);
tone(9, 1020, 80);
delay(150);
tone(9, 1020, 80);
delay(75);
tone(9, 1020, 80);
delay(150);
tone(9, 380, 100);
delay(150);
tone(9, 500, 100);
delay(150);
tone(9, 760, 100);
delay(50);
tone(9, 720, 100);
delay(75);
tone(9, 680, 100);
delay(75);
tone(9, 620, 150);
delay(150);
tone(9, 650, 150);
delay(150);
tone(9, 380, 100);
delay(75);
tone(9, 430, 100);
delay(75);
tone(9, 500, 100);
delay(150);
tone(9, 430, 100);
delay(75);
tone(9, 500, 100);
```

```
  delay(50);
  tone(9, 570, 100);
  delay(210);
  tone(9, 585, 100);
  delay(275);
  tone(9, 550, 100);
  delay(210);
  tone(9, 500, 100);
  delay(180);
  tone(9, 380, 100);
  delay(150);
  tone(9, 500, 100);
  delay(150);
  tone(9, 500, 100);
  delay(75);
  tone(9, 500, 100);
  delay(150);
  tone(9, 500, 60);
  delay(75);
  tone(9, 500, 80);
  delay(150);
  tone(9, 500, 60);
  delay(175);
  tone(9, 500, 80);
  delay(75);
  tone(9, 580, 80);
  delay(175);
  tone(9, 660, 80);
  delay(75);
  tone(9, 500, 80);
  delay(150);
  tone(9, 430, 80);
  delay(75);
  tone(9, 380, 80);
  delay(300);
  tone(9, 500, 60);

  delay(75);
  tone(9, 500, 80);
  delay(150);
  tone(9, 500, 60);
  delay(175);
  tone(9, 500, 80);
  delay(75);
  tone(9, 580, 80);
  delay(75);
  tone(9, 660, 80);
  delay(225);
  tone(9, 870, 80);
  delay(162);
  tone(9, 760, 80);
```

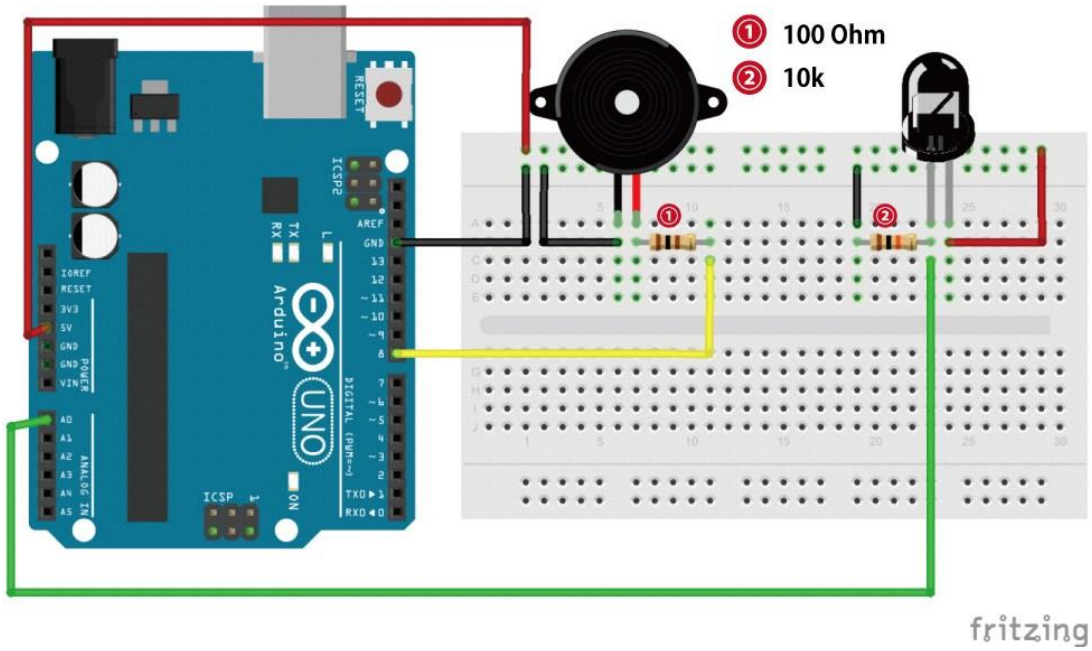
```
  delay(300);
  tone(9, 500, 60);
  delay(75);
  tone(9, 500, 80);
  delay(150);
  tone(9, 500, 60);
  delay(175);
  tone(9, 500, 80);
  delay(75);
  tone(9, 580, 80);
  delay(175);
  tone(9, 660, 80);
  delay(75);
  tone(9, 500, 80);
  delay(150);
  tone(9, 430, 80);
  delay(75);
  tone(9, 380, 80);
  delay(300);
  tone(9, 660, 100);
  delay(75);
  tone(9, 660, 100);
  delay(150);
  tone(9, 660, 100);
  delay(150);
  tone(9, 510, 100);
  delay(50);
  tone(9, 660, 100);
  delay(150);
  tone(9, 770, 100);
  delay(225);
  tone(9, 380, 100);

  delay(1000);
  tone(9, 440, 200);
  delay(200);
  delay(200);
  tone(9, 440, 400);
  delay(200);
  delay(200);
  delay(5000);

}
```

• 빛 감지 알람

준비물 : Uno R3, USB cable, 브레드보드, 빛 센서, 버저, 10K 저항, 100 Ohm 저항, 점퍼선

배경 : 빛 센서는 적외선을 이용하여 불에서 내는 빛을 이용하여 불의 근원을 찾아내는데 사용 할 수 있습니다. 외형은 LED 처럼 생겼는데 검은색입니다. LED 와 비슷하게 긴 다리가 anode(+) 짧은 다리가 cathode(-) 입니다.

회로: 빛이 나면 버저가 울립니다.



코드(light_alarm):

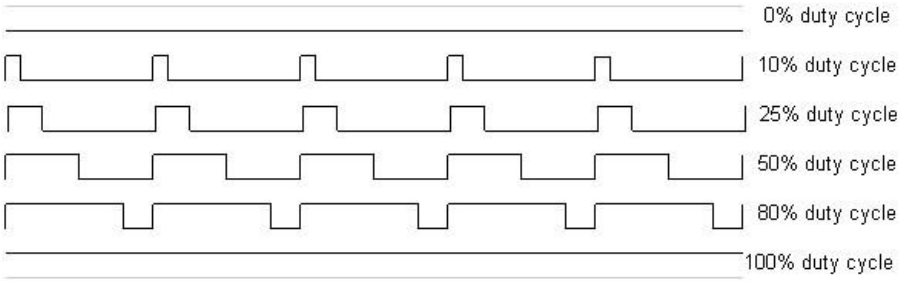
```
int flame = A0;
int Buzzer = 8;
int val = 0;

void setup()
{
  pinMode(Buzzer, OUTPUT); //Set LED as output
  pinMode(flame, INPUT); //Set buzzer as input
  Serial.begin(9600);  //Set baud rate as 9600
}

void loop()
{
  val = analogRead(flame); //Read the analog value
  Serial.println(val);  //Output analog value and print it out
  if (val >= 600) // When analog value 〉 600,   buzzer make sound
  {
```

```
    digitalWrite(Buzzer, HIGH);
  }
  else
  {
    digitalWrite(Buzzer, LOW);
  }
}
```

- **모터**

- **Pulse Width Modulation 이란?**

Pulse width modulation(이하 PWM)은 펄스의 폭을 컨트롤 하는 주기 제어방법입니다. “On”되는 시간에 따라 그 주기가 달라집니다. 주기가 낮다면 그에 따라 전압이 약해집니다. 왜냐하면 전압이 꺼지는 시간이 대부분이기 때문입니다. 다음 그림은 주기를 퍼센트로 나타낸 것입니다.



PWM 의 사용 용도:

- LED 흐리게 하기
- 아날로그 출력하기
- 오디오 신호 만들기
- 모터 공급용 속도조절하기

• 어떻게 서보모터가 회전하는가

반이중 시리얼 통신을 사용하는 서보모터(Dynamixel 이나 Herkulex)를 제외하면 대부분의 서보모터는 PWM 으로 제어됩니다.

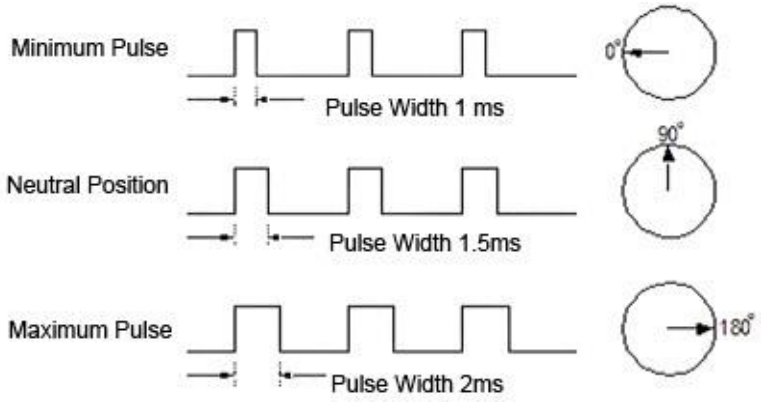


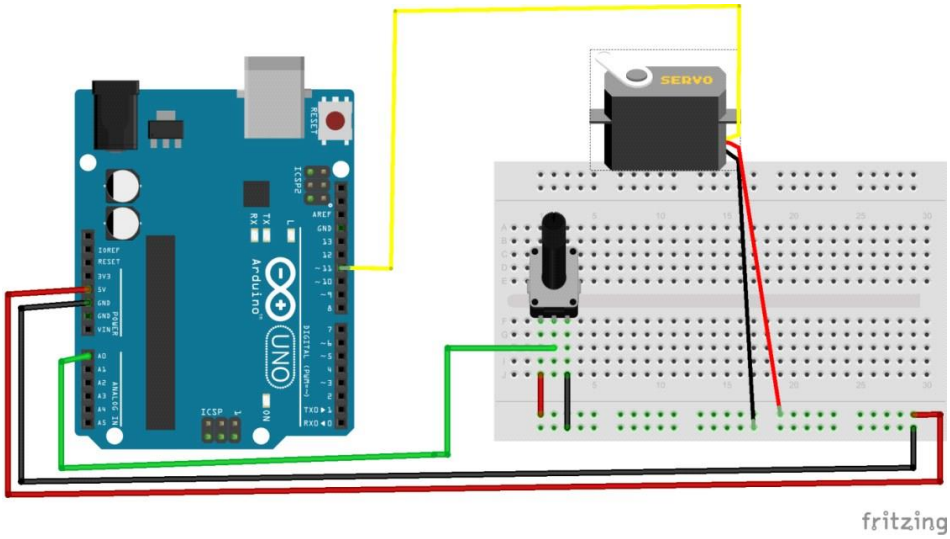
Fig. 27. 펄스의 주기로 서보모터의 각 조절하기

위에서 알아 낸 것은 어떻게 서보모터가 회전하는가 입니다. 서보모터는 공급되는 PWM 신호로 위치 제어가 가능하도록 회로가 내장되어 있습니다.

• 포텐시오미터로 서보모터 조종하기

준비물 : Uno R3, 브레드보드, USB cable, 미니 서보모터, 포텐시오미터, 점퍼선

배경 : Servo.h 라이브러리는 서보모터를 제어하기 더 편하게 해줍니다. Servo.h 는 아두이노 보드로 12 개를 지원하며 메가는 48 개를 지원합니다. 서보모터는 3 개의 선이 있습니다. 전원, 그라운드, 신호선입니다. 보통 그라운드는 검은색이나 갈색이며 전원선은 빨강, 신호는 노랑이나 주황입니다. 미니 서보는 4 ~ 7.2V 를 전원으로 사용할 수 있으며 여기선 약 1.2 kg/cm(16.7 oz/in)의 토크를 가집니다. 만약 이 서보(SG90)로 더 큰 힘을 내고 싶다면 외부 전원선을 연결하면 됩니다. **회로:** 포텐시오미터를 돌릴 때 서보가 같이 회전합니다.



코드(servo):

```
#include <Servo.h>

Servo myservo; // create servo object to control a servo

int potpin = 0; // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin

void setup() {
  myservo.attach(11); // attaches the servo on pin 9 to the servo object
}

void loop() {
  val = analogRead(potpin); // reads the value of the potentiometer
  val = map(val, 0, 1023, 0, 179);     // scale it to use it with the servo
  myservo.write(val);       // sets the servo position according to the scaled value
  delay(15);              // waits for the servo to get there
}
```