

Chapter 1. 아두이노

1. 아두이노 하드웨어

최근 마이크로컨트롤러와 관련하여 가장 주목을 받는 단어는 아두이노가 아닐까 싶다. 하지만 아두이노가 무엇인지 정의하기는 쉽지 않다. 아두이노를 한 마디로 정의하기 어려운 이유는 아두이노가 마이크로컨트롤러를 이용하여 만들어진 개발 보드인 하드웨어와, 하드웨어를 동작 시키기 위해 필요한 개발 환경까지 함께 가리키기 때문이다. 먼저 하드웨어에 대해 살펴보자. 아두이노는 ‘마이크로컨트롤러를 사용하여 만들어진 개발 보드’라고 정의할 수 있다. 주의할 점은 ‘개발 보드’이지 ‘마이크로컨트롤러’가 아니라는 점이다. 마이크로컨트롤러는 컴퓨터의 본체에 해당하는 기능을 하나의 칩으로 집약시켜 놓은 낮은 성능의 컴퓨터를 말한다. 컴퓨터의 본체에 전원만 꽂아주면 컴퓨터로 기능할 수 있는 것처럼 마이크로컨트롤러 역시 전원만 주어진다면 마이크로컨트롤러로서 동작할 수 있다. 하지만 아두이노에는 마이크로컨트롤러 이외에 많은 것들이 추가되어 있다. 아두이노 중 가장 많이 사용되는 아두이노 우노를 살펴보다 라도 아두이노 우노에서 사용된 ATmega328 마이크로컨트롤러 이외에도 여러 가지 부품들이 자리하고 있음을 발견할 수 있다.

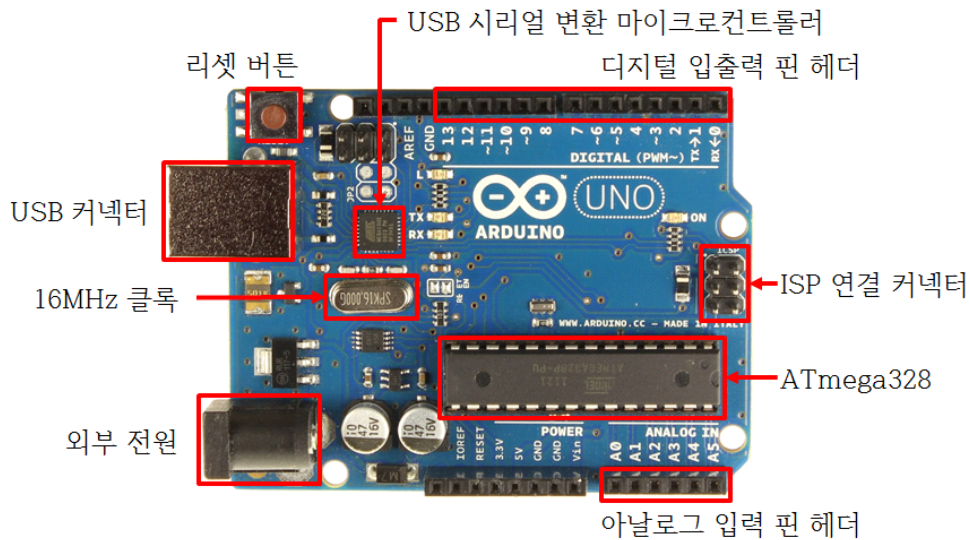


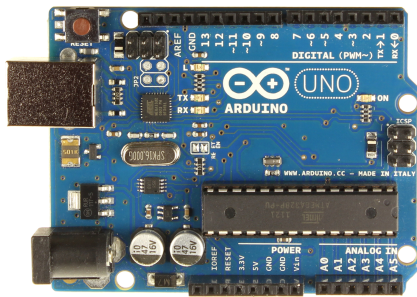
그림 1. 아두이노 우노¹⁾

1) 이미지 출처: <http://arduino.cc/en/Main/ArduinoBoardUno>

아두이노는 마이크로컨트롤러를 위한 개발보드이다. 아두이노 우노 보드에는 핵심이라 할 수 있는 ATmega328 마이크로컨트롤러 이외의 부가적인 부분들이 더 많은 부분을 차지하고 있다. 이처럼 아두이노는 ‘마이크로컨트롤러 + α = 마이크로컨트롤러 보드’를 의미한다.

또 한 가지 아두이노를 이해하기 어렵게 만드는 점은 아두이노가 한 가지 개발 보드만을 지칭하는 것이 아니라는 점이다. 아두이노의 홈페이지²⁾를 살펴보면 10여 종류가 넘는 보드들이 나열되어 있으며 이들 보드는 크기와 형태가 다양하여 외형적으로 공통점을 쉽게 찾아볼 수 없다. 아두이노의 공통점은 공통된 개발환경, 즉 동일한 프로그램을 사용하여 프로그램을 개발할 수 있다는 점에서 찾아야 한다. 마이크로컨트롤러는 종류에 따라 프로그램을 작성하는 방법은 조금씩 다르다. 하지만 아두이노라고 이름 붙은 개발 보드들은 모두 동일한 방식으로 프로그램을 작성하고 동작시킬 수 있다는 점이 아두이노의 가장 큰 장점이라 할 수 있다.

아두이노를 특징짓는 요소 중 한 가지가 개발 환경에 있는 것은 사실이지만 이는 하드웨어의 지원이 바탕이 되어야 하는 것은 당연하다. 아두이노 보드들은 형태에 따라 핀 헤더가 존재하는 보드와 존재하지 않는 보드로 나누어 볼 수 있다.



(a) 아두이노 우노³⁾



(b) 아두이노 나노⁴⁾

그림 2. 아두이노 보드의 종류 1

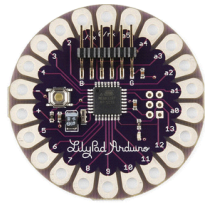
그림 2의 아두이노 우노와 아두이노 나노는 동일한 마이크로컨트롤러를 사용하여 만들어진 보드이다. 동일한 마이크로컨트롤러를 사용한다면 굳이 서로 다른 보드를 만들 필요는 무엇일까? 바로 사용 목적의 차이이다. 아두이노 우노는 아두이노의 대표적인 보드 중 하나로 ‘개발 보드’의 목적에 충실하게 주변장치나 컴퓨터와 간편하기 연결할 수 있도록 다양한 부가장치들이 포함되어 있다. 하지만 아두이노 나노는 개발 목적이라기보다는 개발이 완료된 후 실제 적용하는 단계에서 사용할 수 있도록 아두이노 우노와 동일한 기능을 하는 보드를 소형으로 만

2) <http://arduino.cc/en/Main/Products>

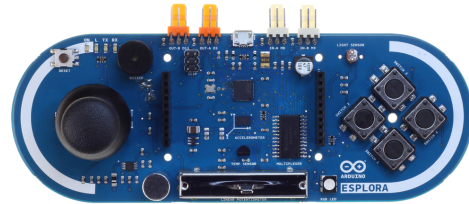
3) 이미지 출처: <http://arduino.cc/en/Main/ArduinoBoardUno>

4) 이미지 출처: <http://arduino.cc/en/Main/ArduinoBoardNano>

들어 놓은 것에 해당한다. 이외에도 웨어러블 기기 개발을 위한 릴리패드 아두이노, 센서가 결합되어 있는 아두이노 에스플로라 등 다양한 전용 보드들이 존재한다.



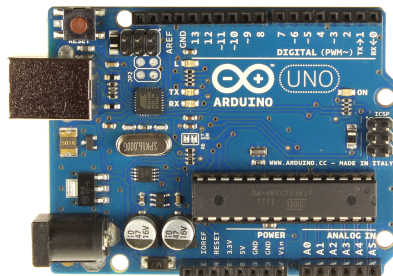
(a) 릴리패드 아두이노⁵⁾



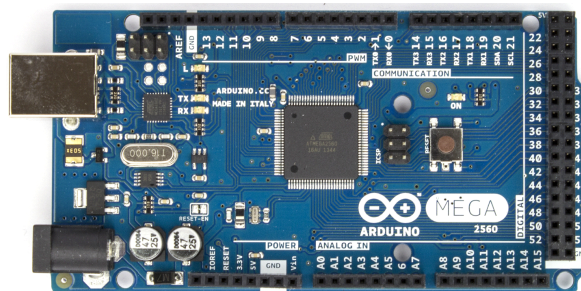
(b) 아두이노 에스플로라⁶⁾

그림 3. 전용 목적의 아두이노 보드

아두이노 보드들의 또 다른 공통점을 찾아내었는가? 핀 헤더의 위치를 유심히 살펴보자.



(a) 아두이노 우노⁷⁾



(b) 아두이노 메가2560⁸⁾

그림 4. 아두이노 보드의 종류 2

5) 이미지 출처: <http://arduino.cc/en/Main/ArduinoBoardLilyPad>

6) 이미지 출처: <http://arduino.cc/en/Main/ArduinoBoardEsplora>

7) 이미지 출처: <http://arduino.cc/en/Main/ArduinoBoardUno>

8) 이미지 출처: <http://arduino.cc/en/Main/ArduinoBoardMega2560>

아두이노 메가2560에 사용된 마이크로컨트롤러는 아두이노 우노에 사용된 마이크로컨트롤러에 비해 더 많은 수의 입출력 핀을 제공하고 있다. 하지만 그림 4에서 아두이노 메가2560 보드의 왼쪽에 있는 핀 헤더들은 아두이노 우노와 동일한 배열을 가지고 있다. 이러한 핀 배열의 공통점은 아두이노 보드들에 공통으로 사용할 수 있는 확장 보드인 쉴드(shield) 제작을 가능하게 하였다. 아두이노 홈페이지에도 여러 종류의 쉴드가 소개되고 있으며 그림 5는 그 중 일부를 보여주고 있다.

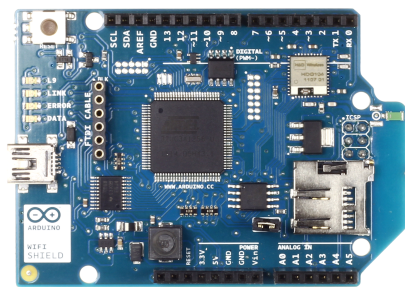
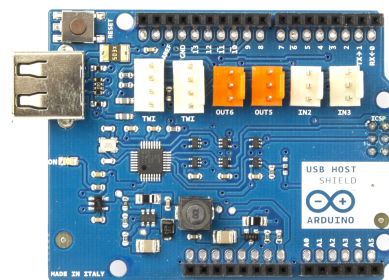
(a) 와이파이 쉴드⁹⁾(b) USB 호스트 쉴드¹⁰⁾

그림 5. 아두이노 쉴드

이외에도 아두이노 보드들은 아트멜(Atmel)에서 제작한 마이크로컨트롤러를 사용한다는 공통점이 있다. 아트멜에서 제작하고 아두이노에 사용된 마이크로컨트롤러 역시 크게 두 종류로 나눌 수 있다. 아두이노 우노, 아두이노 레오나르도, 아두이노 메가2560 등은 아트멜의 AVR 시리즈 마이크로컨트롤러를 사용하고 있다. 반면 아두이노 듀에와 아두이노 제로는 Cortex-M 마이크로컨트롤러를 사용하고 있다. 애초 아두이노는 아트멜의 AVR 시리즈 마이크로컨트롤러를 사용하여 제작되었다. 하지만 높은 사양의 마이크로컨트롤러에 대한 요구가 증가함에 따라 Cortex-M 기반의 아두이노 보드 역시 출시되고 있다. 표 1은 아트멜에서 제작하고 아두이노에서 사용되고 있는 마이크로컨트롤러의 종류를 요약한 것이다.

9) 이미지 출처: <http://arduino.cc/en/Main/ArduinoWiFiShield>

10) 이미지 출처: <http://arduino.cc/en/Main/ArduinoUSBHostShield>

아두이노	마이크로컨트롤러	아키텍처	CPU 비트 수
우노	ATmega328	AVR	8비트
레오나르도	ATmega32u4	AVR	
메가2560	ATmega2560	AVR	
제로	ATSAMD21G18	Cortex M0+	32비트
듀에	AT91SAM3X8E	Cortex M3	

표 1. 아두이노 보드의 종류 3

2. 아두이노 우노

아두이노에 많은 보드들이 있지만 가장 많이 사용되는 보드는 아두이노 우노로 아두이노의 기본 보드로 자리 잡고 있다. 아두이노 우노는 핀 헤더를 통해 20개의 입출력 핀을 사용할 수 있다. 마이크로컨트롤러의 기본 기능은 디지털 또는 아날로그 데이터를 입출력하는 기능이다. 디지털 데이터의 입출력을 위해 아두이노에서는 0번에서 19번까지 핀 단위로 번호를 지정하여 사용한다. 아날로그 데이터의 경우에는 마이크로컨트롤러에서 직접 처리할 수 없으므로 디지털 데이터로의 변환이 먼저 이루어져야 하며 이를 위해 ADC(Analog-Digital Converter)가 필요하다. ATmega328에는 10비트 해상도에 6 채널의 ADC가 포함되어 있고 A0에서 A5까지 상수가 지정되어 있다. 한 가지 주의할 점은 디지털 입출력을 위한 핀 중 14번부터 19번까지 핀에 A0부터 A5까지 상수가 할당되어 있다는 점이다. 즉, 디지털 입출력 핀의 '최대' 개수가 20개이다.

아날로그 데이터 출력을 위해서는 ADC와는 반대로 DAC(Digital-Analog Converter)가 필요하지만 ATmega328에는 DAC가 포함되어 있지 않으므로 아날로그 신호를 직접 출력할 수는 없다. 대신 PWM (Pulse Width Modulation) 신호를 통해 아날로그와 유사한 기능을 하는 디지털 신호를 출력할 수 있다. PWM 신호는 디지털 입출력 핀 중 6개의 핀을 통해서만 사용 가능하며 핀 번호 앞에 물결표(~)를 붙여 PWM 신호 출력이 가능한 핀을 구별하고 있다.

항목	내용	비고
마이크로컨트롤러	ATmega328	ATmega328P-PU
동작 전압	5V	
입력 전압	7V~12V	추천 입력 범위
디지털 입출력 핀	14개	6개 PWM 출력 핀 (3, 5, 6, 9, 10, 11번 핀)
아날로그 입력 핀	6개	14번에서 19번까지의 디지털 입출력 핀
플래시 메모리	32KB	ATmega328, 부트로더 0.5KB
SRAM	2KB	ATmega328
EEPROM	1KB	ATmega328
클록 주파수	16MHz	

표 2. 아두이노 우노

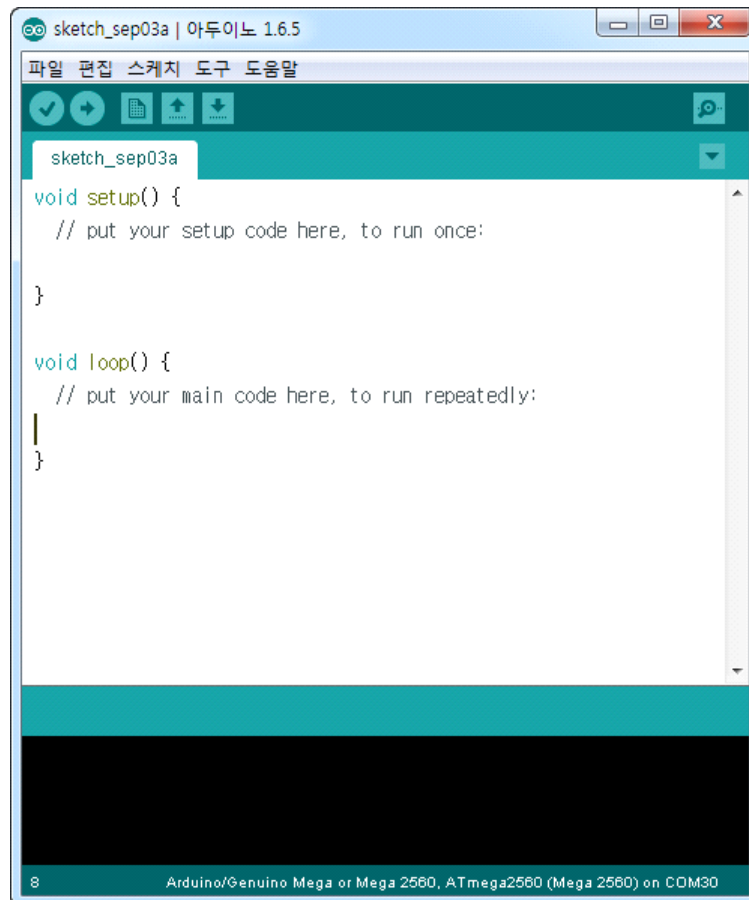
아두이노가 다른 마이크로컨트롤러 보드와 또 다른 점은 프로그램 업로드 방법에 있다. 일반적으로 마이크로컨트롤러에 프로그램 업로드는 ISP (In System Programming) 방식으로 이루어지며 아두이노에서도 ISP 방식의 업로드가 가능하다. 하지만 일반적으로 아두이노에서는 부트로더(bootloader)를 통해 시리얼 방식으로 프로그램을 업로드 한다. 표 2에서도 알 수 있듯이 32 KByte의 플래시 메모리 중 부트로더가 0.5 KByte를 차지하고 있다. 하지만 부트로더만 있다고 프로그램 업데이트가 가능한 것은 아니며 하드웨어적인 지원이 필요하다. 그림 1에서 USB-시리얼 변환을 위해 ATmega16u2 마이크로컨트롤러가 포함되어 있는 것을 확인할 수 있다. 프로그램 업로드에 사용되지 않는 경우 ATmega16u2 마이크로컨트롤러는 컴퓨터와 UART 시리얼 통신을 수행하기 위해 사용될 수 있다. 아두이노 우노 보드가 준비되었다면 필요한 것은 아두이노 우노의 USB 커넥터와 컴퓨터의 USB 커넥터를 연결하는 것뿐이다. 물론 아두이노 IDE를 설치하고 컴퓨터와의 통신을 위해 드라이버가 설치되어 있어야 한다.

3. 아두이노 소프트웨어

아두이노에는 다양한 보드들이 존재하지만 모두 동일한 환경에서 프로그램을 개발할 수 있는 점이 아두이노의 장점 중 하나라고 이야기 하였다. 아두이노 개발 환경은 통합 개발 환경 (IDE, Integrated Development Environment)과 그 아래에 숨겨진 아두이노 함수 및 라이브러리로 이루어진다. 먼저 아두이노 IDE를 살펴보자. 아두이노 IDE는 아두이노 홈페이지에서 무료로 다운받을 수 있다¹¹⁾. 현재 아두이노 정식 버전은 1.6.x이며¹²⁾ Windows를 포함하여 다양한 운영체제를 지원하므로 운영체제에 맞는 프로그램을 다운받아 설치하면 된다¹³⁾. 윈도

우즈 환경을 사용한다면 인스톨 프로그램 역시 제공하고 있다. 인스톨 파일의 경우 IDE 설치 이외에 드라이버 설정 등을 자동으로 해주므로 윈도우즈 환경을 사용한다면 인스톨 파일을 추천한다. 인스톨 파일을 다운받아 설치해 보자. 기본 옵션으로 설치하고 장치 드라이버 설치 메시지가 나타나면 ‘설치’ 버튼만 눌러주면 설치는 끝난다.

설치가 끝났으면 아두이노 우노나 아두이노 메가2560 보드를 컴퓨터에 연결해 보자. 처음 아두이노 우노나 아두이노 메가2560 보드를 컴퓨터와 연결하였다면 드라이버 설정을 통해 추가 작업 없이 자동으로 인식된다. 설치된 프로그램을 실행시켜 보면 간단한 IDE를 확인할 수 있다. C 언어를 사용해본 경험이 있다면 아두이노 IDE가 어딘가 허전해 보일 수도 있지만, 아두이노는 비전문가들을 위한 마이크로컨트롤러 보드로 시작되었으므로 필요한 기본 기능들만을 제공하도록 간단하게 만들어져 있다는 점을 기억하자.



11) <http://arduino.cc/en/Main/Software> 아두이노 IDE는 arduino.cc 이외에도 arduino.org에서도 다운받을 수 있다. 하지만 두 사이트는 별개의 사이트로 독자적으로 운영되고 있다. 이 책에서는 arduino.cc에서 다운 받은 IDE를 기준으로 설명한다.

12) 2015년 9월 기준

13) 이 책의 모든 스케치들은 Windows 7, 64bit 환경에서 테스트되었다.

그림 6. 아두이노 통합 개발 환경

간단한 통합 개발 환경과 함께 서로 다른 아두이노 보드들을 동일한 방식으로 프로그래밍 할 수 있도록 해주는 아두이노의 함수와 라이브러리는 아두이노의 가장 큰 장점 중 하나이다. 아두이노 메가2560의 13번 핀에 연결된 LED를 켜는 프로그램을 ATmega2560의 레지스터를 통해 구현하면 다음과 같다.

```
DDRB |= 0x80;  
PORTB |= 0x80;
```

동일한 기능을 아두이노에서 제공하는 추상화된 함수를 통해 구현하면 다음과 같다.

```
pinMode(13, OUTPUT);  
digitalWrite(13, HIGH);
```

어느 것이 더 이해하기 쉬운가? 아두이노 우노의 13번 핀에 연결된 LED를 켜는 프로그램을 ATmega328의 레지스터를 통해 구현한 예를 살펴보자.

```
DDRB |= 0x20;  
PORTB |= 0x20;
```

ATmega2560과 비교하면 약간의 차이가 있다. 하지만 아두이노의 추상화된 함수를 사용하면 아두이노 메가2560과 아두이노 우노는 동일한 코드를 통해 디지털 13번 핀에 연결된 LED를 켜는 것이 가능하다.

추상화된 함수와 더불어 아두이노에서는 주변장치를 쉽게 사용할 수 있도록 다양한 라이브러리를 제공하고 있다. EEPROM의 100번지에 10의 값을 쓰고 싶다면 다음과 같이 한 문장으로 충분하다.

```
EEPROM.write(100, 10);          // (번지, 값)
```

물론 EEPROM 클래스의 이면에서는 많은 일들이 벌어지고 있지만 프로그래머가 신경 써야 할 부분은 그리 많지 않다. 아두이노가 설치된 디렉터리 아래의 'libraries' 디렉터리를 살펴보면 사용하고자 하는 주변장치에 따라 별도의 디렉터리로 만들어진 라이브러리들을 확인할 수 있다. 겨우 10여개의 라이브러리뿐이어서 실망할 수도 있겠지만 실망하기는 이르다. 아두이노 설치 디렉터리 아래 존재하는 라이브러리들은 기본 라이브러리라고 불리며 아두이노에서 공식적으로 제공하는 라이브러리들이다. 하지만 하드웨어의 기능을 확장하기 위해 설드를 제작하여 사용하는 것과 마찬가지로, 기본 라이브러리에서 제공하지 않는 기능들 역시 라이브러리로 만들어 사용할 수 있으며 이를 확장 라이브러리라고 한다. 확장 라이브러리는 기본 라이브러리 디렉터리에 설치하여 사용할 수도 있지만 스케치북 디렉터리 아래 별도의 libraries 디렉터리가 존재하므로 그 아래에 설치하는 것이 일반적이다. 스케치북 디렉터리는 '파일 → 환경 설정' 메뉴에서 확인할 수 있다. 아두이노를 위해 작성한 프로그램은 '스케치(sketch)'라고 불린다. 스케치북은 스케치들을 모아놓은 것을 말한다. 여러 개의 스케치들은 스케치북 디렉터리 아래에 라이브러리와 마찬가지로 디렉터리 별로 저장되고 관리된다. 스케치북 디렉터리 아래에는 libraries 디렉터리가 자동으로 생성되며, 확장 라이브러리가 설치되는 위치가 바로 그곳이다.

아두이노는 애초 비전문가들을 위한 마이크로컨트롤러로 만들어졌다. 간단한 하드웨어와 간단한 IDE, 그리고 추상화된 함수를 통해 쉽고 빠르게 프로토타입을 만들어볼 수 있는 점이 아두이노의 가장 큰 장점이다. 하지만 간과할 수 없는 부분이 바로 확장 라이브러리이다. 지금도 누군가 기존 라이브러리보다 더 편리한 라이브러리를 만들기 위해 고민에 쌓여 있을 것이며 아두이노의 영토는 점점 넓어지고 있다.

4. 아두이노 프로그래밍

아두이노 메가2560을 USB 연결선으로 컴퓨터와 연결하고 프로그램 설치도 끝났다면 아두이노를 시작할 준비는 끝났다. 아두이노 프로그램을 실행하고 '파일 → 예제 → 01.Basics → Blink'를 선택해 보자. Blink 예제는 13번 핀에 연결된 내장 LED를 1초 간격으로 점멸하는 스케치에 해당한다.



그림 7. Blink 예제

스케치를 컴파일하고 아두이노 우노로 업로드 하는 방법은 간단하다. 다만 업로드 이전에 확인해야 할 점이 몇 가지 있다.

- '도구 → 보드' 메뉴에서 사용하고자 하는 아두이노 보드가 선택되었는지 확인한다. 아두이노 메가2560 보드라고 가정한다면 'Arduino/Genuino Mega or Mega 2560'을 선택하면 된다.
- '도구 → 프로세서' 메뉴에서 마이크로컨트롤러를 선택한다. 아두이노 메가2560의 경우 ATmega2560 마이크로컨트롤러를 사용하고 있지만 이전 버전의 보드는 ATmega1280을 사용하여 만들어졌다. 따라서 사용한 프로세서를 선택해 주어야 한다. 아두이노 우노 보드를 선택하였다면 '도구 → 프로세서' 메뉴 항목은 나타나지 않는다.
- '도구 → 포트' 메뉴에서 아두이노 보드에 할당된 시리얼 통신 포트를 선택한다. 아두이노 메가2560이 연결된 포트 번호 옆에 'Arduino/Genuino Mega or Mega 2560'라는 표시가 나타나므로 쉽게 확인할 수 있다. 포트 번호는 컴퓨터에 따라 달라질 수 있으며 제어판

의 장치관리자에서도 확인할 수 있다.

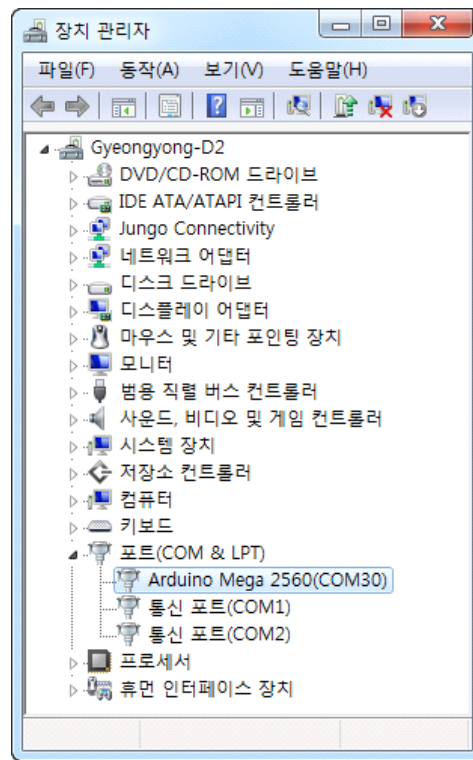


그림 8. 아두이노 보드에 할당된 포트

설정이 끝났으면 먼저 스케치를 컴파일 해 보자. 프로그래밍 과정에서 필요한 기본적인 기능들은 아두이노 프로그램의 툴바 버튼을 통해 사용할 수 있다.

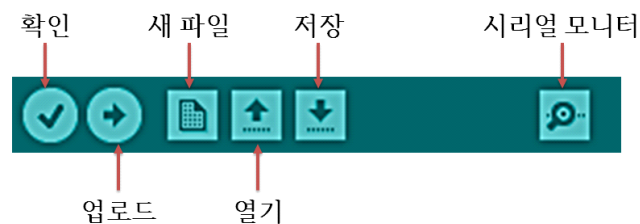


그림 9. 아두이노 IDE의 툴바

툴바의 버튼 중 가장 왼쪽에 있는 버튼은 ‘확인’ 버튼으로 스케치의 문법적인 오류를 검사하기 위해 사용한다. 오류가 없다면 이제 아두이노로 Blink 스케치를 업로드 해보자. ‘확인’ 버튼 옆에 있는 ‘업로드’ 버튼으로 생성된 기계어 파일을 시리얼 포트를 통해 아두이노로 업로드 할 수 있다. 업로드에 성공하면 13번 핀에 연결된 LED가 1초 간격으로 깜빡거리는 것을

확인할 수 있다.

시리얼 포트는 프로그램 업로드 이외에도 컴퓨터와의 통신을 통해 데이터를 주고받기 위해 사용될 수 있다고 설명하였다. 툴바의 가장 오른쪽에 있는 ‘시리얼 모니터’ 버튼을 눌러보자.

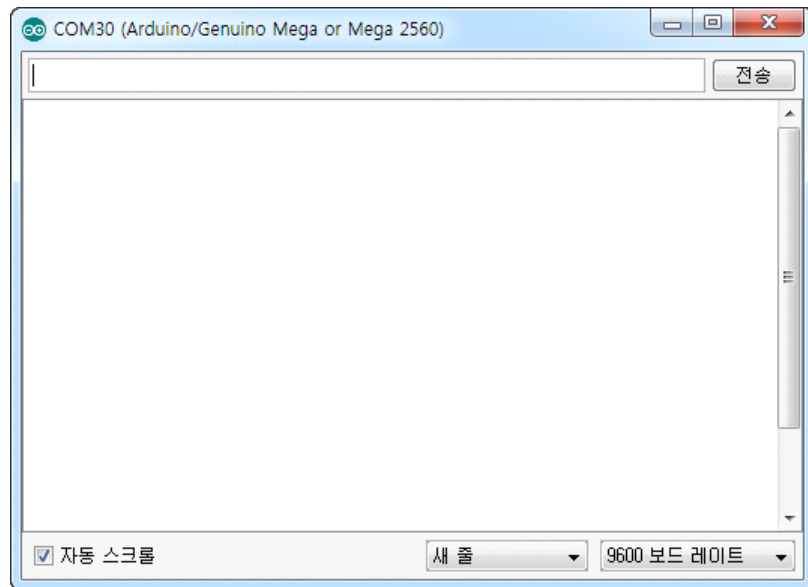


그림 10. 시리얼 모니터

시리얼 모니터는 아두이노로 전달할 데이터를 입력하는 윗부분과 아두이노에서 전달된 데이터를 표시하는 아랫부분으로 나뉘어져 있다. 아두이노의 출력을 표시하는 부분은 현재 아두이노의 실행 상태 등을 표시하기 위해 사용될 수 있다. ‘파일 → 예제 → 04.Communication → ASCIITable’을 선택하자. ASCIITable은 아스키코드 문자들을 다양한 형식으로 시리얼 모니터로 출력하는 스케치의 예이다. 스케치를 업로드 한 후 시리얼 모니터를 실행시키면 아두이노에서 컴퓨터로 전달된 데이터들이 시리얼 모니터로 출력되는 것을 확인할 수 있다.



그림 11. ASCIITable 스케치 실행 결과

5. 스케치의 구조

스케치를 업로드 하고 실행 결과까지 확인해 보았다. 마지막으로 언급하고 넘어가야 할 부분은 바로 스케치의 구조이다. 아두이노의 스케치는 C/C++ 언어로 만들어지며, C/C++ 언어의 경우 프로그램에서 제일 먼저 실행되는 main 함수가 반드시 필요하다. 하지만 그림 7에서는 main 함수를 찾아볼 수 없다. 일반적으로 마이크로컨트롤러에는 오직 하나의 프로그램만 설치될 수 있으며 설치된 프로그램은 전원이 주어지는 동안 끝나지 않는 무한루프를 통해 데이터를 처리한다. 이외에도 필요한 헤더 파일을 포함시키거나 상수를 정의하는 등의 작업을 수행하는 전처리 부분, 마이크로컨트롤러를 설정하는 초기화 부분 등이 필요하다.

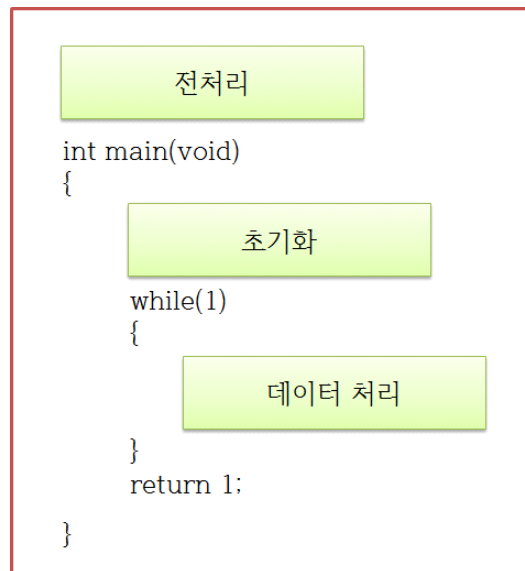


그림 12. 마이크로컨트롤러를 위한 프로그램의 구조

아두이노의 스케치는 프로그램의 구조를 보다 직관적으로 이해할 수 있도록 초기화 부분과 데이터 처리 부분을 `setup`과 `loop`라는 별개의 함수로 분리하고 있다.

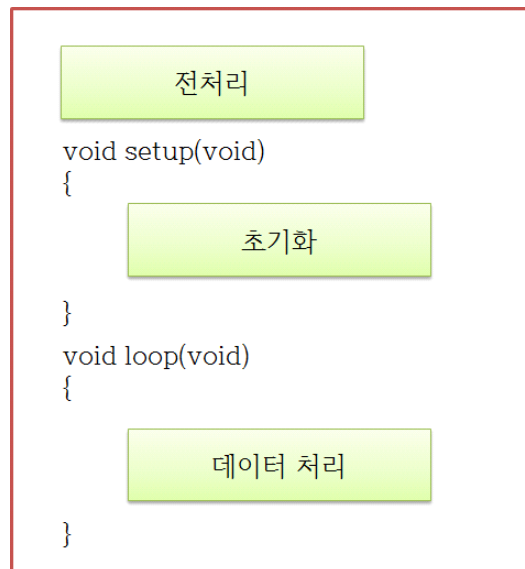


그림 13. 아두이노를 위한 스케치의 구조

내용이 바뀐 것은 아니다. 다만 모양이 바뀌었을 뿐이다. 아두이노에서 초기화를 담당하는 함수는 `setup` 함수이며 스케치가 실행될 때 한 번만 실행된다. `loop` 함수는 반복적인 실행이 필요한 내용을 추가하는 부분으로 데이터의 입력과 출력을 담당하게 된다. 그림 6에서도 볼

수 있듯이 아두이노 프로그램에서 새로운 스케치 파일이 만들어지면 두 함수에 대한 원형이 자동으로 추가된다. 함수에 포함되어야 하는 내용이 없는 경우에도 두 함수는 반드시 스케치에 정의되어 있어야 한다는 점은 명심하여야 한다.