



제천지역 일반고 연합 해커톤

# 아두이노 기초

with Arduino IDE

## 아두이노 기초 강의 (Arduino Basic Lecture)

본 강의 자료는 **2025 제천지역 일반고 연합 해커톤** 중 진행되는 교육용 자료로써, 고등학생(1~2학년)을 대상으로 아두이노 기초 개념과 센서·출력 장치 제어 방법을 학습하기 위해 제작되었습니다.

\* PDF로 보는 것보다 [깃허브 페이지](#)에서 보는 것을 추천합니다.

### 목차 (Table of Contents)



## Chapter 1

### 아두이노 기초 이해

#### Chapter 1. 아두이노 기초 이해

1. [아두이노란?](#)
2. [개발 환경 설정하기 \(Arduino IDE\)](#)
3. [아두이노 핀 구조와 기능](#)
4. [브레드보드 구조와 기능](#)



## Chapter 2

### 다양한 센서와 통신 방식

#### Chapter 2. 다양한 센서와 통신 방식

1. 센서의 역할과 분류
  2. 디지털 vs 아날로그 신호
  3. 아두이노와 센서의 연결 방식/선 종류
  4. 아두이노와 센서 연결 예시
- 



## Chapter 3

### 기본 입출력 실습

#### Chapter 3. 기본 입출력 실습

1. 신호등 LED 제어
  2. RGB LED 제어
  3. 조도센서 응용
  4. 초음파센서 응용
  5. 온/습도센서
- 



## Chapter 4

### 고급 출력 장치 제어

#### Chapter 4. 고급 출력 장치 제어

1. 수동부저 소리 재생

2. 서보모터 회전 제어
  3. I2C LCD 화면 출력
  4. 7세그먼트 숫자 출력
- 



## Chapter 5

### 아두이노에서 주의할 여러 점들

#### Chapter 5. 아두이노에서 주의할 여러 점들

1. 코드 업로드 오류
  - 보드/포트 설정 문제, 케이블 불량 또는 연결 불량, 드라이버 설치 문제
1. 시리얼 모니터 값이 이상하게 나오는 경우
  - 통신 속도(baud rate) 불일치, 센서 연결 문제
2. 센서 값이 변하지 않는 경우
  - 핀 번호 오류, 전원/그라운드 연결 누락, 센서 손상 여부 확인
3. 아두이노/LED/부저 등 장치가 동작하지 않는 경우
  - 극성(+) (-) 반대로 연결, 전압 부족, 코드 로직 오류
4. 서보모터 떨림 또는 움직이지 않음
  - 전원 공급 문제, PWM 핀 확인, 코드 딜레이 설정 문제
5. I2C 장치(LCD 등) 화면이 안 나오는 경우
  - SDA/SCL 핀 연결 확인, I2C 주소 확인 및 변경 방법, 라이브러리 설치 여부 확인
6. 기타 예상치 못한 오류 상황
  - 리셋 버튼 활용, 코드 최소화하여 원인 추적, 공식 문서 및 커뮤니티 검색 방법



# Chapter 1

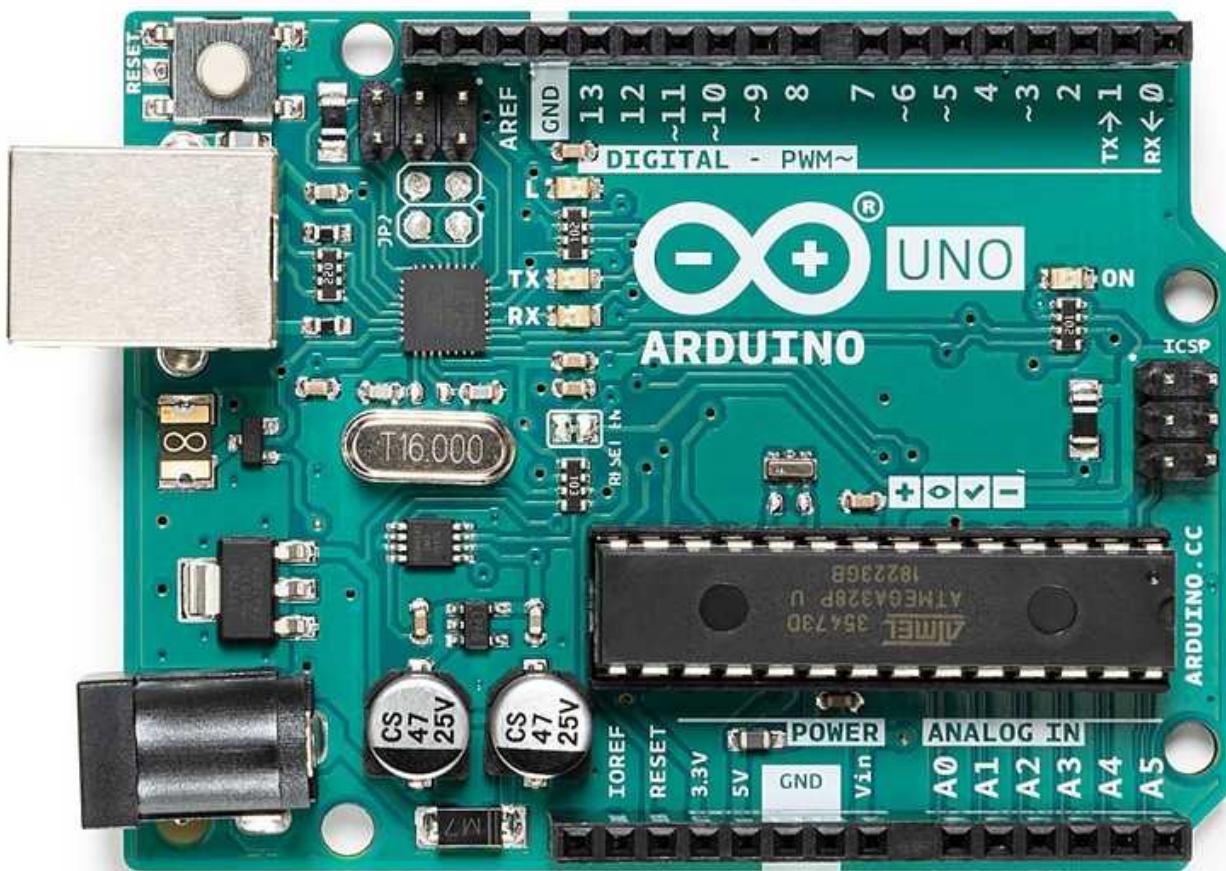
## 아두이노 기초 이해

### 아두이노란? (What is Arduino?)

[!NOTE] 이 문서는 \*\*아두이노(Arduino)\*\*에 대해 설명합니다.

#### 1. 아두이노란?

오픈 소스를 기반으로 한 단일 보드 마이크로컨트롤러로 완성 된 보드와 관련 개발 도구 및 환경을 의미합니다.



가장 대중적인 아두이노 우노 보드

#### 특징

- 저렴한 가격

- 간편한 개발 환경
- 오픈 소스 하드웨어 및 소프트웨어
- 다양한 센서 및 부품과의 호환성

## 기술 사양 (공식 아두이노 우노 기준)

항목	값
마이크로컨트롤러	ATmega328P
동작 전압	5V
입력 전압 (권장)	7-12V
디지털 I/O 핀	14개 (6개는 PWM 출력 가능)
아날로그 입력 핀	6개
플래시 메모리	32KB
SRAM	2KB
EEPROM	1KB
클럭 속도	16MHz

## 2. 아두이노 프로그래밍의 기본 구조

```
void setup() {
    // 코드를 실행하기 전, 처음에 한 번만 실행됩니다.
    // 핀 모드를 설정하거나, 라이브러리를 초기화하는 데 사용됩니다.
}

void loop() {
    // setup() 함수가 실행된 후, 계속해서 반복적으로 실행됩니다.
    // 아두이노의 주요 기능이 이 곳에 작성됩니다.
}
```



# Chapter 1

## 아두이노 기초 이해

### 개발 환경 설정하기 (Arduino IDE)

[!NOTE] 이 문서는 **아두이노 통합 개발 환경(IDE)** 설정 방법에 대해 설명합니다.

#### 1. 아두이노 IDE란?

아두이노 보드에 코드를 작성하고 업로드하기 위한 공식 소프트웨어입니다.

 A screenshot of the Arduino IDE 2.3.6 interface. The top menu bar includes File, Edit, Sketch, Tools, and Help. The main area shows a code editor with a sketch named 'sketch\_aug15a.ino' containing the standard setup and loop functions. To the left is a sidebar with icons for file operations like Open, Save, and Find. On the right, there's a 'Select Board' dropdown and some status indicators. Below the code editor is an 'Output' window displaying library installation logs, such as 'Processing Adafruit NeoPixel@1.15.1: Installed Adafruit NeoPixel@1.15.1' and 'Updating libraries...'. At the bottom, status bars show 'Ln 10, Col 1', 'No board selected', and other system information.
 

```
sketch_aug15a | Arduino IDE 2.3.6
File Edit Sketch Tools Help
Select Board
sketch_aug15a.ino
1 void setup() {
2   // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7   // put your main code here, to run repeatedly:
8
9 }
10
```

Output

```
Installed Firmata@2.5.9
Downloading Servo@1.2.2
Servo@1.2.2
Installing Servo@1.2.2
Installed Servo@1.2.2
Downloading LiquidCrystal@1.0.7
LiquidCrystal@1.0.7
Installing LiquidCrystal@1.0.7
Installed LiquidCrystal@1.0.7
```

Processing Adafruit NeoPixel@1.15.1: Installed Adafruit NeoPixel@1.15.1

Updating libraries...

Ln 10, Col 1 No board selected 2 / 2

아두이노 IDE 2.x 버전의 인터페이스

#### 주요 기능

- 코드 에디터: C/C++ 기반의 아두이노 코드를 작성하는 공간
- 시리얼 모니터: 아두이노 보드와 시리얼 통신을 통해 데이터를 주고받는 기능
- 라이브러리 매니저: 다양한 추가 기능을 쉽게 설치하고 관리
- 보드 매니저: 아두이노 공식 보드 외 다른 보드를 추가하고 관리

#### 2. 아두이노 IDE 설치 과정

## 1. 공식 홈페이지 접속

### 2. 자신의 운영체제에 맞는 버전 다운로드

- Windows, macOS, Linux 등 다양한 OS 지원

### 3. 설치 파일 실행 및 설치 진행

- 설치 중 드라이버 설치 등의 창이 나타나면 '예'를 클릭하여 설치. (모든 것을 설치, 동의, 예 등의 긍정적인 선택)

## 3. 아두이노 보드 연결 및 설정

### 1. 아두이노 보드와 PC를 USB 케이블로 연결

### 2. 아두이노 IDE 실행

### 3. 보드 선택

- 툴 > 보드 메뉴에서 자신의 아두이노 보드 선택 (예: Arduino Uno)

### 4. 포트 선택

- 툴 > 포트 메뉴에서 아두이노가 연결된 COM 포트 선택

## 4. 기본 예제 코드 업로드 테스트

```
void setup() {  
    Serial.begin(9600); // 9600 bps 속도로 시리얼 통신 시작  
}  
  
void loop() {  
    Serial.println("Hello, Arduino!"); // "Hello, Arduino!" 메시지 출력  
    delay(1000); // 1초 대기 (ms)  
}
```

위 코드를 업로드한 후, IDE 우측 상단의 돌보기 아이콘(시리얼 모니터)을 클릭하면 1초마다 메시지가 출력되는 것을 확인할 수 있습니다.

[!TIP] 2초마다 메시지가 출력되도록 수정해보세요!



# Chapter 1

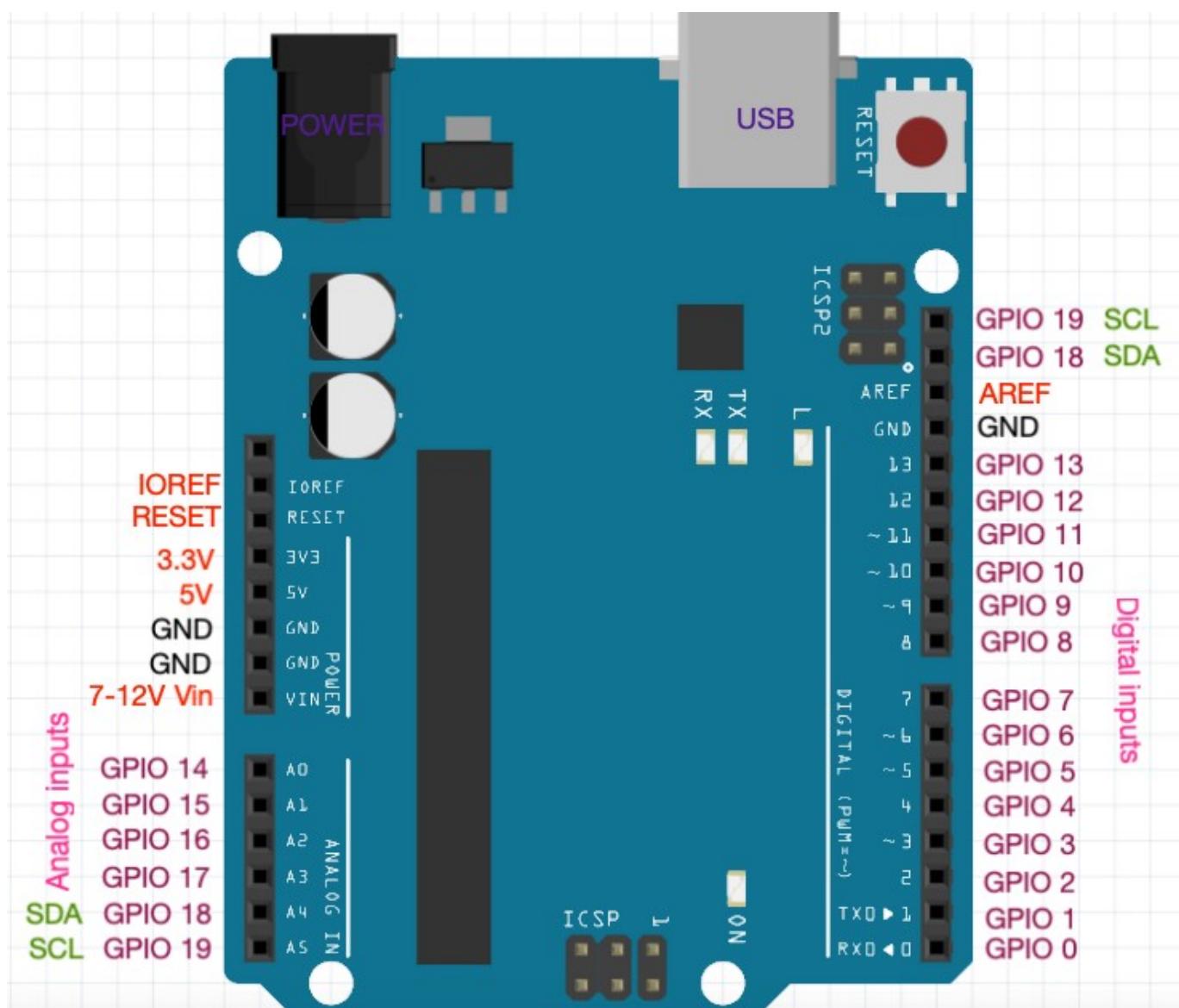
## 아두이노 기초 이해

### 아두이노 핀 구조와 기능

[!NOTE] 이 문서는 아두이노 우노(Uno) 보드의 핀 구조와 기능에 대해 설명합니다.

#### 1. 아두이노 핀의 종류

아두이노 보드는 다양한 종류의 핀을 통해 외부 전자 부품과 상호작용합니다.



아두이노 우노 보드의 핀 배치도

## 전원 핀 (Power Pins)

핀 이름	기능
3.3V	3.3V 전원 공급
5V	5V 전원 공급
GND	접지(Ground) 핀
VIN	외부 전원 입력 (7-12V)

## 아날로그 핀 (Analog Pins)

- **A0 ~ A5**: 아날로그 신호를 입력받는 핀 (0 ~ 1023 사이의 값으로 변환)
- 조도 센서, 가변 저항 등 아날로그 센서 연결에 사용

## 디지털 핀 (Digital Pins)

- **0 ~ 13**: 디지털 신호(HIGH/LOW)를 입/출력하는 핀
- LED, 스위치, 부저 등 디지털 부품 연결에 사용
- **PWM (Pulse Width Modulation)**: ~ 표시가 있는 핀(3, 5, 6, 9, 10, 11)은 아날로그 출력 효과를 낼 수 있음 (예: LED 밝기 조절, 모터 속도 제어)
- **특수 기능 핀**:
  - **0 (RX), 1 (TX)**: 시리얼 통신에 사용
  - **2, 3**: 외부 인터럽트(Interrupt) 사용 가능
  - **10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK)**: SPI 통신에 사용
  - **SDA, SCL**: I2C 통신에 사용

## 2. 핀 모드(Pin Mode) 설정

setup() 함수 내에서 pinMode() 함수를 사용하여 각 핀의 역할을 설정해야 합니다.

```
int ledPin = 13;
int buttonPin = 7;

void setup() {
    pinMode(ledPin, OUTPUT);      // 13번 핀을 출력으로 설정
    pinMode(buttonPin, INPUT);    // 7번 핀을 입력으로 설정
    // pinMode(buttonPin, INPUT_PULLUP); // 내부 풀업 저항을 사용하는 입력으로 설정
}

void loop() {
    // ...
}
```



# Chapter 1

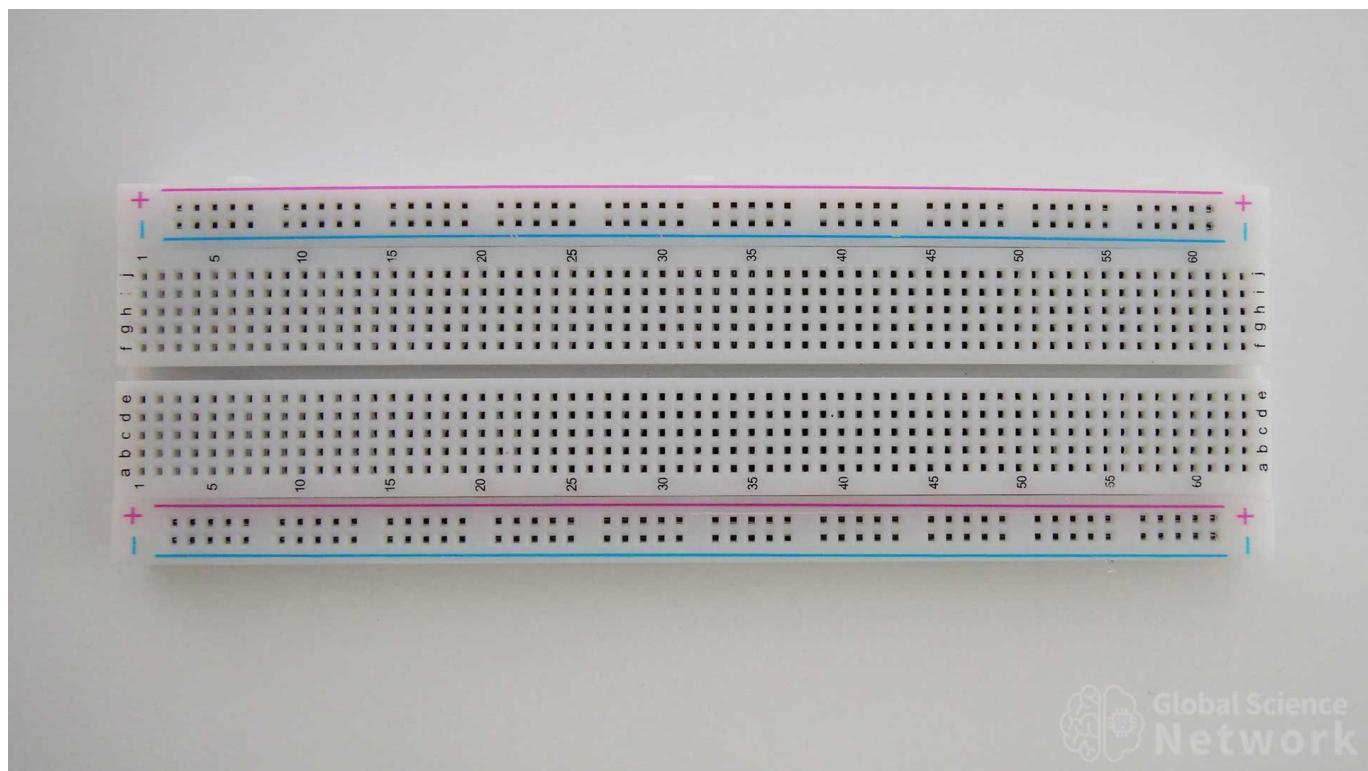
## 아두이노 기초 이해

### 브레드보드 구조와 기능

[!NOTE] 이 문서는 \*\*브레드보드(Breadboard)\*\*의 구조와 사용 방법에 대해 설명합니다.

#### 1. 브레드보드란?

납땜 없이 전자 부품과 전선을 쉽게 꽂아 회로를 구성할 수 있는 실험용 기판입니다. (마치 콘센트에 코드를 꽂는 것처럼)



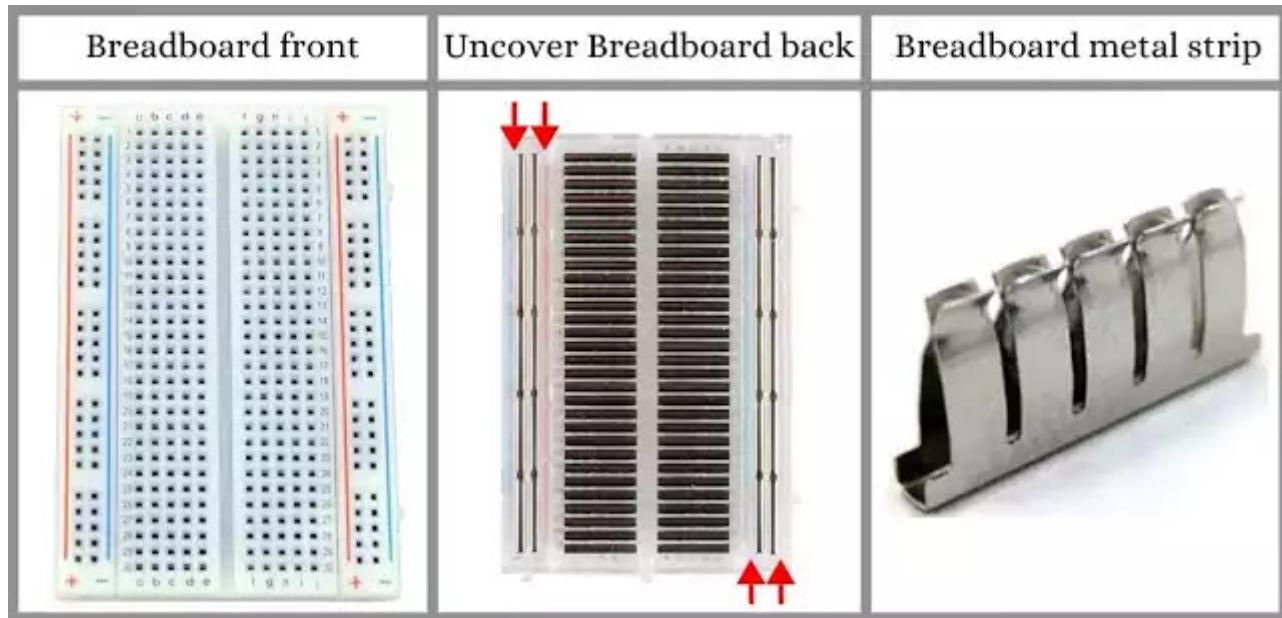
일반적인 풀사이즈 브레드보드

#### 특징

- 프로토타이핑에 용이
- 부품 재사용 가능
- 회로 변경 및 수정이 간편

#### 2. 브레드보드의 내부 구조

브레드보드 내부는 금속 클립으로 연결되어 있어, 특정 규칙에 따라 전기가 통합니다.



### 브레드보드의 내부 연결 구조

#### 전원 버스 (Power Rails)

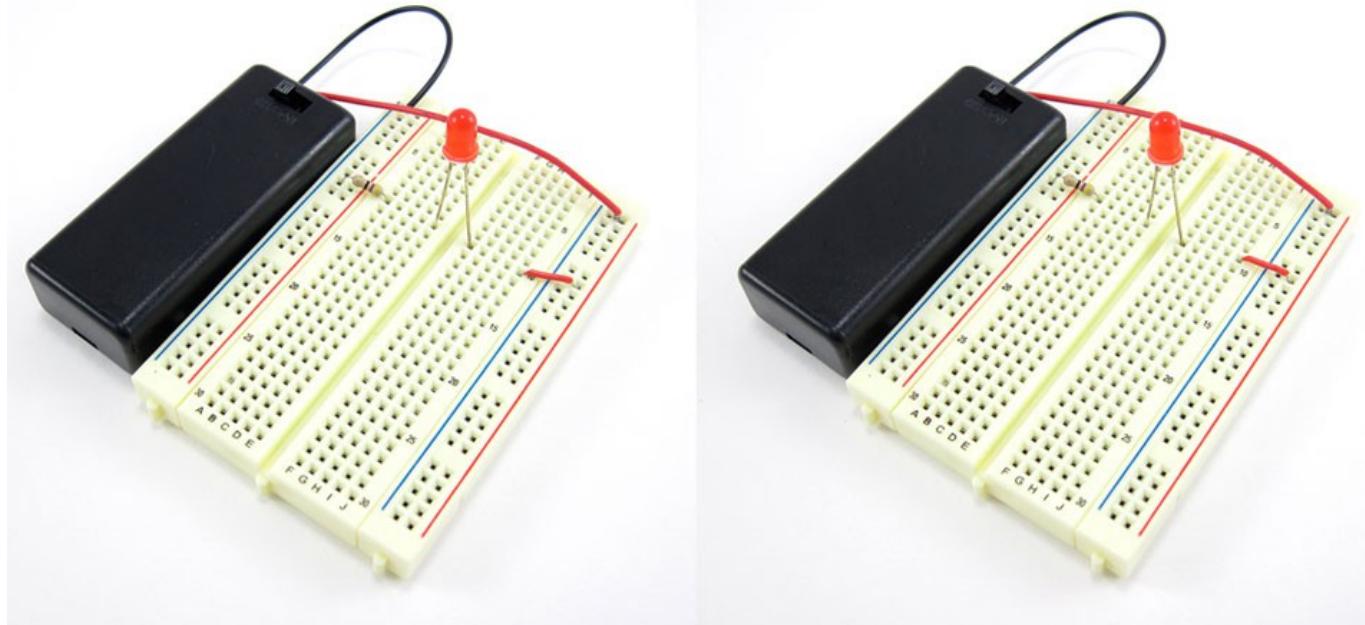
- 브레드보드 양쪽 가장자리에 위치한 **빨간색**과 **파란색** 라인
- **세로 방향**으로 길게 연결되어 있음
- 주로 전원(VCC)과 접지(GND)를 연결하여 회로 전체에 쉽게 공급하는 용도로 사용

#### 단자 스트립 (Terminal Strips)

- 브레드보드 중앙에 위치한 구멍들
- **가로 방향**으로 5개씩 짧게 연결되어 있음
- 중앙의 홈을 기준으로 위아래가 분리되어 있음
- 전자 부품(IC, 저항, LED 등)의 다리를 꽂아 서로 연결하는 데 사용

### 3. 브레드보드 사용 예시

LED와 저항을 연결하는 간단한 회로 예시입니다.



```
// 아두이노 코드
void setup() {
    pinMode(13, OUTPUT); // 13번 핀을 출력으로 설정
}

void loop() {
    digitalWrite(13, HIGH); // LED 켜기
}
```

1. 아두이노의 **5V** 핀을 브레드보드의 **빨간색 전원 버스**에 연결합니다.
2. 아두이노의 **GND** 핀을 브레드보드의 **파란색 전원 버스**에 연결합니다.
3. 저항의 한쪽 다리를 **단자 스트립**에 끒습니다.
4. \*\*LED의 긴 다리(양극)\*\*를 저항과 같은 가로 라인에 끚습니다.
5. \*\*LED의 짧은 다리(음극)\*\*를 다른 가로 라인에 끚고, 그 라인을 \*\*파란색 전원 버스(GND)\*\*에 연결합니다.
6. 아두이노의 **13번** 핀을 저항의 다른 쪽 다리와 같은 가로 라인에 연결합니다.



# Chapter 2

## 다양한 센서와 통신 방식

### 센서의 역할과 분류

[!NOTE] 이 문서는 아두이노에서 사용되는 센서의 역할과 종류에 대해 설명합니다.

#### 1. 센서(Sensor)란?

물리적 또는 환경적 변화를 감지하여 전기적 신호로 변환하는 장치입니다.

제품	가격	판매처
SparkFun Electronics... Sensor kit	₩10,061	DigiKey Sou...
SunFounder ST0050 IR... Modules	₩11,596	DigiKey Sou...
SunFounder ST0254 SW... Projects	₩11,596	DigiKey Sou...
DFRobot KIT011... Using arduino uno	₩114,672	DigiKey Sou...
Arduino A00005... Starter kit	₩36,885	DigiKey Sou...
장애물 감지 아... Sunfounder	₩2,200	아두이노 적외...
Seeed Technology... 37 in	₩5,023	DigiKey Sou...
Arduino ABX00021... Arduino mega	₩79,510	DigiKey Sou...
아두이노 초음... Temperature sensor	₩1,100	아두이노
Arduino ABX00003... 오마이	₩70,038	DigiKey Sou...
아두이노 가스... 아두이노 드 소리	₩22,000	에듀이노
센서 9종 세... 아두이노 예디이노	₩800	에디이노

Google은 상품 판매의 당사자가 아닙니다.

Ubuy South Korea  
98 태양 광 발전소의 태...

Random Nerd Tutorials  
21 Arduino Modules You Can Buy For L...

예24  
전자책] Arduino Se...

Makerguides.com  
Arduino Sensors For Everyone

DIYables products  
Sensors for Arduino and other MCUs - ...

Tutorial45  
Top Arduino Sensors - ...

FMUSER  
Arduino 센서란 무엇인가...

#### 아두이노에서 자주 사용되는 다양한 센서들

#### 센서의 역할

- 데이터 수집:** 주변 환경(온도, 습도, 밝기, 거리 등)의 정보를 수집합니다.
- 상호작용:** 사용자의 입력(버튼, 터치)이나 움직임을 감지하여 시스템과 상호작용합니다.
- 자동 제어:** 특정 조건(예: 어두워지면 조명 켜기)을 감지하여 시스템이 자동으로 동작하게 합니다.

#### 2. 센서의 분류

##### 입력 신호에 따른 분류

구분	설명	예시
아날로그 센서	연속적인 값을 출력하는 센서	조도 센서, 온도 센서(TMP36), 가변 저항
디지털 센서	특정 상태(ON/OFF, HIGH/LOW)를 출력하는 센서	버튼, 적외선 장애물 감지 센서, PIR 인체 감지 센서

### 측정 대상에 따른 분류

- **환경 센서**: 온도, 습도, 기압, 미세먼지, 가스 등
- **광학 센서**: 밝기(조도), 색상, 적외선 등
- **음향 센서**: 소리 크기, 주파수 등
- **기계적 센서**: 압력, 힘, 가속도, 자이로, 거리(초음파) 등
- **생체 센서**: 심박수, 지문 등

## 3. 아두이노와 센서 연결

대부분의 센서는 VCC(전원), GND(접지), SIG(신호) 3개의 핀으로 구성됩니다.

```
// 조도 센서 값 읽기 예시
int sensorPin = A0; // 아날로그 0번 핀에 센서 연결

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

void loop() {
    int sensorValue = analogRead(sensorPin); // 센서 값 읽기
    Serial.println(sensorValue); // 시리얼 모니터에 출력
    delay(100);
}
```



## Chapter 2

### 다양한 센서와 통신 방식

## 디지털 vs 아날로그 신호

[!NOTE] 이 문서는 디지털(Digital) 신호와 아날로그(Analog) 신호의 차이점에 대해 설명합니다.

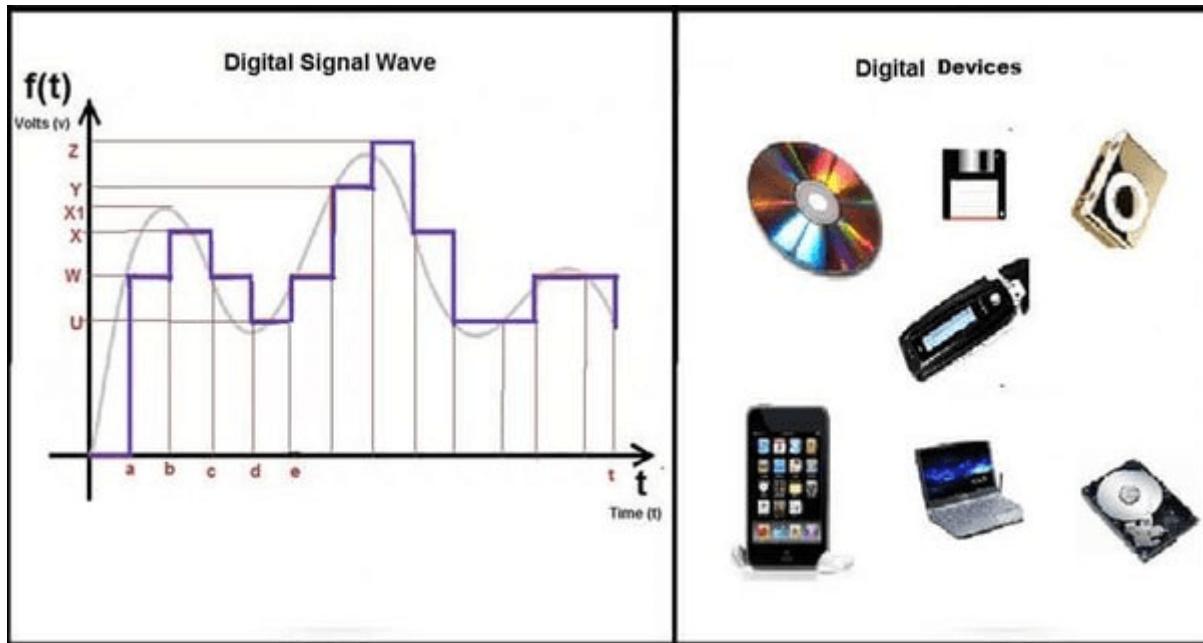


23:99:59



### 1. 신호(Signal)란?

정보를 전달하기 위한 전기적 또는 물리적 표현입니다. 아두이노에서는 주로 전압의 형태로 신호를 다룹니다.



아날로그 신호(위)와 디지털 신호(아래)의 파형 비교

## 2. 아날로그 신호 (Analog Signal)

연속적인 값을 가지는 신호입니다.

### 특징

- 연속성**: 신호의 크기가 끊기지 않고 연속적으로 변합니다.
- 정밀성**: 이론적으로 무한한 정밀도를 가질 수 있습니다.
- 외부 영향**: 노이즈(Noise)에 취약하여 신호가 왜곡되기 쉽습니다.
- 예시**: 목소리, 빛의 밝기, 온도의 변화 등 자연계의 대부분의 신호

### 아두이노에서의 아날로그 입력

- analogRead()** 함수를 사용하여 아날로그 핀(A0~A5)으로 들어오는 신호를 읽습니다.
- 0V ~ 5V 사이의 전압을 0 ~ 1023 사이의 정수 값으로 변환하여 반환합니다.

```
int sensorPin = A0;

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

void loop() {
  int sensorValue = analogRead(sensorPin); // 0~1023 사이의 값
  Serial.println(sensorValue);
  delay(100);
}
```

## 3. 디지털 신호 (Digital Signal)

이산적인 값(두 가지 상태)을 가지는 신호입니다.

## 특징

- **이산성**: 0과 1, 또는 LOW와 HIGH 두 가지 상태만 존재합니다.
- **명확성**: 신호의 상태가 명확하게 구분됩니다.
- **내구성**: 노이즈에 강하여 데이터의 신뢰도가 높습니다.
- **예시**: 스위치의 ON/OFF 상태, 컴퓨터의 데이터 처리(0과 1)

## 아두이노에서의 디지털 입출력

- `digitalRead()` 함수로 디지털 핀의 상태(HIGH 또는 LOW)를 읽습니다.
- `digitalWrite()` 함수로 디지털 핀에 HIGH(5V) 또는 LOW(0V) 신호를 출력합니다.

```
int ledPin = 13;
int buttonPin = 7;

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

void loop() {
    int buttonState = digitalRead(buttonPin);
    if (buttonState == HIGH) {
        digitalWrite(ledPin, HIGH);
    } else {
        digitalWrite(ledPin, LOW);
    }
}
```



# Chapter 2

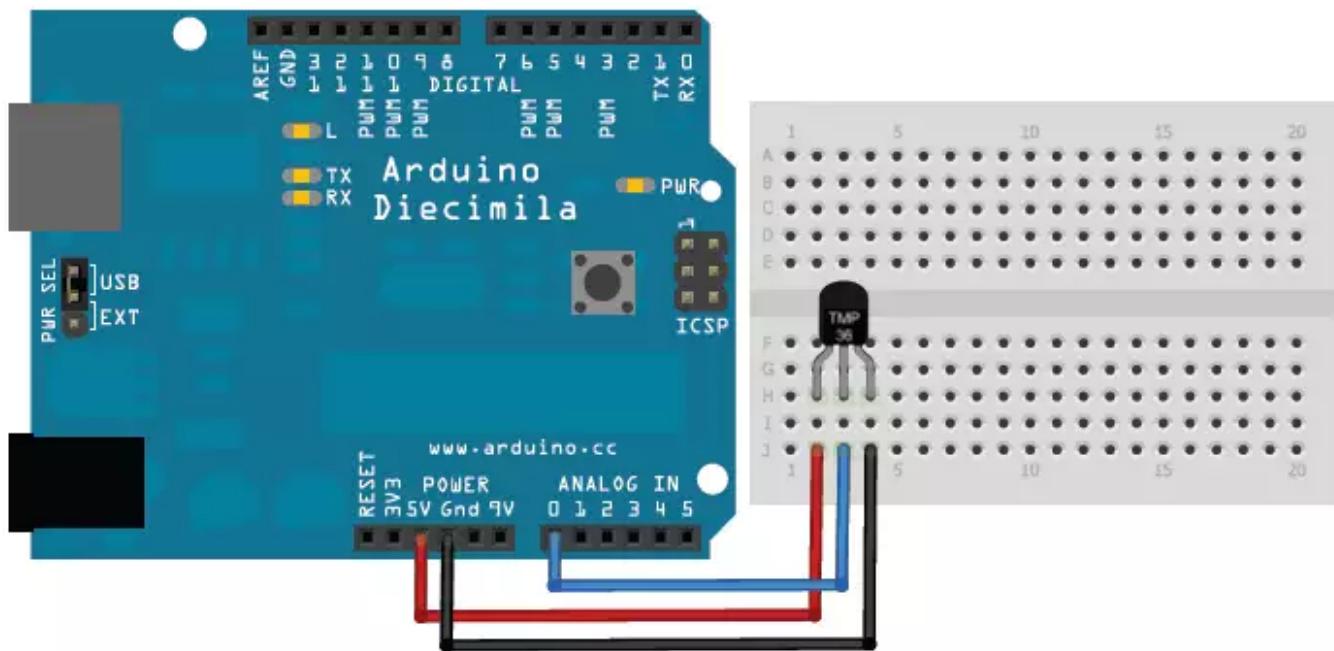
## 다양한 센서와 통신 방식

### 아두이노와 센서의 연결 방식/선 종류

[!NOTE] 이 문서는 **아두이노와 센서를 연결하는 기본적인 방법과 사용되는 전선 종류에 대해 설명합니다.**

#### 1. 기본적인 센서 연결

대부분의 센서 모듈은 3개 또는 4개의 핀을 가지고 있으며, 아두이노와 다음과 같이 연결됩니다.



일반적인 3핀 센서(온도 센서)의 연결 예시

#### 필수 연결 핀

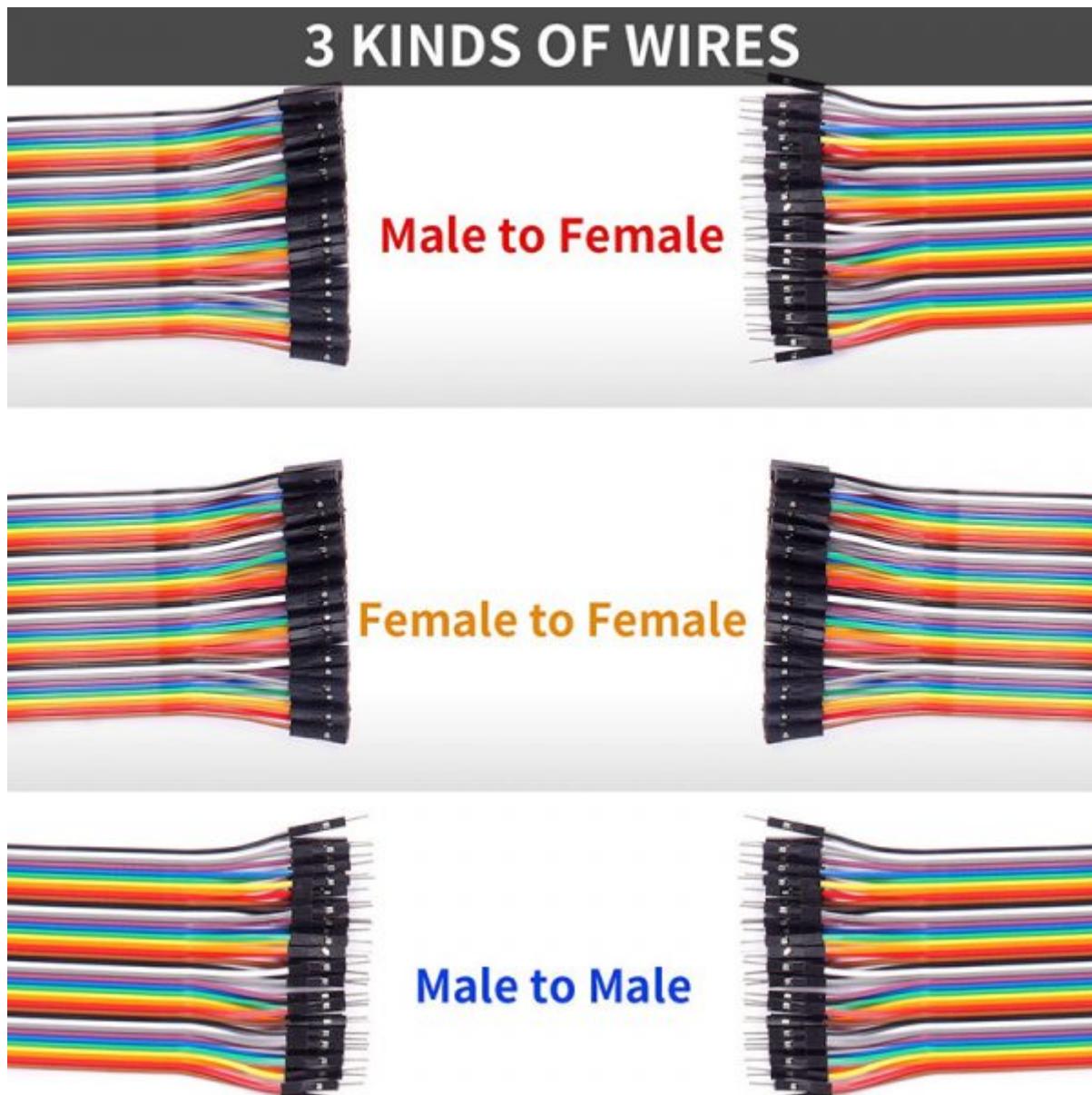
핀 이름	설명	아두이노 연결 핀
VCC 또는 5V	전원 공급 핀	5V
GND	접지 핀	GND
SIG 또는 OUT	신호 출력 핀	디지털 또는 아날로그 핀

추가 기능 핀 (센서에 따라 존재)

- **EN (Enable)**: 센서 동작을 활성화/비활성화하는 핀
- **SDA/SCL**: I2C 통신용 핀
- **TX/RX**: UART 시리얼 통신용 핀

## 2. 사용되는 전선(점퍼 와이어)의 종류

브레드보드와 아두이노, 센서를 연결할 때 주로 점퍼 와이어(Jumper Wire)를 사용합니다.



세 가지 종류의 점퍼 와이어

종류 및 용도

종류	형태	주요 용도
수-수 ( <b>Male-to-Male</b> )	양쪽 끝이 튀어나온 핀 형태	브레드보드 내부 연결, 아두이노-브레드보드 연결

종류	형태	주요 용도
수-암 ( <b>Male-to-Female</b> )	한쪽은 핀, 다른 쪽은 소켓 형태	아두이노 핀과 센서 모듈 핀 직접 연결
암-암 ( <b>Female-to-Female</b> )	양쪽 끝이 소켓 형태	핀 헤더가 있는 모듈 간의 연결

### 3. 연결 시 주의사항

#### [!NOTE]

[Chapter 5](#)에서 다양한 연결 오류에 대해 설명합니다.

- **전원 극성 확인:** VCC와 GND를 반대로 연결하지 않도록 주의합니다. 부품이 손상될 수 있습니다.
- **전압 확인:** 센서가 요구하는 동작 전압(5V 또는 3.3V)을 확인하고 아두이노의 해당 전원 핀에 연결합니다.
- **신호 핀 확인:** 아날로그 센서는 아날로그 핀(A0-A5)에, 디지털 센서는 디지털 핀에 연결합니다.
- **결선 상태 확인:** 선이 헐겁게 연결되지 않았는지 확인합니다. 불안정한 연결은 오작동의 원인이 됩니다.



# Chapter 2

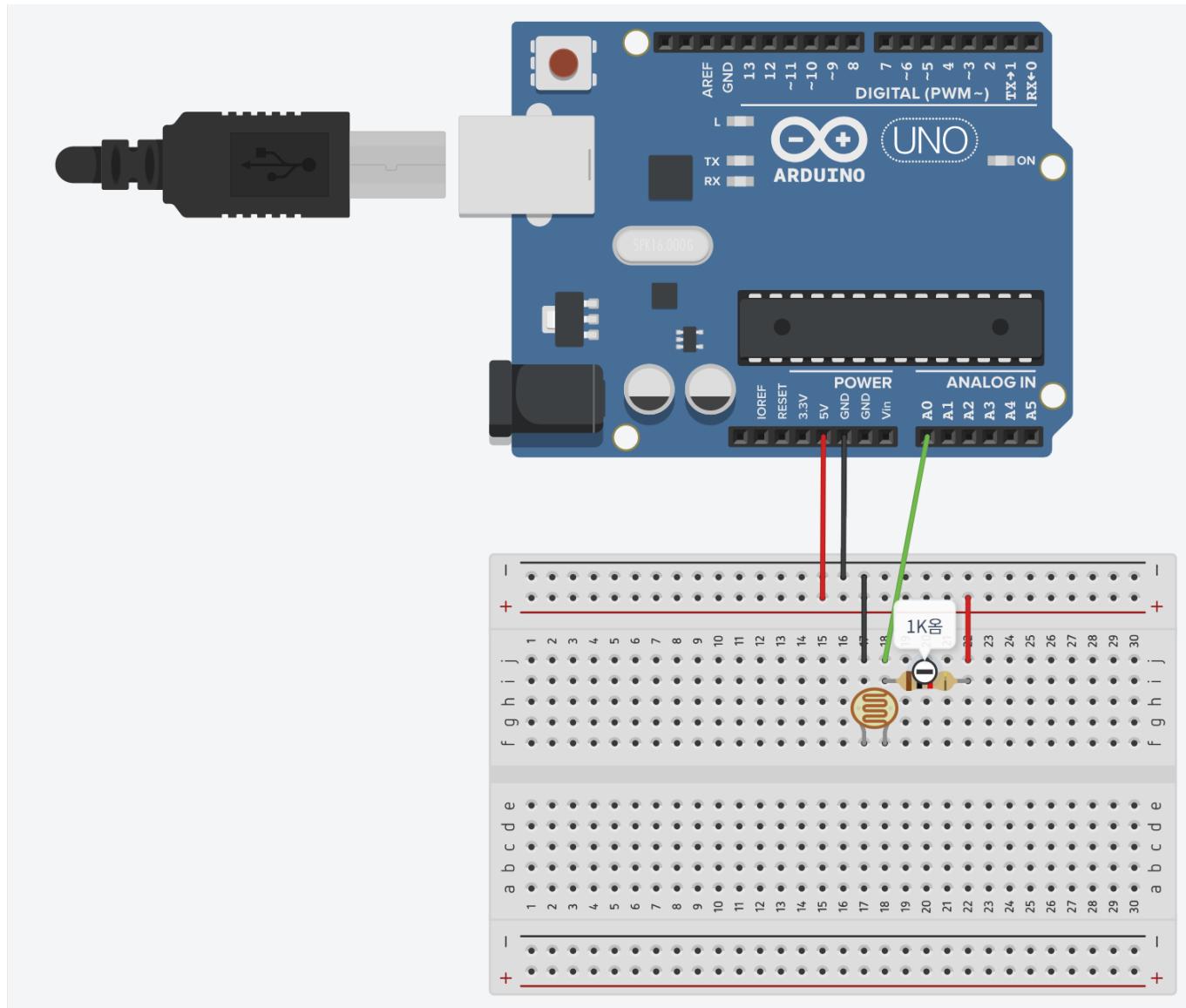
## 다양한 센서와 통신 방식

### 아두이노와 센서 연결 예시

[!NOTE] 이 문서는 대표적인 아두이노 센서들의 연결 예시를 보여줍니다.

#### 1. 조도 센서 (Analog)

빛의 밝기를 감지하는 아날로그 센서입니다.



[!TIP] 시각적으로 연결 회로를 빠르게 판단하기 위해 **VCC**는 빨간색, **GND**는 검정색 케이블로 연결합니다.

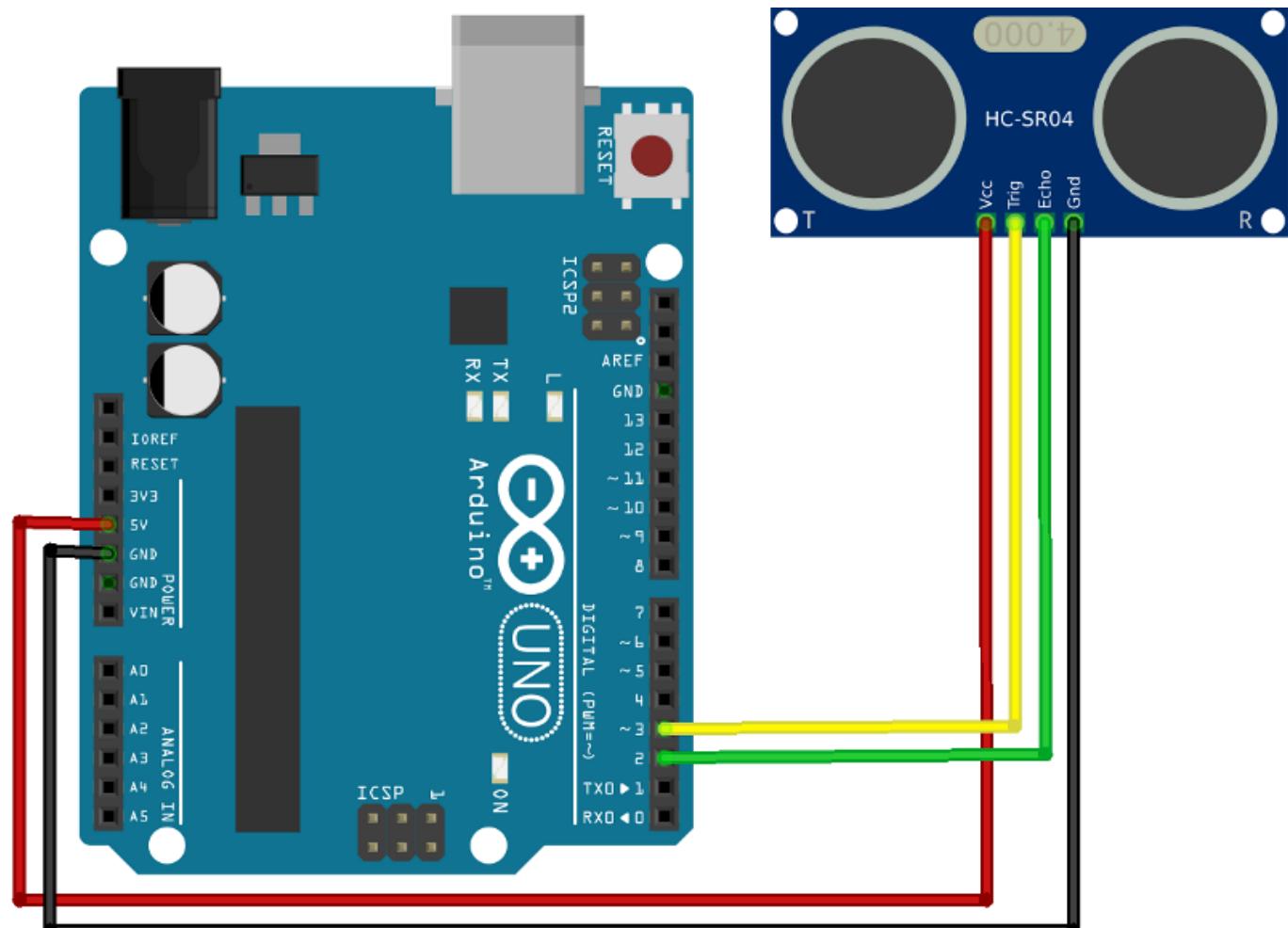
### 센서 핀 아두이노 연결 핀

VCC	5V
GND	GND
SIG	A0 (아날로그 핀)

```
int cdsPin = A0;
void setup() {
    Serial.begin(9600);
}
void loop() {
    int value = analogRead(cdsPin);
    Serial.println(value);
    delay(200);
}
```

## 2. 초음파 센서 (Digital)

초음파를 이용해 거리를 측정하는 디지털 센서입니다.



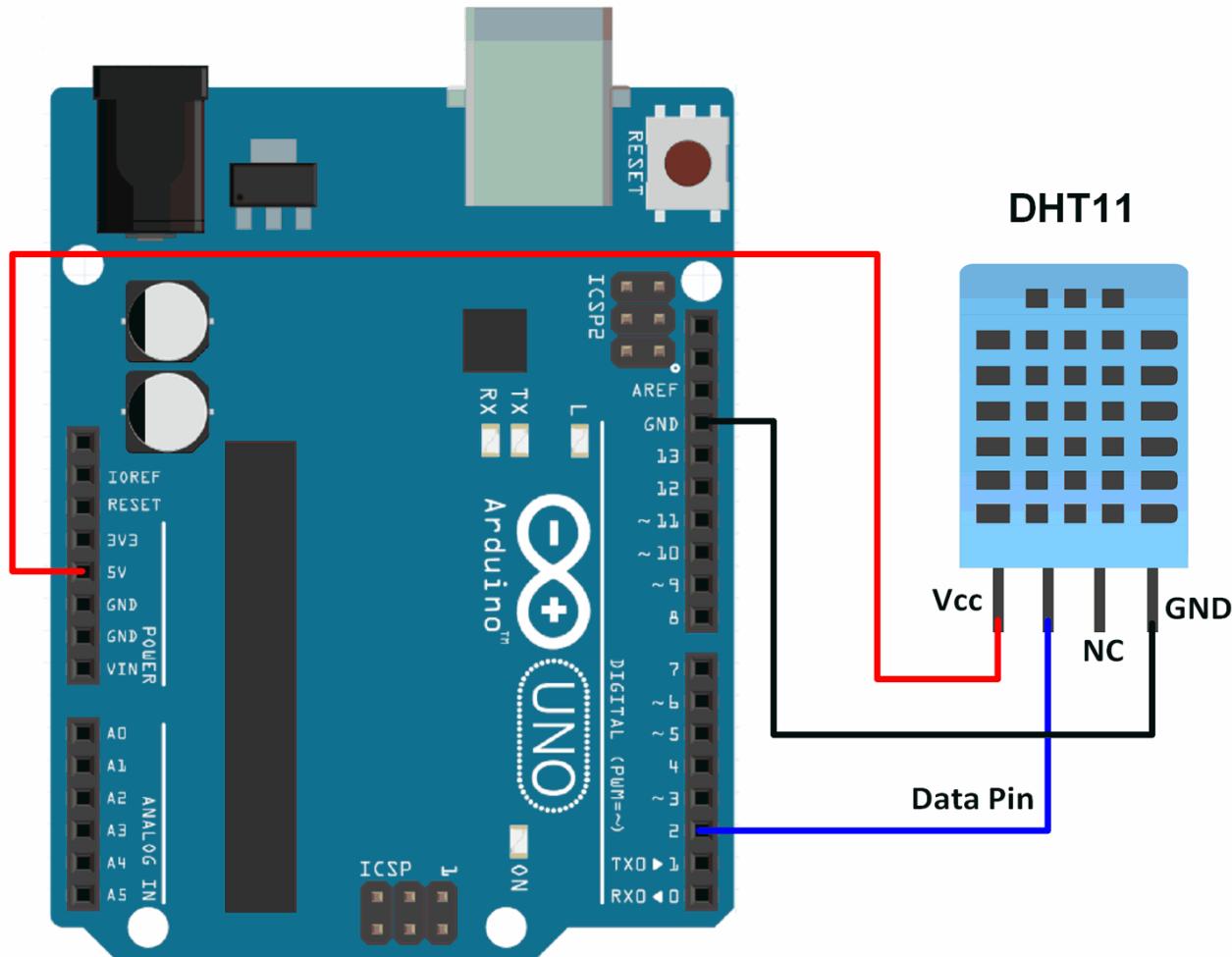
## 센서 핀 아두이노 연결 핀

VCC	5V
GND	GND
Trig	2 (디지털 핀)
Echo	3 (디지털 핀)

```
// 변수 이름은 꼭 직관적이게 지어야합니다.  
// a, aa, aaa 처럼 지으면 나중에 구분하기 어렵겠죠?  
int trigPin = 2;  
int echoPin = 3;  
void setup() {  
    Serial.begin(9600);  
    pinMode(trigPin, OUTPUT);  
    pinMode(echoPin, INPUT);  
}  
void loop() {  
    digitalWrite(trigPin, LOW);  
    delayMicroseconds(2);  
    digitalWrite(trigPin, HIGH);  
    delayMicroseconds(10);  
    digitalWrite(trigPin, LOW);  
    long duration = pulseIn(echoPin, HIGH);  
    int distance = duration * 0.034 / 2;  
    Serial.print(distance);  
    Serial.println(" cm");  
    delay(500);  
}
```

## 3. 온습도 센서 (Digital, DHT11)

온도와 습도를 함께 측정하는 디지털 센서입니다. (별도의 라이브러리 필요)



### 센서 핀 아두이노 연결 핀

VCC	5V
GND	GND
DATA	2 (디지털 핀)

```
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

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

void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
}
```

```
Serial.print("Humidity: ");
Serial.print(h);
Serial.print(" %\t");
Serial.print("Temperature: ");
Serial.print(t);
Serial.println(" *C");
delay(2000);
}
```



# Chapter 3

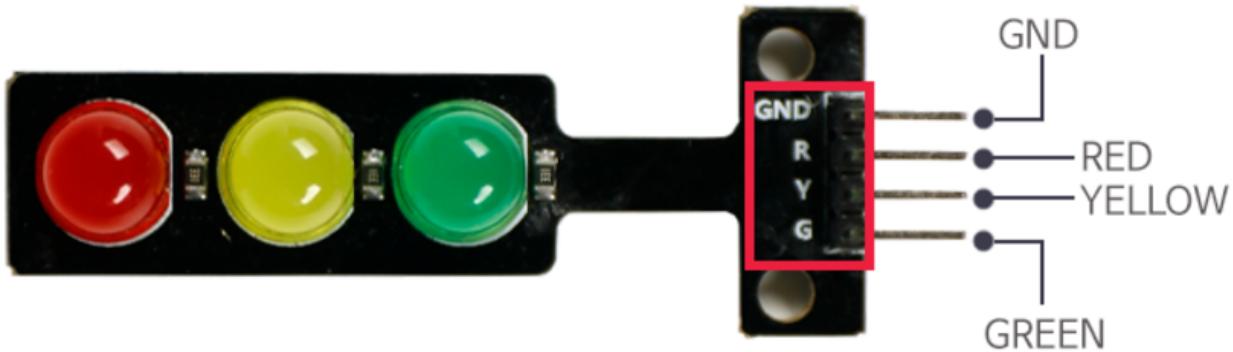
## 기본 입출력 실습

### 신호등 LED 제어

[!NOTE] 이 문서는 **3색 통합 LED**를 이용해 신호등을 만드는 실습에 대해 설명합니다.

#### 1. 실습 목표

빨강, 노랑, 초록 LED를 순차적으로 제어하여 실제 신호등처럼 동작하는 회로를 구성하고 프로그래밍합니다.



3색 LED 신호등 연결 핀 모습

#### 준비물

- 아두이노 우노
- 브레드보드
- LED (빨강, 노랑, 초록 통합)
- 점퍼 와이어

#### 2. 회로 구성

1. 3개의 LED를 각각 브레드보드에 꽂습니다.
2. 빨강 LED의 다리를 아두이노 디지털 **11번** 핀에 연결합니다.
3. 노랑 LED의 다리를 아두이노 디지털 **12번** 핀에 연결합니다.
4. 초록 LED의 다리를 아두이노 디지털 **13번** 핀에 연결합니다.

#### 3. 코드 작성

각 LED가 순서대로 켜지고 꺼지도록 코드를 작성합니다.

```
int redPin = 11;
int yellowPin = 12;
int greenPin = 13;

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

void loop() {
    // 1. 초록불 켜기 (3초)
    digitalWrite(greenPin, HIGH);
    delay(3000);
    digitalWrite(greenPin, LOW);

    // 2. 노란불 켜기 (1초)
    digitalWrite(yellowPin, HIGH);
    delay(1000);
    digitalWrite(yellowPin, LOW);

    // 3. 빨간불 켜기 (3초)
    digitalWrite(redPin, HIGH);
    delay(3000);
    digitalWrite(redPin, LOW);
}
```

## 동작 설명

1. `setup()` 함수에서 각 LED 핀을 출력으로 설정합니다.
2. `loop()` 함수가 시작되면 초록 LED가 3초간 켜졌다가 꺼집니다.
3. 이어서 노랑 LED가 1초간 켜졌다가 꺼집니다.
4. 마지막으로 빨강 LED가 3초간 켜졌다가 꺼집니다.
5. `loop()` 함수에 의해 이 과정이 계속 반복됩니다.

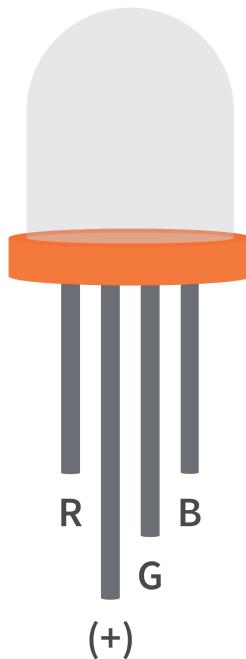


# Chapter 3

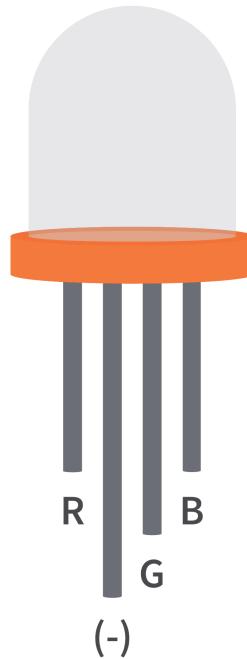
## 기본 입출력 실습

### RGB LED 제어

[!NOTE] 이 문서는 \*\*3가지 색(빨강, 초록, 파랑)이 들어있는 3색 LED(RGB LED)\*\*를 제어하는 실습에 대해 설명합니다.



Common Anode

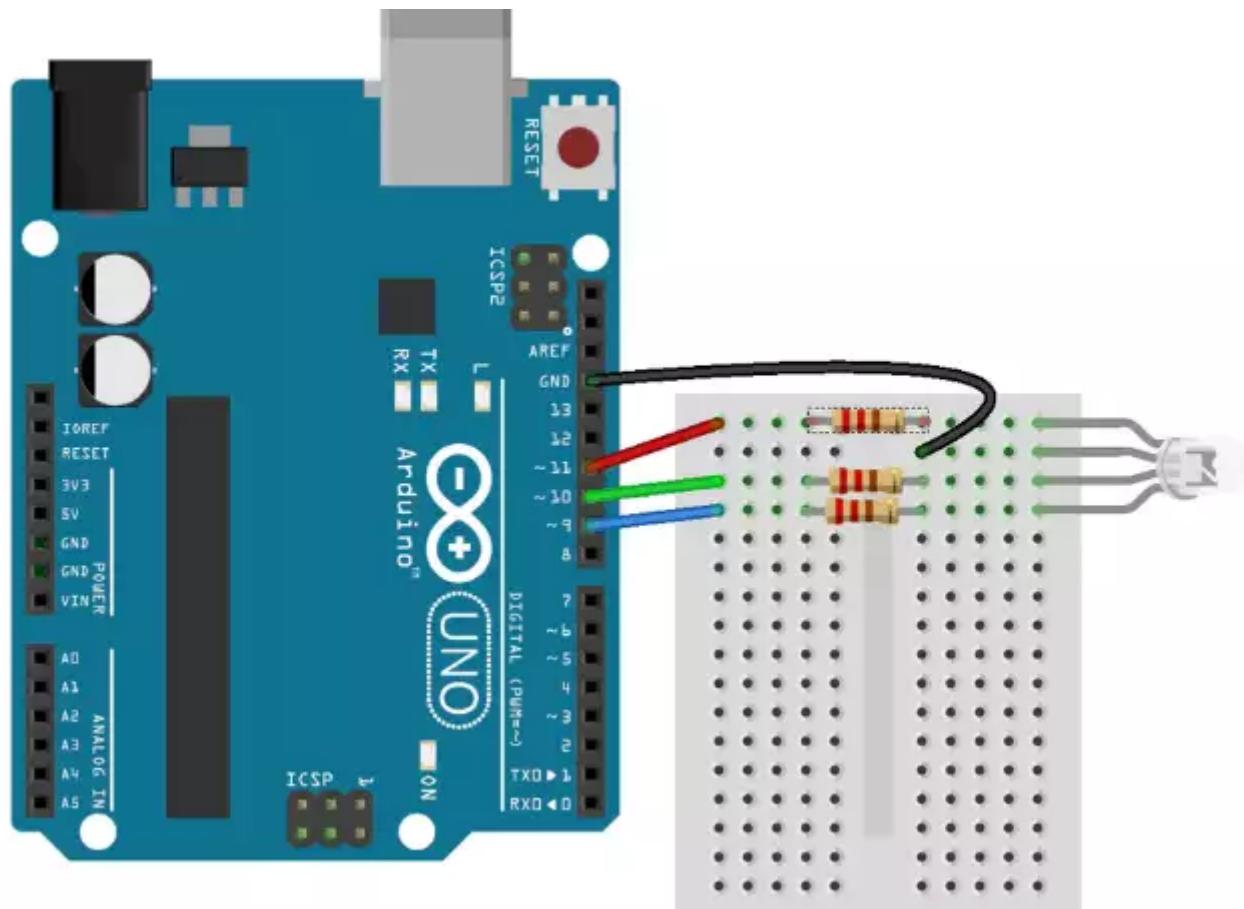


Common Cathode

[!WARNING] 본인의 RGB LED가 어떤 형태인지 먼저 확인해야합니다.

#### 1. 실습 목표

3색 LED를 이용하여 다양한 색상을 조합하고 표현하는 방법을 학습합니다.



### 3색 LED(공통 음극 탑입) 회로 구성 예시

#### 준비물

- 아두이노 우노
- 브레드보드
- 3색 LED (RGB LED, 공통 음극 탑입)
- 220Ω 저항 3개
- 점퍼 와이어

## 2. 3색 LED란?

빛의 삼원색인 빨강(Red), 초록(Green), 파랑(Blue) LED가 하나의 패키지에 합쳐진 부품입니다. 각 색의 밝기를 조절하여 다양한 색상을 만들 수 있습니다.

- **공통 양극(Common Anode):** 공통 핀을 VCC에 연결. 각 색상 핀에 LOW 신호를 주면 해당 색상이 켜짐.
- **공통 음극(Common Cathode):** 공통 핀을 GND에 연결. 각 색상 핀에 HIGH 신호를 주면 해당 색상이 켜짐. (본 실습에서는 공통 음극 사용)

## 3. 회로 구성 (공통 음극 기준)

1. 3색 LED의 가장 긴 다리(공통 음극)를 아두이노의 **GND**에 연결합니다.
2. 빨강(R) 핀을 220Ω 저항을 거쳐 아두이노 디지털 **9번** 핀(PWM)에 연결합니다.
3. 초록(G) 핀을 220Ω 저항을 거쳐 아두이노 디지털 **10번** 핀(PWM)에 연결합니다.
4. 파랑(B) 핀을 220Ω 저항을 거쳐 아두이노 디지털 **11번** 핀(PWM)에 연결합니다.

## 4. 코드 작성

[analogWrite()] 함수를 사용하여 각 색상의 밝기를 0~255 단계로 조절합니다.

```
int redPin = 9;
int greenPin = 10;
int bluePin = 11;

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

void loop() {
    setColor(255, 0, 0); // 빨강색
    delay(1000);
    setColor(0, 255, 0); // 초록색
    delay(1000);
    setColor(0, 0, 255); // 파랑색
    delay(1000);
    setColor(255, 255, 0); // 노랑색 (빨강 + 초록)
    delay(1000);
    setColor(128, 0, 128); // 보라색 (빨강 + 파랑)
    delay(1000);
    setColor(0, 255, 255); // 청록색 (초록 + 파랑)
    delay(1000);
    setColor(255, 255, 255); // 흰색 (모든 색상)
    delay(1000);
}

void setColor(int red, int green, int blue) {
    analogWrite(redPin, red);
    analogWrite(greenPin, green);
    analogWrite(bluePin, blue);
}
```

[!IMPORTANT] 신호등 LED에서는 저항을 사용하지 않았는데 왜 RGB LED는 저항을 사용했을까요? 겉모습을 보면 이유를 생각해봐요.



# Chapter 3

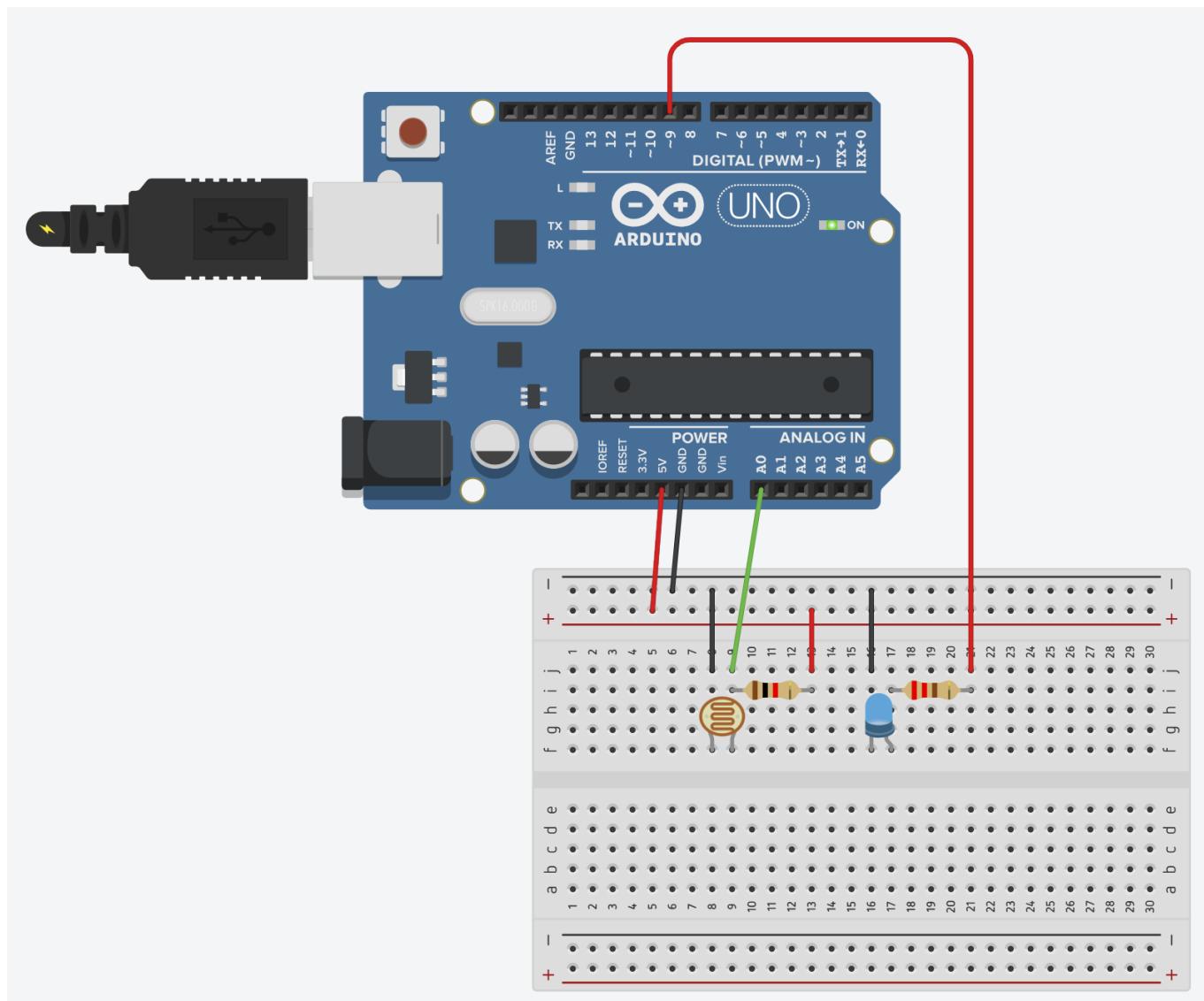
## 기본 입출력 실습

### 조도센서 응용

[!NOTE] 이 문서는 \*\*조도 센서(CdS, Photoresistor)\*\*와 **LED**를 사용하여 빛의 밝기를 측정하는 실습에 대해 설명합니다.

#### 1. 실습 목표

조도 센서의 값을 아날로그 입력으로 읽어와 시리얼 모니터에 출력하고, 특정 밝기 값에 따라 LED를 제어합니다.



## 조도 센서와 LED를 함께 사용한 회로 예시

### 준비물

- 아두이노 우노
- 브레드보드
- 조도 센서 (CdS)
- 10kΩ 저항 1개 (풀다운 저항용)
- LED 1개
- 220Ω 저항 1개 (LED 보호용)
- 점퍼 와이어

## 2. 조도 센서(CdS)란?

빛의 양에 따라 저항값이 변하는 소자입니다. 밝을수록 저항이 낮아지고, 어두울수록 저항이 높아집니다. 아두이노는 저항값을 직접 읽을 수 없으므로, 전압 분배 법칙을 이용하여 변화하는 전압을 측정합니다.

## 3. 회로 구성

1. 아두이노 **5V** 핀을 브레드보드의 한쪽 단자 스트립 라인에 연결합니다.
2. 조도 센서의 한쪽 다리를 위에서 연결한 5V 라인에 연결합니다.
3. 조도 센서의 다른 쪽 다리를 아두이노 **A0** 핀과 10kΩ 저항의 한쪽 끝에 함께 연결합니다.
4. 10kΩ 저항의 다른 쪽 끝을 아두이노 **GND**에 연결합니다. (풀다운 저항)
5. LED의 긴 다리를 220Ω 저항을 거쳐 아두이노 디지털 **9번** 핀에 연결합니다.
6. LED의 짧은 다리를 아두이노 **GND**에 연결합니다.

## 4. 코드 작성

조도 센서 값에 따라 LED가 켜지고 꺼지도록 코드를 작성합니다.

```
int cdsPin = A0;
int ledPin = 9;

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

void loop() {
    int sensorValue = analogRead(cdsPin);
    Serial.println(sensorValue);

    // 어두워지면(센서 값이 특정 값 이하이면) LED 켜기
    // 기준값(예: 300)은 주변 환경에 따라 조절 필요
    if (sensorValue < 300) {
        digitalWrite(ledPin, HIGH); // LED 켜기
    } else {
        digitalWrite(ledPin, LOW); // LED 끄기
    }
}
```

```
    delay(100);  
}
```

## 동작 설명

1. `analogRead(cdsPin)`를 통해 조도 센서의 아날로그 값을 읽어옵니다.
2. 읽어온 값은 시리얼 모니터에 출력됩니다. (주변 밝기에 따라 값이 어떻게 변하는지 확인)
3. `if` 문을 사용하여 센서 값이 300보다 작아지면(어두워지면) 9번 핀에 연결된 LED가 켜지고, 그렇지 않으면(밝으면) 꺼집니다.



# Chapter 3

## 기본 입출력 실습

### 초음파센서 응용

[!NOTE] 이 문서는 \*\*초음파 센서(HC-SR04)\*\*를 사용하여 거리를 측정하는 실습에 대해 설명합니다.

#### 1. 실습 목표

초음파 센서로 측정한 거리를 시리얼 모니터에 출력하고, 특정 거리 이내에 물체가 감지되면 LED를 켜는 프로그램을 작성합니다.

지금까지 배운 내용으로 직접 회로를 구현해봐요

#### 준비물

- 아두이노 우노
- 브레드보드
- 초음파 센서 (HC-SR04)
- LED 1개
- $220\Omega$  저항 1개
- 점퍼 와이어

#### 2. 초음파 센서란?

사람이 들을 수 없는 초음파를 발사하고, 물체에 반사되어 돌아오는 시간을 측정하여 거리를 계산하는 센서입니다.

- **Trig (Trigger)**: 초음파를 발사시키는 핀
- **Echo (Echo)**: 반사된 초음파를 수신하는 핀
- 거리 계산 공식:  $\text{거리(cm)} = (\text{초음파 왕복 시간}(\mu\text{s}) * 0.034) / 2$

#### 3. 회로 구성

1. 초음파 센서의 **VCC** 핀을 아두이노 **5V**에, **GND** 핀을 **GND**에 연결합니다.
2. 초음파 센서의 **Trig** 핀을 아두이노 디지털 **9번** 핀에 연결합니다.
3. 초음파 센서의 **Echo** 핀을 아두이노 디지털 **10번** 핀에 연결합니다.
4. LED의 긴 다리를  $220\Omega$  저항을 거쳐 아두이노 디지털 **11번** 핀에 연결합니다.
5. LED의 짧은 다리를 아두이노 **GND**에 연결합니다.

#### 4. 코드 작성

측정된 거리에 따라 LED가 켜지고 꺼지도록 코드를 작성합니다.

```
int trigPin = 9;
int echoPin = 10;
int ledPin = 11;

void setup() {
    Serial.begin(9600);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    pinMode(ledPin, OUTPUT);
}

void loop() {
    // 1. 초음파 발사
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    // 2. 초음파 수신 및 시간 측정
    long duration = pulseIn(echoPin, HIGH);

    // 3. 거리 계산
    int distance = duration * 0.034 / 2;

    // 4. 시리얼 모니터에 거리 출력
    Serial.print(distance);
    Serial.println(" cm");

    // 5. 거리가 10cm 이내이면 LED 켜기
    if (distance < 10) {
        digitalWrite(ledPin, HIGH);
    } else {
        digitalWrite(ledPin, LOW);
    }

    delay(200);
}
```

## 동작 설명

1. Trig 핀에 짧은 HIGH 신호를 보내 초음파를 발사합니다.
2. pulseIn() 함수를 사용하여 Echo 핀으로 초음파가 돌아올 때까지의 시간을 측정합니다.
3. 측정된 시간을 거리 계산 공식에 대입하여 cm 단위의 거리를 구합니다.
4. 계산된 거리가 10cm보다 가까우면 LED가 켜지고, 그렇지 않으면 꺼집니다.



# Chapter 3

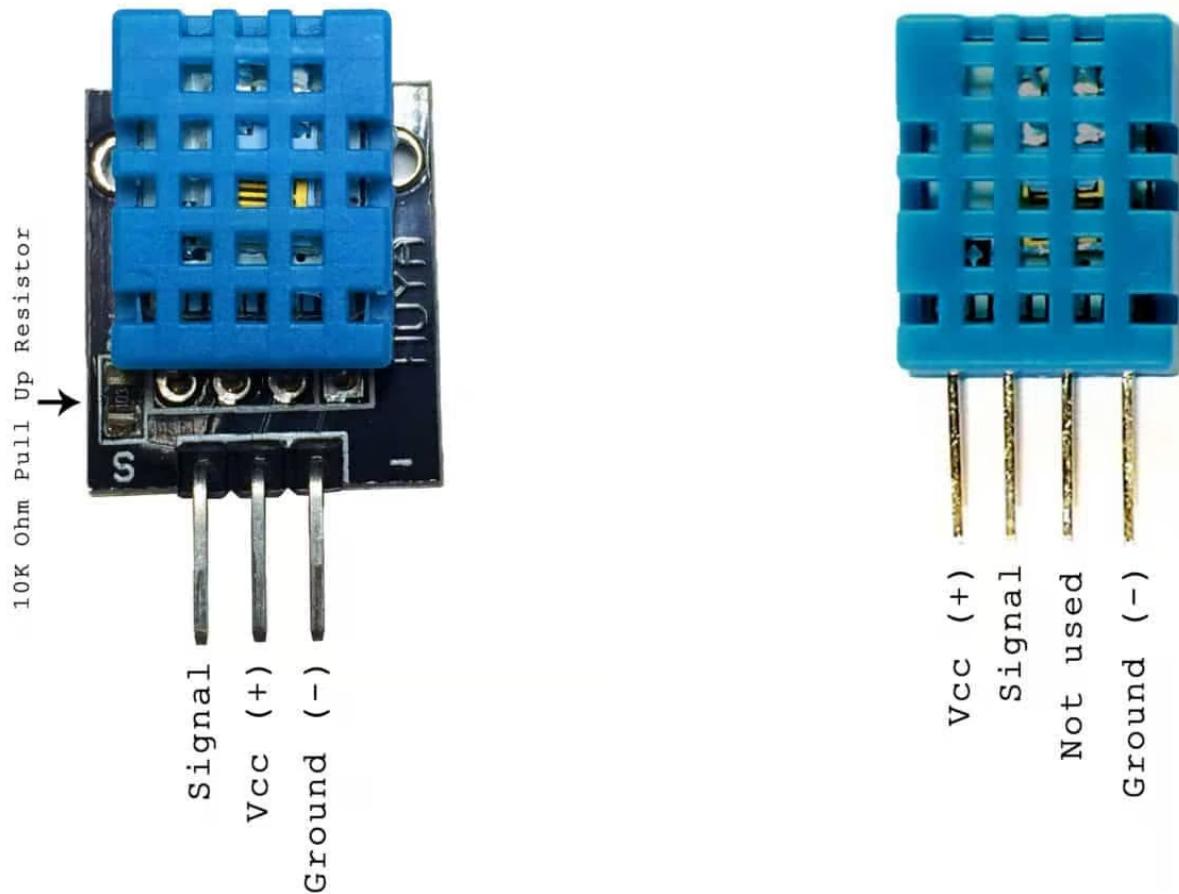
## 기본 입출력 실습

### 온/습도 센서

[!NOTE] 이 문서는 \*\*온/습도 센서(DHT11)\*\*를 사용하여 온도와 습도를 측정하는 실습에 대해 설명합니다.

#### 1. 실습 목표

DHT11 온/습도 센서를 아두이노에 연결하고, 관련 라이브러리를 설치하여 온도와 습도 값을 시리얼 모니터에 출력합니다.



DHT11에 10K 저항이 달려있는 경우도 있으니 확인

준비물

- 아두이노 우노
- 브레드보드
- 온/습도 센서 (DHT11)
- 점퍼 와이어
- (필요시) 10kΩ 저항 1개 (풀업 저항용, 모듈에 내장된 경우 생략 가능)

## 2. DHT11 센서란?

온도와 습도를 측정하여 디지털 신호로 출력하는 복합 센서입니다. 자체적인 통신 프로토콜을 사용하므로, 제어를 위해 별도의 라이브러리가 필요합니다.

- **측정 범위:** 온도 0~50°C (오차 ±2°C), 습도 20~90% (오차 ±5%)
- **동작 전압:** 3.3V ~ 5V

## 3. 라이브러리 설치

1. 아두이노 IDE를 실행합니다.
2. **스케치 > 라이브러리 포함하기 > 라이브러리 관리** 메뉴를 엽니다.
3. 라이브러리 매니저 검색창에 **DHT sensor library**를 검색합니다.
4. **Adafruit**에서 제작한 라이브러리를 찾아 **설치** 버튼을 클릭합니다.
5. 의존성 라이브러리(**Adafruit Unified Sensor**)도 함께 설치할지 묻는 창이 나타나면 **Install all**을 클릭합니다.

## 4. 회로 구성

1. DHT11 센서의 **VCC(또는 +)** 핀을 아두이노 **5V**에 연결합니다.
2. DHT11 센서의 **GND(또는 -)** 핀을 아두이노 **GND**에 연결합니다.
3. DHT11 센서의 **DATA(또는 OUT)** 핀을 아두이노 디지털 **2번** 핀에 연결합니다.

## 5. 코드 작성

설치한 DHT 라이브러리를 사용하여 온도와 습도 값을 읽어옵니다.

```
// 1. 라이브러리 포함
#include "DHT.h"

// 2. 핀 및 타입 설정
#define DHTPIN 2          // 센서가 연결된 핀 번호
#define DHTTYPE DHT11     // 센서 타입 (DHT11)

// 3. DHT 객체 생성
DHT dht(DHTPIN, DHTTYPE);

void setup() {
    Serial.begin(9600);
    Serial.println("DHT11 test!");

    dht.begin(); // 센서 초기화
}
```

```
void loop() {
    // 2초마다 센서 값 읽기
    delay(2000);

    // 4. 습도 및 온도 값 읽기
    float h = dht.readHumidity();
    float t = dht.readTemperature(); // 섭씨(Celsius)
    // float f = dht.readTemperature(true); // 화씨(Fahrenheit)

    // 5. 값 확인 및 출력
    if (isnan(h) || isnan(t)) {
        Serial.println("Failed to read from DHT sensor!");
        return;
    }

    Serial.print("Humidity: ");
    Serial.print(h);
    Serial.print(" %\t");
    Serial.print("Temperature: ");
    Serial.print(t);
    Serial.println(" *C");
}
```



# Chapter 4

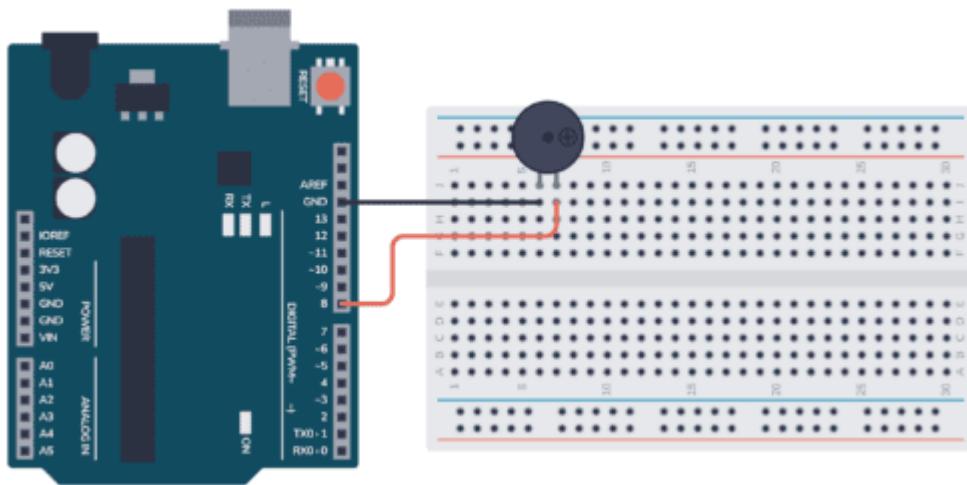
## 고급 출력 장치 제어

### 수동부저 소리 재생

[!NOTE] 이 문서는 \*\*수동 부저(Passive Buzzer)\*\*를 사용하여 특정 음계를 재생하는 실습에 대해 설명합니다.

#### 1. 실습 목표

`tone()` 함수를 이용하여 수동 부저로 '도레미파솔라시도' 음계를 연주하는 프로그램을 작성합니다.



수동 부저 회로 구성 예시

#### 준비물

- 아두이노 우노
- 브레드보드
- 수동 부저 (Passive Buzzer)
- 점퍼 와이어

#### 2. 수동 부저 vs 능동 부저

- **능동 부저 (Active Buzzer):** 전원만 연결하면 내장된 회로에 의해 정해진 단일음이 발생합니다. (`digitalWrite()`로 제어)
- **수동 부저 (Passive Buzzer):** 특정 주파수의 신호를 주어야 소리가 발생하며, 주파수를 바꾸어 다양한 음을 만들 수 있습니다. (`tone()` 함수로 제어)

### 3. 회로 구성

- 수동 부저의 양극(+) 핀 (또는 긴 다리)을 아두이노 디지털 **8번** 핀에 연결합니다.
- 수동 부저의 음극(-) 핀 (또는 짧은 다리)을 아두이노 **GND**에 연결합니다.

### 4. 코드 작성

각 음계에 해당하는 주파수 값을 배열에 저장하고, **tone()** 함수를 이용해 순서대로 재생합니다.

NOTE FREQUENCY CHART | HEROIC AUDIO

	Octave 0	Octave 1	Octave 2	Octave 3	Octave 4	Octave 5	Octave 6	Octave 7	Octave 8	Octave 9	Octave 10
C	16.35	32.70	65.41	130.81	261.63	523.25	1046.50	2093.00	4186.01	8372.02	16744.04
C#	17.32	34.65	69.30	138.59	277.18	554.37	1108.73	2217.46	4434.92	8869.84	17739.69
D	18.35	36.71	73.42	146.83	293.66	587.33	1174.66	2349.32	4698.64	9397.27	18794.55
D#	19.45	38.89	77.78	155.56	311.13	622.25	1244.51	2489.02	4978.03	9956.06	19912.13
E	20.60	41.20	82.41	164.81	329.63	659.26	1318.51	2637.02	5274.04	10548.08	
F	21.83	43.65	87.31	174.61	349.23	698.46	1396.91	2793.83	5587.65	11175.30	
F#	23.12	46.25	92.50	185.00	369.99	739.99	1479.98	2959.96	5919.91	11839.82	
G	24.50	49.00	98.00	196.00	392.00	783.99	1567.98	3135.96	6271.93	12543.86	
G#	25.96	51.91	103.83	207.65	415.30	830.61	1661.22	3322.44	6644.88	13289.75	
A	27.50	55.00	110.00	220.00	440.00	880.00	1760.00	3520.00	7040.00	14080.00	
A#	29.14	58.27	116.54	233.08	466.16	932.33	1864.66	3729.31	7458.62	14917.24	
B	30.87	61.74	123.47	246.94	493.88	987.77	1975.53	3951.07	7902.13	15804.26	

```
int buzzerPin = 8;

// 음계별 주파수 (4옥타브 기준)
// 도, 레, 미, 파, 솔, 라, 시, 도
int scale[] = {262, 294, 330, 349, 392, 440, 494, 523};

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

void loop() {
    // '도'부터 순서대로 음계 연주
    for (int i = 0; i < 8; i++) {
        // tone(핀 번호, 주파수, 지속시간(ms));
        tone(buzzerPin, scale[i], 500);
        delay(500); // 다음 음과의 간격
    }

    // 2초간 휴식 후 반복
    delay(2000);
}
```

## tone() 함수 사용법

- `tone(pin, frequency)`: 지정된 `pin`에서 `frequency`(Hz)의 소리를 계속 재생합니다.
- `tone(pin, frequency, duration)`: 지정된 `pin`에서 `frequency`(Hz)의 소리를 `duration`(ms)만큼 재생합니다.
- `noTone(pin)`: 지정된 `pin`의 소리 재생을 중지합니다.

## 동작 설명

1. `scale` 배열에 '도'부터 높은 '도'까지 8개의 음계에 해당하는 주파수 값을 저장합니다.
2. `for` 반복문을 사용하여 배열의 첫 번째 값부터 마지막 값까지 순차적으로 접근합니다.
3. `tone()` 함수가 호출되어 `buzzerPin`(8번 핀)에서 해당 주파수(`scale[i]`)의 소리를 0.5초(500ms)간 재생합니다.
4. 0.5초의 딜레이 후 다음 음을 연주하며, 모든 음계 연주가 끝나면 2초 쉬고 다시 반복합니다.



# Chapter 4

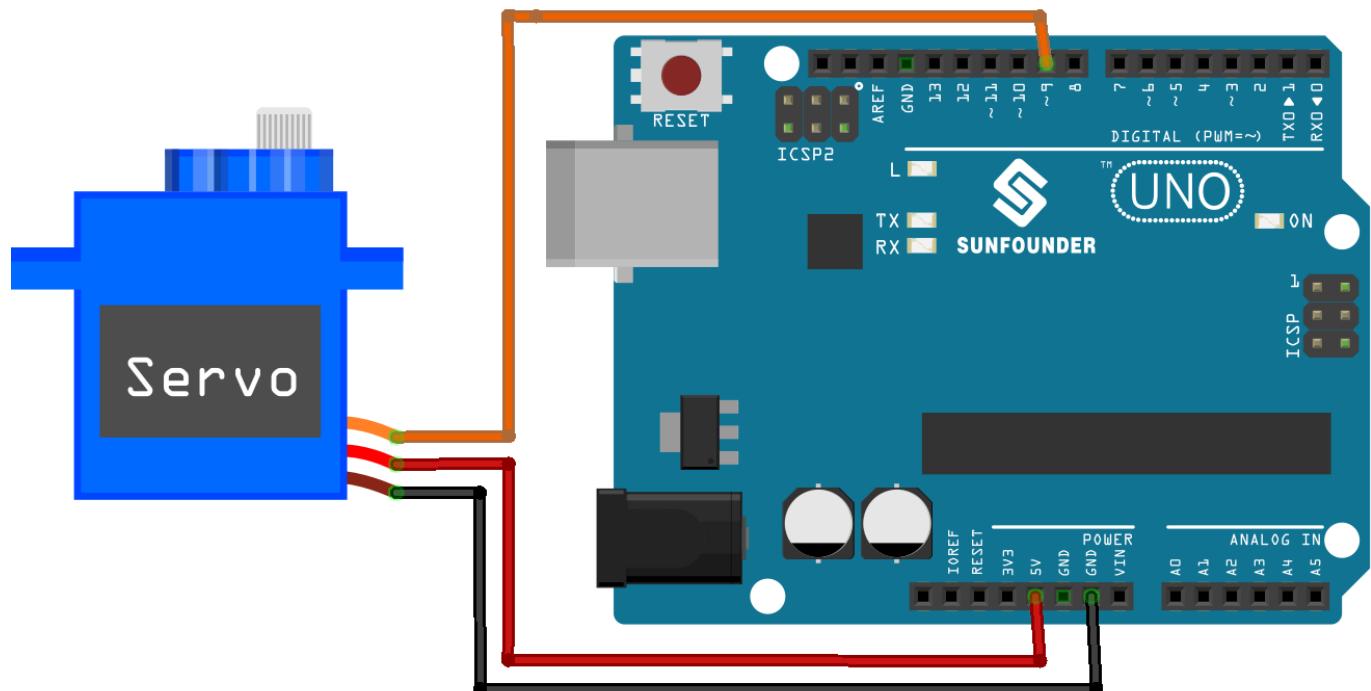
## 고급 출력 장치 제어

### 서보모터 회전 제어

[!NOTE] 이 문서는 \*\*서보 모터(Servo Motor)\*\*를 제어하여 원하는 각도로 회전시키는 실습에 대해 설명합니다.

#### 1. 실습 목표

서보 모터 라이브러리를 사용하여 모터를 0도, 90도, 180도로 반복적으로 회전시키는 프로그램을 작성합니다. 단, 180도 이상 회전할 수 없습니다.



서보 모터(SG90) 회로 구성 예시

#### 준비물

- 아두이노 우노
- 브레드보드
- 서보 모터 (SG90)
- 점퍼 와이어

#### 2. 서보 모터란?

일반 모터와 달리, 원하는 각도로 정밀하게 회전하고 그 위치를 유지할 수 있는 모터입니다. 로봇 팔, RC 카의 조향 장치 등 특정 각도 제어가 필요한 곳에 널리 사용됩니다.

- **제어 방식:** PWM(펄스 폭 변조) 신호의 펄스 폭(길이)에 따라 모터의 각도가 결정됩니다.
- **기본 라이브러리:** 아두이노 IDE에 **Servo.h** 라이브러리가 내장되어 있어 쉽게 제어할 수 있습니다.

### 3. 회로 구성

서보 모터는 3개의 선으로 구성되어 있습니다.

선 색상	핀 이름	기능	아두이노 연결 핀
갈색 또는 검정	GND	접지	GND
빨강	VCC	전원	5V
주황 또는 노랑	Signal	신호선	디지털 9번 핀 (PWM)

**주의:** 여러 개의 서보 모터를 사용하거나 큰 서보 모터를 사용할 경우, 아두이노의 5V 전원만으로는 부족할 수 있습니다. 이 경우 외부 전원을 별도로 연결해야 합니다.

### 4. 코드 작성

**Servo.h** 라이브러리를 사용하여 서보 모터를 제어합니다.

```
// 1. 서보 라이브러리 포함
#include <Servo.h>

// 2. 서보 객체 생성
Servo myServo;

int servoPin = 9;

void setup() {
    // 3. 서보 모터를 지정된 핀에 연결
    myServo.attach(servoPin);
}

void loop() {
    // 4. write() 함수로 각도 이동
    myServo.write(0);    // 0도로 이동
    delay(1000);        // 1초 대기

    myServo.write(90);   // 90도로 이동
    delay(1000);        // 1초 대기

    myServo.write(180); // 180도로 이동
    delay(1000);        // 1초 대기
}
```

**Servo** 라이브러리 주요 함수

- `attach(pin)`: 서보 모터를 제어할 핀을 설정합니다.
- `write(angle)`: 서보 모터를 `angle` (0 ~ 180) 각도로 이동시킵니다.
- `writeMicroseconds(us)`: 펠스 폭을 마이크로초 단위로 직접 설정하여 모터를 제어합니다. (일반적으로 1000: 0도, 1500: 90도, 2000: 180도)
- `read()`: 현재 서보 모터의 각도 값을 반환합니다.
- `detach()`: 서보 모터와 핀의 연결을 해제합니다. 모터에 전력 공급이 중단됩니다.



# Chapter 4

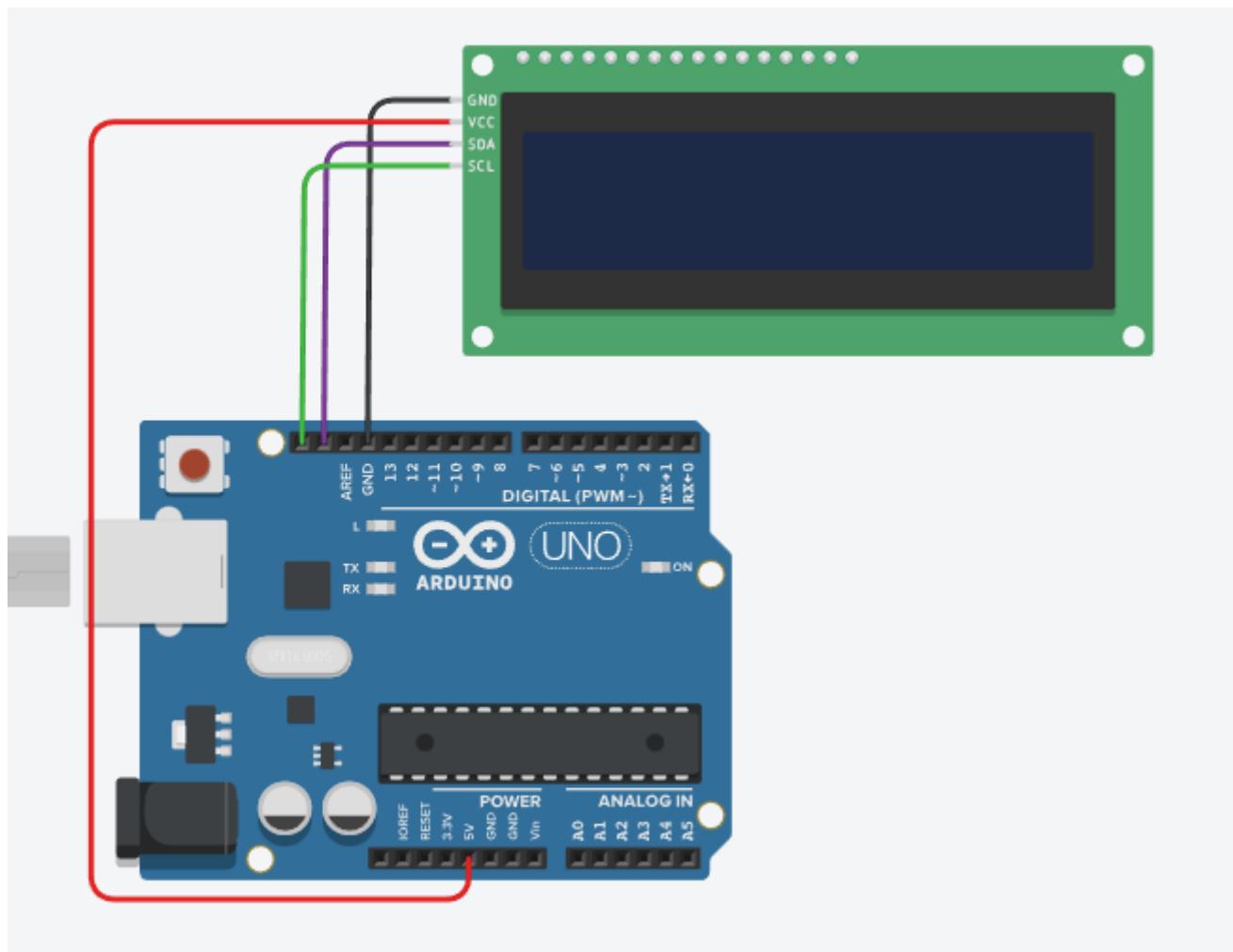
## 고급 출력 장치 제어

### I2C LCD 화면 출력

[!NOTE] 이 문서는 **I2C 모듈이 부착된 LCD**를 사용하여 원하는 텍스트를 출력하는 실습에 대해 설명합니다.

#### 1. 실습 목표

I2C LCD 라이브러리를 설치하고, 2개의 선만으로 LCD에 "Hello, World!"와 같은 문자열을 출력하는 프로그램을 작성합니다.



I2C LCD 회로 구성 예시

준비물

- 아두이노 우노
- I2C LCD 모듈 (16x2 또는 20x4)
- 점퍼 와이어 (4가닥)

## 2. I2C LCD란?

기존의 병렬 방식 LCD는 여러 개의 핀(최소 6개 이상)을 연결해야 해서 회로가 복잡했지만, **I2C 통신 모듈**을 사용하면 단 2개의 통신선(SDA, SCL)과 전원선만으로 LCD를 제어할 수 있어 매우 편리합니다.

- **I2C (Inter-Integrated Circuit)**: 단 2개의 선으로 여러 장치와 통신할 수 있는 직렬 통신 방식
- **SDA (Serial Data)**: 데이터 전송 라인
- **SCL (Serial Clock)**: 클럭 신호 라인

## 3. 라이브러리 설치

1. 아두이노 IDE에서 [스케치 > 라이브러리 포함하기 > 라이브러리 관리](#)로 이동합니다.
2. 검색창에 [LiquidCrystal\\_I2C](#)를 검색합니다.
3. [Frank de Brabander](#)가 만든 [LiquidCrystal\\_I2C](#) 라이브러리를 찾아 설치합니다.

## 4. 회로 구성

I2C LCD 모듈의 4개 핀을 아두이노 우노 보드의 정해진 I2C 핀에 연결합니다.

### I2C LCD 핀      아두이노 우노 핀

GND	<a href="#">GND</a>
VCC	<a href="#">5V</a>
SDA	<a href="#">A4 또는 SDA</a> 핀
SCL	<a href="#">A5 또는 SCL</a> 핀

## 5. 코드 작성

I2C LCD의 주소(일반적으로 0x27 또는 0x3F)와 크기(16x2)를 설정하고 텍스트를 출력합니다.

```
// 1. 라이브러리 포함
#include <LiquidCrystal_I2C.h>

// 2. LCD 객체 생성 (주소, 열, 행)
// 주소는 LCD 모듈 뒷면의 가변저항 등으로 확인 가능 (일반적으로 0x27)
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
    // 3. LCD 초기화 및 백라이트 켜기
    lcd.init();
    lcd.backlight();
}

void loop() {
    // 4. 커서 위치 설정 (0, 0) -> 첫 번째 줄, 첫 번째 칸
}
```

```

lcd.setCursor(0, 0);
// 5. 텍스트 출력
lcd.print("Hello, World!");

// 두 번째 줄에 카운터 출력
lcd.setCursor(0, 1);
lcd.print("Count: ");
lcd.print(millis() / 1000);

delay(1000);
}

```

## I2C 주소 확인하는 방법

LCD가 정상적으로 동작하지 않는 가장 흔한 원인 중 하나는 I2C 주소가 잘못되었기 때문입니다. LCD 모듈마다 사용되는 칩이 달라 주소가 0x27이 아닐 수 있습니다. 이럴 때는 **I2C 스캐너** 코드를 아두이노에 업로드하여 정확한 주소를 직접 확인할 수 있습니다.

### 1. I2C 스캐너 코드

아래 코드를 아두이노 IDE에 복사하여 붙여넣고, I2C LCD가 연결된 상태에서 업로드합니다.

```

#include <Wire.h>

void setup() {
    Wire.begin();
    Serial.begin(9600);
    while (!Serial); // 시리얼 포트가 열릴 때까지 대기
    Serial.println("\nI2C Scanner");
}

void loop() {
    byte error, address;
    int nDevices;

    Serial.println("Scanning...");

    nDevices = 0;
    for(address = 1; address < 127; address++ ) {
        Wire.beginTransmission(address);
        error = Wire.endTransmission();

        if (error == 0) {
            Serial.print("I2C device found at address 0x");
            if (address < 16) {
                Serial.print("0");
            }
            Serial.print(address, HEX);
            Serial.println(" !");
            nDevices++;
        }
    }
}

```

```
else if (error == 4) {
    Serial.print("Unknown error at address 0x");
    if (address < 16) {
        Serial.print("0");
    }
    Serial.println(address, HEX);
}
if (nDevices == 0) {
    Serial.println("No I2C devices found\n");
}
else {
    Serial.println("done\n");
}
delay(5000); // 5초마다 다시 스캔
}
```

## 2. 주소 확인 절차

1. 회로 연결: 위에서 설명한 대로 I2C LCD를 아두이노에 연결합니다.
2. 코드 업로드: I2C 스캐너 코드를 아두이노에 업로드합니다.
3. 시리얼 모니터 실행:
  - 아두이노 IDE의 오른쪽 상단에 있는 돋보기 아이콘(ocular icon)을 클릭하거나, 툴 > 시리얼 모니터를 선택합니다.
  - 시리얼 모니터 창의 오른쪽 하단에서 보드레이트(baud rate)를 9600으로 설정합니다.
4. 주소 확인: 시리얼 모니터에 "Scanning..."이라는 메시지와 함께 0x로 시작하는 16진수 주소가 나타납니다. 이 주소가 현재 연결된 I2C LCD의 주소입니다.
5. 코드 수정: 확인된 주소를 원래 코드의 LiquidCrystal\_I2C lcd(0x27, 16, 2); 부분에 있는 0x27 대신 넣고 다시 업로드하면 LCD가 정상적으로 작동합니다.



# Chapter 4

## 고급 출력 장치 제어

### 7세그먼트와 CD4511 디코더로 숫자 출력

[!NOTE] 이 문서는 **CD4511** BCD-to-7-Segment 디코더를 사용하여 7세그먼트 표시 장치에 숫자를 출력하는 방법에 대해 설명합니다. 이 방법을 사용하면 아두이노 핀 사용을 크게 줄일 수 있습니다.

#### 1. 실습 목표

CD4511 7세그먼트 디코더의 작동 원리를 이해하고, 4개의 아두이노 핀만을 사용하여 0부터 9까지의 숫자를 7세그먼트에 순차적으로 출력하는 프로그램을 작성합니다.

#### 준비물

<p>7세그먼트의 구성</p>	<p>7세그먼트의 단자 확인</p>
------------------	---------------------

- 아두이노 우노
- 브레드보드
- **CD4511 BCD to 7-Segment 디코더**
- **공통 음극(Common Cathode)** 7세그먼트 표시 장치

- 220Ω 저항 7개
- 점퍼 와이어

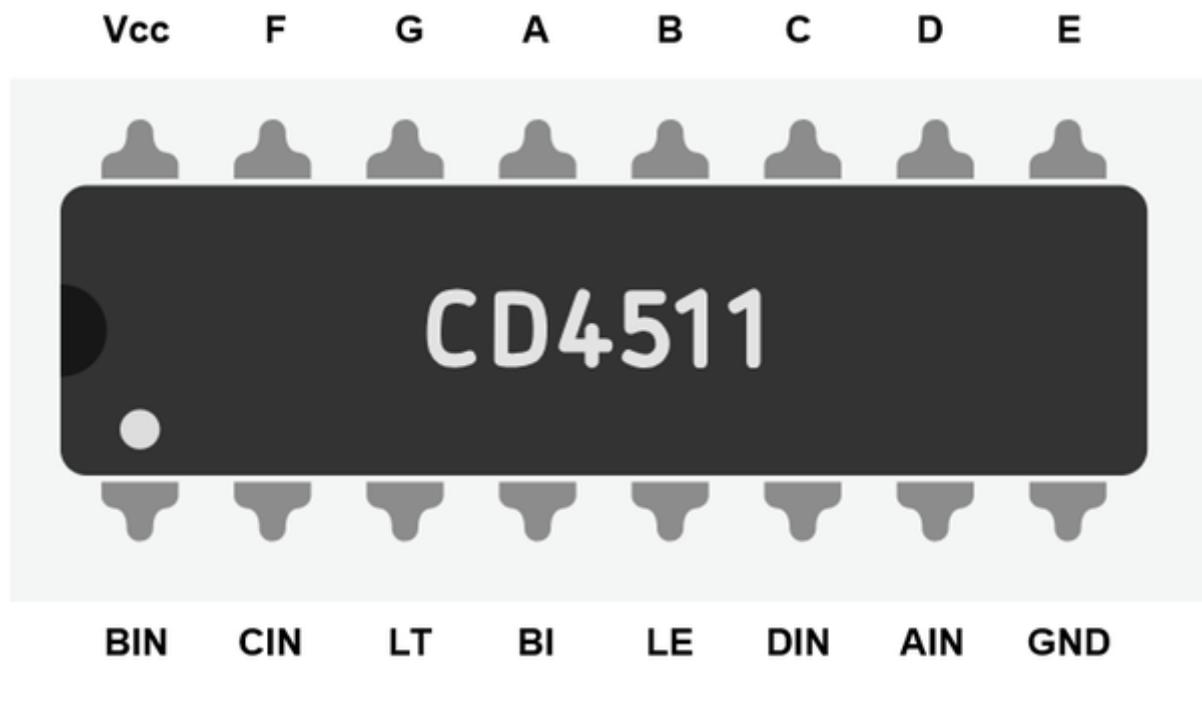
## 2. CD4511 7세그먼트 디코더란?

7세그먼트를 직접 제어하려면 최소 7~8개의 아두이노 핀이 필요하지만, **CD4511 디코더 IC**를 사용하면 4개의 핀만으로 숫자를 제어할 수 있습니다.

- **BCD (Binary Coded Decimal)**: 10진수(0~9)를 4비트의 2진수로 표현하는 방식입니다. 예를 들어, 10진수 5는 BCD로 0101이 됩니다.
- **CD4511 디코더**: 4비트의 BCD 입력을 받아, 해당 숫자를 7세그먼트에 표시하기 위한 7개의 출력 신호 (a~g)로 변환해주는 역할을 합니다. **공통 음극** 7세그먼트와 함께 사용하도록 설계되었습니다.

## 3. 회로 구성

아두이노의 디지털 핀 4개를 CD4511 디코더의 BCD 입력 핀(D0~D3)에 연결하고, 디코더의 출력 핀 (a~g)을 7세그먼트의 각 핀에 저항을 거쳐 연결합니다.



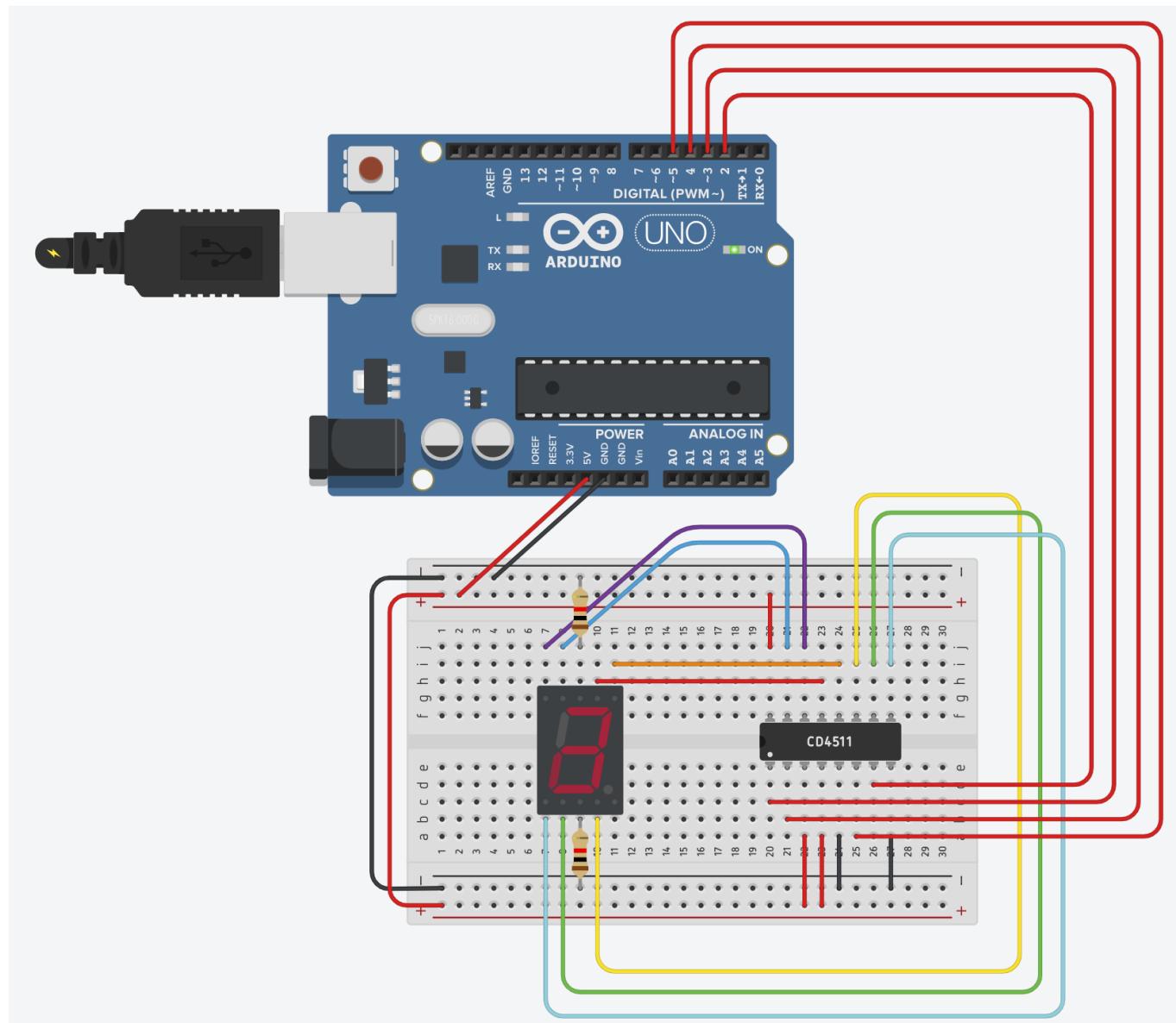
### 아두이노와 CD4511 디코더 연결

아두이노 핀	CD4511 핀	설명
D2	7 (D0/A)	BCD 입력 A (가장 낮은 비트, LSB)
D3	1 (D1/B)	BCD 입력 B
D4	2 (D2/C)	BCD 입력 C
D5	6 (D3/D)	BCD 입력 D (가장 높은 비트, MSB)
5V	16(VDD), 3(LT), 4(BL)	전원 및 기능 핀
GND	8(VSS), 5(LE)	접지 및 기능 핀

- **LT (Lamp Test):** **LOW**일 때 모든 세그먼트를 켭니다. 정상 작동을 위해 **5V(HIGH)**에 연결합니다.
- **BL (Blanking):** **LOW**일 때 모든 세그먼트를 끕니다. 정상 작동을 위해 **5V(HIGH)**에 연결합니다.
- **LE (Latch Enable):** **HIGH**일 때 현재 BCD 입력을 저장(래치)하여 출력을 고정시킵니다. 입력에 따라 출력이 계속 바뀌게 하려면 **GND(LOW)**에 연결합니다.

## CD4511 디코더와 7세그먼트 연결

- CD4511의 출력 핀 **a**부터 **g**까지(13, 12, 11, 10, 9, 15, 14번 핀)를 각각 220Ω 저항을 거쳐 7세그먼트의 **a**부터 **g** 핀에 연결합니다.
- 7세그먼트의 **공통 음극(COM)** 핀을 **GND**에 연결합니다.



## 4. 코드 작성

**bitRead()** 함수를 사용하여 숫자의 각 비트를 읽어와 4개의 BCD 핀으로 전송합니다. 디코더가 나머지 복잡한 작업을 처리해줍니다.

```
// CD4511 디코더의 BCD 입력 핀에 연결된 아두이노 핀
const int bcdA = 2; // LSB
const int bcdB = 3;
```

```
const int bcdC = 4;
const int bcdD = 5; // MSB

void setup() {
    // BCD 입력 핀들을 출력으로 설정
    pinMode(bcdA, OUTPUT);
    pinMode(bcdB, OUTPUT);
    pinMode(bcdC, OUTPUT);
    pinMode(bcdD, OUTPUT);
}

void loop() {
    // 0부터 9까지 숫자를 1초 간격으로 표시
    for (int number = 0; number <= 9; number++) {
        displayNumber(number);
        delay(1000);
    }
}

// 숫자를 BCD 형태로 디코더에 전송하는 함수
void displayNumber(int number) {
    digitalWrite(bcdA, bitRead(number, 0)); // 1번째 비트 (LSB)
    digitalWrite(bcdB, bitRead(number, 1)); // 2번째 비트
    digitalWrite(bcdC, bitRead(number, 2)); // 3번째 비트
    digitalWrite(bcdD, bitRead(number, 3)); // 4번째 비트 (MSB)
}
```

## 코드 설명

### 1. `displayNumber(int number)` 함수:

- `bitRead(number, 0)`: `number`의 0번째 비트(가장 오른쪽) 값을 읽습니다.
- `digitalWrite(bcdA, ...)`: 읽어온 비트 값(0 또는 1)을 `bcdA` 핀으로 출력하여 디코더에 BCD 신호를 보냅니다.
- 이 과정을 4개의 비트에 대해 반복하여 완전한 BCD 코드를 전송합니다.



# Chapter 5

## 아두이노에서 주의할 여러 점들

### 코드 업로드 오류

[!WARNING] 이 문서는 아두이노 코드 업로드 시 발생할 수 있는 일반적인 오류와 해결 방법에 대해 설명합니다.

#### 1. 오류 현상

- 아두이노 IDE 하단 콘솔 창에 주황색 글씨로 오류 메시지가 나타납니다.
- avrdude: stk500\_getsync() not in sync: resp=0x00
- avrdude: ser\_open(): can't open device "\.\COMx": The system cannot find the file specified.
- Problem uploading to board. 등의 메시지가 표시됩니다.

Problem uploading to board. See <http://www.arduino.cc/en/Guide/Troubleshooting#upload> for suggestions.

```
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 3 of 10: not in sync: resp=0x00
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 4 of 10: not in sync: resp=0x00
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 5 of 10: not in sync: resp=0x00
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 6 of 10: not in sync: resp=0x00
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 7 of 10: not in sync: resp=0x00
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 8 of 10: not in sync: resp=0x00
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 9 of 10: not in sync: resp=0x00
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 10 of 10: not in sync: resp=0x00
Problem uploading to board. See http://www.arduino.cc/en/Guide/Troubleshooting#upload for suggestions.
```

대표적인 업로드 오류 메시지

#### 2. 주요 원인 및 해결 방법

원인 1: 보드 또는 포트 설정 문제

IDE에 선택된 보드나 포트가 실제 연결된 아두이노와 다를 경우 발생합니다.

해결 방법

- 올바른 보드 선택:** 툴 > 보드 메뉴에서 사용 중인 아두이노 보드(예: Arduino Uno)가 정확히 선택되었는지 확인합니다.
- 올바른 포트 선택:** 툴 > 포트 메뉴에서 아두이노가 연결된 COM 포트가 올바르게 선택되었는지 확인합니다. 포트를 잘 모를 경우, 아두이노를 분리했다가 다시 연결했을 때 나타나는 포트를 선택합니다.

## 원인 2: 케이블 불량 또는 연결 불량

사용하는 USB 케이블이 데이터 전송 기능이 없는 충전 전용 케이블이거나, 케이블 내부가 손상된 경우, 또는 USB 포트와의 연결이 헐거운 경우 발생합니다.

### 해결 방법

- 데이터 통신용 케이블 사용:** 다른 장치와 데이터 통신이 정상적으로 되었던 USB 케이블로 교체해 봅니다.
- USB 포트 변경:** PC의 다른 USB 포트에 연결해 봅니다. (특히 USB 허브를 사용 중이라면 PC 본체의 포트에 직접 연결)
- 연결 상태 확인:** 케이블이 아두이노와 PC에 단단히 연결되었는지 확인합니다.

## 원인 3: 드라이버 설치 문제

CH340/CH341 칩을 사용하는 일부 저렴한 아두이노 호환 보드는 PC와 통신하기 위해 별도의 드라이버 설치가 필요합니다.

### 해결 방법

- 장치 관리자 확인:** Windows 키 + X > 장치 관리자 를 열고 포트 (COM & LPT) 항목을 확인합니다. USB-SERIAL CH340 또는 알 수 없는 장치로 표시된다면 드라이버가 필요합니다.
- 드라이버 검색 및 설치:** 구글에서 CH340 driver 를 검색하여 자신의 운영체제에 맞는 드라이버를 다운로드하고 설치합니다.

## 원인 4: 다른 프로그램이 포트 사용 중

시리얼 모니터, 다른 통신 프로그램, 3D 프린터 제어 프로그램 등이 아두이노가 연결된 COM 포트를 이미 사용하고 있을 경우 발생합니다.

### 해결 방법

- 시리얼 모니터 닫기:** 아두이노 IDE의 시리얼 모니터가 열려 있다면 닫고 다시 업로드를 시도합니다.
- 관련 프로그램 종료:** 해당 COM 포트를 사용할 가능성이 있는 다른 모든 프로그램을 종료하고 다시 시도 합니다.



# Chapter 5

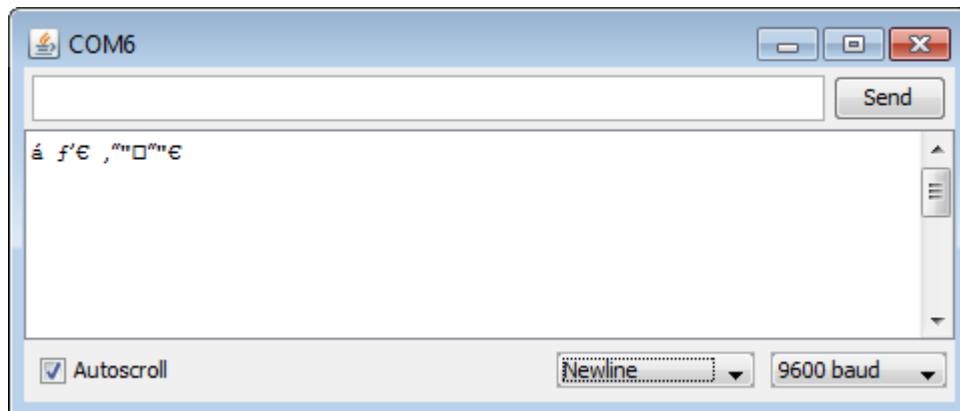
## 아두이노에서 주의할 여러 점들

### 시리얼 모니터 값이 이상하게 나오는 경우

[!WARNING] 이 문서는 아두이노 시리얼 모니터에 출력되는 값이 깨지거나 알아볼 수 없는 문자로 나타나는 문제의 해결 방법에 대해 설명합니다.

#### 1. 오류 현상

- 시리얼 모니터에 `Hello, WorlD!`와 같은 정상적인 텍스트 대신 `⸮⸮⸮⸮⸮` 와 같이 의미를 알 수 없는 문자들이 출력됩니다.
- 숫자 값이 예상과 전혀 다른 큰 값이나 음수 값으로 출력됩니다.



시리얼 모니터 글자 깨짐 현상

#### 2. 주요 원인 및 해결 방법

원인 1: 통신 속도(Baud Rate) 불일치

아두이노 코드에 설정된 통신 속도와 시리얼 모니터 창에 설정된 통신 속도가 일치하지 않는 것이 가장 흔한 원인입니다.

#### 해결 방법

- 코드의 통신 속도 확인:** `setup()` 함수 안에 있는 `Serial.begin()` 함수의 인자 값을 확인합니다. 일반적으로 `Serial.begin(9600);` 과 같이 설정되어 있습니다.

```
void setup() {
    Serial.begin(9600); // 통신 속도를 9600 bps로 설정
```

```
}
```

2. **시리얼 모니터 통신 속도 설정:** 아두이노 IDE의 시리얼 모니터 창 우측 하단에 있는 드롭다운 메뉴를 클릭하여 코드에 설정된 값과 \*\*동일한 값 (예: 9600 baud)\*\*으로 맞춰줍니다.
3. **추가 확장 모듈의 통신 속도 설정:** 일부 모듈(블루투스 등)은 별도로 통신 속도를 지정해줘야합니다. 이때 모듈의 통신 속도를 `Serial.begin()`에서 입력한 숫자와 동일하게 맞추어 설정해줍니다. (모듈의 통신 속도를 어떻게 변경하는지는 구글에 검색 필요)

## 원인 2: 불안정한 회로 연결

센서나 다른 부품의 연결이 헐겁거나 잘못 연결된 경우, 전기적으로 불안정한 신호가 아두이노에 입력되어 시리얼 통신에 영향을 줄 수 있습니다.

### 해결 방법

1. **전원 및 접지(GND) 확인:** 모든 부품의 VCC와 GND가 올바르게 연결되었는지 다시 확인합니다.
2. **점퍼 와이어 연결 확인:** 브레드보드나 핀에 꽂힌 점퍼 와이어가 헐겁지 않은지 확인하고, 의심되는 경우 다른 선으로 교체해 봅니다.
3. **회로 단순화:** 문제가 발생할 경우, 연결된 부품을 최소화하여 어느 부분에서 문제가 발생하는지 범위를 좁혀 나갑니다.

## 원인 3: `delay()` 함수의 부재

`loop()` 함수 안에 `delay()` 없이 너무 빠른 속도로 `Serial.print()`를 호출하면, 아두이노와 PC 간의 데이터 처리 속도 차이로 인해 데이터가 누락되거나 깨져 보일 수 있습니다.

### 해결 방법

- `loop()` 함수 마지막에 적절한 `delay()`를 추가하여 약간의 지연 시간을 줍니다.

```
void loop() {
    int sensorValue = analogRead(A0);
    Serial.println(sensorValue);
    delay(100); // 0.1초 정도의 딜레이를 주어 안정적인 출력 유도
}
```



# Chapter 5

### 아두이노에서 주의할 여러 점들

센서 값이 변하지 않는 경우

[!WARNING] 이 문서는 센서를 연결했지만 시리얼 모니터의 값이 고정되어 있거나, 0 또는 1023과 같은 극단적인 값만 나오는 문제의 해결 방법에 대해 설명합니다.

## 1. 오류 현상

- 조도 센서를 손으로 가리거나 빛을 비춰도 `analogRead()` 값이 거의 변하지 않습니다.
  - 초음파 센서 앞에 물체를 가까이하거나 멀리해도 측정되는 거리가 항상 동일합니다.
  - 센서 값이 0, 1023(아날로그) 또는 0, 1(디지털) 등 특정 값에 고정되어 있습니다.

센서 값이 1017으로 고정되어 변하지 않는 모습

## 2. 주요 원인 및 해결 방법

### 원인 1: 핀 번호 설정 오류

코드에 선언된 핀 번호와 실제 센서가 연결된 아두이노의 핀 번호가 일치하지 않는 경우입니다.

## 해결 방법

- **코드와 실제 회로 대조:** 코드 상단의 `int sensorPin = A0;` 와 같은 핀 번호 선언부와, 점퍼 와이어가 실제로 꽂힌 아두이노 보드의 핀 번호가 일치하는지 꼼꼼히 확인합니다.

원인 2· 전월/그라운드 연결 누락 또는 불량

센서에 VCC(5V) 또는 GND(접지)가 제대로 연결되지 않으면 센서가 정상적으로 동작하지 않아 불안정한 값을 출력합니다.

## 해결 방법

- VCC/GND 연결 확인:** 센서 모듈의 VCC 핀이 아두이노 5V에, GND 핀이 아두이노 GND에 잘 연결되었는지 확인합니다.
- 브레드보드 전원 라인 확인:** 브레드보드의 전원 버스(+/- 라인)를 사용했다면, 아두이노의 5V/GND가 브레드보드의 전원 버스에 잘 연결되었는지 확인합니다. 브레드보드에 따라 중앙에서 전원 라인이 끊어진 경우도 있으니 주의합니다.

## 원인 3: 회로 구성 오류 (특히 아날로그 센서)

조도 센서, 가변 저항과 같은 아날로그 센서는 전압 분배 법칙을 이용하기 위해 풀업 또는 풀다운 저항과 함께 사용해야 합니다. 이 저항이 없거나 잘못 연결되면 값이 한쪽으로 치우쳐 나옵니다.

## 해결 방법

- 회로도 재확인:** 해당 센서의 데이터시트나 정상적으로 동작하는 회로 예시를 다시 보고, 자신의 회로에 풀업/풀다운 저항이 올바르게 연결되었는지 확인합니다. (예: 조도 센서의 경우, 신호선과 GND 사이에 10kΩ 저항 연결)

## 원인 4: 센서 자체의 손상

드물지만, 정전기나 잘못된 전압 인가 등으로 인해 센서 자체가 물리적으로 손상되었을 수 있습니다.

## 해결 방법

- 가장 간단한 예제 실행:** 해당 센서를 사용하는 가장 기본적인 예제 코드를 실행하여 센서의 동작 여부를 최소한의 환경에서 테스트합니다.
- 동일한 다른 센서로 교체:** 가능하다면, 동일한 종류의 다른 센서로 교체하여 테스트해 봅니다. 교체 후 정상 동작한다면 기존 센서의 고장일 가능성성이 높습니다.



# Chapter 5

## 아두이노에서 주의할 여러 점들

### 아두이노/LED/부저 등 장치가 동작하지 않는 경우

[!WARNING] 이 문서는 코드를 업로드했지만 LED, 부저, 모터 등 출력 장치가 전혀 반응하지 않는 문제의 해결 방법에 대해 설명합니다.

#### 1. 오류 현상

- `digitalWrite(ledPin, HIGH);` 코드를 실행해도 LED에 불이 들어오지 않습니다.
- `tone(buzzerPin, 262);` 코드를 실행해도 부저에서 소리가 나지 않습니다.
- 서보 모터나 DC 모터가 전혀 움직이지 않습니다.

#### 2. 주요 원인 및 해결 방법

##### 원인 1: 극성(+/-)을 반대로 연결한 경우

LED, 다이오드, 커패시터, 부저, IC 등 극성이 있는 부품을 반대로 연결하면 동작하지 않거나 부품이 손상될 수 있습니다.

##### 해결 방법

- **부품의 극성 확인:**
  - **LED:** 긴 다리가 양극(+), 짧은 다리가 음극(-)입니다.
  - **부저:** + 표시가 있는 쪽이 양극입니다.
  - **커패시터:** 흰색 띠가 있는 쪽이 음극(-)입니다.
- **회로 재확인:** 양극은 아두이노의 출력 핀(또는 5V)에, 음극은 GND에 올바르게 연결되었는지 확인합니다.

##### 원인 2: 전류 부족 또는 전압 부족

여러 개의 LED나 모터와 같이 상대적으로 많은 전류를 소모하는 부품을 동시에 동작시키려 할 때, 아두이노 보드의 5V 핀에서 공급하는 전류만으로는 부족할 수 있습니다.

##### 해결 방법

1. **외부 전원 사용:** 모터, 다수의 LED 스트립 등 많은 전류를 필요로 하는 부품은 아두이노 전원이 아닌, 별도의 어댑터나 배터리 팩과 같은 외부 전원을 사용해야 합니다. 이때, **외부 전원의 GND와 아두이노의 GND는 반드시 공통으로 연결해야 합니다.**
2. **동작 전압 확인:** 사용하는 부품이 5V가 아닌 3.3V 또는 그 이상의 전압에서 동작하는지 확인하고, 올바른 전압을 공급해야 합니다.

## 원인 3: 코드 로직 오류

회로에는 문제가 없지만, 코드의 논리적인 흐름이 잘못되어 출력 명령이 실행되지 않는 경우입니다.

### 해결 방법

1. **pinMode() 설정 확인**: `setup()` 함수 내에서 해당 핀을 `pinMode(pin, OUTPUT);`으로 올바르게 설정했는지 확인합니다.
2. **조건문 확인**: `if`문과 같은 조건문 안에 출력 코드가 있다면, 해당 조건이 실제로 참(true)이 되는지 `Serial.print()` 등을 이용해 중간 과정을 출력하며 확인합니다.
3. **함수 호출 확인**: 직접 만든 함수 안에 출력 코드가 있다면, `loop()` 함수 등에서 해당 함수가 정상적으로 호출되는지 확인합니다.

## 원인 4: 저항 미사용 또는 잘못된 저항 사용

LED에 전류 제한 저항 없이 직접 5V를 연결하면 LED가 즉시 타버릴 수 있습니다. 반대로 너무 큰 저항을 연결하면 전류가 약해져 불이 켜지지 않거나 매우 어둡게 켜집니다.

### 해결 방법

- **적절한 저항 사용**: LED에는 보통  $220\Omega \sim 1k\Omega$  사이의 저항을 직렬로 연결하는 것이 일반적입니다. 부품에 맞는 적절한 저항을 사용했는지 확인합니다.



## Chapter 5

### 아두이노에서 주의할 여러 점들

## 서보모터 떨림 또는 움직이지 않음

[!WARNING] 이 문서는 서보 모터가 특정 각도로 이동한 후 부르르 떨거나, 전혀 움직이지 않거나, 힘없이 흔들리는 문제의 해결 방법에 대해 설명합니다.

### 1. 오류 현상

- `myServo.write(90);` 명령 후, 모터가 90도 위치에서 계속 "드르륵" 또는 "윙" 하는 소리를 내며 미세하게 떨립니다.
- 서보 모터가 지정된 각도로 이동하지 않고 힘없이 쭉 늘어집니다.
- 여러 개의 서보 모터를 연결하자 모든 모터가 제대로 동작하지 않습니다.

### 2. 주요 원인 및 해결 방법

#### 원인 1: 전원 공급 부족 (가장 흔한 원인)

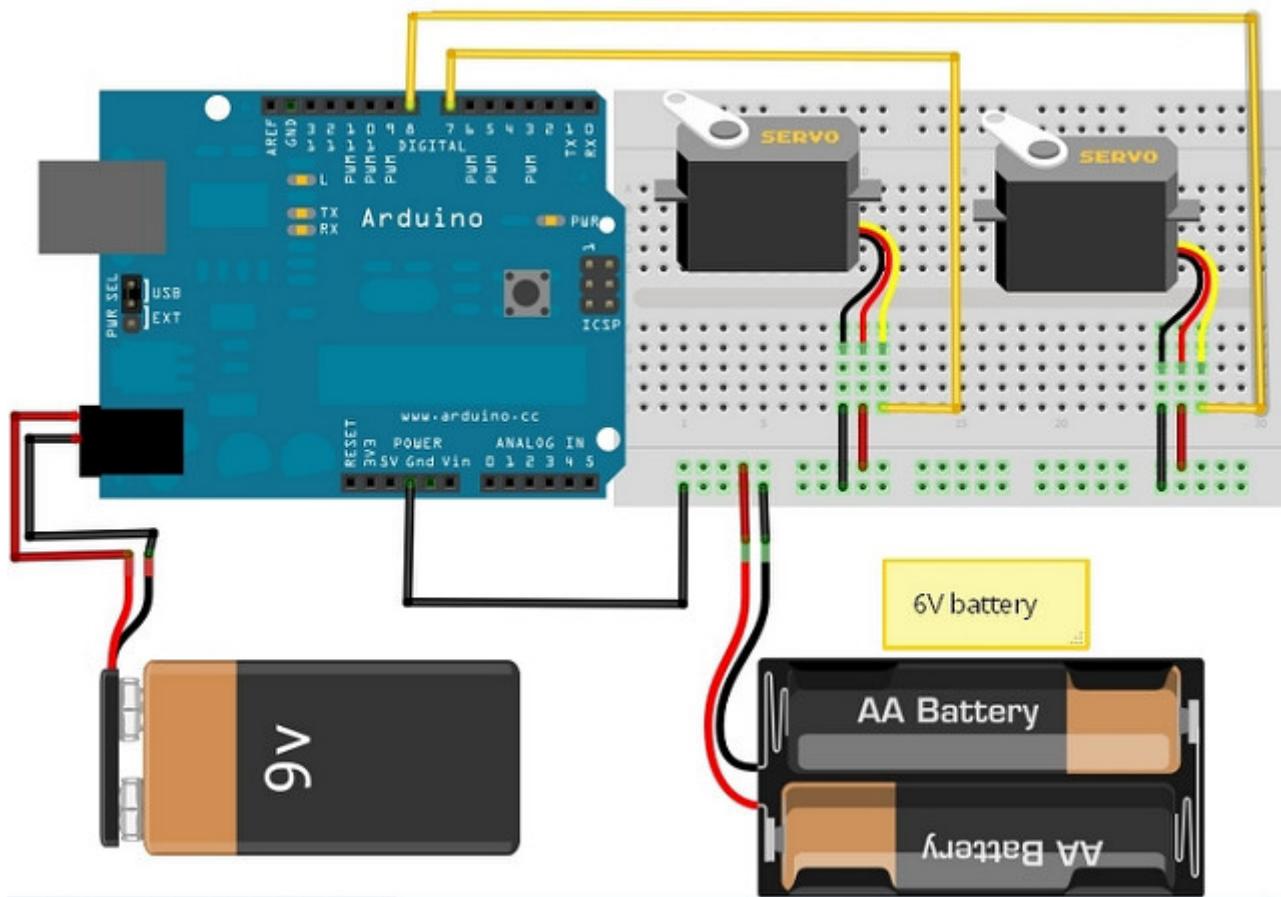
서보 모터는 위치를 유지하기 위해 계속해서 전류를 소모합니다. 아두이노 보드의 5V 레귤레이터는 공급할 수 있는 전류(약 500mA)에 한계가 있어, 서보 모터 1~2개를 겨우 감당하거나 그마저도 부족한 경우가 많습니다.

#### 해결 방법

1. **외부 전원 사용:** 서보 모터를 위한 \*\*별도의 5V 전원(예: 5V 2A 어댑터, AA 건전지 4개 팩 등)\*\*을 사용하는 것이 가장 확실한 해결책입니다.

2. **외부 전원 연결 시 주의사항:**

- 외부 전원의 + 극을 서보 모터의 VCC(빨간색 선)에 연결합니다.
- 외부 전원의 - 극(GND)을 서보 모터의 GND(갈색 선) 그리고 아두이노의 GND 핀에 반드시 함께 연결해야 합니다. (→ **공통 접지**)
- 아두이노의 신호선(주황색 선)만 서보 모터의 신호 핀에 연결합니다.



### 서보 모터에 외부 전원을 연결한 회로 예시

#### 원인 2: 잘못된 핀 연결

서보 모터 제어를 위한 PWM 신호는 모든 디지털 핀에서 생성할 수 있지만, `Servo.h` 라이브러리는 특정 타이머와 연결된 핀에서 가장 안정적으로 동작합니다. (Uno 기준 대부분의 핀에서 잘 동작함)

#### 해결 방법

- PWM 지원 핀 사용:** 가급적 디지털 핀 중 ~ 표시가 있는 PWM 핀(3, 5, 6, 9, 10, 11)에 연결하는 것을 권장 합니다.
- 라이브러리 attach 확인:** `myServo.attach(pinNumber);` 코드의 `pinNumber`가 실제 신호선이 연결된 핀 번호와 일치하는지 다시 확인합니다.

#### 원인 3: `delay()` 함수와 Servo 라이브러리의 충돌

`Servo.h` 라이브러리는 내부적으로 타이머를 사용하여 PWM 신호를 만듭니다. 매우 긴 `delay()` 함수 (수 초 이상)나 타이머를 직접 제어하는 다른 라이브러리와 함께 사용할 경우, `Servo` 라이브러리의 동작이 방해받아 떨림이 발생할 수 있습니다.

#### 해결 방법

- delay() 최소화:** 코드 내의 불필요하게 긴 `delay()`를 줄이거나, `millis()` 함수를 사용한 비동기 방식으로 코드를 변경하는 것을 고려합니다.
- 라이브러리 충돌 확인:** 서보 모터 외에 다른 라이브러리(특히 소리를 다루거나, 적외선 통신을 하는 라이브러리)를 사용 중이라면, 해당 라이브러리와 `Servo.h`의 타이머 충돌 여부를 검색해 봅니다.



# Chapter 5

## 아두이노에서 주의할 여러 점들

### I2C 장치(LCD 등) 화면이 안 나오는 경우

[!WARNING] 이 문서는 I2C 통신을 사용하는 LCD 등의 장치가 동작하지 않거나 화면에 아무것도 표시되지 않는 문제의 해결 방법에 대해 설명합니다.

#### 1. 오류 현상

- I2C LCD에 `lcd.print()` 명령을 실행해도 아무 글자도 나타나지 않습니다.
- 백라이트는 켜져 있지만, 글씨가 보이지 않습니다.
- 백라이트조차 켜지지 않습니다.
- I2C 통신을 사용하는 다른 센서(자이로 센서, 기압 센서 등)의 값이 읽어지지 않습니다.

#### 2. 주요 원인 및 해결 방법

##### 원인 1: SDA/SCL 핀 연결 오류

I2C 통신은 정해진 특정 핀(SDA, SCL)으로만 가능합니다. 이 핀을 잘못 연결하면 통신 자체가 이루어지지 않습니다.

##### 해결 방법

- 보드별 I2C 핀 확인:** 사용 중인 아두이노 보드의 I2C 핀 위치를 정확히 확인하고 연결합니다.
  - Arduino Uno:** A4 (SDA), A5 (SCL)
  - Arduino Mega:** 20 (SDA), 21 (SCL)
  - Arduino Nano:** A4 (SDA), A5 (SCL)
- SDA, SCL 교차 연결 확인:** 장치의 SDA는 아두이노의 SDA에, SCL은 아두이노의 SCL에 연결되었는지 확인합니다. (간혹 반대로 연결하는 실수 발생)

##### 원인 2: 잘못된 I2C 주소 사용

모든 I2C 장치는 고유한 주소(Address)를 가집니다. 코드에 설정된 주소와 실제 장치의 주소가 다르면 장치를 찾을 수 없어 동작하지 않습니다.

##### 해결 방법

- I2C 스캐너 실행:** 인터넷에서 **I2C Scanner** 예제 코드를 찾아 아두이노에 업로드합니다.
- 시리얼 모니터 확인:** 시리얼 모니터를 열어 스캔된 장치의 주소를 확인합니다. (`I2C device found at address 0x27`과 같은 메시지 출력)

3. 코드에 주소 반영: 확인된 주소를 `LiquidCrystal_I2C lcd(0x27, 16, 2);` 와 같이 코드에 정확하게 입력합니다. (I2C LCD 모듈은 보통 `0x27` 또는 `0x3F` 주소를 가집니다.)

The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, Help, and a status bar showing Bluetooth, WiFi, battery level (42%), and the date (01.34). The title bar says "ESP32\_I2c\_scanner | Arduino 1.8.13". The code area contains the `I2C Scanner` sketch. The serial monitor window titled "/dev/ttyUSB0" displays the output of the I2C scanning process. The output shows multiple I2C devices found at address `0x3C`. The bottom status bar shows "Done uploading.", "Leaving...", and "Hard resetting via RTS pin...".

```

void loop()
{
    byte error, address;
    int nDevices;

    Serial.println("Scanning...");

    nDevices = 0;
    for(address = 1; address < 127; address++)
    {
        // The i2c_scanner uses the return value of
        // the Wire.endTransmission() to see if
        // a device did acknowledge to the address.
        Wire.beginTransmission(address);
        error = Wire.endTransmission();

        if (error == 0)
        {
            Serial.print("I2C device found at address 0x");
            if (address<16)
                Serial.print("0");
            Serial.print(address,HEX);
            Serial.println(" !");

            nDevices++;
        }
        else if (error==4)
        {
            Serial.print("Unknown error at address 0x");
            if (address<16)
                Serial.print("0");
            Serial.println(address,HEX);
        }
    }
    if (nDevices == 0)
        Serial.println("No I2C devices found\n");
    else
        Serial.println("done\n");

    delay(5000); // wait 5 seconds for next scan
}

```

### I2C 스캐너를 통해 장치 주소를 확인하는 모습

#### 원인 3: 라이브러리 미설치 또는 충돌

해당 I2C 장치를 제어하기 위한 라이브러리가 없거나, 다른 라이브러리와 충돌하는 경우 문제가 발생할 수 있습니다.

#### 해결 방법

- 필수 라이브러리 설치:** 장치 제조사나 데이터시트에서 권장하는 라이브러리를 아두이노 IDE의 라이브러리 매니저를 통해 정확히 설치했는지 확인합니다. (예: I2C LCD의 경우 `LiquidCrystal_I2C`)
- 라이브러리 예제 실행:** 설치한 라이브러리에 포함된 기본 예제 코드를 실행하여 하드웨어와 라이브러리의 기본적인 동작 여부를 먼저 확인합니다.

#### 원인 4: LCD 명암 조절 문제 (LCD 한정)

I2C LCD 모듈 뒷면에는 파란색 사각형 모양의 **가변 저항**이 있습니다. 이 저항은 화면의 명암(Contrast)을 조절하는 역할을 합니다.

#### 해결 방법

- 드라이버로 저항 조절:** 작은 십자 드라이버를 사용하여 이 가변 저항을 천천히 좌우로 돌려보면서 글씨가 선명하게 보이는 지점을 찾습니다. 저항이 한쪽으로 끝까지 돌아가 있으면 글씨가 너무 희미하거나 너무 진해서 보이지 않을 수 있습니다.



# Chapter 5

## 아두이노에서 주의할 여러 점들

### 기타 예상치 못한 오류 상황

[!TIP] 이 문서는 위에서 언급되지 않은 다양한 문제 상황에 대처하고, 스스로 문제를 해결하는 디버깅 방법에 대해 설명합니다.

#### 1. 원인을 알 수 없는 오작동

코드는 정상적인 것 같고 회로 연결도 문제가 없어 보이는데, 아두이노가 멈추거나 예상과 전혀 다르게 동작하는 경우 시도해볼 수 있는 방법들입니다.

##### 해결 방법 1: 리셋 버튼 활용

- 소프트 리셋:** 아두이노 보드에 있는 빨간색 또는 검은색의 작은 리셋 버튼을 눌러보세요. 프로그램이 처음부터 다시 시작되면서 일시적인 소프트웨어 문제를 해결할 수 있습니다.
- 전원 리셋:** USB 케이블을 뽑았다가 다시 꽂아 아두이노의 전원을 완전히 차단하고 재공급합니다. 하드웨어적인 상태를 초기화하는 데 도움이 됩니다.

##### 해결 방법 2: 코드 최소화하여 원인 추적

복잡한 코드 전체를 한 번에 디버깅하는 것은 어렵습니다. 문제가 발생한 부분을 찾기 위해 코드를 점진적으로 줄여나가거나, 반대로 최소한의 코드에서 시작하여 기능을 하나씩 추가하며 테스트합니다.

- 새로운 스케치 작성:** 문제가 되는 부분과 관련된 최소한의 코드만 남기고 새로운 스케치 파일에 복사합니다. (예: 특정 센서 값만 읽어와서 출력하는 코드)
- 기능 단위 테스트:** 이 최소화된 코드가 정상적으로 동작하는지 확인합니다. 여기서도 문제가 발생한다면 문제는 해당 코드나 하드웨어에 있는 것입니다.
- 점진적 추가/주석 처리:** 최소화된 코드가 정상이라면, 원래 코드에서 다른 기능들을 하나씩 주석 처리(// 또는 /\* ... \*/) 해제하거나 추가하면서 어느 부분에서 문제가 발생하는지 범위를 좁혀 나갑니다.

#### 2. 문제 해결을 위한 정보 검색

스스로 해결하기 어려운 문제는 이미 다른 사람들도 겪었을 가능성이 높습니다. 효과적으로 정보를 검색하는 방법을 알아두는 것이 중요합니다.

##### 방법 1: 오류 메시지로 검색하기

- 정확한 오류 메시지 복사:** IDE 콘솔에 출력된 주황색 오류 메시지(avrdude: ..., error: ... 등)를 그대로 복사하여 구글에 검색하는 것이 가장 빠르고 정확한 방법입니다.

- **키워드 추가:** `arduino`라는 키워드와 함께 센서 이름, 라이브러리 이름 등을 추가하여 검색하면 더 관련성이 높은 결과를 얻을 수 있습니다. (예: `arduino dht11 Failed to read from DHT sensor!` )

## 방법 2: 공식 문서 및 커뮤니티 활용

- **아두이노 공식 홈페이지 (Arduino Docs):** 함수나 라이브러리의 정확한 사용법을 찾아볼 수 있습니다.
- **아두이노 포럼 (Arduino Forum):** 전 세계 사용자들이 질문하고 답변하는 곳으로, 유사한 문제에 대한 해결책을 찾거나 직접 질문을 올릴 수 있습니다.
- **Stack Overflow:** 개발자들을 위한 질의응답 사이트로, 아두이노 관련 질문도 많이 올라옵니다.
- **제조사 데이터시트:** 사용 중인 센서나 부품의 제조사에서 제공하는 데이터시트(Datasheet)에는 가장 정확한 전기적 특성과 사용법이 명시되어 있습니다.