

· 창의융합형 인재 교육 · S/W 코딩 교육 · S/W 협업 프로젝트

# Smart Mini City 교재

- 기초 예제 소스
- 건물 조립 매뉴얼
  - 응용 예제
- 협업 시나리오 예제



이씨FrontU

## ■ 아두이노

### ◇ 아두이노

1. 아두이노란?
2. 아두이노 우노(Uno)
3. 브레드보드(빵판) & 점퍼선
4. 아두이노 프로그래밍 툴

### ◇ 기초학습

1. 아두이노를 이용한 LED 제어
2. 아두이노를 이용한 Motor 제어
3. 아두이노를 이용한 Sensor 제어

## ■ 스마트 미니시티

### ◇ 스마트 미니시티

1. 스마트 미니시티이란?
2. 재료 배치도

### ◇ 조립

1. 아두이노 우노(Uno)보드 & 브레드보드(빵판) 연결
2. LED 조립 & 테스트
3. 카페 조립 & 테스트
4. 가로등 조립 & 테스트
5. 크레인 조립 & 테스트
6. 건물 조립 & 테스트

### ◇ 센서 응용

1. 조도 센서
2. 사운드 센서
3. 적외선 센서

### ◇ 미니시티 - 독립

1. 기본 예제 소스
2. 심화 예제 소스

### ◇ 미니시티 - 협업

1. 협업이란?
2. 협업방법
3. 협업 시나리오 예제

## ■ 아두이노

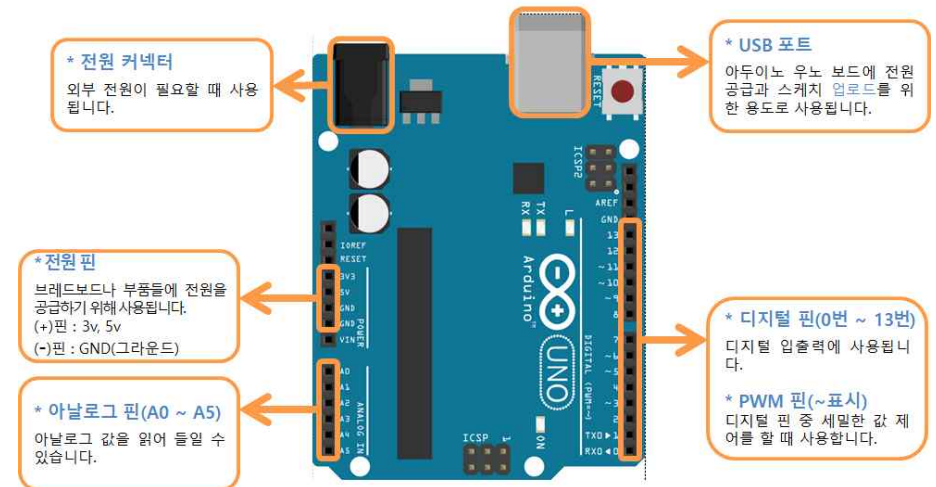
### ◇ 아두이노

#### 1. 아두이노란?

아두이노(Arduino)는 오픈 소스를 기반으로 한 단일 보드 마이크로컨트롤러(MCU)로, 완성 된 보드(상품)와 관련 개발 도구 및 환경을 말합니다. 아두이노는 하드웨어에 익숙지 않은 학생들이 자신들의 디자인 작품을 손쉽게 제어할 수 있도록 고안된 것입니다.

아두이노(Arduino)는 다수의 스위치나 센서로부터 값을 받아들이며, LED나 모터와 같은 외부 전자 장치들을 통제함으로써 환경과 상호작용이 가능한 것을 만들어 낼 수 있습니다. 아두이노는 소프트웨어를 처음 배우는 사용자들이 가장 쉽게 접근할 수 있는 플랫폼이며, 다양한 센서 및 액추에이터를 지원해 드론, 로봇제어, 사물인터넷 등의 제품 개발 및 제작에 최적화된 플랫폼입니다.

#### 2. 아두이노 우노(Uno)



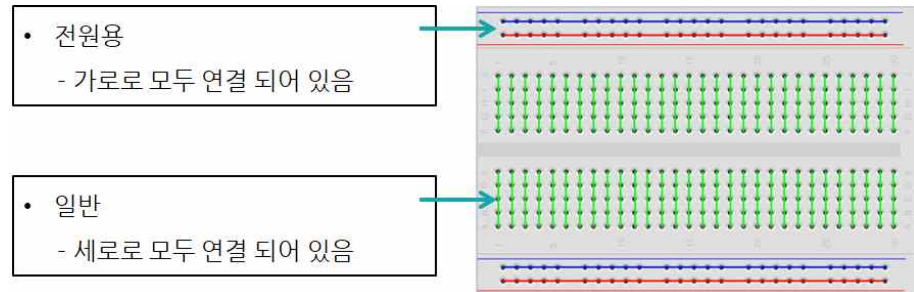
<아두이노 우노보드 구조도>

- ▶ 아두이노 우노(Arduino Uno)는 가장 많이 사용되는 보드로서 거의 표준과 같이 사용되고 있습니다. 우노는 총 44개의 핀과 단자들로 구성되어 있습니다. 각 핀과 단자들은 아두이노와 다른 보드 또는 센서들의 제어에 이용될 수 있으며 다양한 응용을 할 수 있습니다. 또한 안드로이드 표준 보드로, 핀 배열 등이 Uno를 기준으로 맞춰져 있으므로 확장 실드 등을 사용하는데 최적입니다.

- 전원 커넥터 : 아두이노에 외부전원을 공급할 때 사용합니다. (9v 권장)
- 전원 핀 : 아두이노로부터 브레드보드(빵판)나 센서, 모터 등 사용할 모듈에 전원을 공급하기 위해 사용합니다. 3v, 5v를 모듈들의 (+) 혹은 VCC라 표시된 핀에 연결하고 GND는 (-)혹은 GND라 표시된 핀과 연결하면 됩니다.
- USB 포트 : 아두이노와 PC를 연결할 수 있습니다. PC와 연결하여 소스 프로그램을 업로드할 수 있으며, 전원을 공급할 수 있습니다.
- 아날로그 핀 : A0핀 ~ A5핀 까지 총 6개의 핀으로 아날로그 신호를 사용하는 센서 혹은 모듈을 연결하여 사용합니다.
- 디지털 핀 : 0번핀 ~ 13번핀 까지 총 14개의 핀으로 디지털 신호를 사용하는 센서 혹은 모듈을 사용할 때 연결하여 사용합니다.
- PWM 핀 : 디지털 핀중 ~ 표시가 되어 있는 핀으로 3번, 5번, 6번, 9번, 10번, 11번 총 6개의 핀으로 모터와 같이 세밀한 값 제어를 할 때 사용합니다.

### 3. 브레드보드(빵판) & 점퍼선

#### ○ 브레드보드(빵판)



=> 브레드보드는 빵판으로도 불립니다. 납땜을 하지 않고도 전자 부품을 쉽게 꽂아 전자회로를 구성할 수 있는 보드입니다. 재사용이나 수정이 가능합니다. 위 그림과 같이 녹색으로 표시된 곳을 IC 영역이라고 하며 세로로 연결 되어 있으며 회로를 구성할 수 있습니다. 그리고 빨간색 선(+), 파란색선(-)은 버스영역이라 하며 가로로 연결 되어 있고 부품들의 전원을 연결할 때 사용합니다.

#### ○ 점퍼선

=> 점퍼선은 떨어져 있는 두 위치를 도선으로 이어주는 역할을 합니다. 점퍼선은 길이와 타입에 따라 구분할 수 있습니다. 길이는 10cm, 20cm, 30cm 로 나눌 수 있고, 타입은 점퍼선의 끝의 모양에 따라 F타입 또는 M타입으로 나뉩니다.

- F타입 : F타입은 Female의 F자를 의미하며 암단자라고도 불립니다. M타입을 끼울 수 있도록 안으로 들어간 형태입니다.
- M타입 : M타입은 Male의 M자를 의미하며 수단자라고도 불립니다. F타입에 끼울 수 있도록 핀이 나와있는 형태입니다.

점퍼선의 양끝의 타입에 따라 암/암(F/F), 수/수(M/M), 암/수(F/M) 으로 구분 합니다.



### 4. 아두이노 프로그래밍 툴

이 과제에서 사용할 프로그래밍 툴은 아두이노 IDE (스케치)와 스크래치(MBlock)입니다. 두 프로그래밍 툴에 대하여 간단하게 알아보도록 하겠습니다.

아두이노 통합개발환경은 아두이노를 개발하기 위한 거의 모든 기능이 내장된 컴파일러입니다. 다른 말로는 아두이노 IDE 혹은 스케치라고 합니다. C, C++ 기반의 프로그래밍 언어를 사용하여 프로그램을 만들 수 있습니다.

스크래치는 초·중 학생들의 컴퓨팅 사고능력 향상을 목표로 개발된 프로그래밍 교육용 도구입니다. 복잡한 프로그래밍 언어대신 블록을 쌓아 올려 간단한 애니메이션이나 게임과 같은 프로그램을 만들 수 있습니다. mblock 프로그램을 사용하여 아두이노 하드웨어와 연동하여 사용할 수 있습니다. 아두이노 IDE의 모든 기능을 사용할 수 없다는 단점이 있지만 난이도가 쉬워 어린아이도 배울 수 있다는 장점이 있습니다.

○ 아두이노 IDE (스케치)

① 설치

아두이노를 사용하기 위해서는 IDE 소프트웨어(S/W) 설치가 꼭 필요합니다. 아두이노 IDE는 아두이노 홈페이지(<https://www.arduino.cc>)에서 다운받을 수 있습니다.

- 홈페이지에 접속해 상단의 [SOFTWARE] > [DOWNLOADS]로 이동합니다.



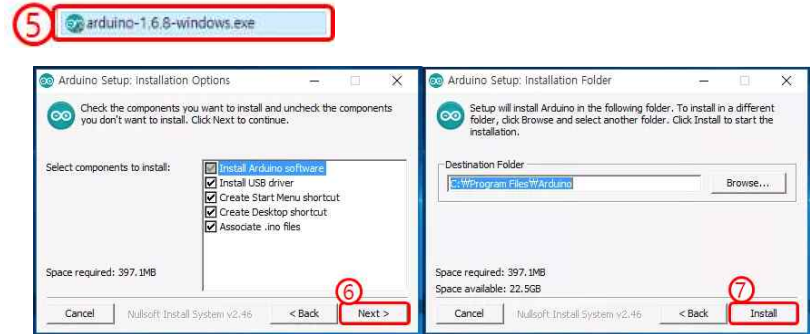
- 자신의 PC 운영체제(OS)환경에 맞는 파일을 마우스로 클릭 합니다.



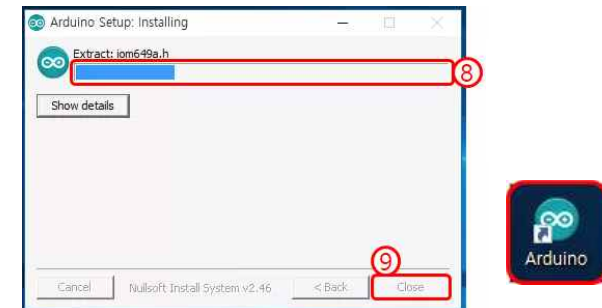
- JUST DOWNLOAD를 클릭하여 파일을 다운로드 합니다.



- 다운 받은 파일을 더블클릭하여 설치를 진행합니다.



- 설치가 완료되면 바탕화면의 Arduino 아이콘을 더블클릭하여 실행합니다.



## ② 연결 및 드라이버 설치



- USB 케이블을 이용하여 아두이노 우노보드와 PC를 연결합니다.
- 연결하면 아래와 같은 메시지가 모니터 화면의 오른쪽 아래에 뜹니다.



- 드라이버 설치가 완료되었다면 스케치를 실행합니다.



- [툴] -> [보드] -> [Arduino/Genuino UNO]를 선택합니다.
- [툴] -> [포트] -> 아두이노 우노 보드와 연결된 포트를 선택합니다.
- 업로드 버튼을 눌러 업로드가 완료되면 정상적으로 연결된 것입니다.

## ③ 화면구성



- 컴파일 (☑️): 소스 에러 체크
- 업로드 (➡️): 아두이노에 소스 업로드
- 새 파일 (📄): 새로운 스케치 창 생성
- 열기 (⬆️): 파일 불러오기
- 저장 (⬇️): 파일 저장하기
- 시리얼모니터 (📡): 데이터 값 확인



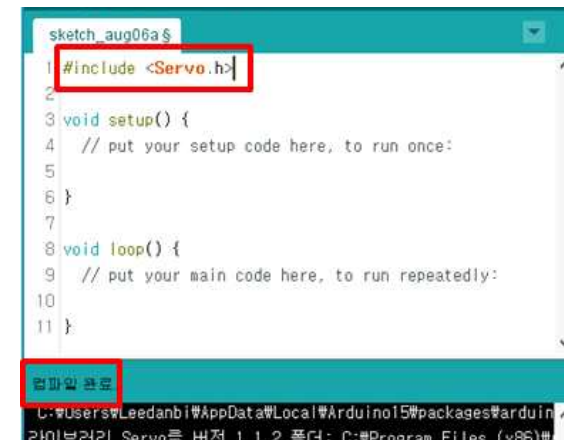
소스 작성 화면

- 초기화 루틴 -> setup() 함수  
아두이노가 실행될 때 최초 1회 실행
- 반복 루틴 -> loop() 함수  
setup()함수 이후 무한반복

컴파일 및 오류 확인

## ④ 라이브러리 사용법

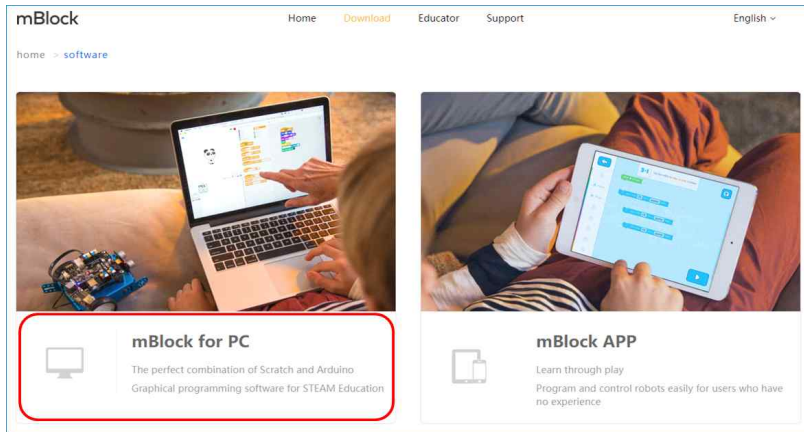
- 아두이노 라이브러리
  - 다양한 센서, 모터, 모듈을 쉽게 제어할 수 있도록 지원하는 파일입니다.
  - 헤더 파일을 include 한 뒤, 초기화해서 사용합니다.
- 아두이노 라이브러리 사용방법에서 -> 서보모터 라이브러리 (뒤에서 자세히 다룰 예정입니다.)
  - #include <Servo.h>를 입력합니다.
  - 컴파일해서 오류가 발생하는지 확인합니다.



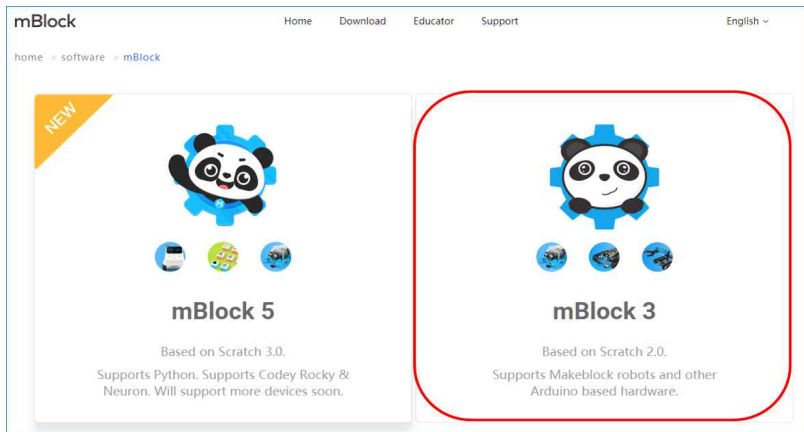
○ 스크래치(MBlock)

① 설치

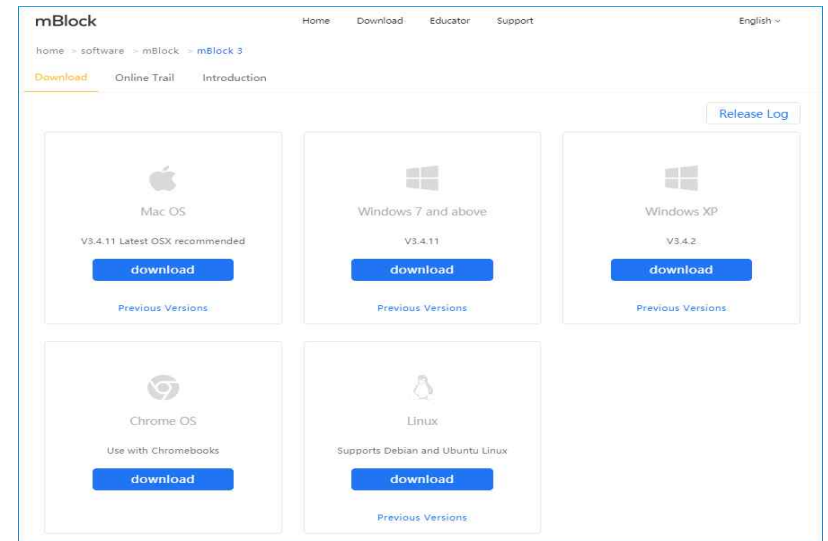
- MBlock 프로그램 다운로드 사이트 <http://www.mblock.cc/software/>에 접속하여 환경에 알맞은 방법을 선택합니다.



- 본 교재는 MBlock3를 기준으로 작성되었습니다. MBlock3를 선택합니다.

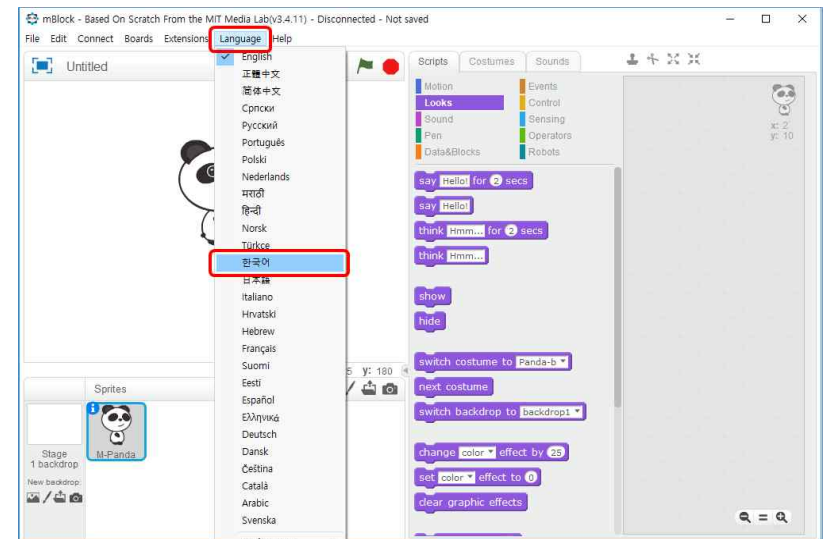


- 자신의 PC 운영체제(OS)환경에 맞는 파일을 마우스로 클릭 합니다. MBlock의 설치는 일반적인 소프트웨어 프로그램 설치처럼 간단하게 진행할 수 있습니다.



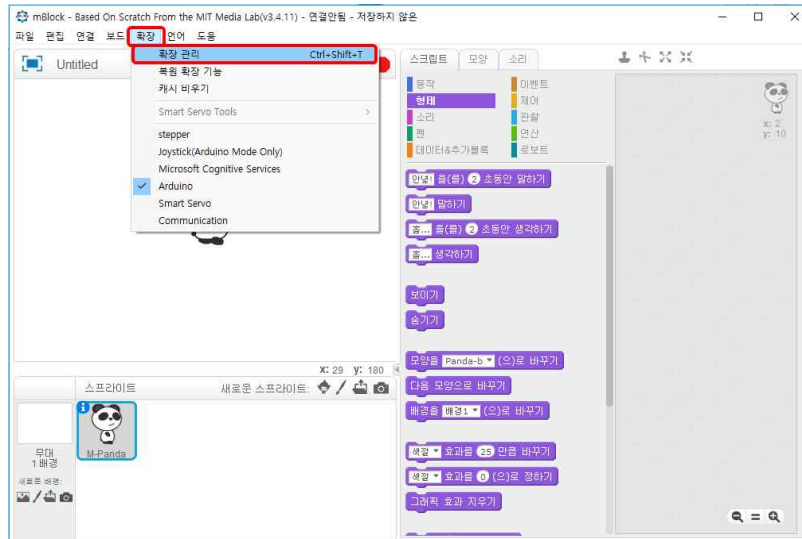
② MBlock 기본 설정

- MBlock 첫 실행화면에서 상단의 [Language] 탭을 클릭하여 언어를 한국어로 설정합니다.

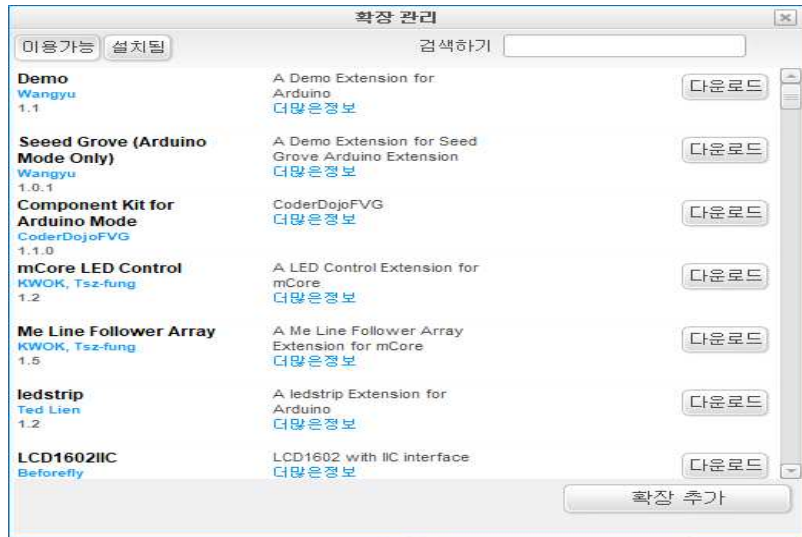


### ③ 라이브러리 사용법

- 상단의 [확장] 탭을 클릭하여 [확장 관리]로 이동합니다.

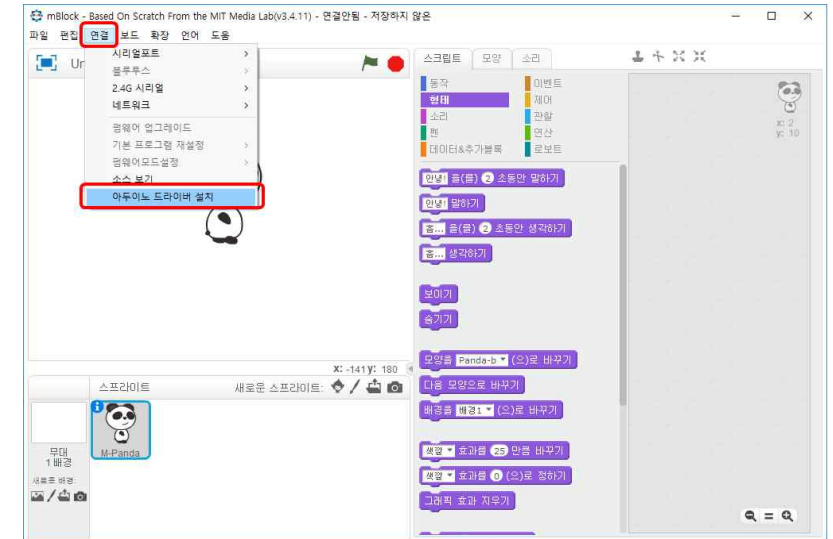


- 확장관리 창에서 필요한 라이브러리를 설치 또는 추가할 수 있습니다.

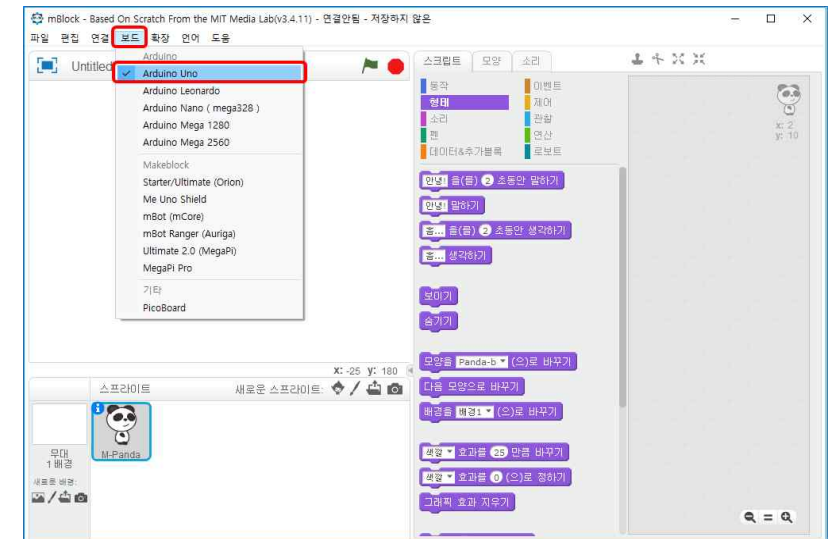


### ④ 아두이노 모드 설정

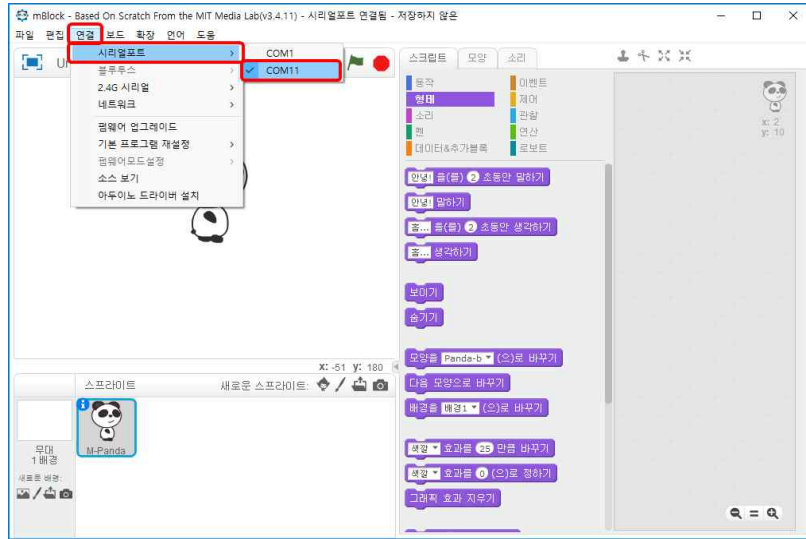
- MBlock 화면에서 상단의 [연결] 탭에서 [아두이노 드라이버 설치]를 클릭하여 설치를 진행합니다.



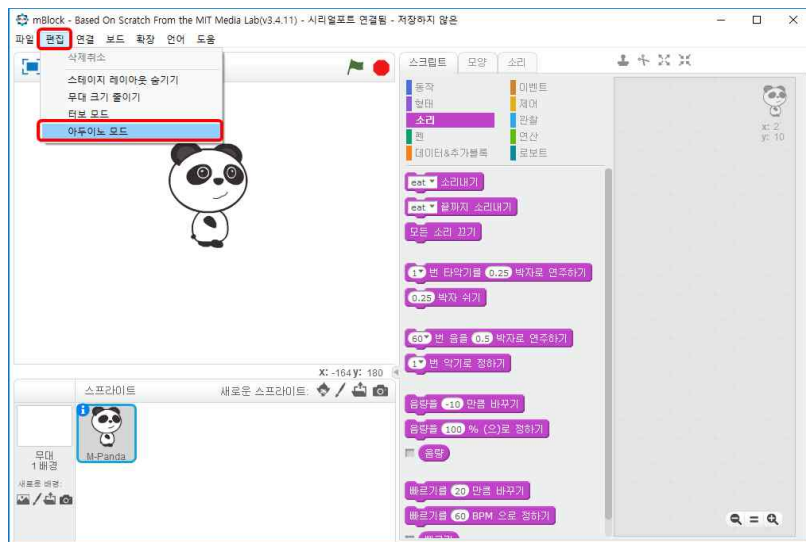
- 상단의 [보드] 탭에서 [Arduino Uno]를 선택합니다.



- 상단의 [연결] 탭에서 [시리얼포트] 메뉴를 열어 현자 아두이노가 연결되어 있는 시리얼 번호를 선택합니다.

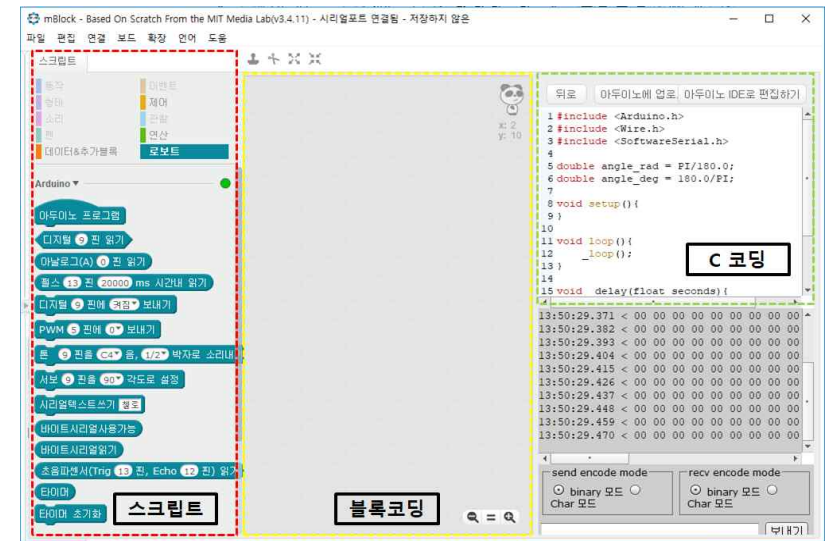


- 상단의 [편집] 탭에서 [아두이노 모드]를 클릭합니다.

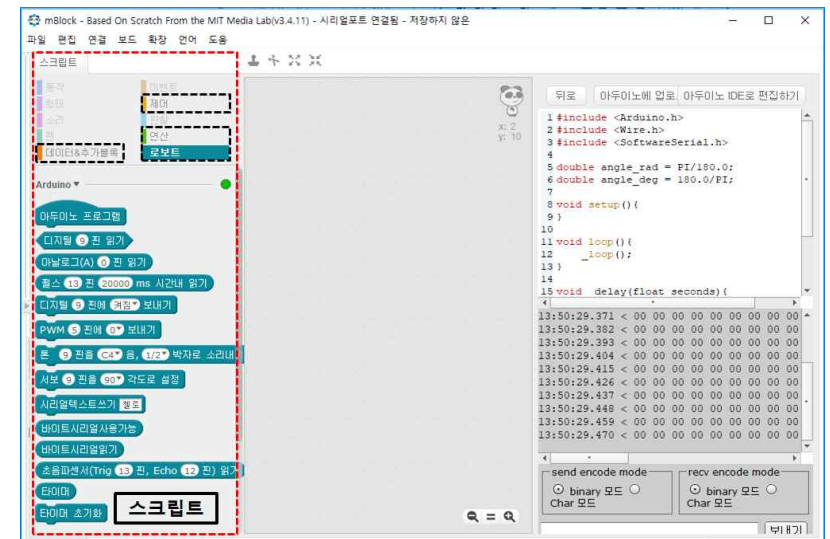


### ⑤ 화면 구성

- MBlock 아두이노 모드의 화면은 크게 [스크립트], [블록코딩], [C코딩] 세 곳으로 나뉩습니다.

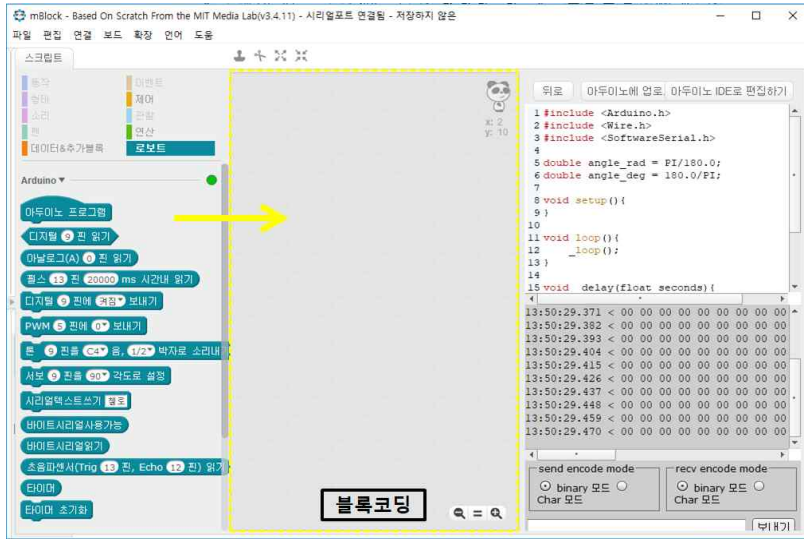


- [스크립트] 창은 사용할 블록들이 있으며, 아두이노 모드에서는 제어, 연산, 데이터&추가블록, 로봇 블록을 사용할 수 있습니다.

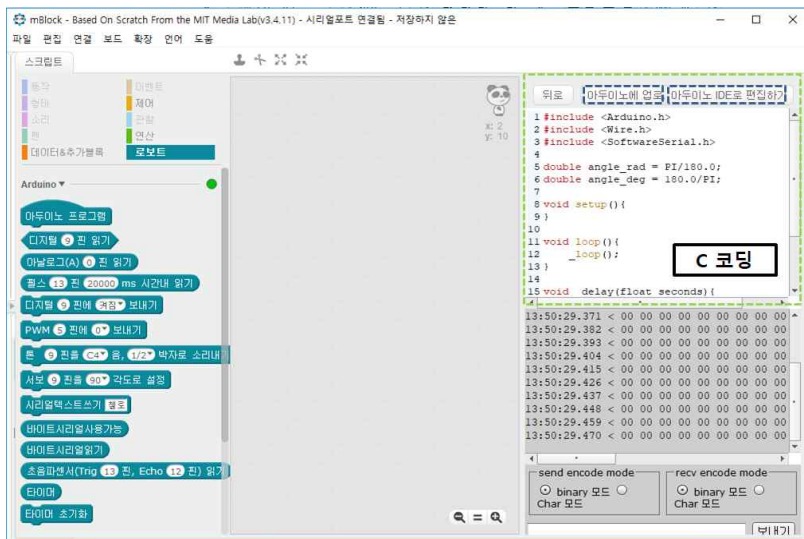




- [블록코딩] 창은 [스크립트] 창에서 블록들을 마우스로 끌어 놓아 작성하는 공간입니다.



- [C코딩] 창은 [블록코딩] 창에서 블록코딩을 하면 C 코딩이 자동으로 입력이 됩니다. 블록에 해당하는 C 소스를 확인 가능하며 [아두이노 IDE로 편집하기] 버튼을 눌러 편집이 가능합니다. 그리고 [아두이노 업로드] 버튼을 눌러 연결된 아두이노에 소스를 업로드할 수 있습니다.
- 아두이노IDE를 이용한 편집은 블록코딩에 반영이 되지 않아 따로 저장하여 관리하면 됩니다.



## ⑥ 블록 설명

### • 제어 블록

명령 블록	설명
	지정된 시간동안 기다렸다가 다음 명령 블록을 실행합니다.
	지정된 횟수만큼만 반복합니다. 반복 횟수가 정해져 있을 때 사용합니다.
	끝나지 않고 계속해서 반복하며, 이후 다른 명령 블록을 연결할 수 없습니다.
	만약 <조건>을 검사하여 참이면(조건을 만족하면) 명령 블록 안에 포함된 명령 블록을 실행합니다. 거짓이면(조건을 만족하지 않으면) 명령 블록을 실행하지 않고 다음 명령 블록을 실행합니다.
	만약 <조건>을 검사하여 조건이 참이면, 블록을 실행하고 조건이 거짓이면 아래에 있는 [아니면] 명령 블록을 실행합니다.
	<조건>이 참일 때까지 기다립니다.
	<조건>을 검사하여 거짓이면 반복을 합니다. 조건이 참이면 반복을 끝냅니다.

• 연산 블록

명령 블록	설명
	첫 번째 값과 두 번째 값을 더합니다.
	첫 번째 값과 두 번째 값을 뺍니다.
	첫 번째 값과 두 번째 값을 곱합니다.
	첫 번째 값과 두 번째 값을 나눕니다.
	첫 번째 값부터 두 번째 값 사이의 난수를 구합니다. 정수를 입력하면 정수 난수를 구하고, 소수를 입력하면 소수 난수를 구합니다.
	첫 번째 값과 두 번째 값을 비교하여 두 번째 값이 크면 '참' 아니면 '거짓'이 됩니다.
	첫 번째 값과 두 번째 값을 비교하여 같으면 '참' 아니면 '거짓'이 됩니다.
	첫 번째 값과 두 번째 값을 비교하여 첫 번째 값이 크면 '참' 아니면 '거짓'이 됩니다.
	첫 번째 식과 두 번째 식이 모두 참이면 '참', 두 식 중 하나의 식이 거짓이면 '거짓'이 됩니다.
	첫 번째 식과 두 번째 식 중 하나라도 참이면 '참', 두 식 모두 거짓이면 '거짓'이 됩니다.
	수식이 참이면 '거짓', 수식이 거짓이면 '참'이 됩니다.
	두 문자열을 결합하여 새로운 문자열을 만듭니다.
	문자열에서 지정된 순서의 문자를 기억합니다.
	문자열의 길이를 기억합니다.
	첫 번째 값을 두 번째 값으로 나눈 나머지를 기억합니다.
	입력된 값을 반올림합니다.
	입력된 값의 수학함수(절대값, 바닥함수, 천장함수, 제곱근, sin, cos, tan, asin 등)에 해당하는 결과 값을 구합니다.

• 데이터&추가블록

명령 블록	설명
	새로운 변수를 만듭니다.
	변수의 값을 보고합니다.
	변수의 값을 지정된 값으로 변경합니다.
	변수의 값을 현재 값에서 지정된 값만큼 변경합니다.
	변수를 화면에 표시합니다.
	변수를 화면에서 숨깁니다.
	새로운 블록을 만듭니다.

• 로봇 블록

명령 블록	설명
	블록 코딩을 C코드로 바꿔줍니다.
	해당 디지털 핀으로 입력받는 값을 읽어 옵니다.
	해당 아날로그 핀으로 입력받는 값을 읽어 옵니다.
	해당 디지털 핀에 켜짐/꺼짐 신호를 보냅니다.
	해당 핀에 PWM 값을 보냅니다.
	해당 핀에 서보모터 연결 및 회전 각도를 입력합니다.
	문자를 출력합니다.
	시리얼 통신으로 데이터를 받았는지 확인합니다.
	시리얼 통신으로 받은 데이터를 읽어옵니다.
	초음파센서를 사용할 때 핀 설정 및 데이터를 읽어옵니다.
	실행 시간 값을 읽어옵니다.
	타이머를 초기화 합니다.

◇ 기초학습

1. 아두이노를 이용한 LED 제어

○ 기본 설명

아두이노를 이용한 LED 켜보기는 프로그램 소스가 업로드 된 보드에 LED를 연결하고 전원을 ON하면 LED 불이 켜지는 기본적인 예제입니다. 일반적인 LED는 저항을 따로 연결해 주어야 하지만 사용할 LED바는 저항이 달려있어 별도의 저항이 필요 없습니다.

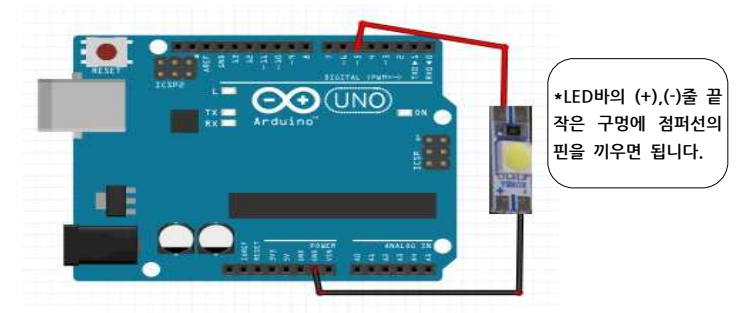
재료 구성

아두이노 우노 보드	LED 바	점퍼선

○ 하나의 LED바 ON

① 연결

아두이노	LED 바
디지털 핀 5	(+)
GND	(-)



<LED바 배선 연결도>

## ② 스케치 작성 및 업로드

```
int ledPin = 5;           // 핀 번호, 이름 설정

void setup() {
  pinMode(ledPin, OUTPUT); // ledPin의 출력(OUTPUT) 설정
}

void loop() {
  digitalWrite(ledPin, HIGH); // LED바 ON
}
```

- int ledPin = 5;  
=> 변수이름을 정해주고 해당 변수의 값을 5로 지정해줍니다.
- pinMode(ledPin, OUTPUT);

```
pinMode(핀 번호, 입/출력);
```

=> pinMode는 사용할 핀들의 입력(INPUT), 출력(OUTPUT)을 설정 해주는 함수로, setup() 안에서 선언해 줍니다. pinMode(ledPin, OUTPUT); 소스의 의미는 5번(ledPin)핀을 출력으로 설정 한다는 의미입니다.

- digitalWrite(ledPin, HIGH);

```
digitalWrite(핀 번호, 입력신호);
```

=> digitalWrite()는 아두이노 보드의 디지털 핀에 디지털 신호(HIGH/LOW)를 주기위해 사용되는 함수입니다. HIGH신호를 주면 LED바의 불이 켜지고 LOW신호를 주면 LED바의 불이 꺼집니다.

## ○ 하나의 LED바 ON/OFF 반복

### ① 연결

연결 구조는 5페이지의 <2-2 하나의 LED바 ON>과 같습니다.

### ② 스케치 작성 및 업로드

```
int ledPin = 5;           // 핀 번호, 이름 설정

void setup() {
  pinMode(ledPin, OUTPUT); // ledPin의 출력(OUTPUT) 설정
}

void loop() {
  digitalWrite(ledPin, HIGH); // LED바의 불이 켜짐
  delay(2000); // 2초 대기
  digitalWrite(ledPin, LOW); // LED바의 불이 꺼짐
  delay(2000); // 2초 대기
}
```

- delay(2000);

```
delay(시간(ms));
```

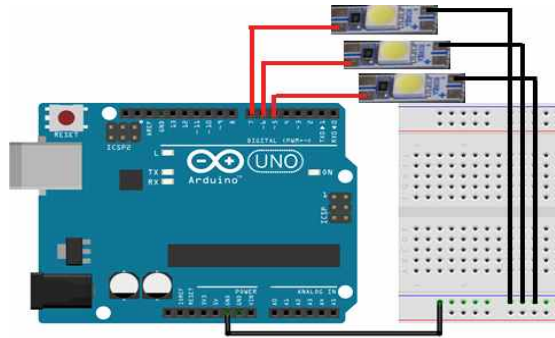
=> delay(ms);는 밀리초 단위로 프로그램을 일시적으로 중지하는 함수입니다. 이 시간 동안에는 센서의 값을 읽거나, 연산, 통신 등의 기능도 중단 됩니다. 위 소스는 2초 동안 LED바의 불이 켜졌다 2초 동안 LED 바의 불이 꺼지는 것을 반복합니다.

\*ms는 밀리초를 말하며 1ms는 1/1000초를 의미합니다. 예를 들어 1000ms는 1초입니다.

## ○ 3개의 LED바 ON

### ① 연결

아두이노	LED 바
디지털 핀 5, 6, 7	(+)
GND	(-)



<3개의 LED바 배선 연결도>

## ② 스케치 작성 및 업로드

```
int ledPin1 = 5;           // 핀 번호, 이름 설정
int ledPin2 = 6;
int ledPin3 = 7;

void setup() {
  pinMode(ledPin1, OUTPUT); // ledPin1의 출력(OUTPUT) 설정
  pinMode(ledPin2, OUTPUT); // ledPin2의 출력(OUTPUT) 설정
  pinMode(ledPin3, OUTPUT); // ledPin3의 출력(OUTPUT) 설정
}

void loop() {
  digitalWrite(ledPin1, HIGH); // LED바1의 불이 켜짐
  digitalWrite(ledPin2, HIGH); // LED바2의 불이 켜짐
  digitalWrite(ledPin3, HIGH); // LED바3의 불이 켜짐
}
```

아두이노 우노보드의 5~7번 핀을 사용하여 3개의 LED바를 연결하고 digitalWrite함수를 이용하여 3개의 LED바에 불이 들어오는 것을 확인할 수 있습니다.

### \* 응용예제 - 3개의 LED ON/OFF 반복하기






=> 3개의 LED바가 2초 동안 켜지고 2초 동안 꺼지는 것을 반복하는 스케치를 작성해 보세요.

## 2. 아두이노를 이용한 Motor 제어

### ○ 기본 설명

모터의 종류는 DC모터, SERVO모터, STEP모터 등이 있습니다. 아두이노에서는 세 가지 모터를 쉽게 제어가 가능합니다. 스마트 미니시티에서 사용할 SERVO모터와 STEP모터에 대해 알아보겠습니다. SERVO모터는 0도 ~ 180도를 회전할 수 있으며, STEP모터는 모터드라이브를 사용하여 360도 회전, 회전속도와 방향을 제어할 수 있습니다.

### 재료 구성

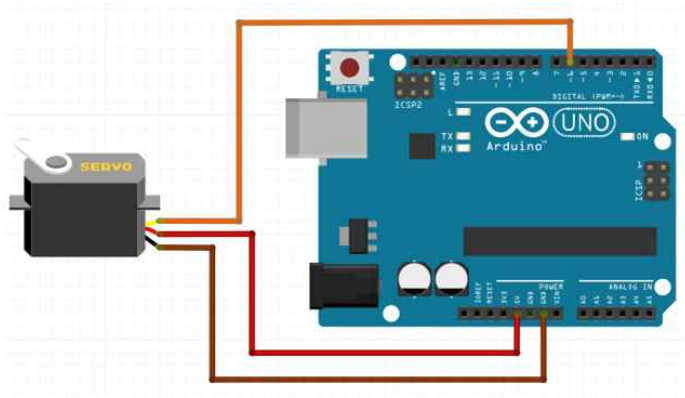
		
아두이노 우노 보드	점퍼선	SERVO모터
		
STEP모터	모터드라이브	

### ○ SERVO모터 동작하기

#### ① 연결

아두이노	SERVO모터 선
디지털 핀 6	주황색
5V	빨간색
GND	갈색

\* 위의 서보모터 선의 역할과 색을 기억하고 10cm수/수 점퍼선을 이어서 연결해주세요.



<서보모터 배선 연결도>

### <서보모터 90도->0도, 0도->90도 반복 회전하는 예제>

```

/* 90도->0도, 0도->90도 반복 회전하는 서보모터 예제 */
#include <Servo.h> // 라이브러리 Servo.h를 포함시키는 명령.
Servo myservo; // 서보모터를 제어하기 위해 myservo라는 객체 생성

void setup() {
  myservo.attach(6); // 6번 핀에 서보모터가 연결되었음을 설정
}

void loop() {
  myservo.write(90); // 서보모터 90도 회전
  delay(5000); // 5초
  myservo.write(0); // 서보모터 0도 회전
  delay(5000); // 5초
}

```

### <0도->180도, 180도->0도로 반복 회전하는 서보모터 예제>

```

/* 0도->180도, 180도->0도로 일정 각도씩 움직여 반복 회전하는 서보모터 예제 */
#include <Servo.h> // 라이브러리 Servo.h를 포함시키는 명령.
Servo myservo; // 서보모터를 제어하기 위해 myservo라는 객체 생성

void setup() {
  myservo.attach(6); // 6번 핀에 서보모터가 연결되었음을 설정
}

void loop() {
  for(int pos = 0; pos <= 180; pos += 10) { // 서보모터가 0도 -> 180도로 10도씩회전
    myservo.write(pos); // pos의 값 만큼 회전
    delay(100); // 0.1초
  }

  delay(1000); // 1초 대기

  for(int pos = 180; pos >= 0; pos -= 10) { // 서보모터가 180도 -> 0도로 10도씩회전
    myservo.write(pos); // pos의 값 만큼 회전
    delay(100); // 0.1초
  }
}

```

## ② 스케치 작성 및 업로드

### <서보모터 90도 회전 예제>

```

/* 90도 회전하는 서보모터 예제 */
#include <Servo.h> // 라이브러리 Servo.h를 포함시키는 명령.
Servo myservo; // 서보모터를 제어하기 위해 myservo라는 객체 생성

void setup() {
  myservo.attach(6); // 6번 핀에 서보모터를 연결하기 위해 설정
}

void loop() {
  myservo.write(90); // 서보모터 90도 회전
}

```

- #include <Servo.h>  
=> #include는 외부 라이브러리를 스케치 안에 포함할 때 사용합니다. Servo.h는 서보모터를 제어하기 위한 아두이노에서 기본적으로 제공하는 라이브러리입니다.
- Servo myservo;  
=> myservo라는 객체를 선언해줍니다.
- myservo.attach(6);  
=> 서보모터의 주황색 선이 연결된 핀을 지정해줍니다. 위의 소스는 6번 핀에 지정된 것입니다.
- myservo.write(90);  
=> 0도 ~ 180도 사이의 각도를 지정해줍니다.

• for문

```
for( 초기화식; 조건식; 증가식 ) {
    ... ..
}
```

=> 위와 같은 프로그래밍 문법을 반복문 for문이라고 합니다. for문은 기초적인 문법중 하나로 식을 활용하여 조건이 만족할 때 까지 반복합니다.

**초기화식** : 조건식에서 참조될 변수의 초기 값을 지정한 식, 위의 소스에서 초기화식은 int pos = 0;이다.

**조건식** : for문을 계속 수행할지 아닌지를 결정하는 부분. 위의 소스에서 조건식은 pos <= 180;이며 pos의 값이 0부터 180이 될 때까지 수행을 반복합니다.

**증가식** : 변수의 값을 증가시키거나 감소시키는 식이 들어갑니다. 예를 들어 pos++는 pos의 값을 1씩 증가시키고 pos--는 pos의 값을 1씩 감소시키는 식입니다. 위의 소스에서 pos+=10과 pos-=10은 변수 값을 10씩 증가시키거나 10씩 감소시키는 것입니다.

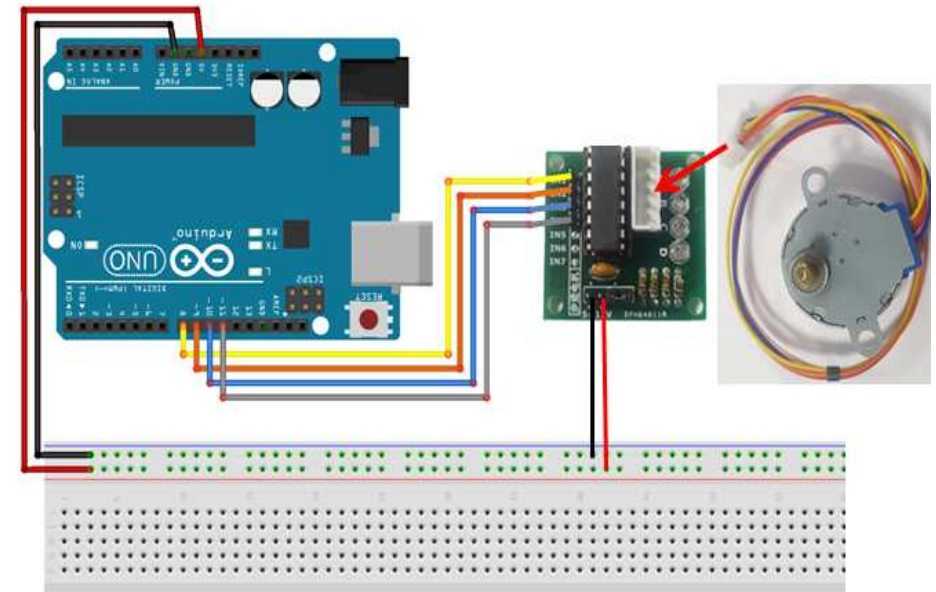
```
for(int pos = 0; pos <= 180; pos += 10) {
    myservo.write(pos);
    delay(100);
}
```

=> 변수 pos의 값이 초기값 0에서 180미만인지 확인 후 for문 안의 내용을 수행 합니다. 그런 다음 증가식에 따라 pos의 값을 10증가시켜 서보모터를 10도 회전 합니다. 이렇게 계속 반복하여 pos의 값이 180이 되었을 때 반복을 중지합니다.

○ STEP모터 동작하기

① 연결

아두이노	모터드라이브	STEP모터
디지털 핀 8	IN1	-
디지털 핀 9	IN2	-
디지털 핀 10	IN3	-
디지털 핀 11	IN4	-
	묶음 핀 5개	묶음 선 5개
5V	(+)	-
GND	(-)	-



<스텝모터 & 모터드라이브 배선 연결도>

\* 스텝모터의 묶음 선 5가닥과 모터드라이브의 묶음 핀 5개를 표시된 방향에 맞게 꽂아주세요.

## ② 스케치 작성 및 업로드

```
#include <Stepper.h> // 라이브러리 Stepper.h를 포함시키는 명령.
// 2048은 360도를 의미하며 스텝모터가 한번 동작할 때 마다 360도 움직이게 됩니다.
const int stepsPerRevolution = 2048;
// myStepper 객체 생성과 회전각과 핀 번호 8,9,10,11 설정
Stepper myStepper(stepsPerRevolution, 8, 10, 9,11);

void setup() {
  myStepper.setSpeed(14); // STEP모터 회전속도
}

void loop() {
  myStepper.step(stepsPerRevolution); // STEP모터 회전
}
```

먼저 #include <Stepper.h>를 통해 스텝모터를 쉽게 제어하기 위한 아두이노에서 기본적으로 제공하는 라이브러리 Stepper.h를 사용하였습니다.

- #include <Stepper.h>  
=> Stepper.h는 스텝모터를 제어하기 위한 아두이노에서 기본적으로 제공하는 라이브러리입니다.
- const int stepsPerRevolution = 2048;  
=> 스텝모터가 360도 회전을 위한 총 스텝수입니다.
- Stepper myStepper(stepsPerRevolution, 8, 10, 9,11);  
=> myStepper라는 객체를 생성하고 위에서 설정한 회전각도와, 핀 번호를 설정해 줍니다.
- myStepper.setSpeed(14);  
=> 스텝수를 2048로 설정할 경우 모터의 회전속도를 10~14로 설정하는 것이 적절합니다.
- myStepper.step(stepsPerRevolution);  
=> stepsPerRevolution앞에 ' - ' 를 붙여주면 반대방향으로 회전하게 됩니다.

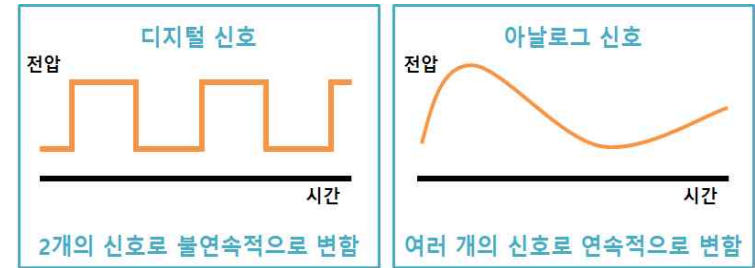
### \* 응용예제 - 반대방향으로 회전하기

=> 시계방향으로 회전하고 2초 후 반시계방향으로 회전하고 2초 후 다시 반복하는 스케치를 작성해서 아두이노 우노보드에 업로드하여 동작을 확인해 보세요.

## 3. 아두이노를 이용한 Sensor 제어

### ○ 기본 설명

센서란 열, 빛, 소리, 움직임 등을 계속하여 신호로 알려주는 부품입니다. 여기서 말하는 신호에 따라 센서를 두 가지로 분류할 수 있습니다. 디지털 신호를 출력해주는 디지털 센서와 아날로그 신호를 출력해주는 아날로그 센서가 있습니다. 디지털 신호는 두 가지 종류의 전압으로만 변화하는 특성을 가지고 있으며, 아날로그 신호는 시간에 따라 연속적으로 변화하는 특성을 가지고 있습니다.



- 조도 센서 : 빛의 밝은 정도를 측정하는 아날로그 센서.  
실생활 예 - 스마트폰(화면 밝기 자동 조절 기능), 자동으로 켜지는 무드등, 가로등 등
- 사운드 센서 : 소리의 크기를 측정하는 아날로그 센서.  
실생활 예 - 소음측정기, 데시벨 측정기 등
- 적외선 센서 : 장애물 감지 측정하는 디지털 센서.  
실생활 예 - 아파트 복도 전등(움직임이 감지되면 일정시간 동안 불이 켜짐) 등

### ○ 조도센서 값 시리얼모니터에서 알아보기

#### ① 조도 센서란?

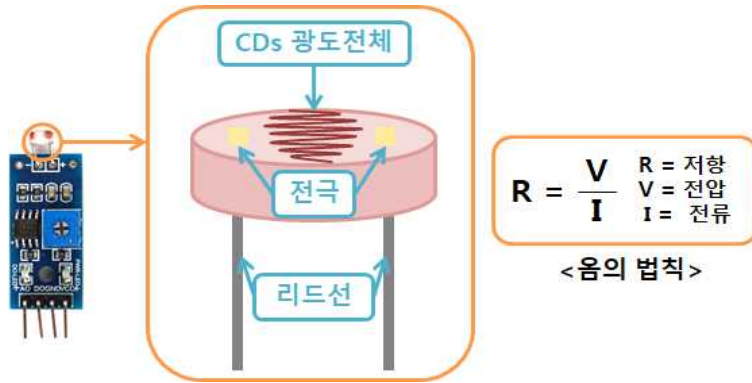


<조도센서 모듈의 기능 및 핀 배열>

- ▶ 조도센서는 측정된 빛의 세기에 따라 저항 값이 변하는 센서입니다. 빛이 강하면 저항 값이 약해지고, 빛이 약하면 저항 값이 높아집니다. 이러한 성질에 따라 아날로그 신호를 발생시키며 아두이노로 밝기 값을 얻어와 LED로 확인할 수 있습니다. 조도센서의 경우 디지털 핀을 사용할 때 감도조절을 합니다.



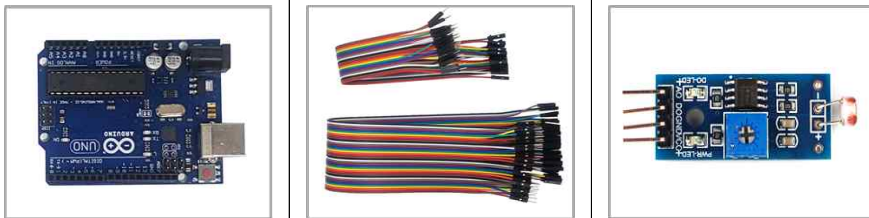
## ② 동작 원리



<동작 원리>

- ▶ 센서에 빛을 비추게 되면 센서 내부의 CDs 광도전체에서 전류가 증가하게 됩니다. 증가된 전류는 옴의 법칙에 따라 저항의 크기를 줄여줍니다. 만약 센서에 빛을 비추지 않게 된다면, 적은 양의 전류를 가지게 되어 저항의 크기가 늘어나게 됩니다. 이 때 생성된 전류는 전극을 타고 리드선으로 흘러 SIG핀으로 흐르게 됩니다.

### 재료 구성



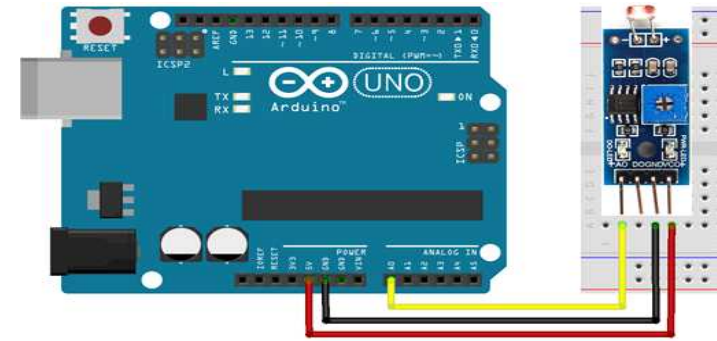
아두이노 우노 보드

점퍼선

조도센서

## ③ 연결

아두이노	조도센서
아날로그 핀 A0	AO(AnalogOut)
5V	VCC
GND	GND



<조도센서 배선 연결도>

## ④ 스케치 작성 및 업로드

```

int cdsPin = A0; // 아날로그 핀 A0을 cdsPin로 설정

void setup() {
  Serial.begin(9600); // 시리얼 통신을 위한 통신속도 설정
  pinMode(cdsPin, INPUT); // A0핀 입력핀 설정
}

void loop() {
  int cds_data = analogRead(cdsPin); // A0 핀으로 입력된 데이터를 cds_data에 저장
  Serial.print("빛의 어두운 정도 : "); // 시리얼모니터에 " "안의 문자 출력
  Serial.println(cds_data); // 시리얼모니터에 cds_data 값 출력
  delay(10);
}
    
```

### \* 시리얼 모니터 사용방법

=> 시리얼 모니터 창을 띄우는 다음 세 가지 방법이 있습니다.

- ① 시리얼 모니터 화면은 스케치하면 상단의 탭 [툴] > [시리얼 모니터]
- ② 단축키[Ctrl+Shift+M]
- ③ 시리얼 모니터 아이콘을 클릭 (6페이지의 아두이노 스케치 화면구성 참고)

- Serial.begin(9600);

Serial.begin( 보드레이트 );

=> 시리얼 데이터 통신을 위해 보드레이트(초당 전송 비트수)를 설정합니다. 보드레이트는 보통 9600으로 설정합니다. 시리얼 모니터에서 데이터를 확인하기 위해서는 소스와 시리얼모니터의 보드레이트를 동일하게 지정해줘야 합니다.

- Serial.print("문자");

```
Serial.print("출력할 문자");
```

=> 입력한 문자를 시리얼 모니터에 출력합니다.

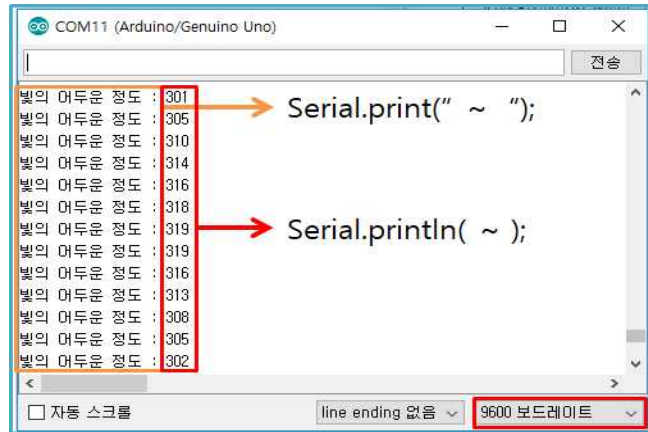
- Serial.print(데이터);

```
Serial.print(출력할 값);
```

=> 시리얼 모니터에 데이터 값을 출력합니다.

- Serial.println( ~ );

=> 기능은 위의 Serial.print()와 같습니다. 뒤에 In을 붙여주면 줄바꿈이 추가되어 시리얼 모니터에서 문자가 출력된 다음 엔터를 한번 쳐주는 효과가 있습니다.



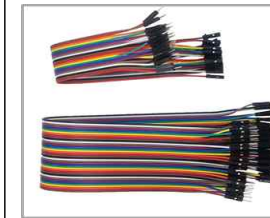
<시리얼 모니터 화면>

## ○ 조도센서를 이용한 LED ON/OFF

### 재료 구성



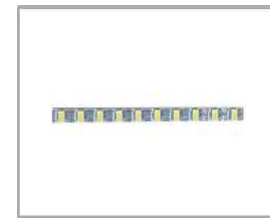
아두이노 우노 보드



점퍼선



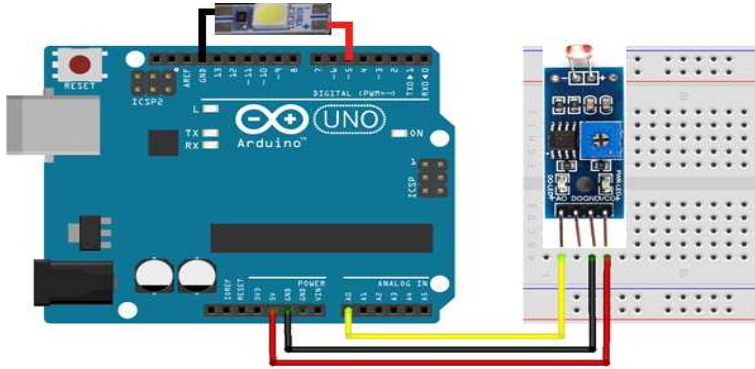
조도센서



LED 바

### ① 연결

아두이노	조도센서	LED 바
아날로그 핀 A0	AO(AnalogOut)	-
디지털 핀 5	-	(+)
5V	VCC	-
GND	GND	(-)



<조도센서, LED바 배선 연결도>

## ② 스케치 작성 및 업로드

```
int cdsPin = A0; // 아날로그 핀 A0을 cdsPin로 설정
int ledPin = 5; // 디지털 핀 5번을 ledPin로 설정

void setup() {
  Serial.begin(9600); // 시리얼 통신을 위한 통신속도 설정
  pinMode(cdsPin, INPUT); // cdsPin 입력핀 설정
  pinMode(ledPin, OUTPUT); // ledPin 출력핀 설정
}

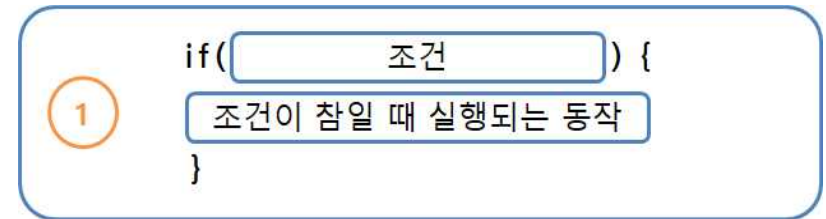
void loop() {
  int cds_data = analogRead(cdsPin); // A0 핀으로 입력된 데이터를 cds_data에 저장

  if(cds_data >= 500) { // cds_data 값이 500 이상이면
    Serial.println("어두움");
    digitalWrite(ledPin, HIGH);
  } else { // cds_data 값이 500보다 작으면
    Serial.println("밝음");
    digitalWrite(ledPin, LOW);
  }
  delay(10);
}
```

빛의 밝기 데이터 값이 500 이상이면 시리얼 모니터에 “어두움”이라는 문자가 출력되면서 LED불이 켜집니다. 500보다 작으면 “밝음”이라는 문자가 출력되면서 LED불이 꺼집니다.

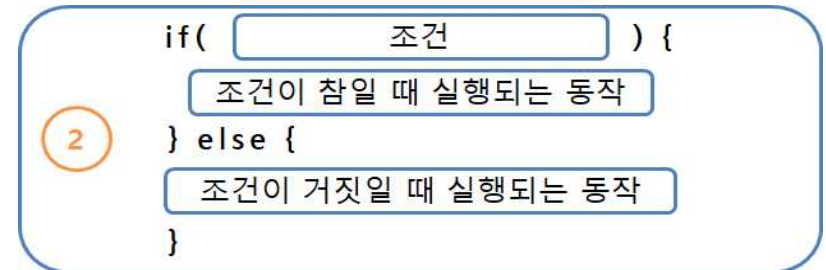
- `if(cds_data >= 500) { ~ }`  
=> 위와 같은 프로그래밍 문법을 **if문**이라고 합니다. if문은 기초적인 문법중 하나입니다. if문은 크게 세 가지 구조로 작성 됩니다.

- if문



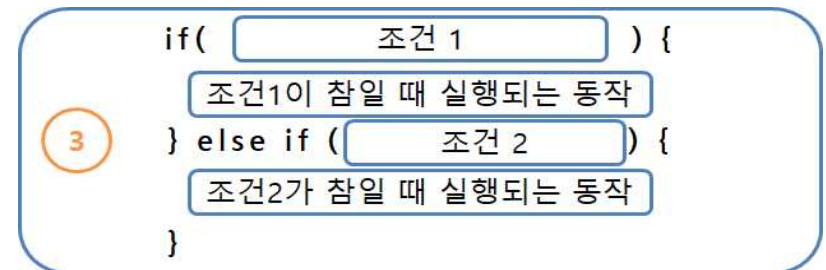
=> 가장 기초적인 if문 구조로서 하나의 조건에 대해 참/거짓을 판별한 다음 조건이 참일 때 동작이 실행됩니다.

- if ~ else문



=> if문에 else문을 추가한 구조입니다. else문은 if문이 실행되지 않을시 반드시 실행되는 블록으로 조건이 참일 경우에는 if문 반드시 실행되고 조건이 거짓일 경우에는 else문이 반드시 실행됩니다.

- if ~ else if문



=> if문에 else if문을 추가한 구조입니다. if문의 조건1이 참일 경우 동작을 실행하고 거짓일 경우 else if문의 조건2의 참/거짓을 판별한 다음 참일 경우 다른 동작을 실행하게 됩니다.

○ 사운드센서 값 시리얼모니터에서 알아보기

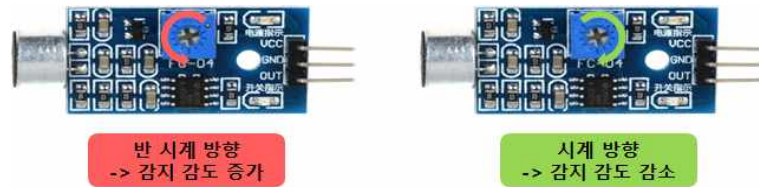
① 사운드 센서란?



<사운드 센서 기능 및 핀 배열>

▶ 사운드센서는 주변에서 발생하는 소리를 마이크로 모아, 소리의 크기를 증폭시켜 출력합니다. 만약 원하는 대상의 소리가 잘 감지가 안 되는 경우에는 가변저항을 통해 감도를 조절할 수 있습니다. 소리 신호는 기본적으로 아날로그 전압 신호로 출력되기 때문에 아두이노를 사용할 경우 아날로그 핀을 통해 측정이 됩니다.

② 감도 조절



<감도 조절부 사진>

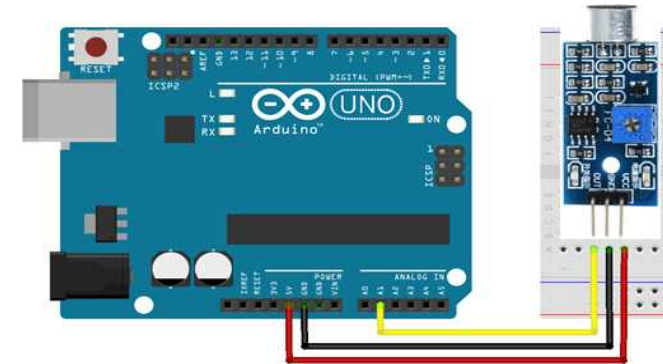
▶ 센서의 감도 조절부를 반시계 방향으로 돌리면 감지감도가 증가하고, 시계 방향으로 돌리면 감지감도가 감소합니다.

재료 구성

		
아두이노 우노 보드	점퍼선	사운드센서

③ 연결

아두이노	사운드센서
아날로그 핀 A1	AO(AnalogOut)
5V	VCC
GND	GND



<사운드 센서 배선 연결도>

④ 스케치 작성 및 업로드

```
int soundPin = A1; // 아날로그 핀 A1을 soundPin으로 설정

void setup() {
  Serial.begin(9600); // 시리얼 통신을 위한 통신속도 설정
  pinMode(soundPin, INPUT); // A1핀 입력 설정
}

void loop() {
  int sound_data = analogRead(soundPin); // sound_data에 값 저장
  Serial.print("소리 크기 : ");
  Serial.println(sound_data);
  delay(10);
}
```

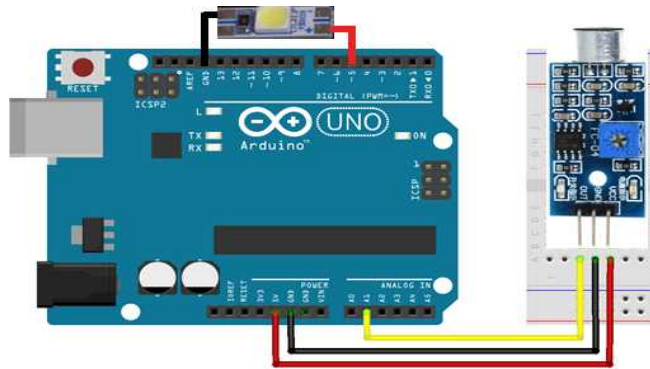
시리얼 모니터를 통해 사운드센서로 측정되는 소리크기에 대한 데이터를 알 수 있습니다.

**\* 감도를 조절하면서 사운드센서 데이터가 정상적으로 측정되는지 꼭 확인해 주세요.**

○ 사운드 센서를 이용한 LED ON/OFF

① 연결

아두이노	사운드센서	LED 바
아날로그 핀 A1	AO(AnalogOut)	-
디지털 핀 5	-	(+)
5V	VCC	-
GND	GND	(-)



<사운드 센서, LED바 배선 연결도>

② 스케치 작성 및 업로드

```
int soundPin = A1; // 아날로그 핀 A1을 soundPin으로 설정
int ledPin = 5;    // 디지털 핀 5번을 ledPin으로 설정

void setup() {
  Serial.begin(9600); // 시리얼 통신을 위한 통신속도 설정
  pinMode(soundPin, INPUT); // soundPin 핀 입력 설정
  pinMode(ledPin, OUTPUT); // ledPin 핀 출력 설정
}

void loop() {
  int sound_data = analogRead(soundPin); // sound_data에 값 저장
  Serial.print("소리 크기 : ");
  Serial.println(sound_data);

  if(sound_data >= 500) { // 500보다 크면
    digitalWrite(ledPin, HIGH);
  }
}
```

```
delay(3000);
} else { // 500보다 작으면
  digitalWrite(ledPin, LOW);
}
delay(10);
}
```

사운드센서를 통해 측정된 값이 500보다 크면 LED의 불이 들어오고 500보다 작으면 LED가 켜지지 않습니다. LED 불이 켜진 것을 확인하기 쉽게 delay()함수를 이용해 켜지고 3초 후에 꺼지도록 합니다.

○ 적외선센서 값 시리얼모니터에서 알아보기

① 적외선 센서란?



<적외선 송수신 센서의 기능 및 핀 배열>

▶ 적외선(IR) 송수신 센서 모듈은 적외선 수광부와 발광부를 가지고 있으며, 가변 저항을 이용해 측정 거리 조절이 가능한 모듈입니다. 측정거리는 2cm ~ 15cm이며, 아두이노와 연결하여 스마트 카 등 다양한 적외선 기반 장애물 인식 및 회피를 위한 용도로 사용됩니다.

② 동작 원리



<적외선 송수신 센서 모듈의 원리>

▶ 우리가 주변에서 쉽게 볼 수 있는 TV 리모컨 등에 내장된 적외선 LED에서 적외선을 방출합니다. 방출된 적외선이 물체에 부딪혀 반사되어 포토트랜지스터가 이를 받아들여 장애물을 감지하는 원리입니다. 장애물이 감지되면 센서 LED가 불이 들어옵니다.

③ 감도 조절



<감도 조절부 사진>

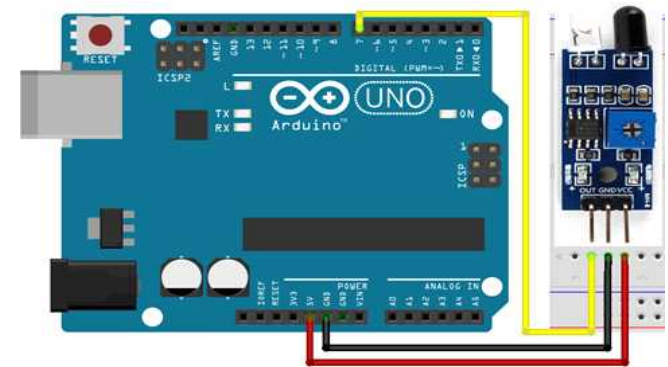
▶ 센서의 감도 조절부를 반시계 방향으로 돌리면 감지거리가 감소하고, 시계 방향으로 돌리면 감지거리가 증가합니다.

재료 구성



④ 연결

아두이노	적외선센서
디지털 핀 7	OUT
5V	VCC
GND	GND



<적외선 센서 배선 연결도>

#### ④ 스케치 작성 및 업로드

```
int irPin = 7;           // 디지털 핀 7번 설정

void setup() {
  Serial.begin(9600);    // 시리얼 통신을 위한 통신속도 설정
  pinMode(irPin, INPUT); // A1핀 입력 설정
}

void loop() {
  int ir_data = digitalRead(irPin);    // ir_data에 값 저장

  Serial.print("감지 체크 : ");
  Serial.println(ir_data);

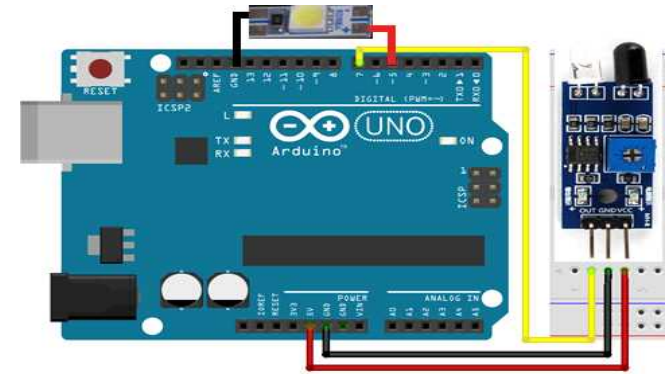
  if(ir_data == HIGH) {                // 장애물 미확인시
    Serial.println("감지된 물체가 없습니다.");
  } else {                              // 장애물 확인시
    Serial.println("물체가 감지되었습니다.");
  }
  delay(100);
}
```

적외선 센서는 디지털 핀으로 연결되어 있기 때문에 HIGH, LOW 신호를 보냅니다. 물체가 감지되면 LOW 신호를, 감지되지 않으면 HIGH 신호를 보냅니다.

#### ○ 적외선 센서를 이용한 LED ON/OFF

##### ① 연결

아두이노	적외선센서	LED 바
디지털 핀 7	OUT	-
디지털 핀 5	-	(+)
5V	VCC	-
GND	GND	(-)



<적외선 센서 배선 연결도>

#### ② 스케치 작성 및 업로드

```
int irPin = 7; // 디지털 핀 7번을 irPin로 설정
int ledPin = 5; // 디지털 핀 5번을 ledPin로 설정

void setup() {
  Serial.begin(9600); // 시리얼 통신을 위한 통신속도 설정
  pinMode(irPin, INPUT); // irPin 핀 입력 설정
  pinMode(ledPin, OUTPUT); // ledPin 핀 출력 설정
}

void loop() {
  int ir_data = digitalRead(irPin); // ir_data에 값 저장

  Serial.print("감지 체크 : ");
  Serial.println(ir_data);

  if(ir_data == HIGH) { // 장애물 미확인시
    Serial.println("감지된 물체가 없습니다.");
    digitalWrite(ledPin, LOW);
  } else { // 장애물 확인시
    Serial.println("물체가 감지되었습니다.");
    digitalWrite(ledPin, HIGH);
  }
  delay(500);
}
```

적외선 센서에 의해 물체가 감지되면 LED바의 불빛이 켜지고 감지되지 않으면 LED바의 불이 꺼지는 것을 확인할 수 있습니다.

## ▣ 스마트 미니시티

### ◇ 스마트 미니시티

#### 1. 스마트 미니시티란?

##### ○ 단계별 실습

- 기초 H/W 실습
- 개인용 Smart Mini City 실습
- 4인 협업 - 팀 프로젝트 실습
- 8인 협업 - 팀 프로젝트 실습

##### ○ 협업 효과

- 협업 프로젝트(2명이상) 가능
- 다양한 협업 시나리오에 따른 차별화된 스마트 미니시티 동작 가능
- 사용자 또는 팀별의 다양한 응용

#### 2. 재료 배치도

**재료 배치도**

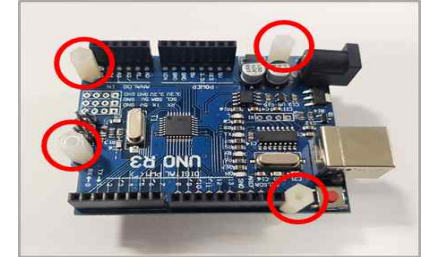
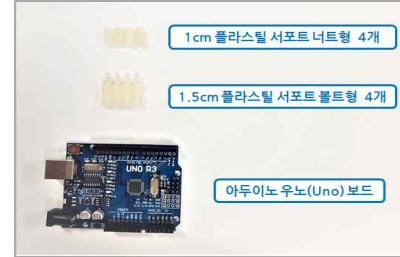
\* 속 패드의 위치와 동일해요 배치도를 보면서 재료를 찾아주세요.

번호	이름	수량	5	2	10	1	1	1	1	1	1
1	서보 모터	1	6	모터 고정판	1	11	직접산 센서	1	16	점퍼선 Set	1
2	스텝 모터	1	7	블루투스 모듈	1	12	아두이노 우노 보드	1	17	LED 배	1
3	모터 드라이버	1	8	5v 릴레이	1	13	USB 케이블	1	18	가방용	6
4	접시니서	2	9	초도 센서	1	14	연장 케이블	1	19	수축 튜브	1

## ◇ 스마트 미니시티 조립

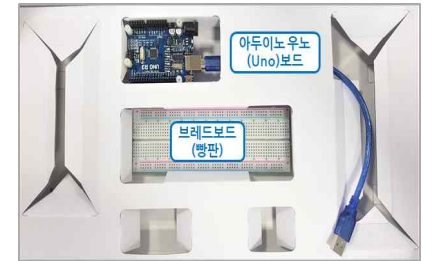
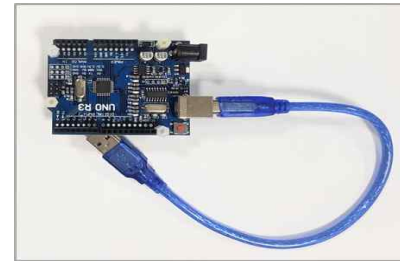
### 1. 아두이노 우노(Uno)보드 & 브레드보드(빵판) 연결

#### ○ 아두이노 우노(Uno)보드



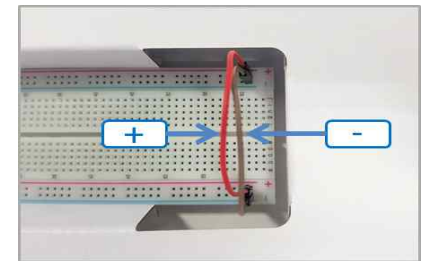
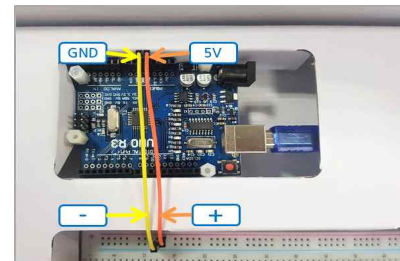
① 플라스틱 서포트 너트형 4개, 볼트형 4개, 아두이노 우노(Uno)보드입니다.

② 볼트형 플라스틱 서포트를 아두이노 우노보드의 구멍에 아래에서 꽂은 다음 너트형 플라스틱 서포트를 보드 위에서 조여 고정합니다.



③ 아두이노 우노(Uno)보드에 USB 케이블을 연결합니다.

④ 사진과 같이 아두이노 우노(Uno)보드와 브레드보드를 박스 안쪽에 배치해주세요.



⑤ 10cm 수/수 점퍼선 2개를 사용하여 아두이노 우노(Uno)보드의 GND, 5V핀과 브레드보드의 (+), (-)버스를 연결 합니다.

⑥ 사진과 같이 브레드보드의 반대쪽 (+), (-) 버스에도 10cm 수/수 점퍼선 2개를 사용하여 확장시켜 줍니다.

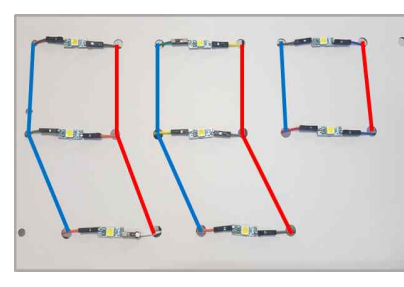
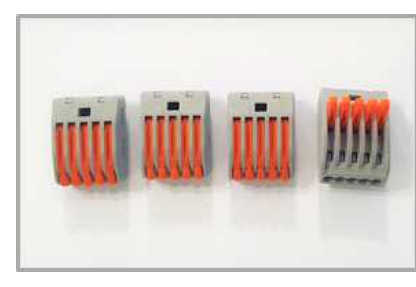
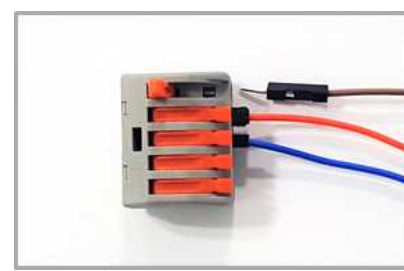

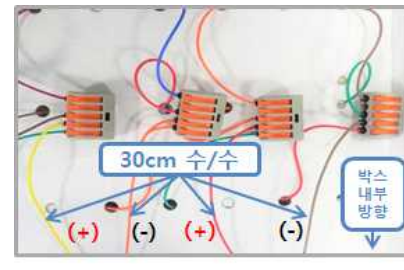



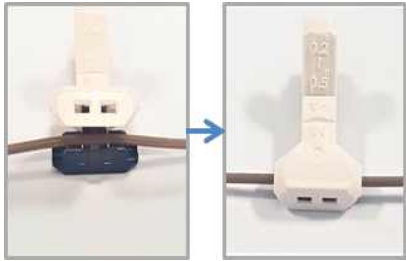
## 2. LED 조립 & 동작 테스트

### ○ LED-점퍼선 연결 및 박스 고정

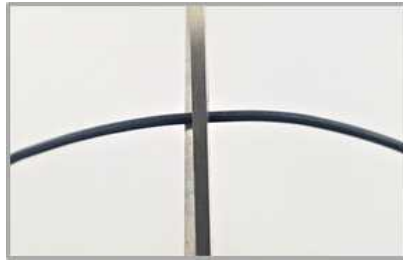
	
<p>① 그림과 같이 노란부분이 중심에 오도록 LED바를 가위로 잘라주세요. *이때 동그란 구멍이 잘리지 않도록 유의해주세요.</p>	<p>② ①에서 자른 LED바 8개와 점퍼선 30cm 수/수 6개, 10cm 수/수 14개를 준비해주세요.</p>
	
<p>③ LED바 뒷면의 양면테이프를 떼 후 7개는 10cm 수/수 점퍼선으로 1개는 30cm 수/수 점퍼선으로 (+), (-)에 끼워주세요. *끼울 때 손을 찌르지 않도록 조심하세요.</p>	<p>④ ③에서 LED바에 넣어준 점퍼선을 사진과 같이 바깥방향으로 구부려 주세요. * 옆에서 봤을 때 위의 그림처럼 점퍼선의 검은 부분이 LED바 밖으로 나가게 해주세요.</p>
	
<p>⑤ ③,④ 과정을 반복하여 10cm점퍼선을 연결한 LED바 7개와 30cm 점퍼선을 연결한 LED바 1개를 만들어주세요.</p>	<p>⑥ LED 부분이 가운데 구멍에 위치하여 붙이고 (+)에 연결된 점퍼선을 LED 중심으로 오른쪽 구멍에 꽂아주세요. *연결된 점퍼선은 양쪽 구멍에 넣어주면 됩니다.</p>

### ○ 점퍼선-커넥터연결

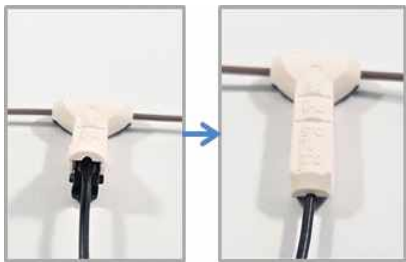
	
<p>① 사진에 표시한 것과 같이 LED바와 연결한 곳의 (-)줄은 파란색, (+)줄은 빨간색처럼 되어있는지 확인해주세요.</p>	<p>② 꽃음형 커넥터 4개를 준비합니다.</p>
	
<p>③ ①번 사진의 첫 번째 파란색(-)줄의 점퍼선3개를 첫 번째 꽃음형 커넥터 1개에 끼워서 연결해줍니다.</p>	<p>④ 그 다음 두 번째 꽃음형 커넥터에 빨간색(+)-줄의 점퍼선 3개를 연결해줍니다.</p>
	
<p>⑤ ③,④와 같은 과정을 반복해 꽃음형 커넥터를 이용해서 4줄을 완성해주세요. 그리고 각 커넥터의 구멍 한 개에 30cm 수수 점퍼선을 끼워주세요.</p>	<p>⑥ ⑤까지 완료했다면 끝에 두 줄이 남습니다. 이 두 줄은 T형 커넥터 2개를 이용해 연결합니다.</p>



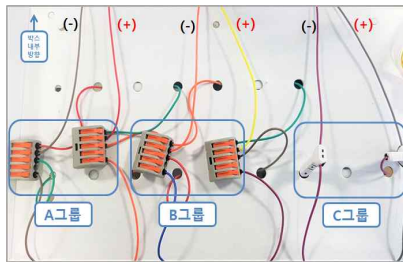
⑦ 먼저 30cm 수/수 점퍼선과 연결한 LED바1개가 있습니다. 점퍼선의 5cm 정도 부분에 T형 커넥터를 연결해주세요.  
\*연결 시 손으로 누르는 것보다 도구를 이용하면 쉽습니다.



⑧ 그 다음 10cm수/수 점퍼선과 연결된 LED바의 점퍼선을 4cm정도 남겨두고 가위로 잘라주세요.



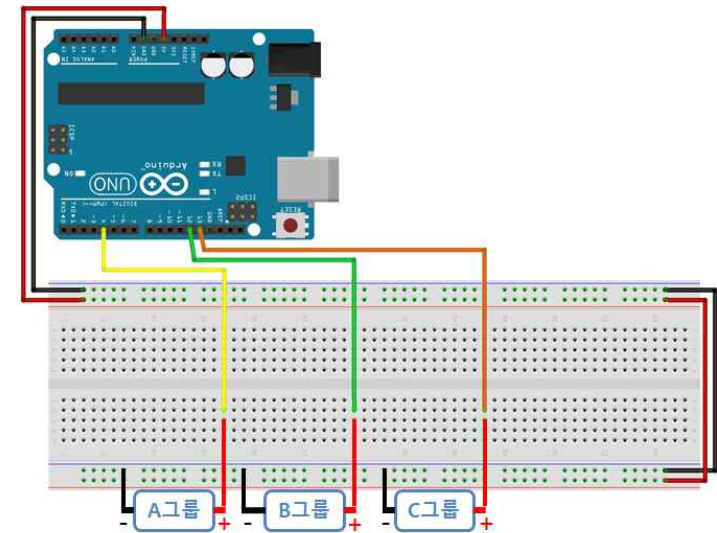
⑨ 자른 점퍼선을 30cm 점퍼선과 연결되어있는 T형 커넥터와 연결해주세요.  
앞에서 준비했던 10cm 암수 점퍼선을 T형 커넥터와 연결된 30cm 점퍼선과 연결하여 연장 해주세요.



⑩ 꽃음힘 커넥터 이용 2그룹 + T형 커넥터 이용 1그룹 총 3개의 LED바 그룹이 만들어졌습니다. 1개의 그룹 당 (+)줄, (-)줄 총 2개의 줄을 연결해 불을 ON/OFF 가능합니다.

## ○ 배선 연결

아두이노	브레드보드	LED바 A, B, C 그룹
디지털 핀 4		A 그룹 (+)
디지털 핀 12		B 그룹 (+)
디지털 핀 13		C 그룹 (+)
5V	(+)	
GND	(-)	A, B, C 그룹 (-)



<건물 LED A, B, C, 그룹 배선 연결도>

- ① 20cm 수/수 점퍼선 3개를 준비합니다.
- ② 20cm 수/수 점퍼선 3개를 아두이노의 4번, 12번, 13번 핀에 연결한 다음 브레드보드에 확장해줍니다.
- ③ 다음과 같이 건물 LED A, B, C 그룹의 선을 브레드보드에 연결합니다.
  - > A그룹 (+)핀 - 4번 핀 확장 / (-)핀 - (-)버스
  - > B그룹 (+)핀 - 12번 핀 확장 / (-)핀 - (-)버스
  - > C그룹 (+)핀 - 13번 핀 확장 / (-)핀 - (-)버스

\*배선 연결이 바르게 되었는지 꼭 확인해주세요.

\*아두이노 보드와 브레드보드의 전원 연결과 전원 버스 확장은 <아두이노 우노(Uno)보드 & 브레드보드 (빵판) 연결> 과정을 참고하세요.

○ 동작 테스트 소스

```

/* 건물LED 동작 테스트 */

int buildingA = 4; // 건물 A그룹
int buildingB = 13; // 건물 B그룹
int buildingC = 12; // 건물 C그룹

void setup() {
  pinMode(buildingA, OUTPUT);
  pinMode(buildingB, OUTPUT);
  pinMode(buildingC, OUTPUT);
}

void loop() {
  digitalWrite(buildingA, HIGH);
  digitalWrite(buildingB, HIGH);
  digitalWrite(buildingC, HIGH);
  delay(2000);

  digitalWrite(buildingA, LOW);
  digitalWrite(buildingB, LOW);
  digitalWrite(buildingC, LOW);
  delay(2000);
}

```

3. 카페 조립 & 동작 테스트

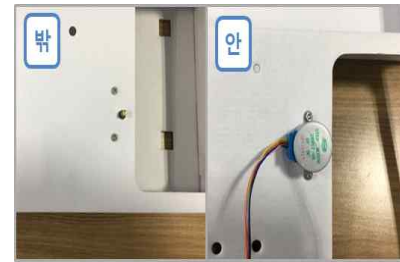
○ 스텝모터 박스고정



① 스텝모터, 접시나사, 너트입니다.



② 사진과 같이 박스 속 뚜껑에 표시한 위치에 스텝모터를 고정 시킵니다.

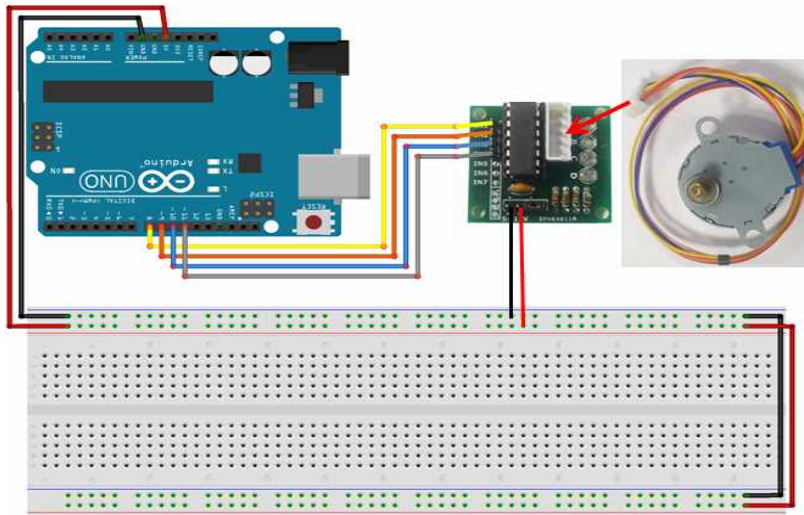


③ 스텝모터를 접시나사와 너트를 이용해 고정시켜 줍니다.

\*카페 건물은 6. 건물 조립 과정 진행 후 조립하시면 됩니다.

○ 배선 연결

아두이노	브레드보드	모터드라이브	스텝모터
디지털 핀 8		IN1	
디지털 핀 9		IN2	
디지털 핀 10		IN3	
디지털 핀 11		IN4	
		핀 5개	핀 5개
5V	(+)	(+)	
GND	(-)	(-)	



<스텝모터&모터드라이브 배선 연결도>

- ① 20cm 암/수 점퍼선 4개, 10cm 암/수 점퍼선 2개 준비합니다.
- ② 20cm 암/수 점퍼선 4개를 아두이노와 모터드라이브를 연결합니다.
  - > 아두이노 8번핀 - 모터드라이브 IN1, 아두이노 9번핀 - 모터드라이브 IN2
  - > 아두이노 10번핀 - 모터드라이브 IN3, 아두이노 11번핀 - 모터드라이브 IN4
- ③ 10cm 암/수 점퍼선 2개를 모터드라이브의 (+),(-)핀과 브레드보드의 (+), (-) 버스를 연결합니다.
- ④ 스텝모터 선 끝의 하얀 커넥터를 표시된 방향에 맞게 모터드라이브의 하얀 핀에 연결합니다.

\*배선 연결이 바르게 되었는지 꼭 확인해주세요.  
 \*아두이노 보드와 브레드보드의 전원 연결과 전원 버스 확장은 <아두이노 우노(Uno)보드 & 브레드보드 (빵판) 연결> 과정을 참고하세요.

○ 동작 테스트 소스

```

/* 전원 연결시 스텝 모터 회전 동작 테스트 소스 */

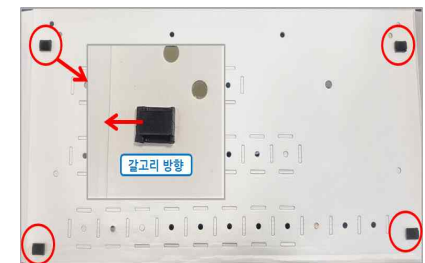
#include <Stepper.h>

const int stepsPerRevolution = 2048;
Stepper myStepper(stepsPerRevolution, 8, 10, 9,11);

void setup() {
  myStepper.setSpeed(14);
}

void loop() {
  myStepper.step(stepsPerRevolution);
}
    
```

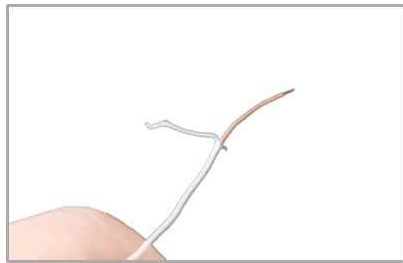
\* 선 정리대 붙이기



- ① 뒷면에 양면테이프가 붙어 있는 선 정리대입니다.
- ② 박스 길 두께 안쪽에 사진과 같은 위치에 4개의 선 정리대를 갈고리(끼우는 부분)가 밖을 향하도록 붙여 줍니다.

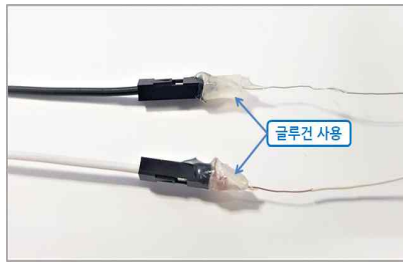
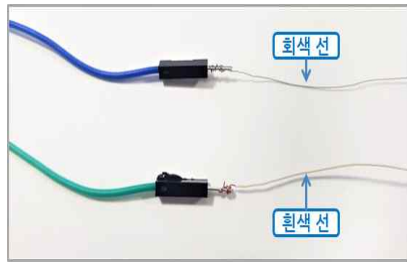
#### 4. 가로등 조립 & 동작 테스트

##### ○ 가로등-점퍼선 연결



① 가로등 모듈 4개, 20cm 수/수 점퍼선 6개, 30cm 수/수 점퍼선 4개, 꽃음형 커넥터 2개  
\*가로등 4개중 2개는 30cm 점퍼선으로 연결하고 나머지 2개는 20cm 점퍼선으로 연결합니다.

② 가로등의 흰색 선의 피복을 손톱으로 눌러 2cm 정도 벗겨주세요.  
\*선이 끊어지지 않도록 주의하세요. 도구를 이용해 선을 눌러주어 피복을 제거할 수 있습니다.



③ 가로등의 흰색 선의 피복을 벗긴 부분을 30cm 수/수 점퍼선 핀에 감아줍니다. 그 다음 회색 선을 다른 30cm 수/수 점퍼선 핀에 감아 줍니다.  
\*선들이 풀리지 않게 꼭 감아주세요.

④ 감은 부분에 글루건을 사용하여 고정해주세요.  
\*글루건을 사용할 때 화상에 주의하세요. 적당량만 사용해주세요.



⑤ 사진과 같이 30cm 수/수 점퍼선과 연결한 가로등 2개 20cm 수/수 점퍼선과 연결한 가로등 2개 총 4개를 만들어 주세요.

##### ○ 박스에 가로등 고정하기



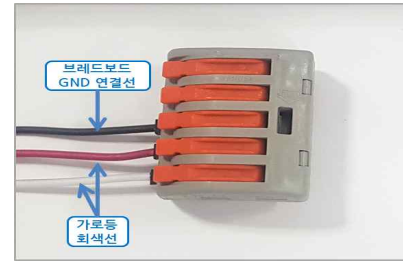
① 총 4개의 가로등을 박스의 가로등 구멍에 선을 하나씩 넣어 끼워줍니다.  
\*30cm와 연결한 가로등을 안쪽구멍에 20cm와 연결한 가로등을 바깥구멍에 끼워주세요.

② 가로등을 모두 알맞게 끼웠다면 글루건을 사용해 고정시켜주세요.  
\*글루건을 사용할 때 화상위험이 있으니 조심하세요.



③ 왼쪽의 선 정리대에는 왼쪽 가로등 2개 오른쪽의 선 정리대에는 오른쪽 가로등 2개 선들을 끼워주세요.

④ 각 가로등의 흰색 선과 연결된 점퍼선에 10cm 암/수 점퍼선을 연결해 길이를 연장해줍니다.  
\*브레드보드에 꽃을 핀입니다.



⑤ 가로등의 회색 선과 연결된 점퍼선을 꽃음형 커넥터에 연결합니다. 그 다음 브레드보드에 (-)(GND)와 연결할 수/수 20cm를 커넥터에 연결합니다.

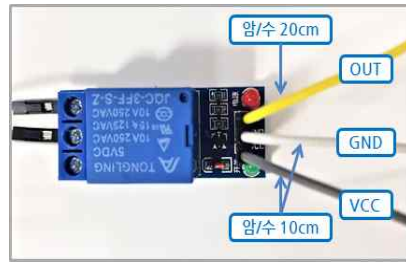
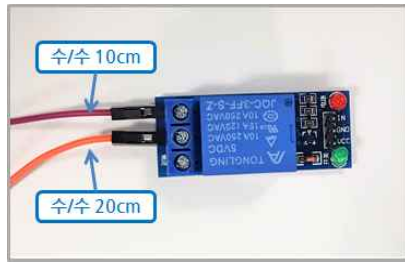
⑥ 왼쪽 가로등 2개와 오른쪽 가로등 2개의 선들을 각각 라벨지로 묶어 표시해 줍니다.

○ 5V릴레이-점퍼선 연결



① 5V릴레이 모듈, 수/수 점퍼선 10cm(1개) 20cm(1개), 암/수 점퍼선 10cm(2개) 20cm(1개)  
\*릴레이 모듈은 가로등의 전원을 제어하기 위해 사용됩니다.

② 드라이버를 사용해 나사를 조절하여 점퍼선 핀을 꽂을 공간을 만들고 꽂은 다음 다시 나사를 조여 고정해주세요.

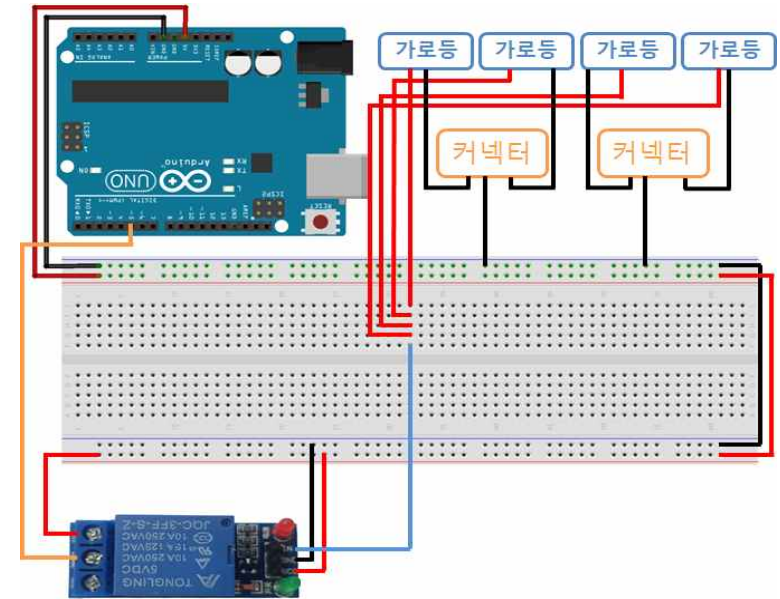


③ 나사를 조절하여 수/수 점퍼선을 연결한 사진입니다.

④ 암/수 점퍼선을 연결한 사진입니다.

○ 배선 연결

아두이노	브레드보드	가로등	릴레이	커넥터
디지털 핀 5			첫 번째 나사구멍	
		(+)	IN	
5V	(+)		VCC, 두 번째 나사구멍	
GND	(-)	(-) : 커넥터연결	(-)	(-)



<가로등&릴레이 배선 연결도>

- 그림과 같이 5v릴레이 오른쪽의 전원 핀(VCC, GND)을 브레드보드의 (+),(-) 버스에 연결합니다.
- 그림과 같이 5v릴레이 IN을 브레드보드에 연결합니다.
- 그림과 같이 5v릴레이 왼쪽의 짧은 점퍼선을 브레드보드의 (+)버스에 연결하고 긴 점퍼선을 아두이노의 5번 핀에 연결합니다.
- 그림과 같이 가로등의 흰색 선(+)은 브레드보드에 연결된 릴레이의 IN핀과 연결합니다. 커넥터의 20cm 수/수 점퍼선을 브레드보드의 (-)버스에 연결합니다.

\*배선 연결이 바르게 되었는지 꼭 확인해주세요.  
\*아두이노 보드와 브레드보드의 전원 연결과 전원 버스 확장은 <아두이노 우노(Uno)보드 & 브레드보드 (빵판) 연결> 과정을 참고하세요.

○ 동작 테스트 소스

```

/* 5V 릴레이 사용하여 가로등 on/off 동작 테스트 소스 */
int streetLightPin = 5;

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

void loop() {
  digitalWrite(streetLightPin, LOW); // 가로등 ON
  delay(2000);
  digitalWrite(streetLightPin, HIGH); // 가로등 OFF
  delay(2000);
}

```

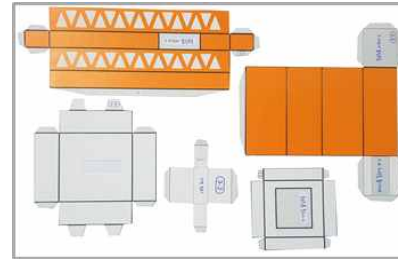
• 5v 릴레이 모듈

=> 릴레이(Relay)란 전류가 흐르면 자력이 발생하여 스위치가 닫힌 상태가 되고 전류가 흐르지 않으면 자력이 없어져 스위치가 열린 상태가 되는 원리를 이용한 모듈입니다. 따라서 적은 양의 전력으로 큰 전력을 제어하거나 하나의 신호로 다수의 전원을 제어할 수 있습니다.

=> 사용할 릴레이 모듈은 Low-Trigger이므로 사용 시 기존의 디지털 신호(HIGH, LOW)가 반대가 됩니다. 예를 들어 HIGH신호를 주면 LED가 꺼지고 LOW신호를 주면 LED가 켜집니다.

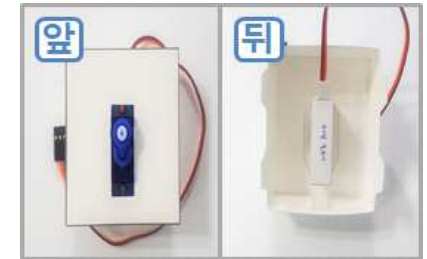
5. 크레인 조립 & 동작 테스트

○ 크레인 건물(3번) 접기



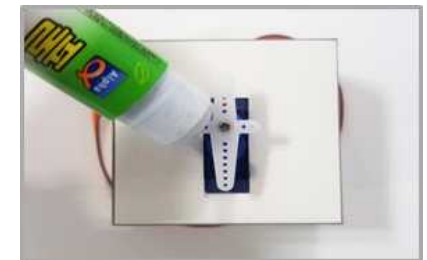
① 도면에서 3-1, 3-2, 3-3, 3-4, 3-5 총 5개 건물을 찢어지지 않게 주의하며 손으로 뜯어 주세요.

② 3-1, 3-2를 접어주세요. 3-2는 날개를 접을 때 사진과 같이 바깥쪽으로 접어주세요. 접은 다음 테이프로 감아주면 더욱 단단하게 고정할 수 있습니다.



③ 3-2의 작은 구멍으로 서보모터의 선을 통과시키고 감싸주세요.  
\*옆면을 테이프로 감싸주시면 편리해요.

④ 사진과 같이 위치에 맞춰 3-1 아래에서 붙여주세요.



⑤ 서보모터에 십자가 모양의 하얀 플라스틱 날개를 끼워주고 같이 들어있던 끈이 뺄수록 작은 나사로 고정시켜 줍니다.  
\*날개의 앞뒤를 구분하여 서보모터에 끼워 줍니다.

⑥ 날개에 풀을 많이 발라 줍니다.

<p>⑦ 3-3을 접어 펼친 날개 위에 붙여주세요.</p>	<p>⑧ 3-3 위에 3-4를 접어 붙여주세요.</p>
<p>⑨ 3-4위에 3-5를 접어 붙여주세요.</p>	<p>⑩ 크레인 완성</p>

○ 크레인-박스 연결

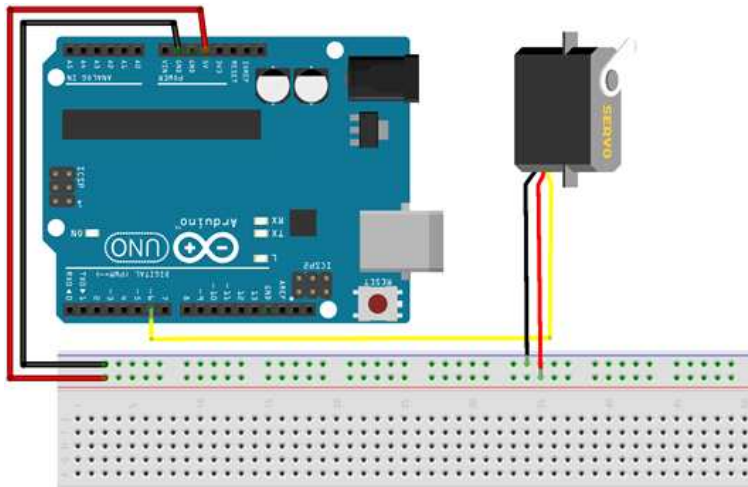
<p>① 위의 사진과 같이 표시된 위치에 크레인을 배치 하세요.</p>	<p>② 서보모터에 연결된 줄을 일자 홈에 끼워 줍니다. *서보모터 선 =&gt; 갈색-GND, 빨강-VCC, 주황-OUT</p>
<p>③ 건물에 달린 지지대를 양쪽의 일자 홈에 끼우고 안쪽으로 접은 다음 테이프로 고정시켜줍니다. * 남은 지지대는 건물 안쪽으로 접어주세요.</p>	<p>④ 선의 끝에 10cm수/수 점퍼선을 끼워 연장해 주 세요. *서보모터 선 =&gt; 갈색-GND, 빨강-VCC, 주황-OUT 센서의 OUT, GND, VCC와 연결했던 선을 기억하 세요.</p>
<p>⑤ 연장된 선을 사진과 같이 선 정리대의 갈고리에 끼워 고정시켜 주세요.</p>	

- 64 -



○ 배선연결

아두이노	SERVO모터 선
디지털 핀 6	주황색
5V	빨간색
GND	갈색



<서보모터 배선 연결도>

① 10cm 암/수 점퍼선 3개를 준비합니다. 서보모터 선의 길이를 준비한 점퍼선으로 연장해줍니다.

② 다음과 같이 연결합니다.

- > 서보모터 갈색(GND)선 - 브레드보드 (-)버스
- > 서보모터 빨강(VCC)선 - 브레드보드 (+)버스
- > 서보모터 주황(OUT)선 - 아두이노 6번 핀

\*배선 연결이 바르게 되었는지 꼭 확인해주세요.

\*아두이노 보드와 브레드보드의 전원 연결과 전원 버스 확장은 <아두이노 우노(Uno)보드 & 브레드보드 (빵판) 연결> 과정을 참고하세요.

○ 동작 테스트 소스

```

/* 크레인 동작 테스트 소스 */

#include <Servo.h>

int servoPin = 6;
Servo myservo;

int pos = 0;

void setup() {
  myservo.attach(servoPin);
}

void loop() {
  for (pos = 0; pos <= 90; pos += 10) {
    myservo.write(pos);
    delay(30);
  }
  delay(3000);

  for (pos = 90; pos >= 0; pos -= 10) {
    myservo.write(pos);
    delay(30);
  }
  delay(3000);
}

```

## 6. 건물 조립 & 동작 테스트

### ○ 조도센서 건물(5번) 접기



① 도면에서 5-1번 건물과 5-5 센서 건물을 찢어지지 않게 주의하며 손으로 뜯어 주세요.



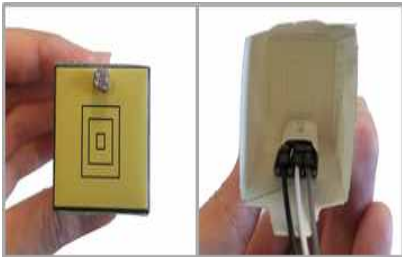
② 조도센서를 5-5로 감싸 폴로 붙여주세요.  
\*센서 바닥면의 튀어나온 핀의 길이가 달라 종이 가 헐렁하거나 모자를 수 있습니다.



③ 조도센서에 30cm암/수 점퍼선 3개를 한 가닥씩 왼쪽부터 AO, GND, VCC에 연결하세요.  
\* DO는 연결하지 마세요.



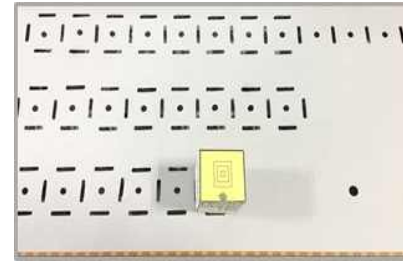
④ 5-1 건물을 풀을 붙여 잘 접어주세요.



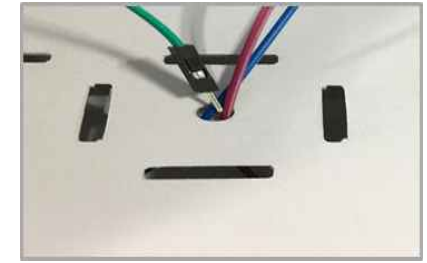
⑤ 5-5 글씨가 적힌 면에 풀칠을 하고 5-1의 건물 안쪽으로 뒷면의 구멍에 센서가 나오도록 끼운 다음 붙여주세요.

\* 센서의 AO, GND, VCC와 연결했던 선을 기억하세요.

### ○ 조도센서 건물 박스고정



① 위의 사진과 같이 표시된 위치에 조도센서 건물을 배치하세요.



② 센서에 연결된 3줄의 선을 하나씩 구멍을 통해 뚜껑 아래로 빼주세요.  
\* 센서의 AO, GND, VCC와 연결했던 선을 기억하세요.



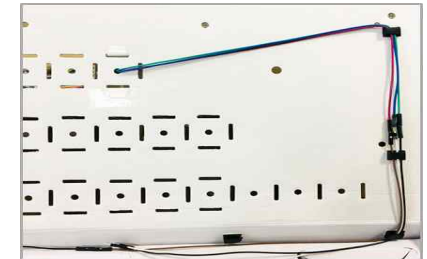
③ 건물에 달린 지지대를 구멍 옆의 일자 홈에 끼워주세요.  
\* 사각형 홈에 끼울 수 없는 위치는 건물 안쪽으로 접어두시면 됩니다.



④ 그 다음 겹 뚜껑 안쪽에서 접어 테이프를 이용해 고정시켜주세요.



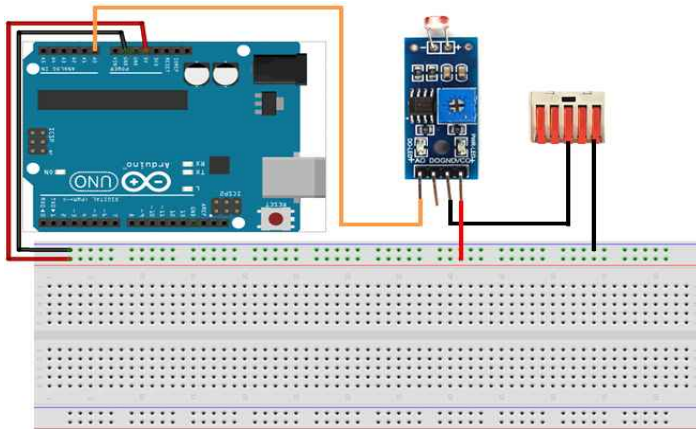
⑤ 뺀 선의 끝에 30cm암/수 점퍼선을 끼워 연장해주세요. 센서의 OUT핀과 연결된 선은 20cm암/수 점퍼선으로 한번 더 연장해주세요.  
\* 센서의 AO, GND, VCC와 연결했던 선을 기억하세요.



⑥ 연장된 선을 사진과 같이 선 정리대의 갈고리에 끼워 고정시켜 주세요.

○ 배선연결 및 동작 테스트

아두이노	조도센서
아날로그 핀 A0	AO(AnalogOut)
5V	VCC
GND	GND



<조도센서 배선 연결도>

① 다음과 같이 연결합니다.

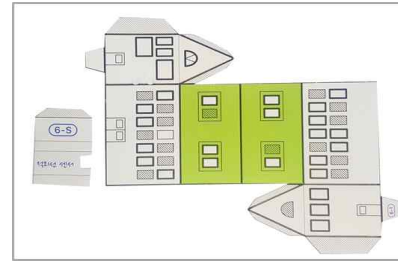
- > 조도센서 AO 점퍼선 - 아두이노 아날로그 핀 A0
- > 조도센서 VCC 점퍼선 - 브레드보드 (+)버스
- > 조도센서 GND 점퍼선 - 오른쪽 꽃음형 커넥터

② 동작테스트소스 -> 기초학습 - 조도센서 예제소스를 참고하세요. (34p)

\*배선 연결이 바르게 되었는지 꼭 확인해주세요.

\*아두이노 보드와 브레드보드의 전원 연결과 전원 버스 확장은 <아두이노 우노(Uno)보드 & 브레드보드 (빵판) 연결> 과정을 참고하세요.

○ 적외선센서 건물(6번) 접기



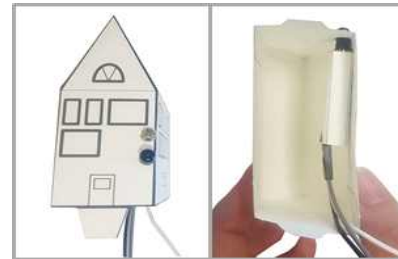
① 도면에서 6-1번 건물과 6-5 센서 건물을 찢어지지 않게 주의하며 손으로 뜯어 주세요.

② 적외선센서를 6-5로 감싸 폴로 붙여주세요.



③ 적외선센서에 30cm암/수 점퍼선 3개를 한 가닥씩 왼쪽부터 OUT, GND, VCC에 연결하세요.

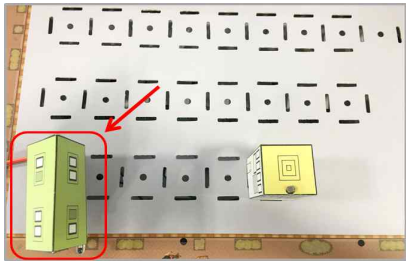
④ 6-1 건물을 폴로 붙여 잘 접어주세요.



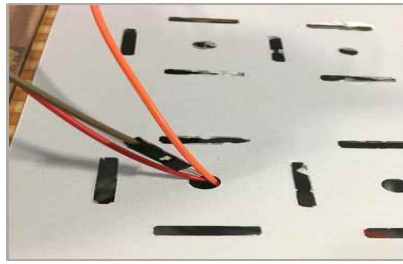
⑤ 6-5 글씨가 적힌 면에 풀칠을 하고 6-1의 건물 안쪽으로 옆면의 구멍에 센서가 나오도록 끼운 다음 붙여주세요.

\* 센서의 OUT, GND, VCC와 연결했던 선을 기억하세요.

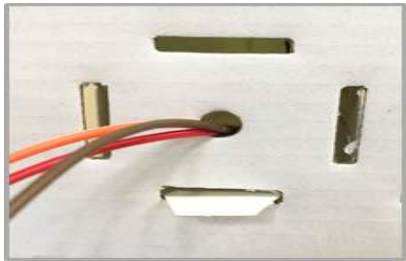
○ 적외선센서 건물 박스고정



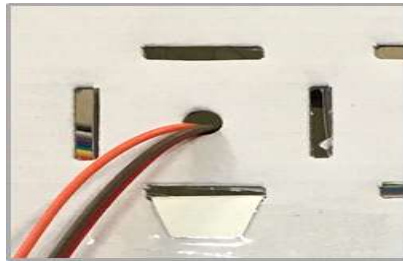
① 위의 사진과 같이 표시된 위치에 적외선센서 건물을 배치하세요.



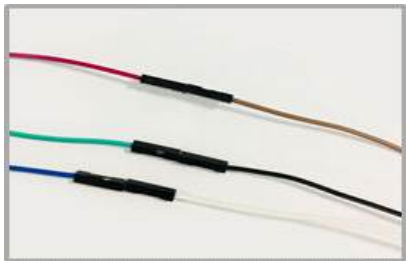
② 센서에 연결된 3줄의 선을 구멍을 통해 뚜껑 아래로 빼주세요.  
\* 센서의 OUT, GND, VCC와 연결했던 선을 기억하세요.



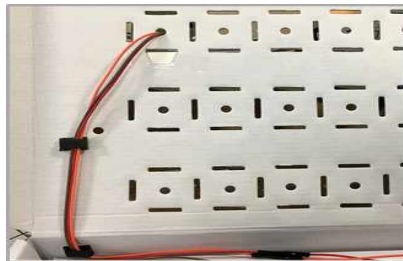
③ 건물에 달린 지지대를 구멍 옆의 일자 홈에 끼워주세요.  
\* 사각형 홈에 끼울 수 없는 위치는 건물 안쪽으로 접어두시면 됩니다.



④ 그 다음 겉 뚜껑 안쪽에서 접어 테이프를 이용해 붙여주세요.



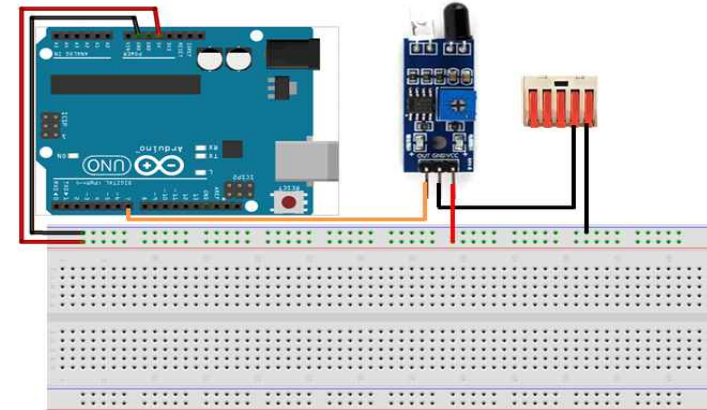
⑤ 뺀 선의 끝에 20cm암/수 점퍼선을 끼워 연장해주세요.  
\* 센서의 OUT, GND, VCC와 연결했던 선을 기억하세요.



⑥ 연장된 선을 사진과 같이 선 정리대의 갈고리에 끼워 고정시켜주세요.

○ 배선연결 및 동작 테스트

아두이노	적외선센서
디지털 핀 7	OUT
5V	VCC
GND	GND



<적외선센서 배선 연결도>

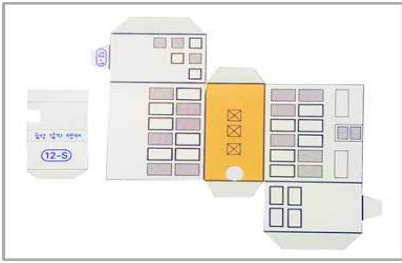
① 다음과 같이 연결합니다.

- > 적외선센서 OUT 점퍼선 - 아두이노 디지털 핀 7
- > 적외선센서 VCC 점퍼선 - 브레드보드 (+)버스
- > 적외선센서 GND 점퍼선 - 왼쪽 꽃음형 커넥터

② 동작테스트소스 -> 기초학습 - 적외선센서 예제소스로 테스트 하면 됩니다. (45p)

\*배선 연결이 바르게 되었는지 꼭 확인해주세요.  
\*아두이노 보드와 브레드보드의 전원 연결과 전원 버스 확장은 <아두이노 우노(Uno)보드 & 브레드보드 (빵판) 연결> 과정을 참고하세요.

○ 사운드센서 건물(12번) 접기



① 도면에서 12-1번 건물과 12-5 센서 건물을 찢어지지 않게 주의하며 손으로 뜯어 주세요.



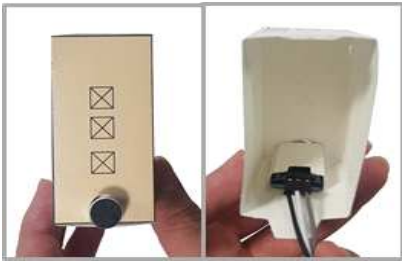
② 사운드센서를 12-5로 감싸 풀로 붙여주세요.



③ 사운드센서에 30cm암/수 점퍼선 3개를 한 가닥씩 왼쪽부터 OUT, GND, VCC에 연결하세요.



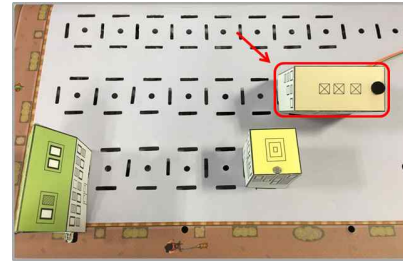
④ 12-1 건물을 풀을 붙여 잘 접어주세요.



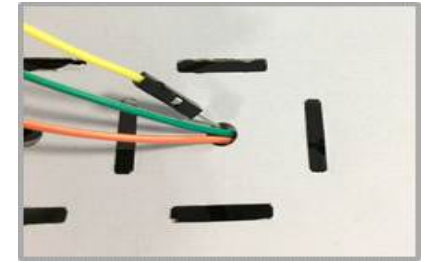
⑤ 12-5 글씨가 적힌 면에 풀칠을 하고 12-1의 건물 안쪽으로 뒷면의 구멍에 센서가 나오도록 끼운 다음 붙여주세요.

\* 센서의 OUT, GND, VCC와 연결했던 선을 기억하세요.

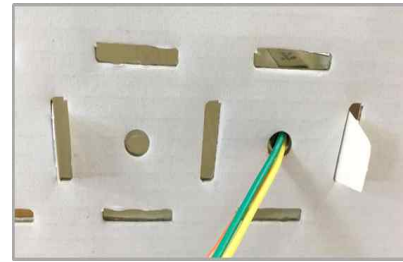
○ 사운드센서 건물 박스고정



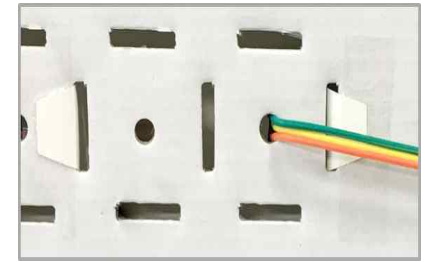
① 위의 사진과 같이 표시된 위치에 사운드센서 건물을 배치하세요.



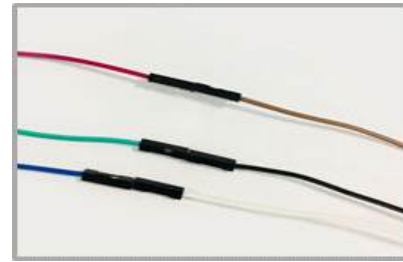
② 센서에 연결된 3줄의 선을 구멍을 통해 뚜껑 아래로 빼주세요.  
\* 센서의 OUT, GND, VCC와 연결했던 선을 기억하세요.



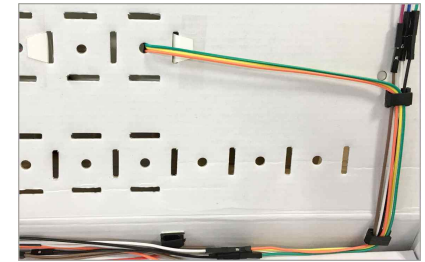
③ 건물에 달린 지지대를 구멍 옆의 일자 홈에 끼워주세요.  
\* 사각형 홈에 끼울 수 없는 위치는 건물 안쪽으로 접어두시면 됩니다.



④ 그 다음 겹 뚜껑 안쪽에서 접어 테이프를 이용해 붙여주세요.



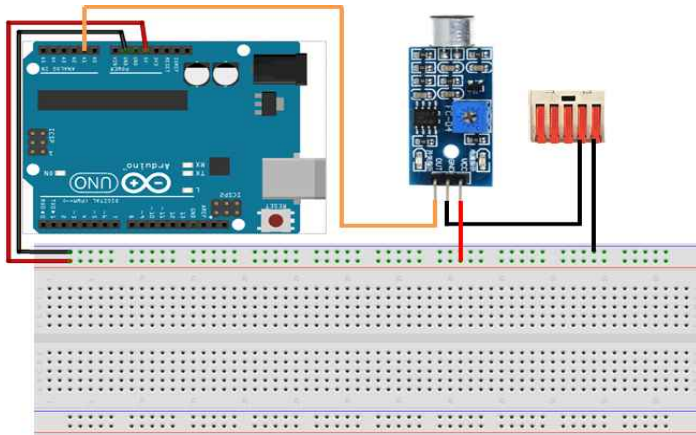
⑤ 뺀 선의 끝에 20cm암/수 점퍼선을 끼워 연장해주세요.  
\* 센서의 OUT, GND, VCC와 연결했던 선을 기억하세요.



⑥ 연장된 선을 사진과 같이 선 정리대의 갈고리에 끼워 고정시켜 주세요.

○ 배선연결 및 동작 테스트

아두이노	사운드센서
아날로그 핀 A1	OUT
5V	VCC
GND	GND



<사운드센서 배선 연결도>

① 다음과 같이 연결합니다.

- > 사운드센서 OUT 점퍼선 - 아두이노 아날로그 핀 A1
- > 사운드센서 VCC 점퍼선 - 브레드보드 (+)버스
- > 사운드센서 GND 점퍼선 - 오른쪽 꽃음형 커넥터

④ 동작테스트소스 -> 기초학습 - 사운드센서 예제소스로 테스트 하면 됩니다. (40p)

\*배선 연결이 바르게 되었는지 꼭 확인해주세요.  
\*아두이노 보드와 브레드보드의 전원 연결과 전원 버스 확장은 <아두이노 우노(Uno)보드 & 브레드보드 (빵판) 연결> 과정을 참고하세요.

○ 일반건물 접기 (완성 사진)



1번 건물



2번 건물



4번 건물



7번 건물



8번 건물



9번 건물

10번 건물	11번 건물
13번 건물	14번 건물

○ 일반건물 박스고정

<p>① 박스의 원하는 위치에 건물을 꽂아줍니다. *LED바 불빛이 나오는 구멍위에 건물을 위치해주세요.</p>	<p>② 박스안쪽에서 껏은 곳을 접고 테이프를 이용해 고정합니다. *다른 일반 건물들도 원하는 위치에 조립해주세요.</p>
<p>③ 카페도로(건물14-4)의 구멍을 모터에 끼우고 동그란 나무원판을 끼워 고정시켜주세요. *나무원판이 종이와 닿는 면에 풀칠을 하면 헛들지 않습니다.</p>	<p>④ 사진과 같이 카페와 터널을 만들어 주세요.</p>
<p>⑤ 터널의 한쪽을 박스에 붙여주고 다른 한쪽을 작은 원판에 붙여 줍니다. 그 다음 카페 건물을 나무원판을 덮도록 붙여주세요.</p>	<p>⑥ 앞의 과정과 같은 방법으로 자신이 원하는 위치에 건물들을 배치해 미니시터를 만들어 주세요.</p>

## ◇ 센서 응용

### 1. 조도센서 응용

#### ① 건물 LED 제어

- 동작 시나리오

=> 조도센서 값이 500보다 크면 건물LED A, B, C 세 그룹의 불이 동시에 ON, 500 보다 작으면 건물 LED A, B, C 세 그룹의 불이 동시에 OFF

- 스케치 소스

```
/* ■ 응용 단계 - 센서 활용
● 조도센서
① 건물LED ON/OFF */

int cdsPin = A0; // 조도센서
int buildingA = 4;
int buildingB = 12;
int buildingC = 13;

void setup() {
  Serial.begin(9600);
  pinMode(cdsPin, INPUT);
  pinMode(buildingA, OUTPUT);
  pinMode(buildingB, OUTPUT);
  pinMode(buildingC, OUTPUT);
}

void loop() {
  int cds_data = analogRead(cdsPin);
  Serial.println(cds_data);
  delay(100); // 0.1초 대기

  if(cds_data >= 500) {
    digitalWrite(buildingA, HIGH);
    digitalWrite(buildingB, HIGH);
    digitalWrite(buildingC, HIGH);
  } else {
    digitalWrite(buildingA, LOW);
    digitalWrite(buildingB, LOW);
    digitalWrite(buildingC, LOW);
  }
}
```

#### ② 카페 회전 제어

- 동작 시나리오

=> 조도센서 값이 500보다 크면 카페 회전

- 스케치 소스

```
/* ■ 응용 단계 - 센서 활용
● 조도센서
② 카페 회전 */

#include <Stepper.h>
int cdsPin = A0; // 조도 센서

const int stepsPerRevolution = 2048;
Stepper myStepper(stepsPerRevolution, 8, 10, 9,11);

void setup() {
  Serial.begin(9600);
  pinMode(cdsPin, INPUT);
  myStepper.setSpeed(14);
}

void loop() {
  int cds_data = analogRead(cdsPin);
  Serial.println(cds_data);
  delay(100); // 0.1초

  if(cds_data >= 500) {
    myStepper.step(stepsPerRevolution);
  }
}
```



### ③ 가로등 제어

- 동작 시나리오

=> 조도센서 값이 500보다 크면 가로등 ON. 그렇지 않으면 OFF

- 스케치 소스

```
/* ■ 응용 단계 - 센서 활용
● 조도센서
③ 가로등 ON/OFF */

int cdsPin = A0; // 조도센서
int streetLightPin = 5;

void setup() {
  Serial.begin(9600);
  pinMode(cdsPin, INPUT);
  pinMode(streetLightPin, OUTPUT);
}

void loop() {
  int cds_data = analogRead(cdsPin);
  Serial.println(cds_data);
  delay(100); // 0.1초

  if(cds_data >= 500) {
    digitalWrite(streetLightPin, LOW);
  } else {
    digitalWrite(streetLightPin, HIGH);
  }
}
```

### ④ 크레인 회전 제어

- 동작 시나리오

=> 조도센서 값이 500보다 크면 크레인 90도 회전, 500보다 작으면 제자리로 회전

- 스케치 소스

```
/* ■ 응용 단계 - 센서 활용
● 조도센서
④ 크레인 회전 */

#include <Servo.h>

int cdsPin = A0; // 조도센서
int servoPin = 6;
Servo myservo;

void setup() {
  Serial.begin(9600);
  pinMode(cdsPin, INPUT);
  myservo.attach(servoPin);
}

void loop() {
  int cds_data = analogRead(cdsPin);
  Serial.println(cds_data);
  delay(100); // 0.1초

  if(cds_data >= 500) {
    myservo.write(90);
  } else {
    myservo.write(0);
  }
}
```

## 2. 사운드센서 응용

### ① 건물 LED 제어

- 동작 시나리오

=> 사운드센서 값이 500보다 크면 건물LED A, B, C 세 그룹의 불이 3초 동안 ON, 500보다 작으면 건물LED A, B, C 세 그룹의 모든 불이 OFF

- 스케치 소스

```
/* ■ 응용 단계 - 센서 활용
   ● 사운드센서
   ① 건물LED ON/OFF */
int soundPin = A1; // 사운드 센서
int buildingA = 4;
int buildingB = 12;
int buildingC = 13;

void setup() {
  Serial.begin(9600);
  pinMode(soundPin, INPUT);
  pinMode(buildingA, OUTPUT);
  pinMode(buildingB, OUTPUT);
  pinMode(buildingC, OUTPUT);
}

void loop() {
  int sound_data = analogRead(soundPin);
  Serial.println(sound_data);
  delay(100); // 0.1초

  if(sound_data >= 500) {
    digitalWrite(buildingA, HIGH);
    digitalWrite(buildingB, HIGH);
    digitalWrite(buildingC, HIGH);
    delay(3000);
  } else {
    digitalWrite(buildingA, LOW);
    digitalWrite(buildingB, LOW);
    digitalWrite(buildingC, LOW);
  }
}
```

### ② 카페 회전 제어

- 동작 시나리오

=> 사운드센서 값이 500보다 크면 카페 회전

- 스케치 소스

```
/* ■ 응용 단계 - 센서 활용
   ● 사운드센서
   ② 카페 회전 */
#include <Stepper.h>
int soundPin = A1;
const int stepsPerRevolution = 2048;
Stepper myStepper(stepsPerRevolution, 8, 10, 9,11);

void setup() {
  Serial.begin(9600);
  pinMode(soundPin, INPUT);
  myStepper.setSpeed(14);
}

void loop() {
  int sound_data = analogRead(soundPin);
  Serial.println(sound_data);
  delay(100); // 0.1초

  if(sound_data >= 500) {
    myStepper.step(stepsPerRevolution);
  }
}
```

### ③ 가로등 제어

- 동작 시나리오

=> 사운드센서 값이 500보다 크면 가로등 3초 동안 ON, 그렇지 않으면 OFF

- 스케치 소스

```
/* ■ 응용 단계 - 센서 활용
● 사운드센서
③ 가로등 ON/OFF */

int soundPin = A1; // 사운드 센서
int streetLightPin = 5;

void setup() {
  Serial.begin(9600);
  pinMode(soundPin, INPUT);
  pinMode(streetLightPin, OUTPUT);
}

void loop() {
  int sound_data = analogRead(soundPin);
  Serial.println(sound_data);
  delay(100);

  if(sound_data >= 500) {
    digitalWrite(streetLightPin, LOW);
    delay(3000);
  } else {
    digitalWrite(streetLightPin, HIGH);
  }
}
```

### ④ 크레인 회전

- 동작 시나리오

=> 사운드센서 값이 500보다 크면 크레인 90도 회전하고 3초 후 제자리로 회전

- 스케치 소스

```
/* ■ 응용 단계 - 센서 활용
● 사운드센서
④ 크레인 회전 */
#include <Servo.h>

int soundPin = A1;
int servoPin = 6;
Servo myservo;

void setup() {
  myservo.attach(servoPin);
}

void loop() {
  int sound_data = analogRead(soundPin);
  Serial.println(sound_data);
  delay(100);

  if(sound_data >= 500) {
    myservo.write(90);
    delay(3000);
    myservo.write(0);
    delay(3000);
  }
}
```

### 3. 적외선센서 응용

#### ① 건물 LED 제어

- 동작 시나리오

=> 물체가 감지되면 건물 LED A, B, C 세 그룹 모두 ON, 감지되지 않으면 모두 OFF

- 스케치 소스

```
/* ■ 응용 단계 - 센서 활용
● 적외선센서
① 건물LED ON/OFF */

int irPin = 7;
int buildingA = 4;
int buildingB = 12;
int buildingC = 13;

void setup() {
  Serial.begin(9600);
  pinMode(irPin, INPUT);
  pinMode(buildingA, OUTPUT);
  pinMode(buildingB, OUTPUT);
  pinMode(buildingC, OUTPUT);
}

void loop() {
  int ir_data = digitalRead(irPin);
  Serial.println(ir_data);
  delay(100);

  if(ir_data == LOW) {
    digitalWrite(buildingA, HIGH);
    digitalWrite(buildingB, HIGH);
    digitalWrite(buildingC, HIGH);
  } else {
    digitalWrite(buildingA, LOW);
    digitalWrite(buildingB, LOW);
    digitalWrite(buildingC, LOW);
  }
}
```

#### ② 카페 회전 제어

- 동작 시나리오

=> 물체가 감지되면 카페 회전

- 스케치 소스

```
/* ■ 응용 단계 - 센서 활용
● 적외선센서
② 카페 회전 */

#include <Stepper.h>
int irPin = 7;
const int stepsPerRevolution = 2048;
Stepper myStepper(stepsPerRevolution, 8, 10, 9,11);

void setup() {
  Serial.begin(9600);
  pinMode(irPin, INPUT);
  myStepper.setSpeed(14);
}

void loop() {
  int ir_data = digitalRead(irPin);
  Serial.println(ir_data);
  delay(100);

  if(ir_data == LOW) {
    myStepper.step(stepsPerRevolution);
  }
}
```

### ③ 가로등 제어

- 동작 시나리오

=> 물체가 감지되면 가로등 ON, 감지되지 않으면 OFF

- 스케치 소스

```
/* ■ 응용 단계 - 센서 활용
● 적외선센서
③ 가로등 ON/OFF */

int streetLightPin = 5;
int irPin = 7;

void setup() {
  Serial.begin(9600);
  pinMode(streetLightPin, OUTPUT);
  pinMode(irPin, INPUT);
}

void loop() {
  int ir_data = digitalRead(irPin);
  Serial.println(ir_data);
  delay(100);

  if(ir_data == LOW) {
    digitalWrite(streetLightPin, LOW);
  } else {
    digitalWrite(streetLightPin, HIGH);
  }
}
```

### ④ 크레인 회전 제어

- 동작 시나리오

=> 물체가 감지되면 크레인 90도 회전, 감지되지 않으면 제자리로 회전

- 스케치 소스

```
/* ■ 응용 단계 - 센서 활용
● 적외선센서
④ 크레인 회전 */

#include <Servo.h>

int irPin = 7;
int servoPin = 6;
Servo myservo;

void setup() {
  Serial.begin(9600);
  pinMode(irPin, INPUT);
  myservo.attach(servoPin);
}

void loop() {
  int ir_data = digitalRead(irPin);
  Serial.println(ir_data);
  delay(100);

  if(ir_data == LOW) {
    myservo.write(90);
  } else {
    myservo.write(0);
  }
}
```

## ◇ 미니시터 - 독립

### 1. 기본 예제 소스

#### ○ 기본 예제 소스 ①

##### • 동작 시나리오

[조도센서] => 어두워지면 모든 건물 LED ON -> 1초 후 가로등 ON -> 1초 후 카페 한 바퀴 회전 -> 1초 후 크레인 90도 회전

##### • 스케치 소스

```
/* ■ 나만의 미니시터 만들기 - 기본
   기초 예제 소스 ① */

#include <Servo.h>    // 서보모터 라이브러리
#include <Stepper.h>  // 스텝모터 라이브러리

// 2048은 360도를 의미, 스텝모터는 한번 동작할 때마다 360도 움직이게 된다.
const int stepsPerRevolution = 2048;
// myStepper 객체생성 & 회전각과 핀 번호 8,9,10,11 설정
Stepper myStepper(stepsPerRevolution, 8, 10, 9,11);
// myservo 객체생성
Servo myservo;

int cdsPin = A0;      // 조도센서
int streetLightPin = 5; // 가로등
int buildingA = 4;    // 건물 A그룹
int buildingB = 12;   // 건물 B그룹
int buildingC = 13;   // 건물 C그룹

void setup() {
  Serial.begin(9600); // 시리얼 통신을 위한 통신속도 설정
  pinMode(cdsPin, INPUT); // cdsPin핀 입력핀 설정
  pinMode(streetLightPin, OUTPUT); // 가로등 출력핀 설정
  pinMode(buildingA, OUTPUT); // 건물 A그룹 출력핀 설정
  pinMode(buildingB, OUTPUT); // 건물 B그룹 출력핀 설정
  pinMode(buildingC, OUTPUT); // 건물 C그룹 출력핀 설정
  myStepper.setSpeed(14); // STEP모터 회전속도
  myservo.attach(6); // 6번핀 서보모터 연결설정
}
```

```
void loop() {
  int cds_data = analogRead(cdsPin); // cdsPin의 데이터값 cds_data에 저장
  if(cds_data >= 500) { // cds_data가 500보다 크면
    lightOn(); // 함수 lightOn() 실행
  } else { // cds_data가 500보다 작으면
    lightOff(); // 함수 lightOff() 실행
  }
  delay(10);
}

void lightOn() {
  digitalWrite(buildingA, HIGH); // 건물A ON
  digitalWrite(buildingB, HIGH); // 건물B ON
  digitalWrite(buildingC, HIGH); // 건물C ON
  delay(1000); // 대기 1초
  digitalWrite(streetLightPin, LOW); // 가로등 ON
  delay(1000); // 대기 1초
  myStepper.step(stepsPerRevolution); // STEP모터 회전
  delay(1000); // 대기 1초
  myservo.write(90); // 서보모터 90도 회전
}

void lightOff() {
  digitalWrite(buildingA, LOW); // 건물A OFF
  digitalWrite(buildingB, LOW); // 건물B OFF
  digitalWrite(buildingC, LOW); // 건물C OFF
  delay(1000); // 대기 1초
  digitalWrite(streetLightPin, HIGH); // 가로등 OFF
  delay(1000); // 대기 1초
  myservo.write(0); // 서보모터 제자리로 회전
}
```

## ○ 기본 예제 소스 ②

### • 동작 시나리오

[사운드센서] => 소리가 감지되면 카페 두 바퀴 회전 -> 1초 후 모든 건물 LED ON -> 1초 후 모든 건물 LED OFF -> 1초 후 가로등 ON -> 1초 후 가로등 OFF -> 1초 후 크레인 90도 회전 -> 1초 후 크레인 제자리로 회전

### • 스케치 소스

```
/* ■ 나만의 미니시티 만들기 - 기본
   기초 예제 소스 ② */

#include <Servo.h> // 서보모터 라이브러리 포함
#include <Stepper.h> // 스텝모터 라이브러리 포함

// 2048은 360도를 의미하며 스텝모터가 한번 동작할 때마다 360도 움직이게 됩니다.
const int stepsPerRevolution = 2048;
// myStepper 객체 생성과 회전각과 핀 번호 8,9,10,11 설정
Stepper myStepper(stepsPerRevolution, 8, 10, 9,11);
// myservo 객체 생성
Servo myservo;

int soundPin = A1; // 사운드센서
int streetLightPin = 5; // 가로등
int buildingA = 4; // 건물 A그룹
int buildingB = 12; // 건물 B그룹
int buildingC = 13; // 건물 C그룹

void setup() {
  Serial.begin(9600); // 시리얼 통신을 위한 통신속도 설정
  pinMode(soundPin, INPUT); // soundPin핀 입력 설정
  pinMode(streetLightPin, OUTPUT); // 가로등 출력핀 설정
  pinMode(buildingA, OUTPUT); // 건물 A그룹 출력핀 설정
  pinMode(buildingB, OUTPUT); // 건물 B그룹 출력핀 설정
  pinMode(buildingC, OUTPUT); // 건물 C그룹 출력핀 설정
  myStepper.setSpeed(14); // STEP모터 회전속도
  myservo.attach(6); // 6번핀 서보모터 연결설정
}
```

```
void loop() {
  int sound_data = analogRead(soundPin); // soundPin의 데이터값 sound_data에 저장

  if(sound_data >= 500) { // sound_data가 500보다 크면
    Serial.println(sound_data); // sound_data 값 출력
    cafe(); // 함수 cafe() 실행
    delay(1000); // 1초 대기
    buildingLED_ON(); // 함수 buildingLED_ON() 실행
    delay(1000); // 1초 대기
    buildingLED_OFF(); // 함수 buildingLED_OFF() 실행
    delay(1000); // 1초 대기
    digitalWrite(streetLightPin, LOW); // 가로등 ON
    delay(1000); // 1초 대기
    digitalWrite(streetLightPin, HIGH); // 가로등 OFF
    delay(1000); // 1초 대기
    myservo.write(90); // 서보모터 90도 회전
    delay(1000); // 1초 대기
    myservo.write(0); // 서보모터 제자리로 회전
  }
  delay(10);
}

void cafe() { // 카페 2바퀴 회전 함수
  for(int i=0; i<2; i++) {
    myStepper.step(stepsPerRevolution);
    delay(100);
  }
}

void buildingLED_ON() { // 건물 A, B, C그룹 ON
  digitalWrite(buildingA, HIGH); // 건물 A그룹 ON
  digitalWrite(buildingB, HIGH); // 건물 B그룹 ON
  digitalWrite(buildingC, HIGH); // 건물 C그룹 ON
}

void buildingLED_OFF() { // 건물 A, B, C그룹 OFF
  digitalWrite(buildingA, LOW); // 건물 A그룹 OFF
  digitalWrite(buildingB, LOW); // 건물 B그룹 OFF
  digitalWrite(buildingC, LOW); // 건물 C그룹 OFF
}
```

### ○ 기본 예제 소스 ③

#### • 동작 시나리오

[적외선센서] => 물체가 감지되면 가로등 ON -> 1초 후 건물 LED ON -> 3초 후 건물 LED OFF -> 1초 후 가로등 OFF -> 1초 후 카페 두 바퀴 회전 -> 1초 후 크레인 90도 회전 -> 1초 후 건물 LED ON -> 3초 후 건물 LED OFF -> 1초 후 크레인 제자리로 회전

#### • 스케치 소스

```

/* ■ 나만의 미니시티 만들기 - 기본
   기초 예제 소스 ③ */

#include <Servo.h> // 서보모터 라이브러리 포함
#include <Stepper.h> // 스텝모터 라이브러리 포함

// 2048은 360도를 의미, 스텝모터는 한번 동작할 때마다 360도 움직이게 된다.
const int stepsPerRevolution = 2048;
// myStepper 객체 생성과 회전각과 핀 번호 8,9,10,11 설정
Stepper myStepper(stepsPerRevolution, 8, 10, 9,11);
// myservo 객체 생성
Servo myservo;

int irPin = 7; // 적외선 센서
int streetLightPin = 5; // 가로등
int buildingA = 4; // 건물 A그룹
int buildingB = 12; // 건물 B그룹
int buildingC = 13; // 건물 C그룹

void setup() {
  Serial.begin(9600); // 시리얼 통신을 위한 통신속도 설정
  pinMode(irPin, INPUT); // irPin핀 입력핀 설정
  pinMode(streetLightPin, OUTPUT); // 가로등
  pinMode(buildingA, OUTPUT); // 건물 A그룹 출력핀 설정
  pinMode(buildingB, OUTPUT); // 건물 B그룹 출력핀 설정
  pinMode(buildingC, OUTPUT); // 건물 C그룹 출력핀 설정
  myStepper.setSpeed(14); // STEP모터 회전속도
  myservo.attach(6); // 6번핀 서보모터 연결설정
}

```

```

void loop() {
  int ir_data = digitalRead(irPin); // irPin의 데이터값을 ir_data에 저장
  if(ir_data == LOW) { // 물체 감지
    digitalWrite(streetLightPin, LOW); // 가로등 ON
    delay(1000); // 1초 대기
    buildingLED_ON(); // 함수 buildingLED_ON() 실행
    delay(3000); // 3초 대기
    buildingLED_OFF(); // 함수 buildingLED_OFF() 실행
    delay(1000); // 1초 대기
    digitalWrite(streetLightPin, HIGH); // 가로등 OFF
    delay(1000); // 1초 대기
    cafe_ON(); // 함수 cafe_ON() 실행
    delay(1000); // 1초 대기
    myservo.write(90); // 서보모터 90도 회전
    delay(1000); // 1초 대기
    buildingLED_ON(); // 함수 buildingLED_ON() 실행
    delay(3000); // 3초 대기
    buildingLED_OFF(); // 함수 buildingLED_OFF() 실행
    delay(1000); // 1초 대기
    myservo.write(0); // 서보모터 제자리로 회전
    delay(1000); // 1초 대기
  }
}

void buildingLED_ON() { // 건물 A, B, C그룹 ON
  digitalWrite(buildingA, HIGH); // 건물 A그룹 ON
  digitalWrite(buildingB, HIGH); // 건물 B그룹 ON
  digitalWrite(buildingC, HIGH); // 건물 C그룹 ON
}

void buildingLED_OFF() { // 건물 A, B, C그룹 OFF
  digitalWrite(buildingA, LOW); // 건물 A그룹 OFF
  digitalWrite(buildingB, LOW); // 건물 B그룹 OFF
  digitalWrite(buildingC, LOW); // 건물 C그룹 OFF
}

void cafe_ON() { // 카페 2바퀴 회전 함수
  for(int i=0; i<2 i++) {
    myStepper.step(stepsPerRevolution);
  }
}

```



## 2. 응용 예제 소스

### ○ 응용 예제 소스 ①

#### • 동작 시나리오

[조도센서] => 어두워지면 건물 A그룹 LED ON -> 2초 후 건물 B그룹 LED ON -> 2초 후 건물 C그룹 LED ON -> 2초 후 건물 LED 깜빡임 -> 5초 후 건물 LED OFF -> 2초 후 가로등 ON -> 1초 후 크레인 90도 회전 -> 1초 후 카페 한 바퀴 회전 -> 1초 후 크레인 제자리로 회전

#### • 스케치 소스

```
/* ■ 나만의 미니시티 만들기 - 기본 응용 예제 소스 ① */
#include <Servo.h> // 서보모터 라이브러리 포함
#include <Stepper.h> // 스텝모터 라이브러리 포함
// 2048은 360도를 의미, 스텝모터는 한번 동작할 때마다 360도 움직이게 된다.
const int stepsPerRevolution = 2048;
// myStepper 객체 생성과 회전각과 핀 번호 8,9,10,11 설정
Stepper myStepper(stepsPerRevolution, 8, 10, 9,11);
// myservo 객체 생성
Servo myservo;

int cdsPin = A0; // 조도센서
int streetLightPin = 5; // 가로등
int buildingA = 4; // 건물 A그룹
int buildingB = 12; // 건물 B그룹
int buildingC = 13; // 건물 C그룹

void setup() {
  Serial.begin(9600); // 시리얼 통신을 위한 통신속도 설정
  pinMode(cdsPin, INPUT); // cdsPin핀 입력핀 설정
  pinMode(streetLightPin, OUTPUT); // 가로등 출력핀 설정
  pinMode(buildingA, OUTPUT); // 건물 A그룹 출력핀 설정
  pinMode(buildingB, OUTPUT); // 건물 B그룹 출력핀 설정
  pinMode(buildingC, OUTPUT); // 건물 C그룹 출력핀 설정
  myStepper.setSpeed(14); // STEP모터 회전속도
  myservo.attach(6); // 6번핀 서보모터 연결설정
}

void loop() {
  int cds_data = analogRead(cdsPin); // cdsPin의 데이터값을 cds_data에 저장
  if(cds_data >= 500) { // cds_data가 500보다 크면
    lightOn(); // 함수 lightOn() 실행
  }
  delay(10);
}
```

```
void lightOn() {
  digitalWrite(buildingA, HIGH); // 건물 A그룹 ON
  delay(2000); // 2초 대기
  digitalWrite(buildingB, HIGH); // 건물 B그룹 ON
  delay(2000); // 2초 대기
  digitalWrite(buildingC, HIGH); // 건물 C그룹 ON
  delay(2000); // 2초 대기
  twinkle(); // 함수 twinkle() 실행
  delay(1000); // 1초 대기
  buildingLED_OFF(); // 함수 buildingLED_OFF() 실행
  delay(2000); // 2초 대기
  digitalWrite(streetLightPin, LOW); // 가로등 ON
  delay(1000); // 1초 대기
  myservo.write(90); // 서보모터 90도 회전
  delay(1000); // 1초 대기
  myStepper.step(stepsPerRevolution); // STEP모터 회전
  delay(1000); // 2초 대기
  myservo.write(0); // 서보모터 제자리로 회전
}

void twinkle() {
  buildingLED_ON();
  delay(1000);
  buildingLED_OFF();
  delay(1000);
  buildingLED_ON();
  delay(1000);
  buildingLED_OFF();
  delay(1000);
  buildingLED_ON();
}

void buildingLED_ON() { // 건물 A, B, C그룹 ON
  digitalWrite(buildingA, HIGH); // 건물 A그룹 ON
  digitalWrite(buildingB, HIGH); // 건물 B그룹 ON
  digitalWrite(buildingC, HIGH); // 건물 C그룹 ON
}

void buildingLED_OFF() { // 건물 A, B, C그룹 OFF
  digitalWrite(buildingA, LOW); // 건물 A그룹 OFF
  digitalWrite(buildingB, LOW); // 건물 B그룹 OFF
  digitalWrite(buildingC, LOW); // 건물 C그룹 OFF
}
```

## ○ 응용 예제 소스 ②

- 동작 시나리오 - (두 개 이상의 센서를 사용할 경우 첫 번째 센서의 시나리오 진행 후 두 번째 센서 시나리오 진행, 동시 불가)

[적외선센서] => 물체가 감지되면 크레인 90도 회전 -> 1초 후 크레인 180도 회전 -> 1초 후 카페 두 바퀴 회전 -> 1초 후 크레인 90도 회전 -> 1초 후 크레인 제자리로 회전

[조도센서] => 어두워지면 가로등 ON -> 1초 후 건물 A 그룹 LED ON -> 1초 후 건물 B 그룹 LED ON -> 1초 후 건물 C 그룹 LED ON -> 1초 후 모든 건물 LED OFF -> 1초 후 모든 건물 LED ON -> 1초 후 모든 건물 LED OFF -> 1초 후 가로 등 OFF

### • 스케치 소스

```

/* ■ 나만의 미니시티 만들기 - 기본
   응용 예제 소스 ② */

#include <Servo.h> // 서보모터 라이브러리
#include <Stepper.h> // 스텝모터 라이브러리

// 2048은 360도를 의미, 스텝모터는 한번 동작할 때마다 360도 움직이게 된다.
const int stepsPerRevolution = 2048;
// myStepper 객체 생성과 회전각과 핀 번호 8,9,10,11 설정
Stepper myStepper(stepsPerRevolution, 8, 10, 9,11);
// myservo 객체 생성
Servo myservo;

int cdsPin = A0; // 조도센서
int irPin = 7; // 적외선 센서
int streetLightPin = 5; // 가로등
int buildingA = 4; // 건물 A그룹
int buildingB = 12; // 건물 B그룹
int buildingC = 13; // 건물 C그룹
int i_state = true; // 적외선센서 분기 변수
int c_state = false; // 조도센서 분기 변수

void setup() {
  Serial.begin(9600); // 시리얼 통신을 위한 통신속도 설정
  pinMode(irPin, INPUT); // irPin핀 입력 설정
  pinMode(cdsPin, INPUT); // cdsPin핀 입력핀 설정
  pinMode(streetLightPin, OUTPUT); // 가로등 출력핀 설정
  pinMode(buildingA, OUTPUT); // 건물 A그룹 출력핀 설정
  pinMode(buildingB, OUTPUT); // 건물 B그룹 출력핀 설정
  pinMode(buildingC, OUTPUT); // 건물 C그룹 출력핀 설정
  myStepper.setSpeed(14); // STEP모터 회전속도
  myservo.attach(6); // 6번핀 서보모터 연결설정
}

```

```

void loop() {
  while(i_state) { // i_state가 true인 동안 실행
    int ir_data = digitalRead(irPin); // irPin의 데이터값을 ir_data에 저장
    if(ir_data == LOW) { // 물체 감지
      Serial.println(ir_data); // ir_data값 출력
      myservo.write(90); // 서보모터 90도 회전
      delay(1000); // 1초 대기
      myservo.write(180); // 서보모터 180도 회전
      delay(1000); // 1초 대기
      cafe_ON(); // 함수 cafe_ON() 실행
      delay(1000); // 1초 대기
      myservo.write(90); // 서보모터 90도 회전
      delay(1000); // 1초 대기
      myservo.write(0); // 서보모터 제자리로 회전
      i_state = false; // i_state값 false
      c_state = true; // c_state값 true
    }
    delay(100);
  }

  while(c_state) { // c_state가 true인 동안 실행
    int cds_data = analogRead(cdsPin); // cdsPin의 데이터값을 cds_data에 저장
    if(cds_data >= 500) { // cds_data가 500보다 크면
      Serial.println(cds_data); // cds_data값 출력
      digitalWrite(streetLightPin, LOW); // 가로등 ON
      delay(1000); // 1초 대기
      digitalWrite(buildingA, HIGH); // 건물 A그룹 ON
      delay(1000); // 1초 대기
      digitalWrite(buildingB, HIGH); // 건물 B그룹 ON
      delay(1000); // 1초 대기
      digitalWrite(buildingC, HIGH); // 건물 C그룹 ON
      delay(1000); // 1초 대기
      buildingLED_OFF(); // 함수 buildingLED_OFF() 실행
      delay(1000); // 1초 대기
      buildingLED_ON(); // 함수 buildingLED_ON() 실행
      delay(3000); // 3초 대기
      buildingLED_OFF(); // 함수 buildingLED_OFF() 실행
      delay(1000); // 1초 대기
      digitalWrite(streetLightPin, HIGH); // 가로등 OFF
      c_state = false; // c_state값 false
      i_state = true; // i_state값 true
    }
    delay(100);
  }
}

```

```

void buildingLED_ON() {           // 건물 A, B, C그룹 ON
    digitalWrite(buildingA, HIGH); // 건물 A그룹 ON
    digitalWrite(buildingB, HIGH); // 건물 B그룹 ON
    digitalWrite(buildingC, HIGH); // 건물 C그룹 ON
}

void buildingLED_OFF() {         // 건물 A, B, C그룹 OFF
    digitalWrite(buildingA, LOW); // 건물 A그룹 OFF
    digitalWrite(buildingB, LOW); // 건물 B그룹 OFF
    digitalWrite(buildingC, LOW); // 건물 C그룹 OFF
}

void cafe_ON() {                 // 카페 2바퀴 회전 함수
    for(int i=0; i<2; i++) {
        myStepper.step(stepsPerRevolution);
    }
}

```

## ○ 응용 예제 소스 ③

### • 동작 시나리오

[사운드센서] => 첫 번째 박수 => 가로등 ON -> 2초 뒤 카페 두 바퀴 회전  
 두 번째 박수 => 크레인 180도 회전 -> 모든 건물 LED ON  
 세 번째 박수 => 가로등 OFF  
 네 번째 박수 => 크레인 제자리로 회전 -> 모든 건물 LED OFF

### • 스케치 소스

```

/* ■ 나만의 미니시티 만들기 - 기본
   응용 예제 소스 ③ */

#include <Servo.h> // 서보모터 라이브러리
#include <Stepper.h> // 스텝모터 라이브러리

// 2048은 360도를 의미, 스텝모터는 한번 동작할 때마다 360도 움직이게 된다.
const int stepsPerRevolution = 2048;
// myStepper 객체 생성과 회전각과 핀 번호 8,9,10,11 설정
Stepper myStepper(stepsPerRevolution, 8, 10, 9,11);
// myservo 객체 생성
Servo myservo;

int soundPin = A1; // 사운드센서
int streetLightPin = 5; // 가로등
int buildingA = 4; // 건물 A그룹
int buildingB = 12; // 건물 B그룹
int buildingC = 13; // 건물 C그룹
int clab_count = 0; // 소리감지카운트

void setup() {
    Serial.begin(9600); // 시리얼 통신을 위한 통신속도 설정
    pinMode(soundPin, INPUT); // soundPin핀 입력 설정
    pinMode(streetLightPin, OUTPUT); // 가로등 출력핀 설정
    pinMode(buildingA, OUTPUT); // 건물 A그룹 출력핀 설정
    pinMode(buildingB, OUTPUT); // 건물 B그룹 출력핀 설정
    pinMode(buildingC, OUTPUT); // 건물 C그룹 출력핀 설정
    myStepper.setSpeed(14); // STEP모터 회전속도
    myservo.attach(6); // 6번핀 서보모터 연결설정
}

```

```

void loop() {
  int sound_data = 0;           // sound_data값 초기화
  sound_data = analogRead(soundPin); // soundPin의 데이터값을 sound_data에 저장
  delay(10);

  if(sound_data >= 500) {      // sound_data가 500보다 크면
    Serial.println(sound_data); // sound_data값 출력
    clab_count = clab_count+1; // 소리감지시 clab_count 1씩 증가
    Serial.println(clab_count); // clab_count값 출력
    if(clab_count == 1){      // clab_count 1일 때
      digitalWrite(streetLightPin, LOW); // 가로등 ON
      delay(2000);           // 2초 대기
      cafe_ON();             // 함수 cafe_ON() 실행
    } else if(clab_count == 2){ // clab_count 2일 때
      myservo.write(180);     // 서보모터 180도 회전
      buildingLED_ON();       // 함수 buildingLED_ON() 실행
    } else if(clab_count == 3){ // clab_count 3일 때
      digitalWrite(streetLightPin, HIGH); // 가로등 OFF
    } else if(clab_count == 4){ // clab_count 4일 때
      myservo.write(0);      // 서보모터 제자리로 회전
      buildingLED_OFF();     // 함수 buildingLED_OFF() 실행
      clab_count = 0;
    }
  }
}

void buildingLED_ON() {      // 건물 A, B, C그룹 ON
  digitalWrite(buildingA, HIGH); // 건물 A그룹 ON
  digitalWrite(buildingB, HIGH); // 건물 B그룹 ON
  digitalWrite(buildingC, HIGH); // 건물 C그룹 ON
}

void buildingLED_OFF() {    // 건물 A, B, C그룹 OFF
  digitalWrite(buildingA, LOW); // 건물 A그룹 OFF
  digitalWrite(buildingB, LOW); // 건물 B그룹 OFF
  digitalWrite(buildingC, LOW); // 건물 C그룹 OFF
}

void cafe_ON() {           // 카페 2바퀴 회전 함수
  for(int i=0; i<2; i++) {
    myStepper.step(stepsPerRevolution);
  }
}

```

## ◇ 미니시터 - 협업

### 1. 협업이란?

독립적으로 동작하던 개개인의 스마트 미니시터를 각 박스 내부에 설치되어있는 아두이노간 유선으로 연결해 시리얼 통신을 이용하여 데이터를 주고받아 동작하는 것입니다.

### 2. 협업 방법

#### • 시리얼통신

시리얼 통신은 데이터 전송을 위한 선(TX) 하나와 수신을 위한 선(RX) 하나로 이루어진다. 아두이노 우노 보드에서는 0번과 1번 핀을 시리얼 통신을 위한 핀으로 사용합니다. 0번, 1번 핀은 USB to Serial 기능을 하는 칩과 연결되어 있어서 USB 케이블을 통해 아두이노에서 PC로 데이터를 전송, 수신할 수 있고, USB케이블을 PC와 연결하지 않았을 경우에 0번, 1번 핀을 다른 기기의 시리얼 통신 포트와 연결하여 데이터를 주고받을 수 있습니다.

#### - 하드웨어 시리얼(HardWare Serial)

▶ 위에서 설명한 일반적인 시리얼 통신을 하드웨어 시리얼(HardWare Serial)이라고 합니다.

명령어	설명
Serial.begin();	setup() 함수 안에서 선언하면 시리얼 통신을 사용할 수 있습니다. 9600은 통신 속도를 의미하며 연결되는 장치와 통신 속도가 일치해야 합니다.
Serial.print();	배열을 제외한 변수를 괄호 안에 넣으면 그 값을 출력해줍니다.
Serial.println();	Serial.print(); 함수에 줄 넘김 문자를 붙여 값을 출력하고 다음 줄로 이동하는 함수입니다.
Serial.write();	1바이트(byte), 1개의 문자, 문자열, 배열 값을 출력합니다.
Serial.available();	아두이노 버퍼에서 데이터를 읽어오며 데이터가 없을시 -1(false)을 반환
Serial.read();	시리얼 통신을 통해 들어온 데이터를 읽습니다.

#### - 소프트웨어 시리얼(SoftWare Serial)

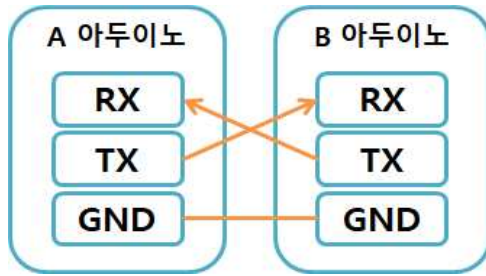
▶ 하드웨어 시리얼 제외한 일반 입출력 핀을 RX, TX핀으로 동작할 수 있게 해주는 방법입니다. 소프트웨어 시리얼(SoftWare Serial) 라이브러리는 기본적으로 제공됩니다.

명령어	설명
#include <SoftwareSerial.h>	소프트웨어 시리얼(SoftWare Serial) 라이브러리를 사용하기 위해서 포함 선언을 반드시 해줘야 합니다.
SoftwareSerial mySerial(2,3);	mySerial.xxx()의 형태로 시리얼 함수들을 사용할 수 있습니다. 2번, 3번 핀을 이용하여 시리얼 통신을 할 수 있습니다.

위 라이브러리 포함선언과 RX, TX핀을 지정 선언을 해주면 하드웨어 시리얼의 명령어를 그대로 사용할 수 있습니다.

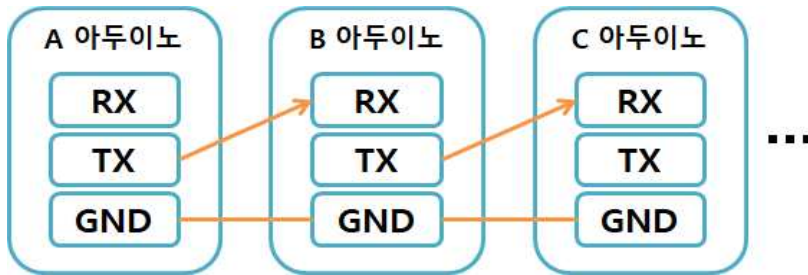
- 배선연결방법

▶ 각 아두이노의 RX, TX, GND를 아래 그림처럼 연결하면 A, B 아두이노는 서로 데이터를 주고받을 수 있습니다.



<두 아두이노간 시리얼 통신 배선 연결도>

▶ 여러 아두이노를 연결하는 방법입니다.



<여러 아두이노 간 시리얼 통신 배선 연결도>

위 그림의 구조는 A아두이노에서 TX핀으로 데이터를 전송하고 B아두이노는 RX핀으로 데이터를 받고 TX핀으로 C아두이노로 데이터를 전송하는 구조입니다.

3. 협업 시나리오 예제

○ 독립적 협업 예시



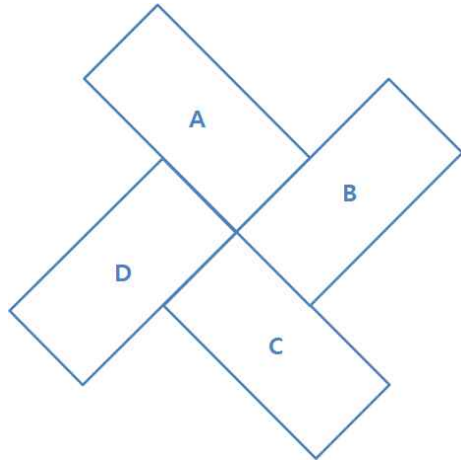
▶ 박스 간의 시리얼 통신 없이 동일한 소스를 업로드하여 각각의 박스마다 센싱을 하여 이벤트가 진행되는 구조입니다. 동일하게 동작하는 박스들을 한 곳에 모아 동시에 움직이는 것처럼 표현할 수 있습니다.

○ 독립적 협업 - 시나리오

① 조도센서 - 어두워지면  
가로등 ON -> 건물 A, B, C 그룹 LED 1초 간격으로 ON

② 사운드센서 - 박수(소리감지) 카운트 횟수에 따라 이벤트 발생  
1. 크레인 90도 회전 후 1초 기다리고 제자리(0도)로 회전  
2. 카페 3바퀴 회전  
3. 가로등, 건물 LED OFF

○ 4개 협업 예시



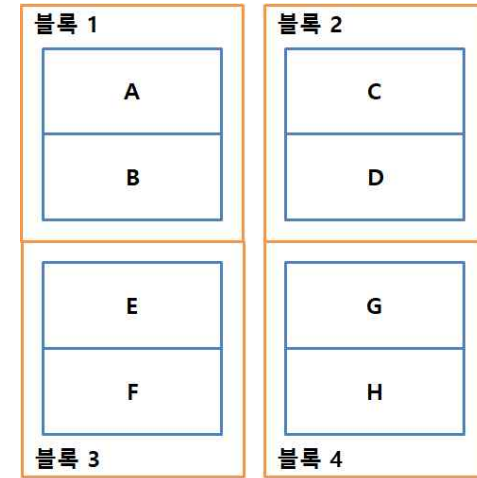
▶ 4개의 박스 A, B, C, D가 시리얼 통신으로 A 박스의 센서에 의해 동작이 진행됩니다. 박스들은 점퍼선으로 RX, TX를 연결합니다. 각각의 박스는 다른 소스 프로그램을 업로드 합니다. 예를 들어 A박스는 센싱과 데이터 전송 소스, B박스는 데이터를 받고 다시 전송하는 소스를 업로드 합니다.

○ 4개 박스 협업 - 시나리오

- 어두워지면
- ① 조도센서
  1. A,B,C,D -건물 A그룹 LED ON(1초)
  2. A,B,C,D -건물 B그룹 LED ON(1초)
  3. A,B,C,D -건물 C그룹 LED ON(1초)
- 첫 번째 박수
- ② 사운드센서
 

<ol style="list-style-type: none"> <li>1. A 크레인 180도 회전(1초)</li> <li>2. B 크레인 180도 회전(1초)</li> <li>3. C 크레인 180도 회전(1초)</li> <li>4. D 크레인 180도 회전(1초)</li> <li>5. A,B,C,D 카페 1바퀴 회전</li> </ol>	<ul style="list-style-type: none"> <li>- 두 번째 박수</li> <ol style="list-style-type: none"> <li>6. D 크레인 제자리로(1초)</li> <li>7. C 크레인 제자리로(1초)</li> <li>8. B 크레인 제자리로(1초)</li> <li>9. A 크레인 제자리로(1초)</li> <li>10. 모든 건물 LED OFF</li> </ol> </ul>
--	---

○ 8개 협업 예시



▶ 2개의 박스를 연결해 하나의 블록이 됩니다. 각 블록의 하나의 박스에서 센싱을 하고 연결된 박스에 값을 전송하여 이벤트가 발생하는 구조입니다.

○ 8개 박스 협업 - 시나리오

- 어두워지면
- ① 조도센서
  - 블록1 -> 블록2 -> 블록3 -> 블록4 순으로 건물 LED ON & 카페 1바퀴 회전
  - => delay() 함수를 이용하여 블록간 순차적인 효과를 준다.
- 박수(소리감지) 카운트 횟수에 따라 이벤트 발생
- ② 사운드센서
  1. 모든 블록 동시에 가로등 ON
  2. 블록별 순차적 크레인 회전
  3. 건물 LED, 가로등 모두 OFF, 크레인 제자리로(0도) 회전