

기초부터 프로젝트 완성까지,

아두이노 “첫 걸음”을 위한

Arduino Bluetooth

Car Control Project

작성자 : 인천대학교 OneScore 임승한

lim_designer@naver.com



Contents

1. Arduino

1.1 아두이노 개발 환경 구성 p.03

아두이노 통합 개발 환경(IDE) 소프트웨어 설치 p.03

아두이노 보드와 컴퓨터 연결 p.09

아두이노 보드 용 드라이버 설치 p.12

1.2 IDE p.19

아두이노 IDE 보드 선택 p.19

아두이노 IDE 포트 선택 p.21

아두이노 스케치 프로그램 작성 p.22

아두이노 스케치 라이프 사이클 p.23

아두이노 스케치 저장 p.25

아두이노 스케치 컴파일 p.28

아두이노 스케치 업로드 p.29

아두이노 스케치 실행 확인 p.31

2. Project p.33

2.1 프로젝트 설명 p.33

2.2 부품 p.33

2.3 무작정 따라하기 p.34

2.4 전체 소스 코드 p.46

2.5 Bluetooth로 실행하기 p.52

3. 회로 구성도 p.55

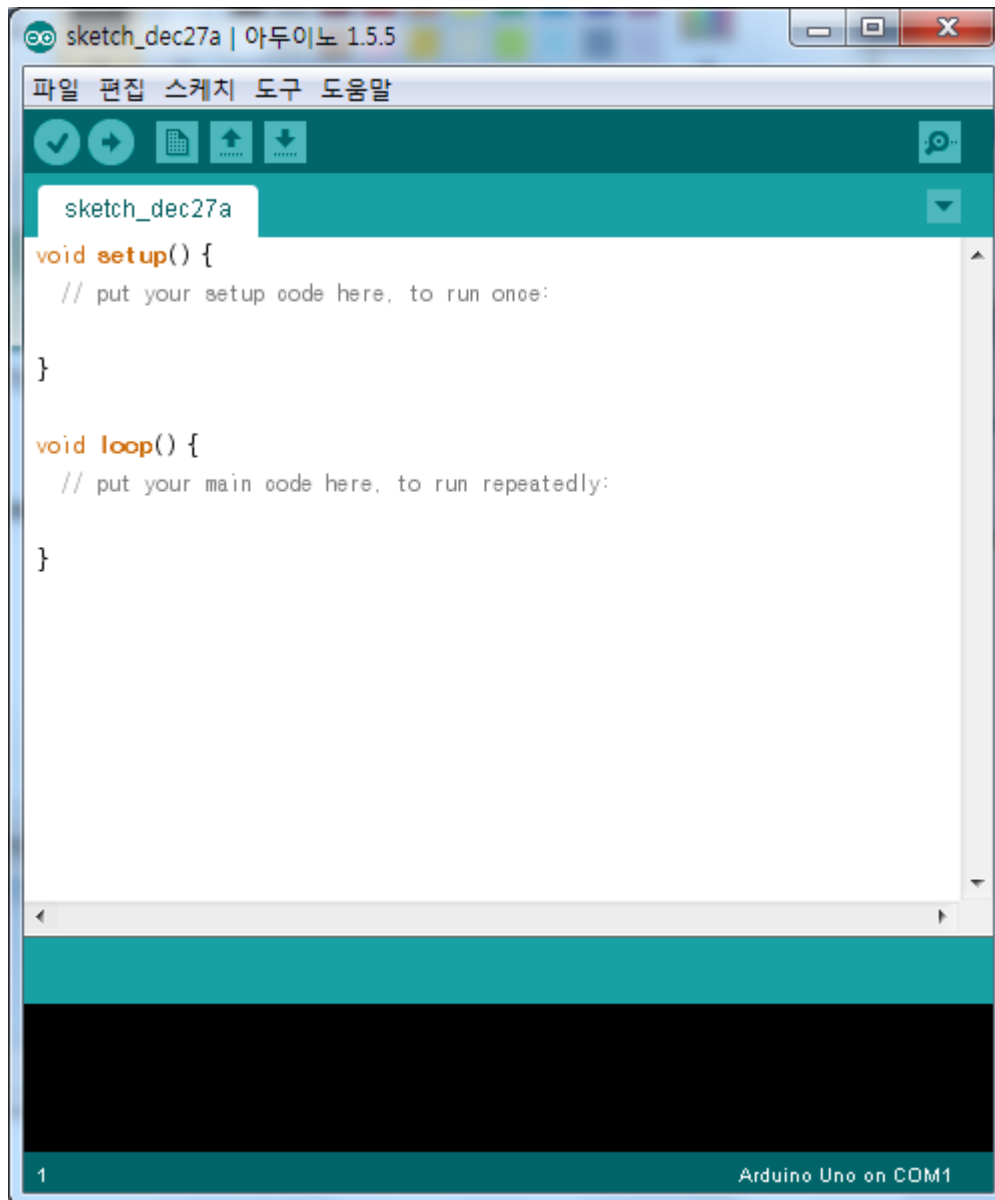
4. 참고문헌 p.56

1. Arduino

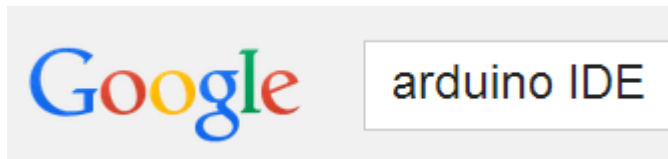
1.1 아두이노 개발 환경 구성

아두이노 통합 개발 환경 소프트웨어 설치

먼저 아두이노 스케치를 작성, 컴파일하고 보드상에 업로드하기 위한 환경을 구성해 보자. 다음과 같은 아두이노 통합 개발 환경을 설치할 것이다.



1. 다음과 같이 구글 사이트 등을 통해서 [아두이노 IDE]를 검색한다. 아두이노 IDE(Integrated Development Environment)는 아두이노 통합 개발 환경을 의미한다.



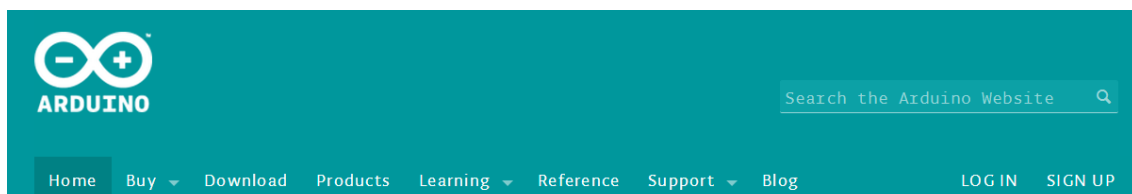
2. 그러면 다음과 같이 아두이노 IDE 소프트웨어를 다운로드 받을 수 있는 사이트를 찾을 수 있다. 이 사이트를 클릭해 들어간다.

[Arduino IDE](#)

arduino.cc/en/main/software ▼ [이 페이지 번역하기](#)

This page contains the download links to the latest **Arduino** integrated development environment (IDE) software, release notes, and additional software not ...

3. 그러면 다음과 같이 ARDUINO 로고가 그려진 아두이노 웹 사이트를 찾을 수 있다. [Download the Arduino Software]라는 문장을 볼 수 있다. 페이지 아래쪽으로 이동해 보자.



Download the Arduino Software

4. 다음 부분이 나올 때까지 페이지를 이동한다. [Windows(ZIP files)]를 선택해 마우스 왼쪽버튼을 클릭한다. 그러면 다운로드가 시작된다.

Arduino 1.5.5 BETA (with support for Arduino Yún and Arduino Due boards)

If you have the Arduino Yún or Due you must download the 1.5.5 version. Refer to [the Yun getting started page](#), or [Due getting started page](#) for specific details about those boards.

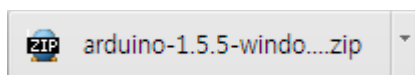
WARNING: This software is a beta version, you may encounter bugs or unexpected behaviours. Please discuss any issues in the [Yún forum](#) or [Due forum](#)

Download

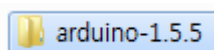
Arduino 1.5.5 ([release notes](#)):

- Windows Installer, [Windows \(ZIP file\)](#)
- Mac OS X
- Linux: [32 bit](#), [64 bit](#)
- [source](#)

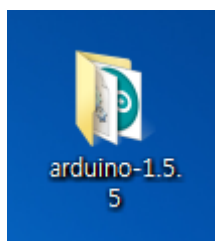
5. 다음과 같이 다운로드가 완료되면, 적당한 곳에 압축을 푼다.



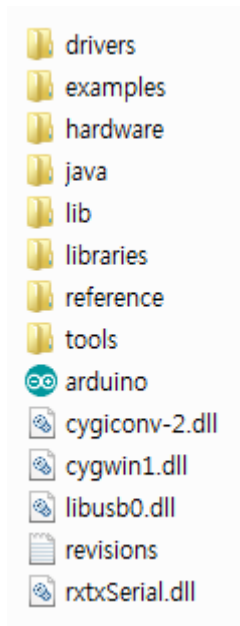
6. 압축을 풀면 다음과 같은 디렉터리를 볼 수 있다.



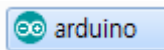
7. 이 디렉터리를 윈도우 바탕화면으로 옮긴다. 다른 적당한 위치로 옮겨도 상관없다.



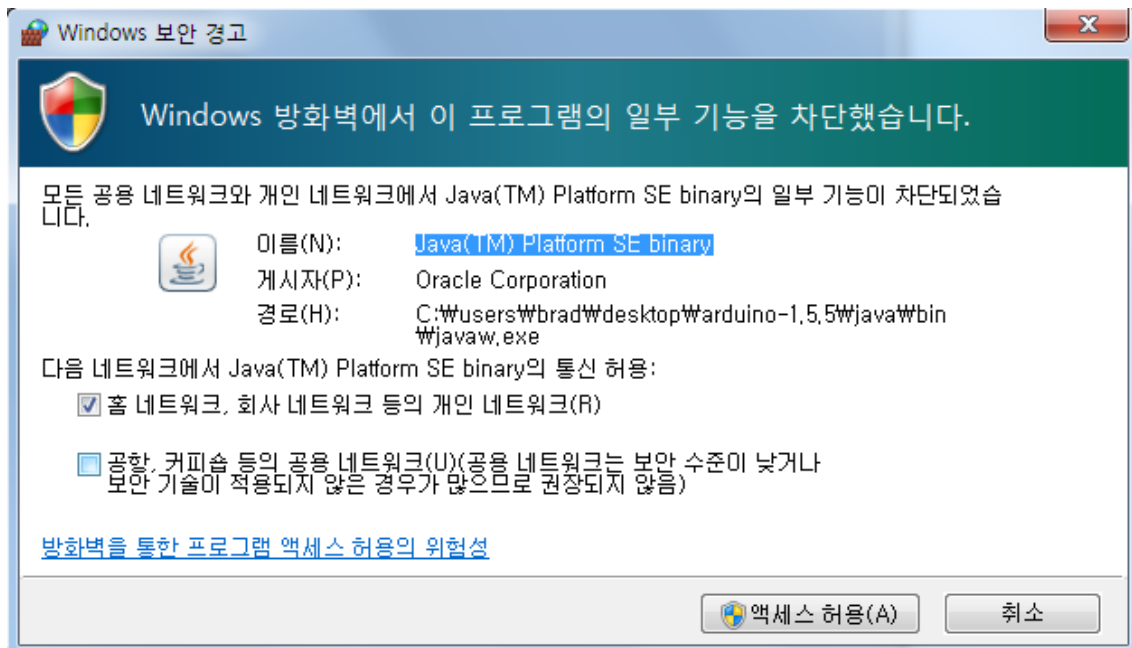
8. 디렉터리 안으로 들어가 보자. 다음과 같은 하위 디렉터리들과 파일들이 있는 것을 볼 수 있다.



9. 이 중에 다음 프로그램을 선택해 마우스 오른쪽 버튼을 클릭한다. 앞으로 이 프로그램을 이용하여 아두이노 IDE 소프트웨어를 구동하게 된다.

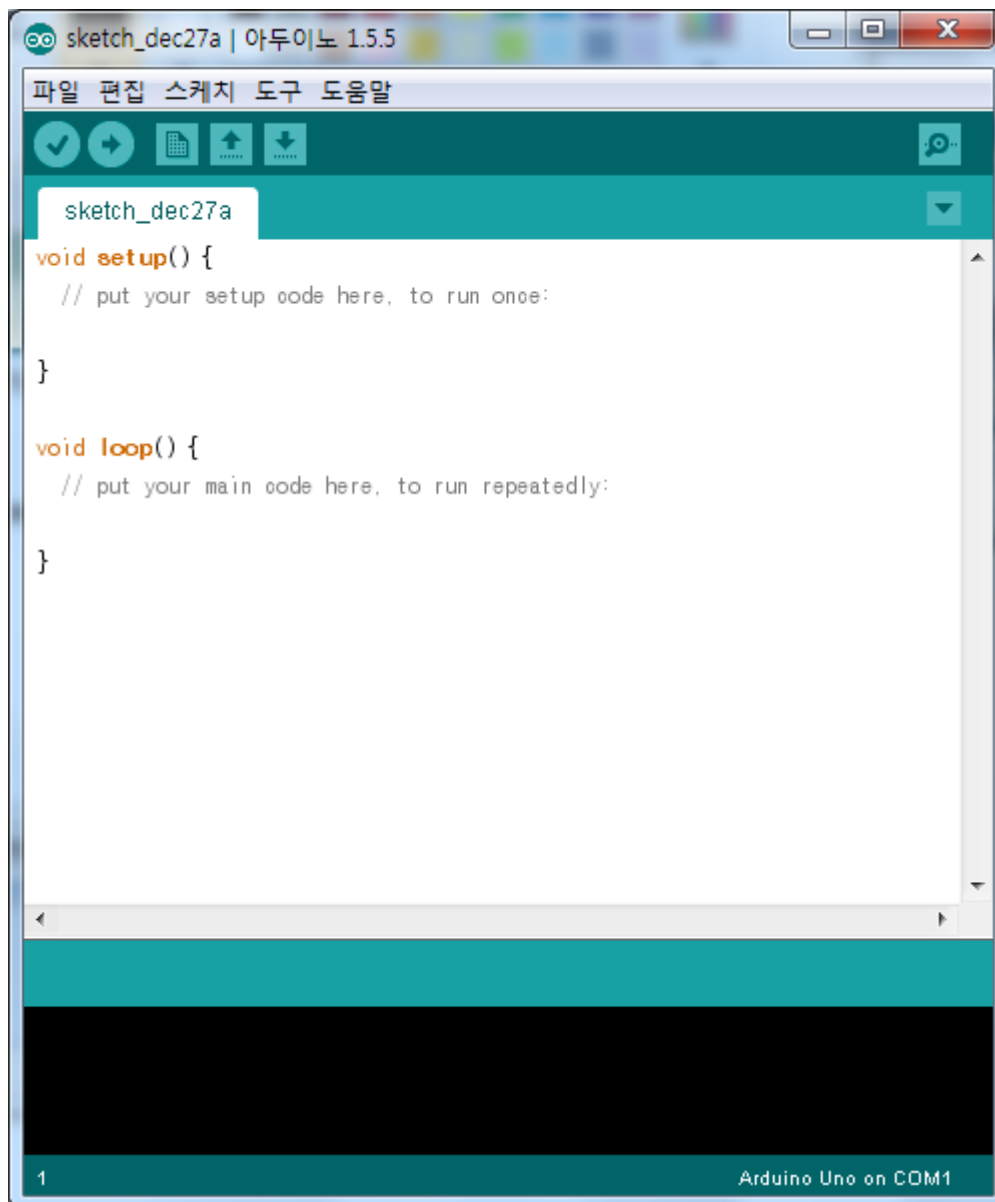


10. 처음엔 다음과 같은 보안 경고 창이 뜬다. 아두이노 IDE 소프트웨어를 사용하기 위해 필요한 부분이기 때문에 액세스 허용 버튼을 누른다.



11. 그러면 다음과 같이 아두이노 IDE 소프트웨어 프로그램이 실행되는 것을 볼 수 있다. 앞으로 우리는 이 프로그램을 이용하여

- 1) 아두이노 스케치를 작성하고,
- 2) 컴파일하고
- 3) 아두이노 보드상에 업로드하게 된다.
- 4) 그리고 보드를 통해서 결과를 확인하게 된다.



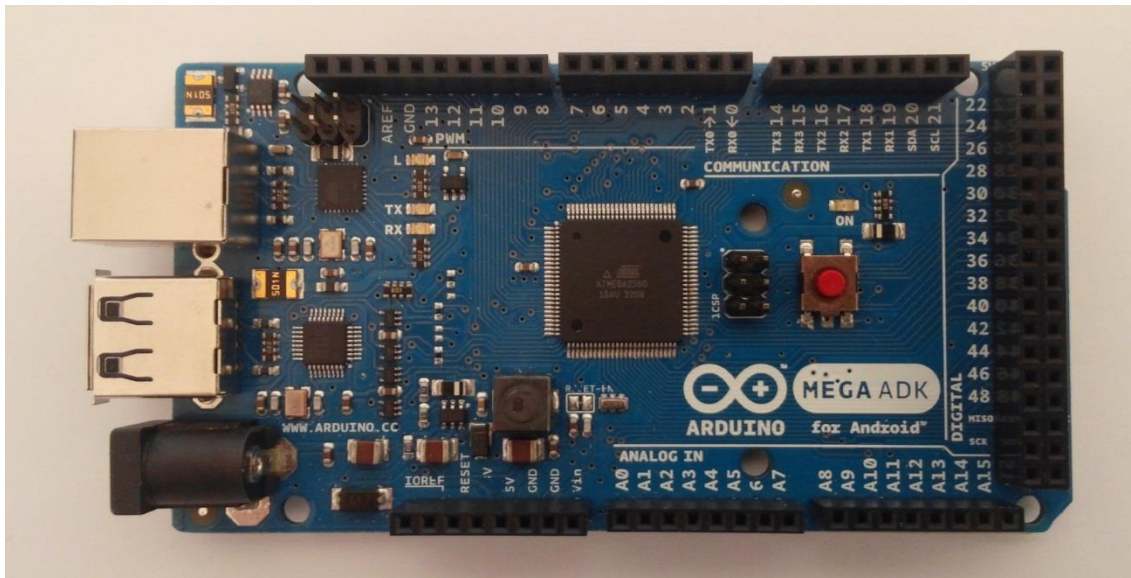
아두이노 보드와 컴퓨터 연결

이번엔 아두이노 보드와 컴퓨터를 연결해 보자. 아두이노 보드의 USB는 다음과 같이 세 가지 기능을 제공한다.

- 1) 먼저 전원을 공급 받을 수 있고,
- 2) 컴파일한 프로그램을 업로드할 수 있고,
- 3) 시리얼 포트를 통해 디버깅 메시지를 볼 수 있다.

이런 이유로 아두이노 보드는 USB 케이블 하나로 컴퓨터로 연결될 수 있으며, 따라서, 아주 간단한 인터페이스를 이용하여, 개발을 진행할 수 있는 장점이 있다. 이러한 기능들은 곧 살펴보게 될 것이다. 조금만 더 가 보도록 하자.

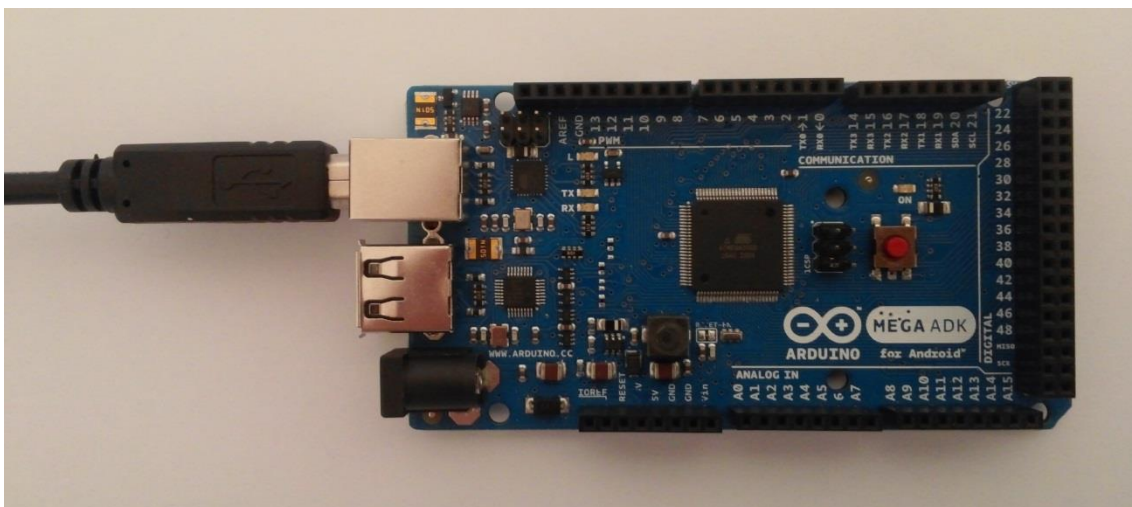
1. 다음과 같이 ARDUINO MEGA ADK for Android 보드를 준비한다. 보드의 이름에서 알 수 있는 것처럼 이 보드는 안드로이드 디바이스와 연결해서 사용할 수 있는 보드이다. 안드로이드 디바이스란 안드로이드 스마트 폰, 안드로이드 태블릿 등 안드로이드 기반 하드웨어를 의미한다.



2. 다음과 같은 모양의 USB 케이블을 준비한다. 이 케이블을 이용하여 아두이노 보드와 컴퓨터를 연결할 것이다.



3. 다음과 같이 USB 케이블의 한쪽 끝을 아두이노 보드에 연결한다.

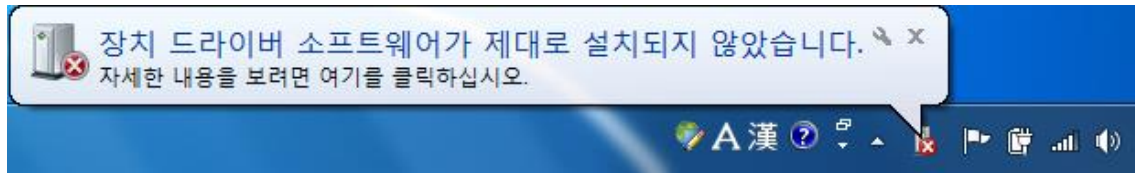


4. 다음과 같이 USB 케이블의 다른 쪽 끝을 컴퓨터에 연결한다.



아두이노 보드용 드라이버 설치

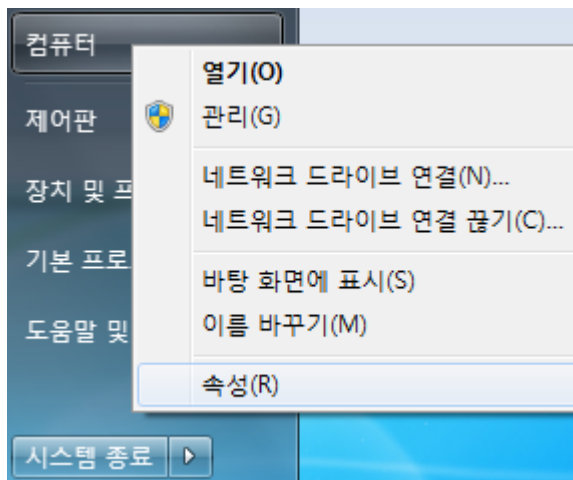
1. 아두이노 보드를 USB 케이블을 통해 처음 연결하면, 다음과 같이 화면에 USB 디바이스 인식 실패 창이 뜨는 것을 볼 수 있다. 장치 드라이버 소프트웨어를 설치해 보자.



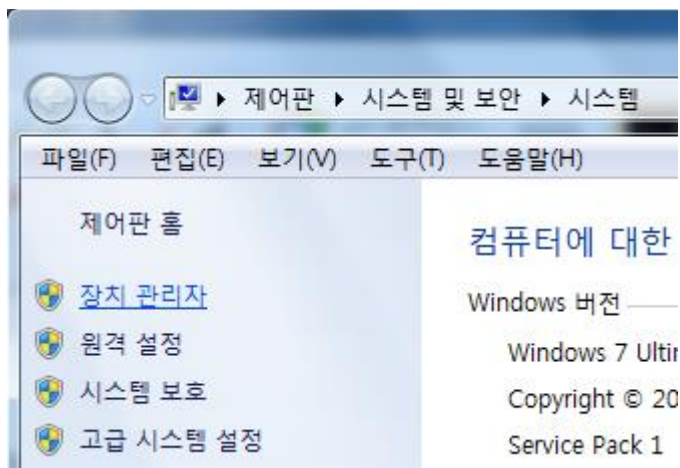
2. 다음과 같은 시작 버튼을 찾아 마우스 왼쪽 클릭한다.



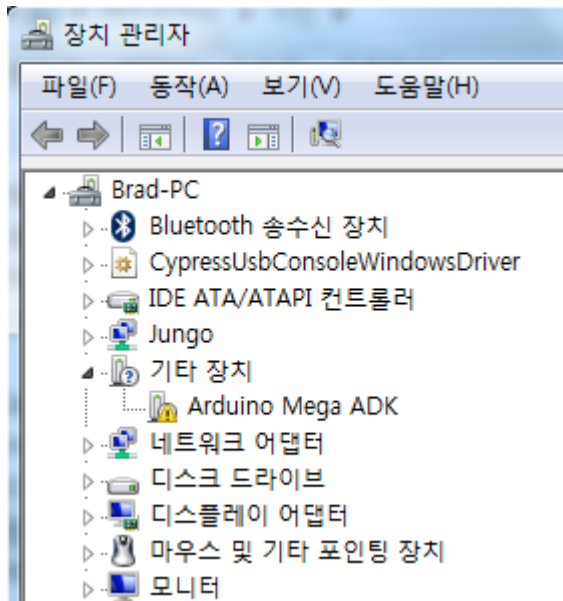
3. 그러면 다음과 같이 메뉴창이 뜬다. 컴퓨터를 마우스 오른쪽 버튼을 누르고 [속성]을 선택한다.



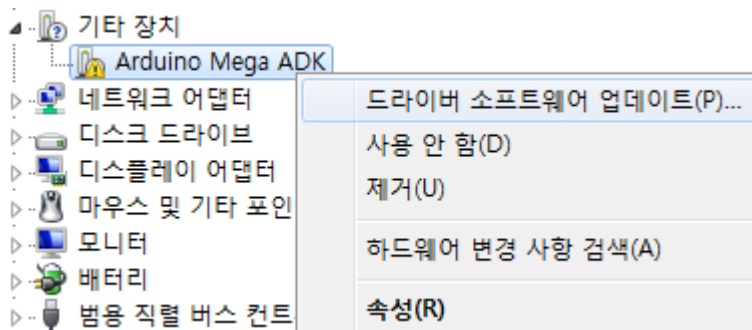
4. 그러면 다음과 같은 창이 뜬다. [장치 관리자]를 선택해 마우스 왼쪽 버튼을 클릭한다.



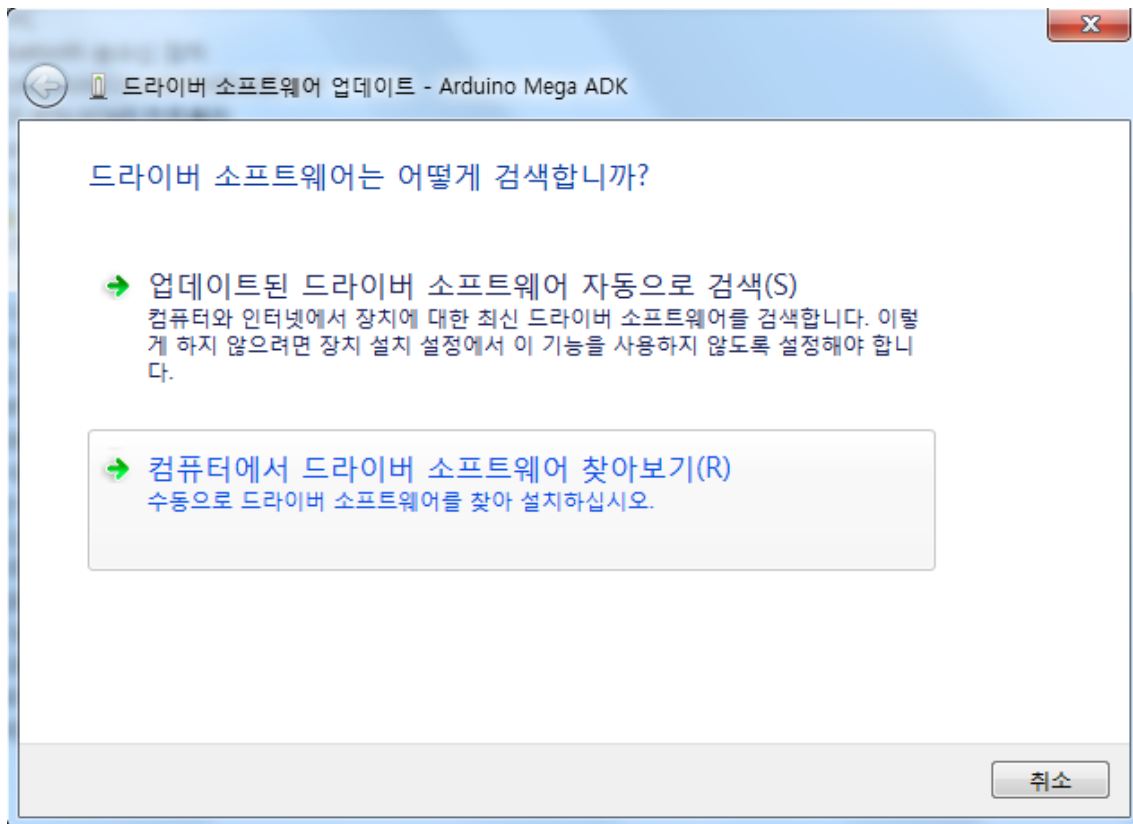
5. 그러면 다음과 같이 [장치 관리자] 창이 뜬다. [기타 장치]에 [Arduino Mega ADK] 장치를 볼 수 있다.



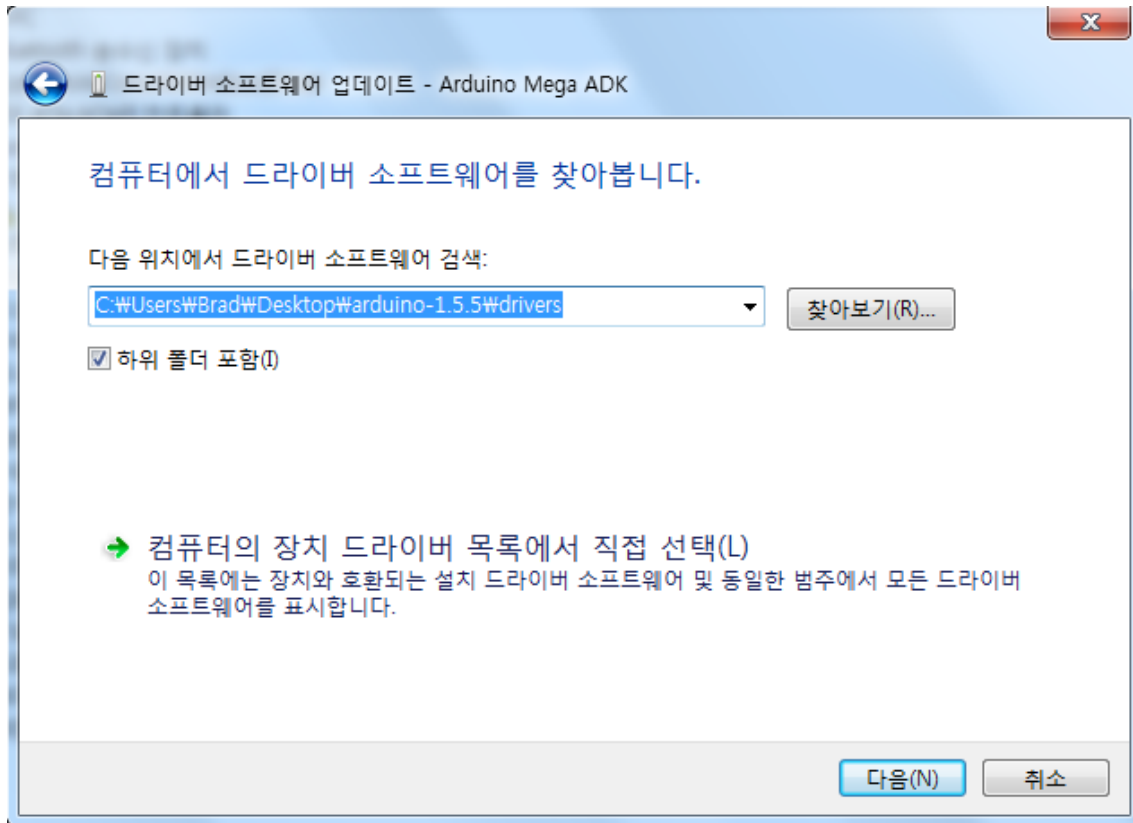
6. 다음과 같이 [Arduino Mega ADK] 항목에서 마우스 오른쪽 버튼을 클릭한 후, [드라이버 소프트웨어 업데이트(P)...] 항목을 선택한다.



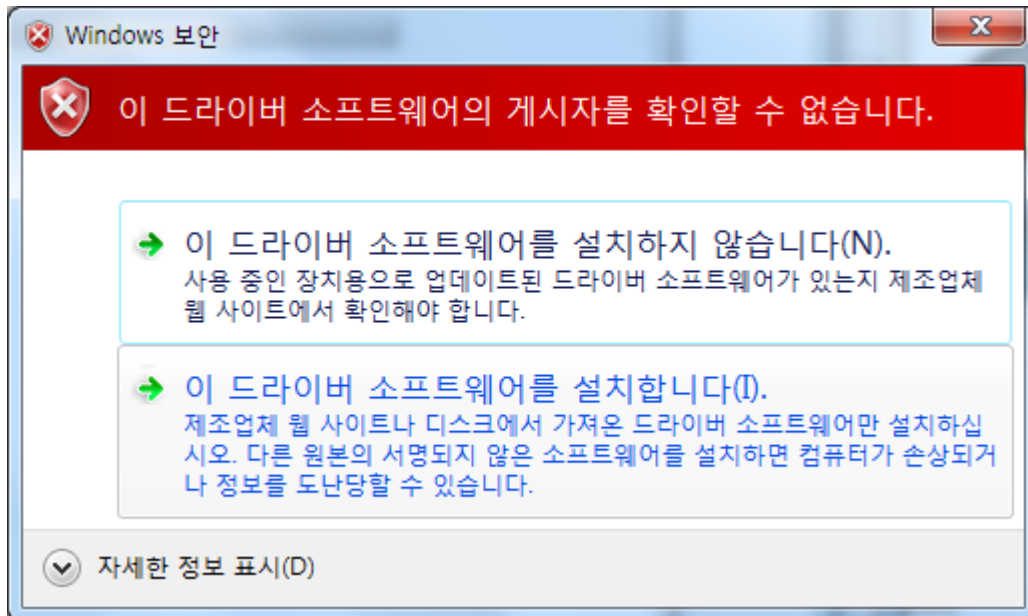
7. 그러면 다음과 같이 [드라이버 소프트웨어 업데이트] 창이 뜬다. 두 번째 항목인 [컴퓨터에서 드라이버 소프트웨어 찾아보기(R)]를 선택한다.



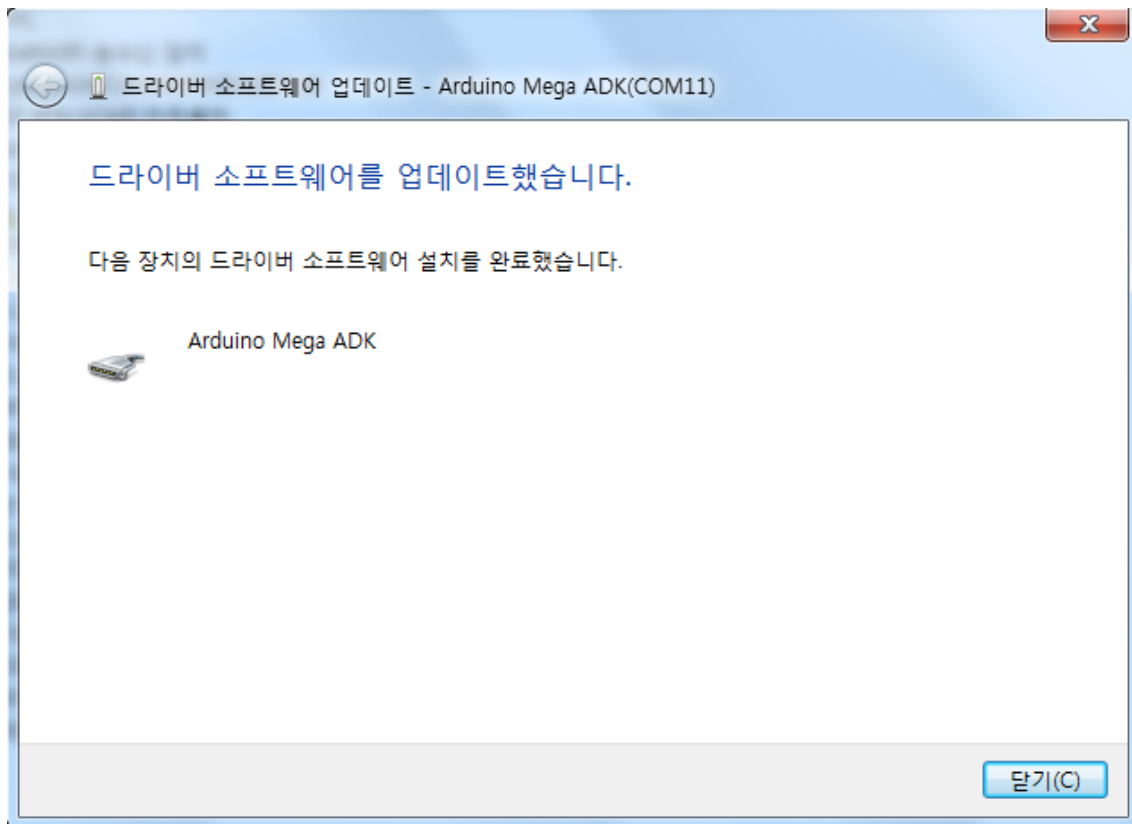
8. 그러면 다음과 같은 창이 뜬다. [찾아보기] 버튼을 눌러 드라이버 위치를 앞에서 설치했던 arduino-1.5.5 디렉터리 내에 있는 drivers 위치를 선택한다. 하위 폴더 포함 부분도 체크해 준다. 이제 다음 버튼을 누르자.



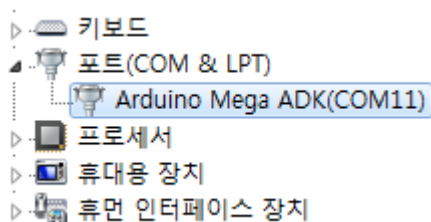
9. 그러면 다음과 같은 Windows 보안 창이 뜬다. 아두이노 보드 용 디바이스 드라이버에 대한 경고 메시지이며, 실제로는 문제가 없으니, 두 번째 항목인 [이 드라이버 소프트웨어를 설치합니다(I)]를 선택하여, 마우스 왼쪽 버튼을 클릭한다.



10. 그러면 다음과 같이 [Arduino Mega ADK] 용 드라이버가 설치된다. 닫기 버튼을 누른다.



11. 창을 닫고 나면, 다음과 같이 포트 항목에 [Arduino Mega ADK(COM11)] 항목이 추가된 것을 볼 수 있다. COM11 부분을 기억하기 바란다. 환경에 따라 독자 여러분들의 포트는 다를 수도 있다. 그 포트 번호를 기억하면 된다.



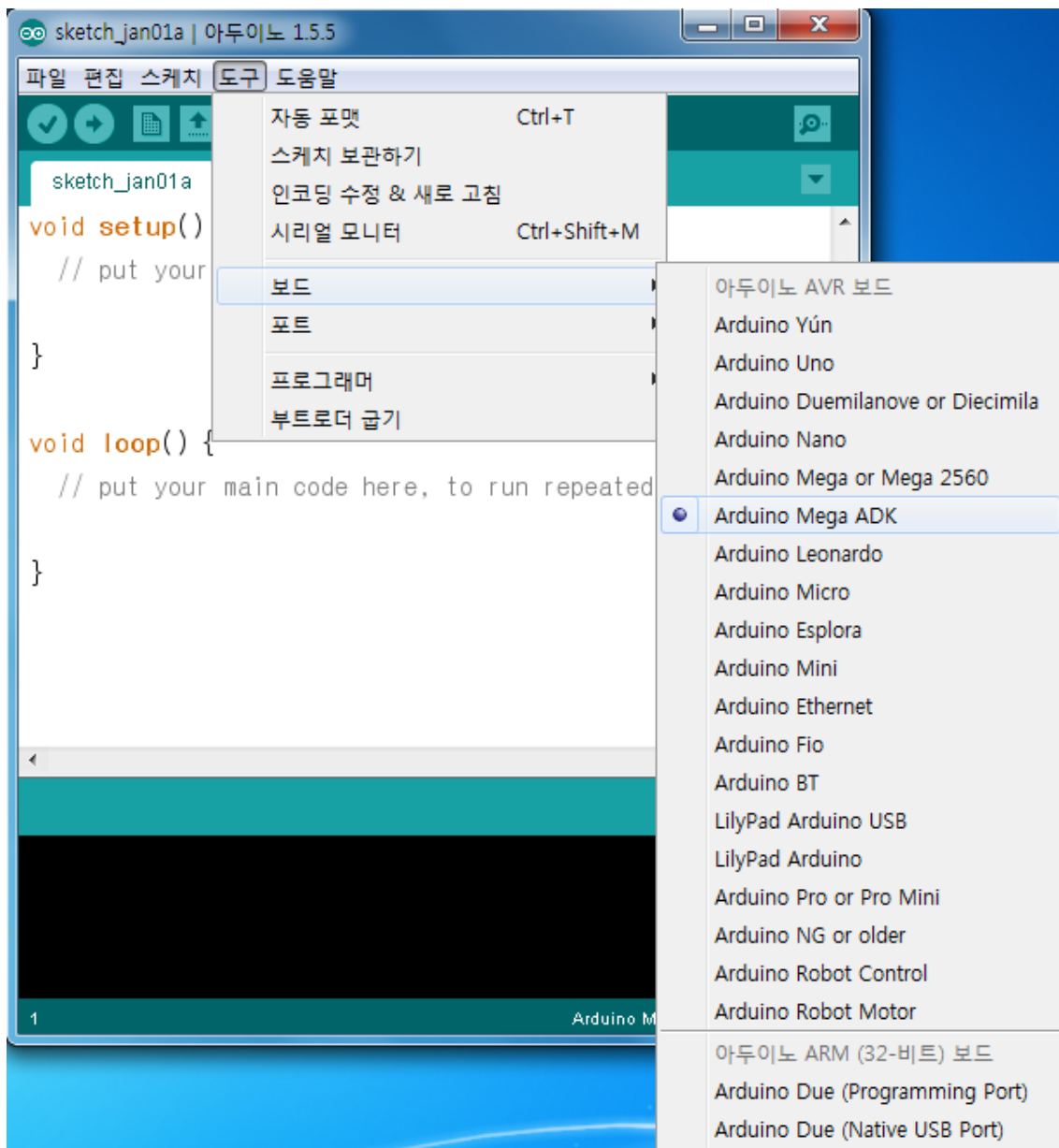
이제 아두이노 보드 드라이버 설치가 완료되었다. 보드 기반의 개발에서는 자주 보게 되는 과정이며, PC 기반의 응용 프로그래밍만 해 본 독자 분들이라면 이 과정이 낯설 수도 있다.

1.2 IDE

여기서는 아두이노 IDE에서 사용할 보드로 MEGA ADK를 선택하고, 아두이노 스케치 프로그램을 다운로드할 포트를 선택하는 방법을 살펴본다. 그리고 간단한 Hello Arduino 스케치 프로그램을 작성한 후, 컴파일하고 보드상으로 업로드하여 테스트하는 과정을 살펴본다.

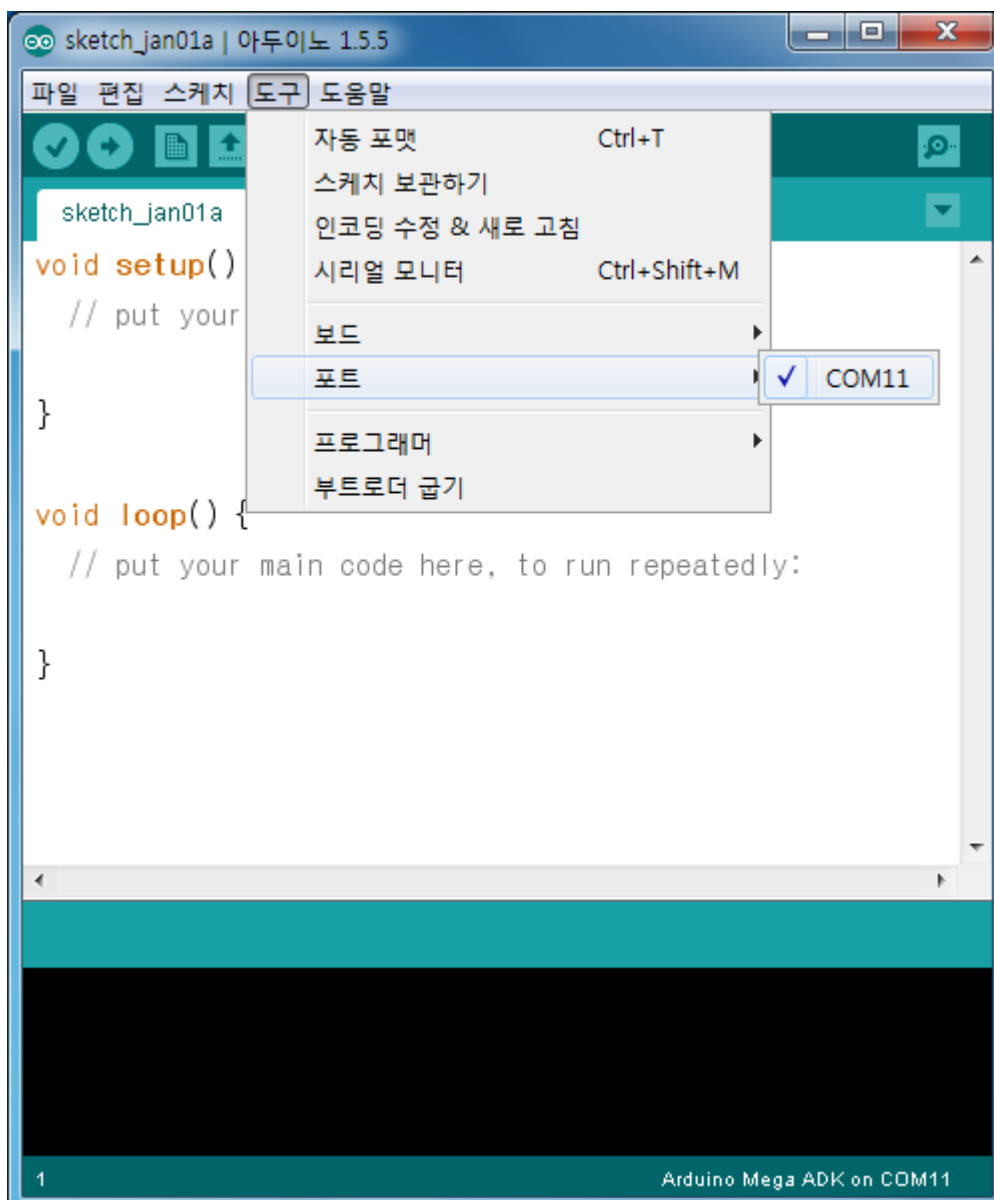
아두이노 IDE 보드 선택

아두이노 IDE에서 사용할 보드를 선택하기 위해 다음과 같이 아두이노 IDE에서 [도구]-[보드]-[Arduino Mega ADK] 보드를 선택한다. 사용하는 보드에 대한 선택은 이 메뉴를 통해서 하면 된다.



아두이노 IDE 포트 선택

다음은 아두이노 스케치 프로그램을 업로드하고, 시리얼을 통한 디버깅 메시지를 확인하기 위해 포트를 선택한다. 다음과 같이 [도구]-[포트]-[COM11]을 선택한다. 독자 여러분의 경우 이 부분이 달라질 수 있다. 드라이버를 설치하는 과정에서 기억했던 COM 포트를 선택해 준다.

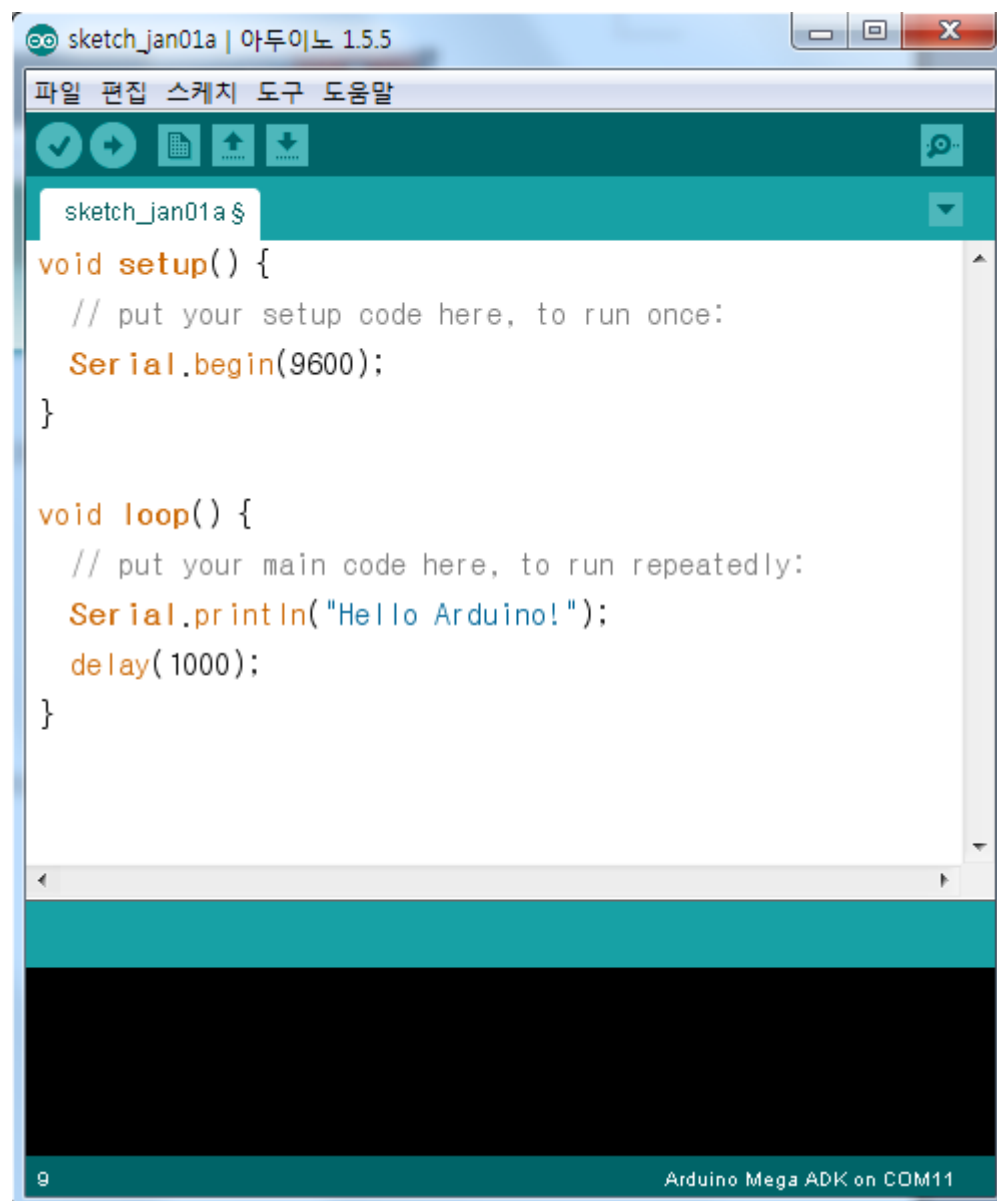


아두이노 스케치 프로그램 작성

이제 간단한 아두이노 스케치 프로그램을 작성해 보자.

1. 다음과 같이 예제를 작성한다.

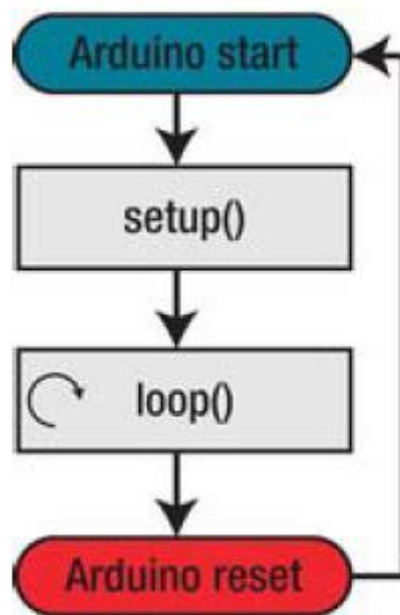
소스 파일 : 부록\Source\Chapter01\01Hello



아두이노 스케치 라이프 사이클

예제의 내용을 살펴 보기 전에 아두이노 보드 상에서 프로그램이 동작하는 방식을 살펴 보도록 하자.

아두이노 스케치 프로그램은 두 개의 중요한 메서드로 구성된다. 하나는 `setup` 메서드이며, 코드 실행을 시작할 때 한 번만 수행된다. 여기에서 하드웨어를 제어하기 위한 초기화 루틴을 작성할 수 있다. 두 번째 메서드는 `loop` 메서드이다. 이 메서드는 보드가 리셋 될 때까지 반복해서 수행되는 루틴이다. 아래 `loop()` 박스 내에 회전 화살표는 반복 수행을 의미한다. 이 부분에서 보드를 구동 시키는 프로그램 로직을 구현하면 되겠다. 다음 그림과 같이 아두이노 스케치 프로그램의 라이프 사이클은 아주 간단하다.



<그림> 아두이노 스케치 라이프 사이클

예제를 살펴 보도록 하자.

```
void setup() {
```

```
// 한 번 실행될 설정 코드를 여기에 놓는다
```

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

```
}
```

여기서는 Serial 클래스의 begin 메서드를 통해 디버깅 메시지를 보게 될 시리얼 디바이스의 통신 속도를 설정하고 있다. 9600 bps의 속도로 시리얼 통신을 하도록 설정한다. 디버깅 메시지를 보는 방법은 조금 뒤에 소개된다. Serial 클래스는 시리얼 통신용 디바이스를 다루기 위한 코드와 데이터들의 집합으로 이해하면 된다.

```
void loop() {
```

```
    // 반복해서 실행될, 주요 코드를 여기에 놓는다.
```

```
    Serial.println("Hello Arduino!");
```

```
    delay(1000);
```

```
}
```

여기서는 Serial 클래스의 println 메서드를 통해 “Hello Arduino!” 메시지를 시리얼 모니터로 보낸다. 그리고 delay 메서드를 통해 1초가 지날 때까지 프로그램의 진행을 멈춘다. 1000은 1000분의 1초 단위이다. 예를 들어 0.1 초 멈추려고 할 경우 100을 delay 메서드에 넘겨주면 된다.

이상 예제를 살펴 보았다.

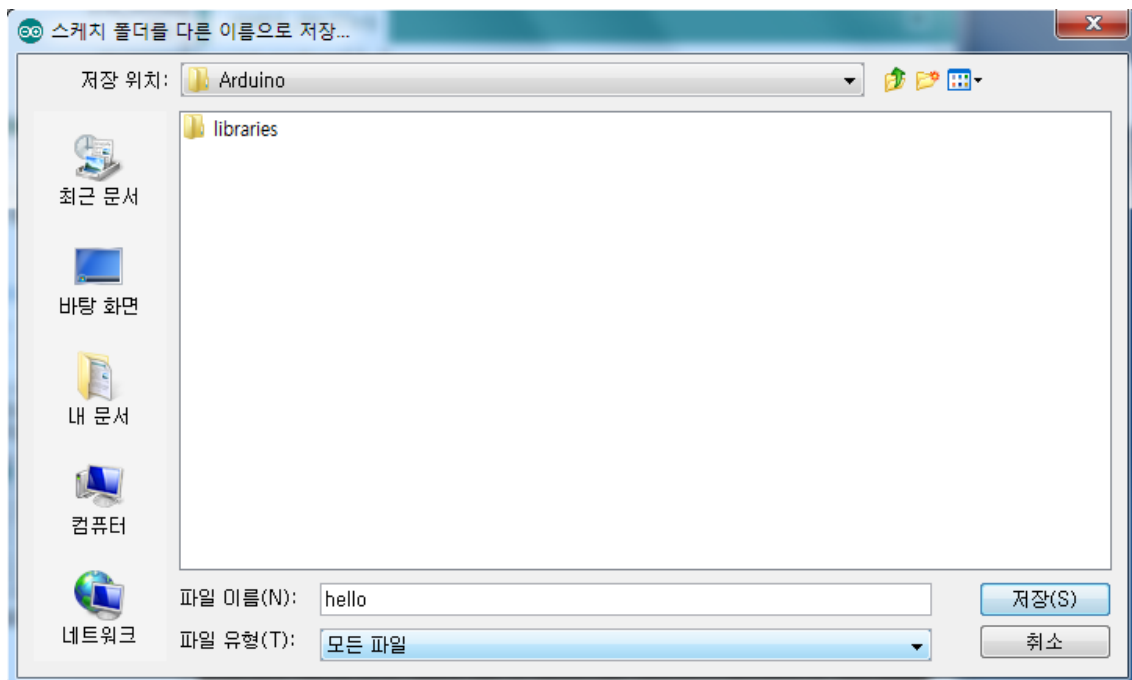
아두이노 스케치 저장

그러면 작성한 Hello Arduino 프로그램을 저장해 보도록 하자.

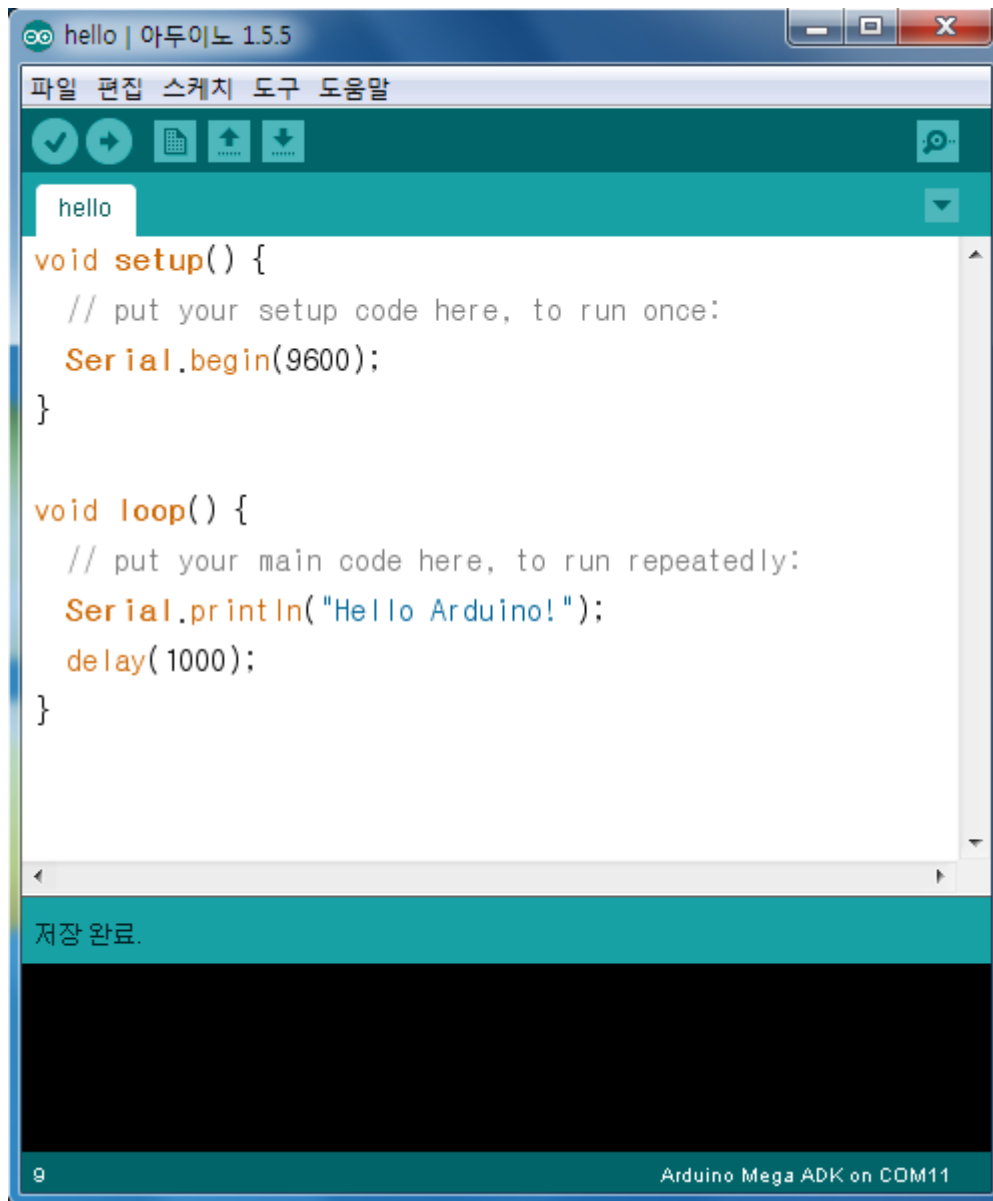
1. 아두이노 IDE 상단에 있는 다음 버튼을 눌러 스케치 프로그램을 저장할 수 있다.



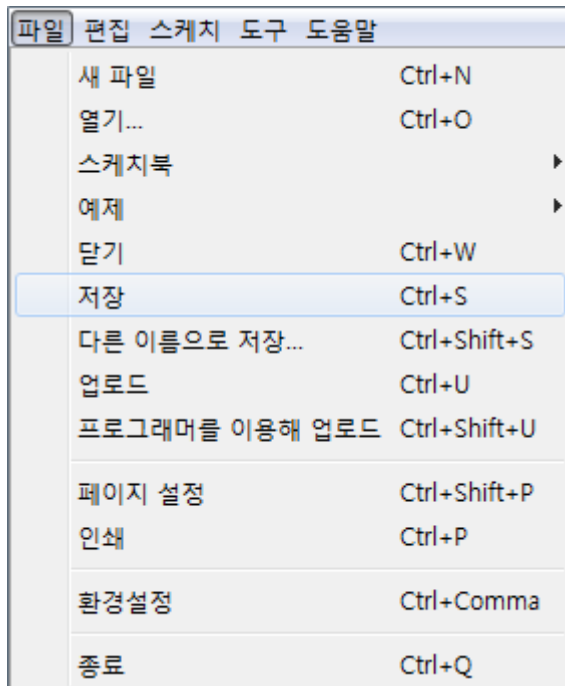
2. 버튼을 누르면 다음과 같이 저장 창이 뜬다. 기본으로 선택되는 디렉토리 내에 hello라는 이름으로 저장하자. 파일 이름 항목에 hello라고 입력하고, 저장 버튼을 누른다.



3. 그러면 다음과 같이 파일의 이름이 hello로 변경되는 것을 볼 수 있다. 이후에 나오는 예제들도 이와 같은 방법으로 저장하면 되겠다.



4. 다음과 같이 [파일]-[저장] 메뉴로 저장할 수도 있다.



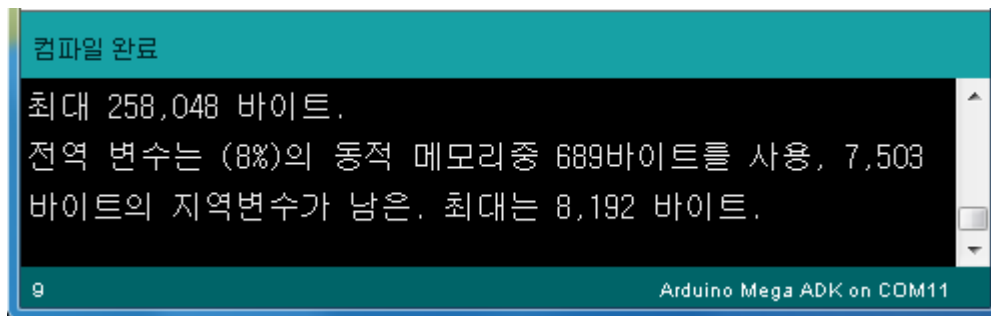
아두이노 스케치 컴파일

이제 아두이노 스케치를 컴파일해 보자. 컴파일이란 우리가 작성한 코드를 아두이노 보드상에 있는 마이크로 컨트롤러가 읽을 수 있는 코드로 변형하는 작업이며, 컴파일러라는 프로그램이 이 작업을 수행하게 된다. 한글로 쓴 소설을 영어로 번역하여, 영어를 사용하는 사람들이 읽을 수 있도록 하는 작업과 같다고 보면 되겠다.

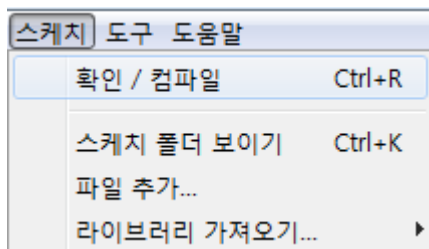
1. 다음 버튼을 눌러 컴파일을 수행한다.



2. 다음과 같이 하단부에 컴파일 완료 메시지가 뜨는 것을 볼 수 있다.



3. 다음과 같이 [스케치]-[확인/컴파일] 메뉴를 통해서도 스케치 파일을 컴파일 할 수 있다.



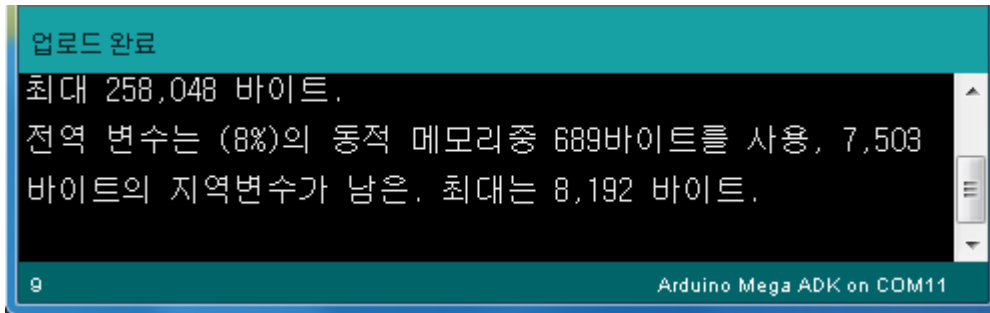
아두이노 스케치 업로드

컴파일이 완료되었으니, 이제 MEGA ADK 상에 컴파일 된 스케치 프로그램을 업로드하여 실행시켜 보자.

1. 다음 버튼을 눌러 보드 상에 업로드를 한다.



2. 다음과 같이 업로드 완료 메시지가 뜨는 것을 볼 수 있다.



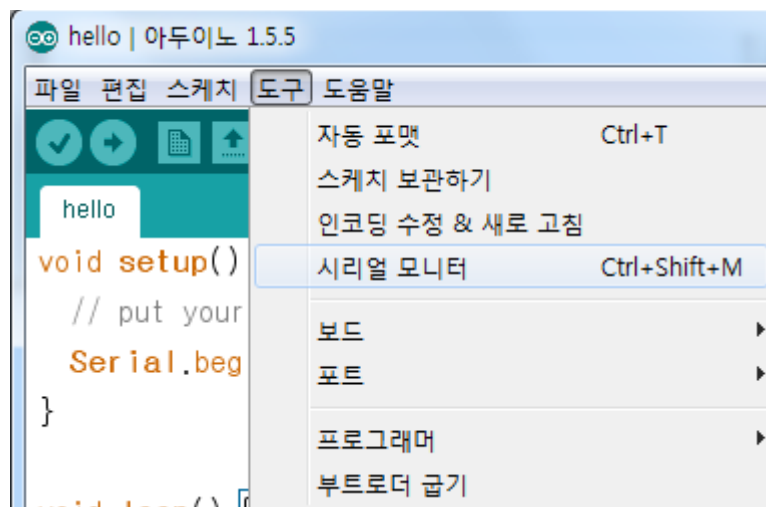
3. 다음과 같이 [파일]-[업로드] 메뉴를 통해서도 보드상에 프로그램을 업로드할 수 있다.

파일	편집	스케치	도구	도움말
새 파일				Ctrl+N
열기...				Ctrl+O
스케치북				▶
예제				▶
닫기				Ctrl+W
저장				Ctrl+S
다른 이름으로 저장...				Ctrl+Shift+S
업로드				Ctrl+U
프로그래머를 이용해 업로드				Ctrl+Shift+U
페이지 설정				Ctrl+Shift+P
인쇄				Ctrl+P
환경설정				Ctrl+Comma
종료				Ctrl+Q

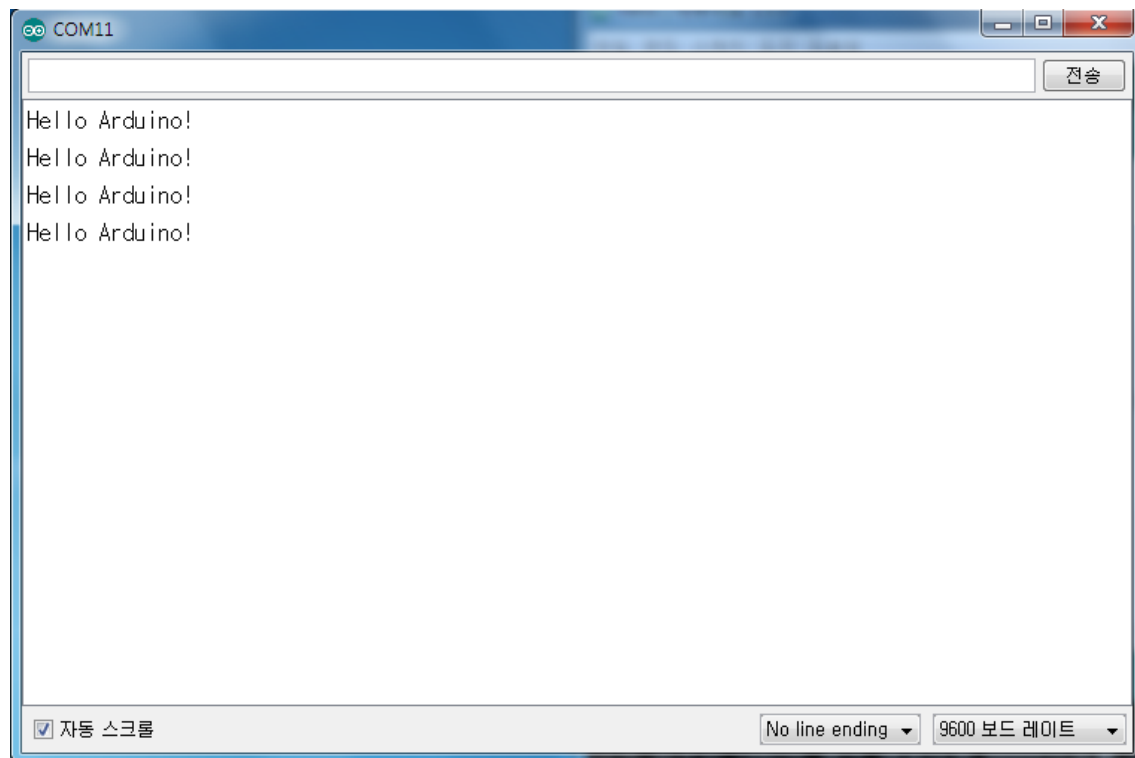
아두이노 스케치 실행 확인

이제 보드상에서 어떻게 동작하는지 확인해 보도록 하자.

1. 보드의 실행 동작을 확인해 보기 위해 다음과 같이 [도구]-[시리얼 모니터] 메뉴를 선택한다.



2. 그러면 다음과 같은 창이 뜨고 COM11을 통해 Hello Arduino! 메시지가 뿌려지는 것을 볼 수 있다. loop 메서드가 반복되서 호출되기 때문에 메시지도 반복되서 뿌려지게 된다.



2. Project

2.1 프로젝트 설명

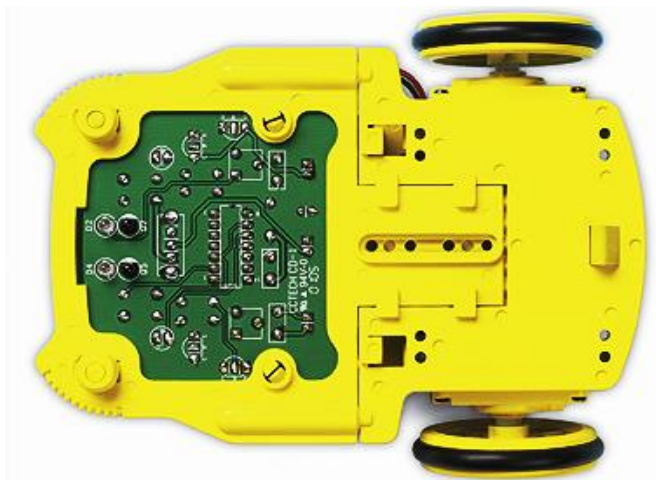
블루투스를 통한 근거리 통신 아두이노 자동차 제어 프로젝트를 개발한다.

2.2 부품

제품명	수량
Arduino Uno R3	1
점퍼 와이어	-
케이블메이트 USB2.0(AM-BM) 케이블	1
블루투스 Slave UART Board	1
소형 브레드보드	1
L293DNE (H 브리지)	1
9v 건전지	1
DC모터	2
자동차 모형 (로봇 라인X-2)	1

* 반드시 로봇 라인X-2 라는 모델일 필요는 없다.

DC모터에 장착이 되는 바퀴와 자동차 모형처럼 보일 수 있는 것이면 O.K.



(로봇 라인X-2)

* 추천 제품 구매 사이트 :

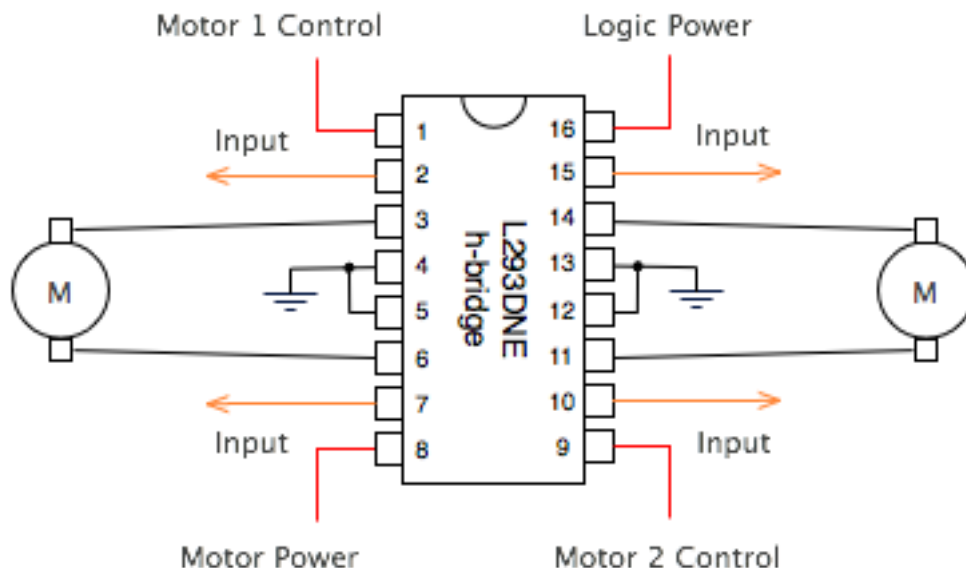
엘레파츠 www.eleparts.co.kr

디바이스마트 www.devicemart.co.kr

2.2 부품 설명

H 브리지

프로젝트에 사용할 H 브리지는 다음과 같으며 두 개의 모터를 제어할 수 있다. 먼저 8번 핀은 9V VCC로 연결하고 4번, 5번 핀은 GND로 연결한다. DC 모터는 3번 핀과 6번 핀에 연결해 준다. 모터의 선은 3번, 6번 핀에 어떻게 연결해도 상관없다. 16번 핀을 VCC로 연결하면 H 브릿지가 동작하게 된다. 2번 핀과 7번 핀은 서로 다른 입력 값을 주어 모터의 방향을 결정하게 된다. 예를 들어 2번 핀에는 LOW, 7번 핀에는 HIGH 값을 주거나, 반대로 2번 핀에는 HIGH, 7번 핀에는 LOW 값을 주면 된다. 두 핀에 같은 값을 입력하면 모터가 멈추게 된다. 1번 핀을 통해서도 모터를 구동 시키거나 멈추게 할 수 있다. 1번 핀에 HIGH 값을 주게 되면 모터가 동작을 시작하며, LOW 값을 주게 되면 모터가 멈추게 된다. 1번 핀에 PWM을 연결하여 모터의 속도를 조절한다.



2.3 무작정 따라하기

1. Arduino IDE 설치

2. notepad++ 설치

플러그인>Plugin Manager>Show Plugin Manager>JSTool 설치

플러그인>JSTool>JSFormat(Ctrl+Alt+M)을 이용하여 소스 정렬할 수 있다.

3. Arduino 실행

```
void setup() {  
    // put your setup code here, to run once:  
  
}  
  
void loop() {
```

```
// put your main code here, to run repeatedly:
```

```
}
```

4. setup() 함수 다음과 같이 수정

```
void setup() {  
  // put your setup code here, to run once:  
  Serial.begin(9600);  
  while(!Serial) {  
    ;  
  }  
  Serial.println(__FUNCTION__);  
}
```

5. 도구 > 보드 > Arduino Uno

도구 > 포트 > COM7(Arduino Uno)

체크 표시로 컴파일

화살표 표시로 업로드

도구 > 시리얼 모니터 띄우기

6. 단축키 이용

ctrl+U -> 컴파일/업로드

ctrl+Shift+M -> 시리얼 모니터

7. 파일 > 환경설정

에디터 글꼴 크기 조정

다음 동작 중 자세한 출력 보이기 : 컴파일, 업로드 체크

줄 번호 표시

업로드 후 코드 검증하기 체크 해제

외부 에디터 사용 나중에 체크 (나중에 코드량이 많아지면, 아두이노 에디터 사용 불편)

다른 부분은 그대로!

8. loop() 함수 다음과 같이 수정

```
void loop() {  
  // put your main code here, to run repeatedly:  
  Serial.println(__FUNCTION__);  
}
```

무한히 반복된다.

다음 파일에서 main() 함수 확인

C:\Program Files(x86)\Arduino\hardware\arduino\avr\cores\arduino\main.cpp

```
int main(void)  
{
```

```
  init();
```

```

    setup();

    for(;;) {
        loop();
        if (serialEventRun) serialEventRun();
    }

    return 0;
}

```

다음 함수 확인

```

init();
setup();
loop();
serialEventRun();

```

9. 파일을 다음과 같이 수정하고, arduino_test로 저장

```

void initSystem(){}
void showMenu(){}
void getUserCmd(){}
void processCmd(){}

```

```

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    while(!Serial) {
        ;
    }
    Serial.println(__FUNCTION__);
    initSystem();
    showMenu();
}

```

```

void loop() {
    // put your main code here, to run repeatedly:
    getUserCmd();
    processCmd();
}

```

10. initSystem(), showMenu() 함수를
다음과 같이 수정한 후 실행시켜 본다.

```

void initSystem()
{
    Serial.println(__FUNCTION__);
}

void showMenu()

```

```
{  
  Serial.println(__FUNCTION__);  
}
```

11. getUserCmd(), processCmd() 함수를
다음과 같이 수정한 후 실행시켜 본다.

```
void getUserCmd()  
{  
  Serial.println(__FUNCTION__);  
}  
void processCmd()  
{  
  Serial.println(__FUNCTION__);  
}
```

12. arduino_test 파일을 다음과 같이 수정한다.

```
void initSystem();  
void showMenu();  
void getUserCmd();  
void processCmd();  
  
void setup() {  
  // put your setup code here, to run once:  
  Serial.begin(9600);  
  while(!Serial) {  
    ;  
  }  
  Serial.println(__FUNCTION__);  
  initSystem();  
  showMenu();  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  getUserCmd();  
  processCmd();  
}
```

13. work라는 파일을 새로 추가한다.

그리고 다음과 같이 함수를 작성한다.

```
void initSystem() {}  
void showMenu() {}  
void getUserCmd() {}  
void processCmd() {}
```

14. showMenu() 함수를 다음과 같이 수정한다.

```
void showMenu()
{
    Serial.println(__FUNCTION__);
    Serial.println("1. menu item 1");
    Serial.println("2. menu item 2");
    Serial.println("3. menu item 3");
    Serial.println("\nINPUT> ");
}
```

15. 예제를 하나 새로 열로 다음 예제를 테스트해 본다.

```
int incomingByte = 0;    // for incoming serial data
```

```
void setup() {
    Serial.begin(9600);    // opens serial port, sets data rate to 9600 bps
}
```

```
void loop() {

    // send data only when you receive data:
    while (Serial.available() > 0) {
        // read the incoming byte:
        incomingByte = Serial.read();

        // say what you got:
        Serial.print("I received: ");
        Serial.println(incomingByte, DEC);
    }
}
```

aaa를 입력해본다.

엔터키를 쳐본다.

16. 다음 사이트를 참조해 사용자 입력을 완성한다.

<http://arduino.cc/en/Tutorial/SerialEvent>

```
/*
```

```
    Serial Event example
```

When new serial data arrives, this sketch adds it to a String.
When a newline is received, the loop prints the string and
clears it.

A good test for this is to try it with a GPS receiver
that sends out NMEA 0183 sentences.

Created 9 May 2011
by Tom Igoe

This example code is in the public domain.

<http://www.arduino.cc/en/Tutorial/SerialEvent>

```
*/

String inputString = "";          // a string to hold incoming data
boolean stringComplete = false;  // whether the string is complete

void setup() {
  // initialize serial:
  Serial.begin(9600);
  // reserve 200 bytes for the inputString:
  inputString.reserve(200);
}

void loop() {
  // print the string when a newline arrives:
  if (stringComplete) {
    Serial.println(inputString);
    // clear the string:
    inputString = "";
    stringComplete = false;
  }
}

/*
  SerialEvent occurs whenever a new data comes in the
  hardware serial RX.  This routine is run between each
  time loop() runs, so using delay inside loop can delay
  response.  Multiple bytes of data may be available.
*/
void serialEvent() {
  while (Serial.available()) {
    // get the new byte:
    char inChar = (char)Serial.read();
    // add it to the inputString:
    inputString += inChar;
    // if the incoming character is a newline, set a flag
    // so the main loop can do something about it:
    if (inChar == '\n') {
      stringComplete = true;
    }
  }
}
```

```

    }
}
}

```

17. 사용자 입력을 받기 위한

cmd, inputString, stringComplete 변수를

work 파일의 맨 위에 다음과 같이 추가한다.

```

static char cmd = '?';
static String inputString = "";
static boolean stringComplete = false;

```

18. getUserCmd() 함수를 다음과 같이 수정한다.

```

void getUserCmd()
{
    if (stringComplete) {
        cmd = inputString[0];
        inputString = "";
        if(cmd != '\r' && cmd != '\n') {
            Serial.println(cmd);
        }
        stringComplete = false;
    }
}

```

19. serialEvent() 함수를 processCmd() 함수 아래에
다음과 같이 추가한다.

컴파일하고 입력해 본다.

```

void serialEvent() {
    while (Serial.available()) {
        char inChar = (char)Serial.read();
        inputString += inChar;
        if (inChar == '\n')
            stringComplete = true;
    }
}

```

20. processCmd() 함수를 다음과 같이 수정한다.

```

void processCmd()
{
    Serial.print("\nprocess command : ");
    Serial.println(cmd);
    Serial.print('\n');
}

```

21. processCmd() 함수를 다음과 같이 수정한다.


```
void processCmd()
{
}

```

21. 메뉴 출력과 메뉴 처리를 위한 루틴을 수정해 보자.

먼저 다음과 같이 루틴을 work 파일 맨 위에 추가해 준다.

```
struct MenuFunc {
    bool enable;
    char * pMenu;
    void (* pFunc)();
};

```

```
void func1(){}
void func2(){}
void func3(){}

```

```
MenuFunc menu_funcs[] = {
    {false, "menu item 1", func1},
    {false, "menu item 2", func2},
    {false, "menu item 3", func3},
};

```

23. showMenu() 함수를 다음과 같이 수정한다.

```
void showMenu()
{
    int i;
    Serial.println(__FUNCTION__);
    for(i=0;i<sizeof(menu_funcs)/sizeof(menu_funcs[0]);i++) {
        Serial.print(i+1);
        Serial.print(". ");
        Serial.println(menu_funcs[i].pMenu);
    }
    Serial.println("\nINPUT> ");
}

```

24. getUserCmd() 함수를 다음과 같이 수정한다.

```
void getUserCmd()
{
    if (stringComplete) {
        cmd = inputString[0];
        inputString = "";
        if(cmd != '\r' && cmd != '\n') {
            int cmdIndex = cmd - '0' - 1;
            int howManyCmds = sizeof(menu_funcs)/sizeof(menu_funcs[0]);
            if(cmdIndex>=0 && cmdIndex<howManyCmds) {
                bool * pFuncEn = &menu_funcs[cmdIndex].enable;

```

```

        *pFuncEn = !*pFuncEn;
    }
}
stringComplete = false;
}
}

```

25. processCmd() 함수를 다음과 같이 수정한다.

```

void processCmd() {
    int cmdIndex;
    int howManyCmds = sizeof(menu_funcs)/sizeof(menu_funcs[0]);

    for(cmdIndex=0;cmdIndex<howManyCmds;cmdIndex++)
        if(menu_funcs[cmdIndex].enable) menu_funcs[cmdIndex].pFunc();
}

```

26. func1(),func2(),func3() 함수를 다음과 같이 수정한다.

```

void func1(){Serial.println(__FUNCTION__);}
void func2(){Serial.println(__FUNCTION__);}
void func3(){Serial.println(__FUNCTION__);}

```

27. funcs 파일을 추가하고

work 파일의 다음 부분을 옮긴다.

```

struct MenuFunc {
    bool enable;
    char * pMenu;
    void (* pFunc)();
};

```

```

void func1(){Serial.println(__FUNCTION__);}
void func2(){Serial.println(__FUNCTION__);}
void func3(){Serial.println(__FUNCTION__);}

```

```

MenuFunc menu_funcs[] = {
    {false, "menu item 1", func1},
    {false, "menu item 2", func2},
    {false, "menu item 3", func3},
};

```

28. work 파일에서 funcs 파일로

옮겨진 부분은 다음과 같이 대체한다.

```

struct MenuFunc;
extern MenuFunc menu_funcs[];

```

29. 4 번 항목을 다음과 같이 추가해 본다.

```
struct MenuFunc {
    bool enable;
    char * pMenu;
    void (* pFunc)();
};

void func1(){Serial.println(__FUNCTION__);}
void func2(){Serial.println(__FUNCTION__);}
void func3(){Serial.println(__FUNCTION__);}
void func4(){Serial.println(__FUNCTION__);}
```

```
MenuFunc menu_funcs[] = {
    {false, "menu item 1", func1},
    {false, "menu item 2", func2},
    {false, "menu item 3", func3},
    {false, "menu item 4", func4},
};
```

30. funcs 파일을 다음과 같이 수정한다.

```
struct MenuFunc {
    bool enable;
    char * pMenu;
    void (* pFunc)();
};

void func1();
void func2();
void func3();
void func4();

MenuFunc menu_funcs[] = {
    {false, "menu item 1", func1},
    {false, "menu item 2", func2},
    {false, "menu item 3", func3},
    {false, "menu item 4", func4},
};

void func1(){Serial.println(__FUNCTION__);}
void func2(){Serial.println(__FUNCTION__);}
void func3(){Serial.println(__FUNCTION__);}
void func4(){Serial.println(__FUNCTION__);}
```

31. Bluetooth Car 기능을 추가하기 위해 work 파일을 다음과 같이 수정한다.

```
struct MenuFunc {
```

```

    bool enable;
    char * pMenu;
    void (* pFunc)();
};

void controlCar();
void func2();
void func3();
void func4();

MenuFunc menu_funcs[] = {
    {false, "Arduino Controlled Bluetooth Car", controlCar},
    {false, "menu item 2", func2},
    {false, "menu item 3", func3},
    {false, "menu item 4", func4},
};

void controlCar(){Serial.println(__FUNCTION__);}
void func2(){Serial.println(__FUNCTION__);}
void func3(){Serial.println(__FUNCTION__);}
void func4(){Serial.println(__FUNCTION__);}

```

32. control_car 함수를 하나 추가하고 다음 함수를 옮겨온다.
void controlCar(){Serial.println(__FUNCTION__);}

33. controlCar() 함수를 다음과 같이 수정한다.

```

void car_init();
char get_dir();
void go_forward();
void go_backward();
void turn_left();
void turn_right();
void stop_car();

void controlCar()
{
    char dir = 's';

    Serial.println("\n\n==Arduino Bluetooth Car Control==");
    Serial.println("f. go forward");
    Serial.println("b. go backward");
    Serial.println("l. trun left");
    Serial.println("r. trun right");
    Serial.println("s. stop");
    Serial.println("q. Exit");
}

```

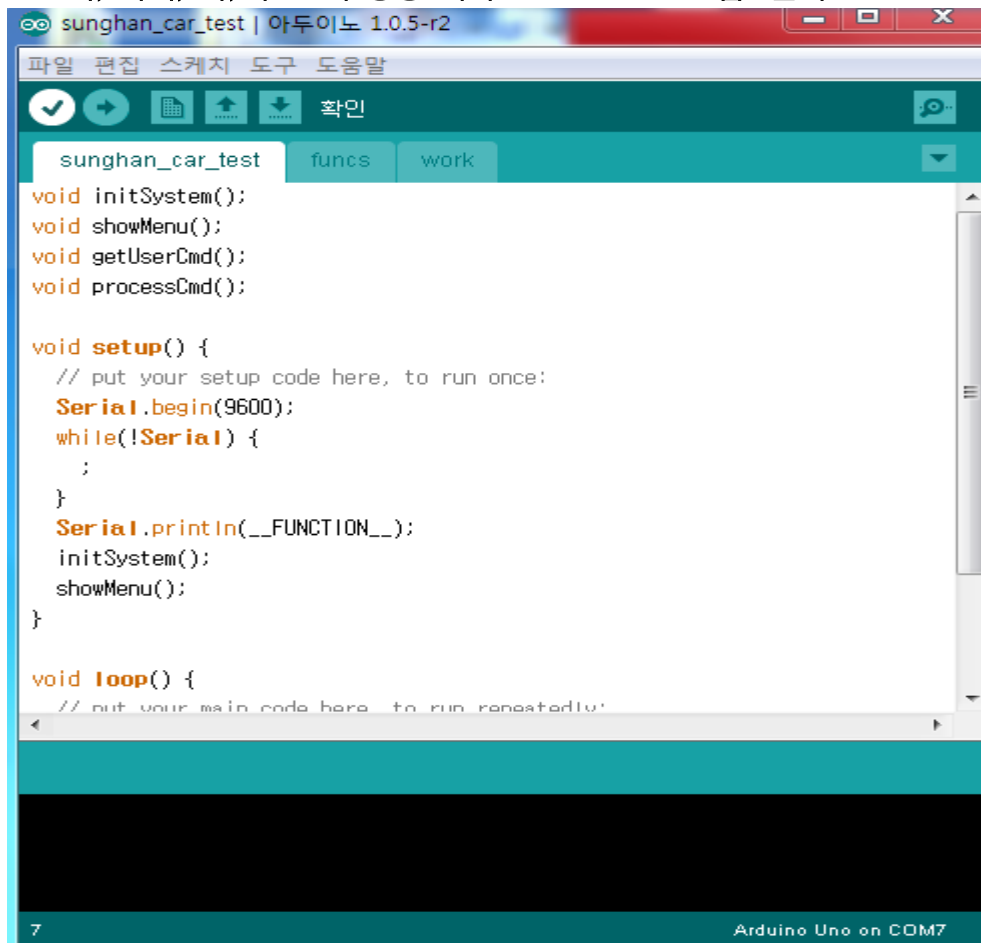
```
Serial.println("\nINPUT> ");

car_init();

while(1) {
    dir = get_dir();
    switch(dir) {
        case 'f': go_forward(); break;
        case 'b': go_backward(); break;
        case 'l': turn_left(); break;
        case 'r': turn_right(); break;
        case 's': stop_car(); break;
        case 'q': return;
        default: break;
    }
}

void car_init(){}
char get_dir(){}
void go_forward(){}
void go_backward(){}
void turn_left(){}
void turn_right(){}
void stop_car(){}
}
```

2.4 (위, 아래, 좌, 우 모터 방향 제어 Source Code 포함) 전체 소스 코드

A screenshot of the Arduino IDE interface. The title bar shows 'sunghan_car_test | 아두이노 1.0.5-r2'. The menu bar includes '파일', '편집', '스케치', '도구', and '도움말'. The toolbar has icons for opening files, saving, compiling, uploading, and verifying. The 'sunghan_car_test' sketch is open, with tabs for 'funcs' and 'work'. The code is as follows:

```
void initSystem();
void showMenu();
void getUserCmd();
void processCmd();

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  while(!Serial) {
    ;
  }
  Serial.println(__FUNCTION__);
  initSystem();
  showMenu();
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

The status bar at the bottom indicates '7' and 'Arduino Uno on COM7'.

<arduino_test>

```
void initSystem();
void showMenu();
void getUserCmd();
void processCmd();

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  while(!Serial) {
    ;
  }
  Serial.println(__FUNCTION__);
  initSystem();
  showMenu();
}

void loop() {
```

```

    // put your main code here, to run repeatedly:
    getUserCmd();
    processCmd();
}

<funcs>

const int dirPin1=2;    //moter_1
const int dirPin2=3;    //moter_1
const int dirPin3=4;    //moter_2
const int dirPin4=5;    //moter_2
const int speedPin1=9;  //moter_1 speed control pin
const int speedPin2=10; //moter_2 speed control pin
int motorSpeed=175;
int motorDir=1;
char inChar;

struct MenuFunc {
    bool enable;
    char * pMenu;
    void (* pFunc)();
};

//프로토타입만 정렬 - - 가독성 - overview효과
void controlCar();
void func2();
void func3();
void func4();
void getDir();

MenuFunc menu_funcs[] = {
    {false, "Arduino Controlled Bluetooth Car", controlCar},
    {false, "menu item 2", func2},
    {false, "menu item 3", func3},
    {false, "menu item 4", func4},
};

void controlCar()
{
    Serial.println("==Arduino Bluetooth Car Contril==");
    Serial.println("w. go forward");
    Serial.println("s. go backward");
    Serial.println("a. turn left");
    Serial.println("d. turn right");
    Serial.println("x. stop");
    Serial.println("q. Exit");
}

```

```

Serial.println("\nINPUT> ");

while(1){
    getDir();
    switch(inChar){
        case 'w': go_forward(); break;
        case 's': go_backward(); break;
        case 'a': turn_left(); break;
        case 'd': turn_right(); break;
        case 'x': stop_car(); break;
        case 'q': showMenu();
        inChar = 0; return;//continue; break;
        default : break;
    }
}

void getDir(){
    while (Serial.available()) {
        inChar = (char)Serial.read();
    }
}

void go_forward() {
    pinMode(dirPin1,OUTPUT);
    pinMode(dirPin2,OUTPUT);
    pinMode(dirPin3,OUTPUT);
    pinMode(dirPin4,OUTPUT);
    motorDir = 1;
    setMotorDir(motorDir);
    setMotorSpeed1(motorSpeed);
    setMotorSpeed2(motorSpeed);
}

void go_backward() {
    pinMode(dirPin1,OUTPUT);
    pinMode(dirPin2,OUTPUT);
    pinMode(dirPin3,OUTPUT);
    pinMode(dirPin4,OUTPUT);
    motorDir = 0;
    setMotorDir(motorDir);
    setMotorSpeed1(motorSpeed);
    setMotorSpeed2(motorSpeed);
}

void turn_left() {
    pinMode(dirPin1,OUTPUT);
    pinMode(dirPin2,OUTPUT);
    pinMode(dirPin3,INPUT);

```



```

    pinMode(dirPin4,INPUT);
    motorDir = 0;
    setMotorDir(motorDir);
    setMotorSpeed1(motorSpeed);
}

void turn_right() {
    pinMode(dirPin1,INPUT);
    pinMode(dirPin2,INPUT);
    pinMode(dirPin3,OUTPUT);
    pinMode(dirPin4,OUTPUT);
    motorDir = 0;
    setMotorDir(motorDir);
    setMotorSpeed2(motorSpeed);
}

void stop_car() {
    setMotorSpeed1(0);
    setMotorSpeed2(0);
}

void func2(){Serial.println(__FUNCTION__);}
void func3(){Serial.println(__FUNCTION__);}
void func4(){Serial.println(__FUNCTION__);}

void setMotorDir(int motorDir) {
    if(motorDir==1) {
        digitalWrite(dirPin1,HIGH);
        digitalWrite(dirPin3,HIGH);
        digitalWrite(dirPin2,LOW);
        digitalWrite(dirPin4,LOW);
    } else {
        digitalWrite(dirPin1,LOW);
        digitalWrite(dirPin3,LOW);
        digitalWrite(dirPin2,HIGH);
        digitalWrite(dirPin4,HIGH);
    }
}

void setMotorSpeed1(int motorSpeed) {
    analogWrite(speedPin1,motorSpeed);
}

void setMotorSpeed2(int motorSpeed) {
    analogWrite(speedPin2,motorSpeed);
}

```

<work>

```
static char cmd = '?';
static String inputString = "";
static boolean stringComplete = false;

struct MenuFunc;
extern MenuFunc menu_funcs[];

void initSystem() {}
void showMenu()
{
    int i;
    // Serial.println(__FUNCTION__);
    for(i=0;i<sizeof(menu_funcs)/sizeof(menu_funcs[0]);i++) {
        Serial.print(i+1);
        Serial.print(" ");
        Serial.println(menu_funcs[i].pMenu);
    }
    Serial.println("\nINPUT> ");
}

void getUserCmd()
{
    if (stringComplete) {
        cmd = inputString[0];
        inputString = "";
        if(cmd != '\r' && cmd != '\n') {
            int cmdIndex = cmd - '0' - 1;
            int howManyCmds = sizeof(menu_funcs)/sizeof(menu_funcs[0]);
            if(cmdIndex>=0 && cmdIndex<howManyCmds) {
                bool * pFuncEn = &menu_funcs[cmdIndex].enable;
                *pFuncEn = !*pFuncEn;
            }
        }
        stringComplete = false;
    }
}

void processCmd() {
    int cmdIndex;
    int howManyCmds = sizeof(menu_funcs)/sizeof(menu_funcs[0]);

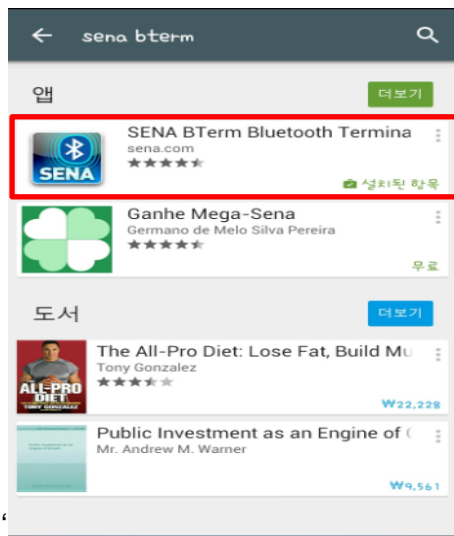
    for(cmdIndex=0;cmdIndex<howManyCmds;cmdIndex++)
        if(menu_funcs[cmdIndex].enable) {
            menu_funcs[cmdIndex].pFunc();
        }
}
```

```
        bool * pFuncEn = &menu_funcs[cmdIndex].enable;
        *pFuncEn = !*pFuncEn;
    }
}

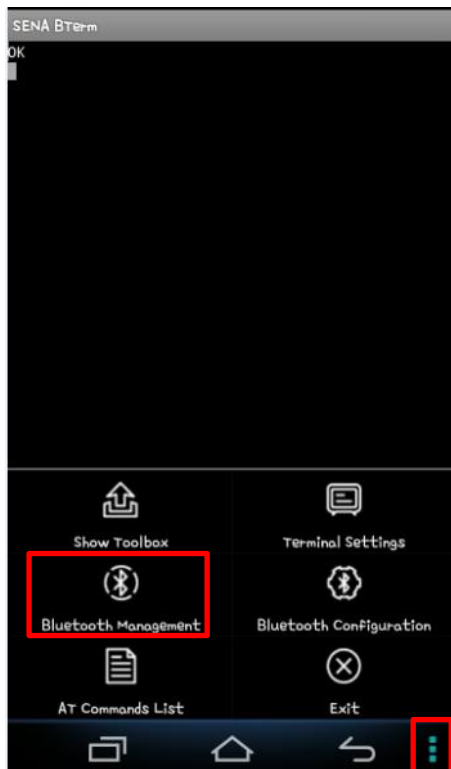
void serialEvent() {
    while (Serial.available()) {
        char inChar = (char)Serial.read();
        inputString += inChar;
        //if (inChar == '\n')
            stringComplete = true;
    }
}
```

2.5 Bluetooth로 실행하기

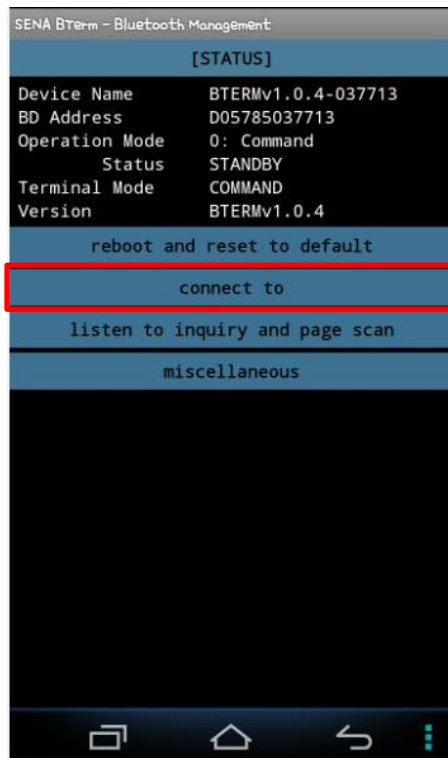
① Plaly Store에서 SENA BTerm 다운로드



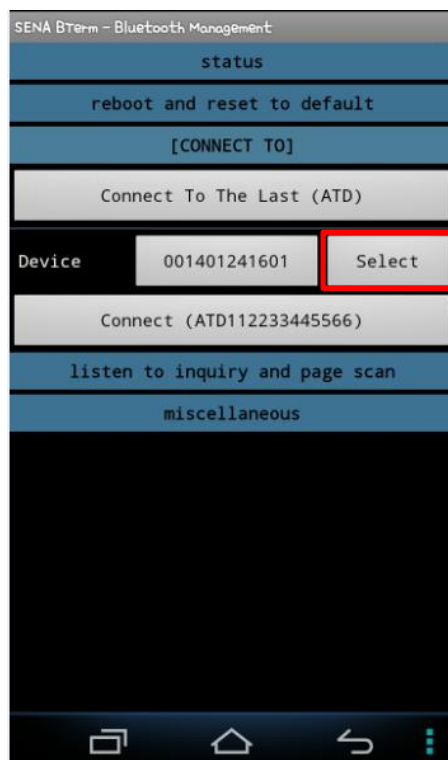
② 메뉴 클릭 후, Bluetooth Management 클릭



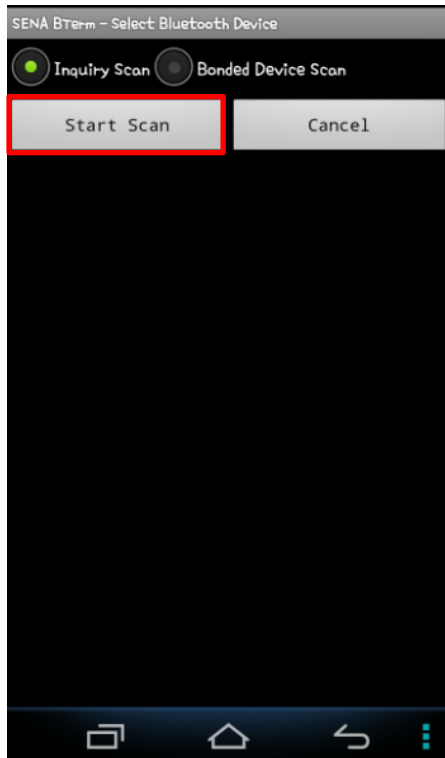
③ Connect to 클릭



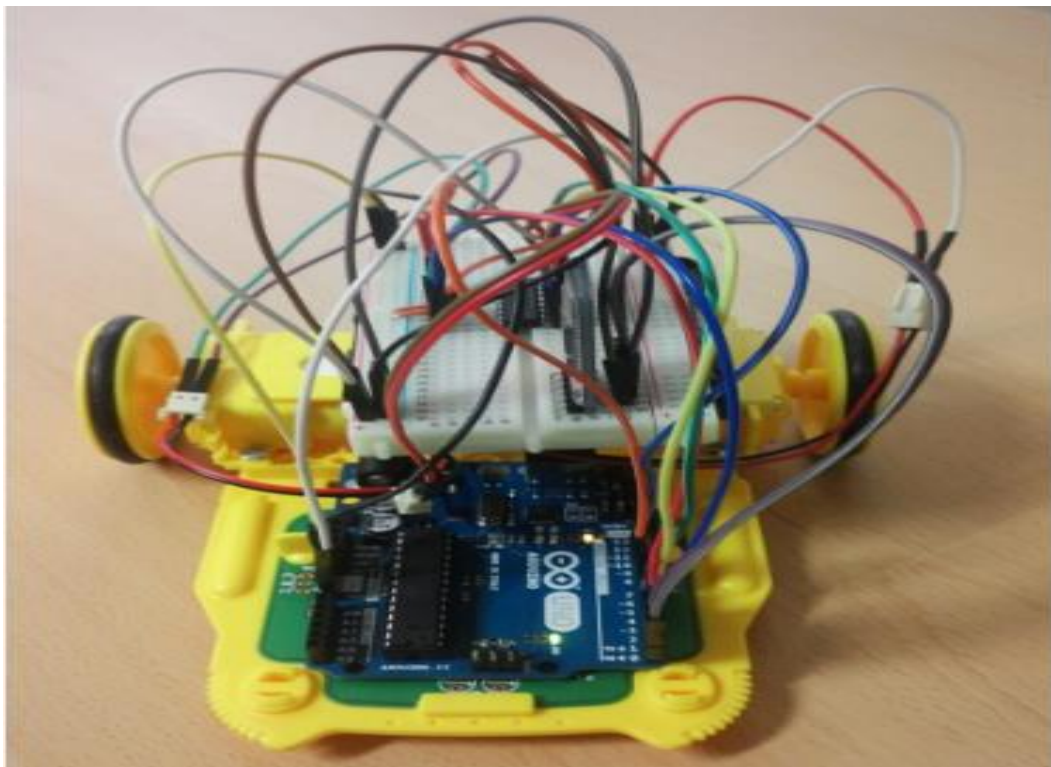
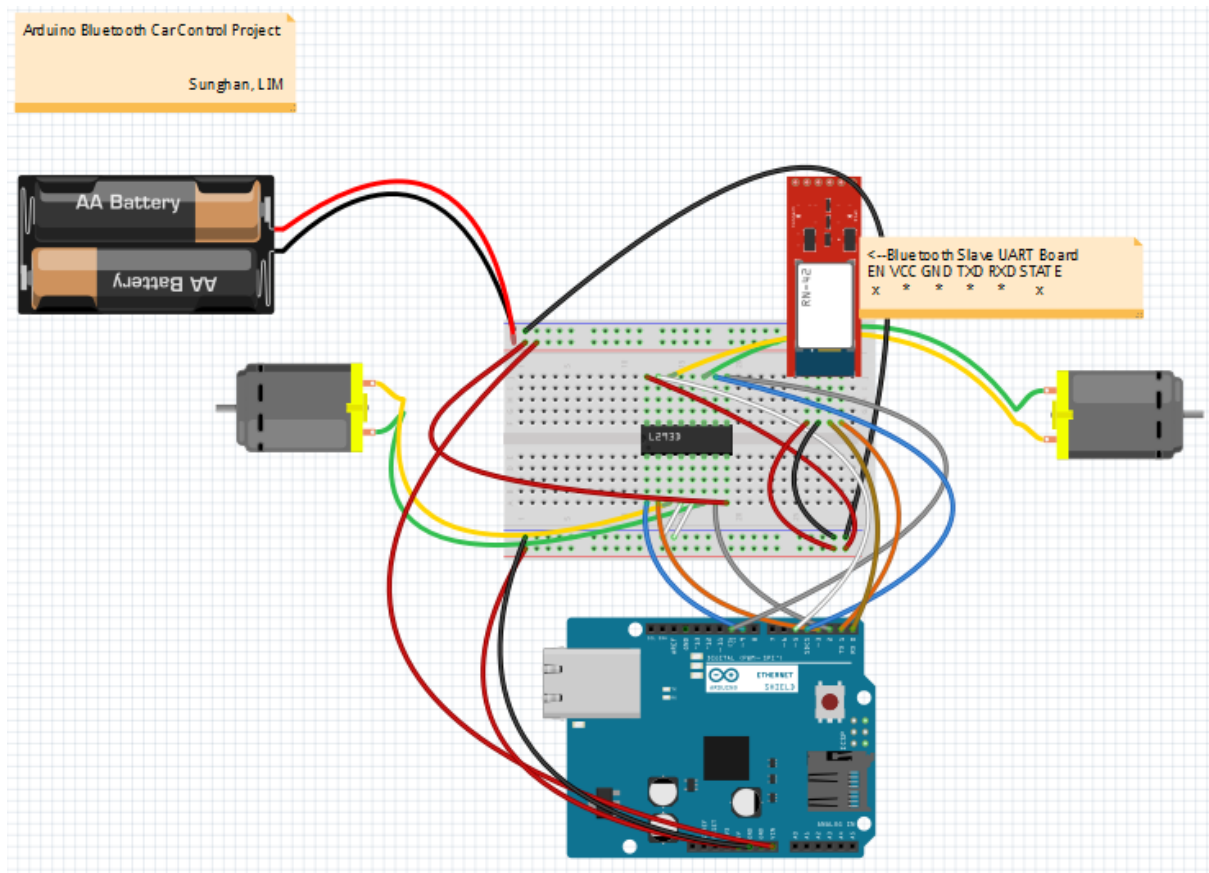
④ Select 선택



⑤ Start Scan을 클릭 후, 해당 Bluetooth 모듈 주소 연결



3. 회로 구성도



* 참고 동영상 : <http://www.youtube.com/watch?v=SZcSKnl0724>

4. 참고문헌

- ✓ 레시피로 배우는 아두이노 쿡북 - 마이클 마골리스
- ✓ 아두이노와 안드로이드로 45개 프로젝트 만들기 - 서민우