Arduino 소개

Lesson 01

아두이노(Arduino)

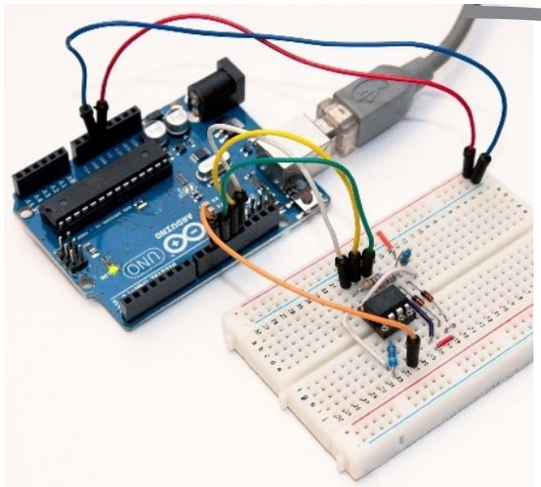
- **오픈소스에 기반한 프로토타이핑(prototyping) 시스템**
  - megaAVR 계열 (ATmega8, Atmega168등)과 ARM 계열의 microcontroller가 사용됨
  - 하드웨어+개발용 소프트웨어를 포함
- **다양한 하드웨어(센서, 네트워크, 입출력장치) 및 라이브러리가 지원됨**
- **2005년 이탈리아의 아두이노 사에서 개발**
- **저가이고 응용프로그램 개발이 편리**
- **많은 개발자 및 커뮤니티 존재**
- **오픈 하드웨어**

Arduino UNO R3 보드

아두이노 구성

- **소프트웨어**
  - Sketch IDE (Integrated Development Environment)
  - 라이브러리
- **하드웨어**
  - 아두이노 보드

아두이노 응용 개발 환경



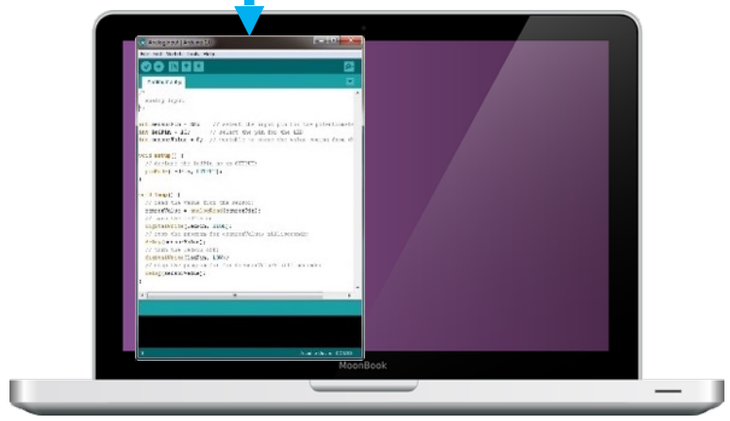
아두이노 보드
(③프로그램 실행)

시리얼 통신

USB

②프로그램 다운로드

Sketch IDE



프로그램 개발용 PC
(①소스 작성, 컴파일)

아두이노 보드(Board)의 종류

	MCU	Arduino 보드
megaAVR	ATmega168	Pro(168), Mini(168), LilyPad(168V)
	ATmega328	UNO, Fio, Nano, Pro(328, Rev5, 5V), Pro Mini, LilyPad (328V)
	ATmega2560	Mega 2560, Mega ADK
	ATmega32U4	Yún, Leonardo, Esplora, Micro
ARM	Cortex-M0+	M0(Zero), M0 Pro(Zero Pro)
	Cortex-M3	Due

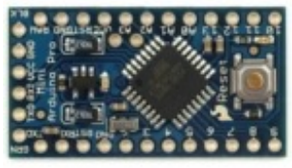
아두이노 UNO 계열 보드

- **아두이노 UNO 보드**
  - 표준 보드
  - R3 버전이 가장 널리 사용되고 있음
    - 8 Bit ATmega328 MCU 사용
    - 동작 전압 5V
    - 클럭 16MHz

- **아두이노 Nano 보드**
  - UNO의 소형 버전
  - UNO와 동일한 CPU 및 칩셋을 사용

아두이노 Pro 계열 보드

- **Pro 계열은 UNO와 동일한 MCU 사용**
- **UNO 보다 작고 얇다**
- **USB 통신기능이 없거나 동작전원이 다름**
  - 5V(16MHz)와 3.3V(8MHz) 용으로 구분됨
- **구분**
  - 아두이노 Pro
    - 전원버튼 및 배터리 커넥터 내장
  - 아두이노 Pro Mini
    - 초소형 저가
  - 아두이노 Pro Micro
    - USB 연결가능



아두이노 보드

- **아두이노 Mega2560**
  - 고성능이며 많은 IO 핀을 제공
  - 고성능이 필요한 로봇 제어, 멀티미디어 처리에 적합

- **아두이노 Leonardo**
  - UNO 보다 많은 IO핀이 제공됨
  - 시리얼 통신을 위한 여분의 핀이 제공
  - 고속의 시리얼 통신 응용에 적합

- **아두이노 Micro**
  - 아두이노 Leonardo의 소형 버전

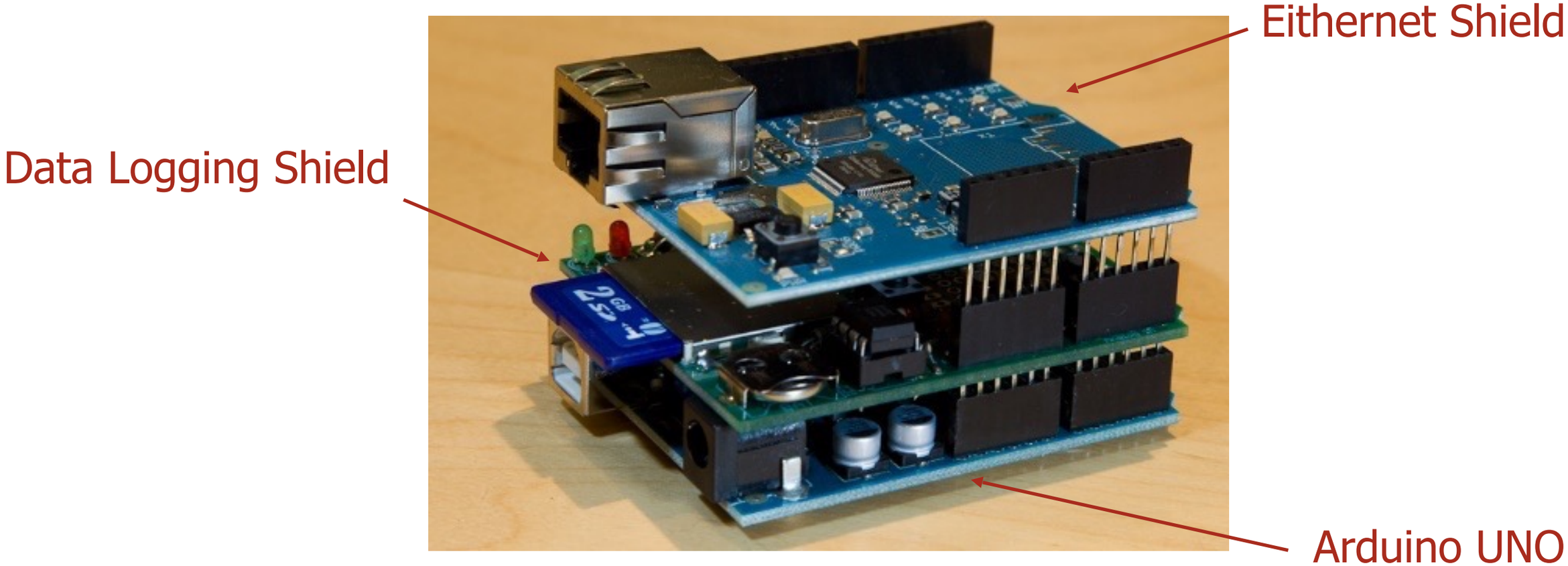
특수용 아두이노 보드

- **Lily Pad**
  - 의류를 비롯한 웨어러블 응용에 특화된 보드

- **FIO (Funnel IO)**
  - 무선통신용 특화 보드
  - Xbee 모듈과 배터리 연결단자를 포함

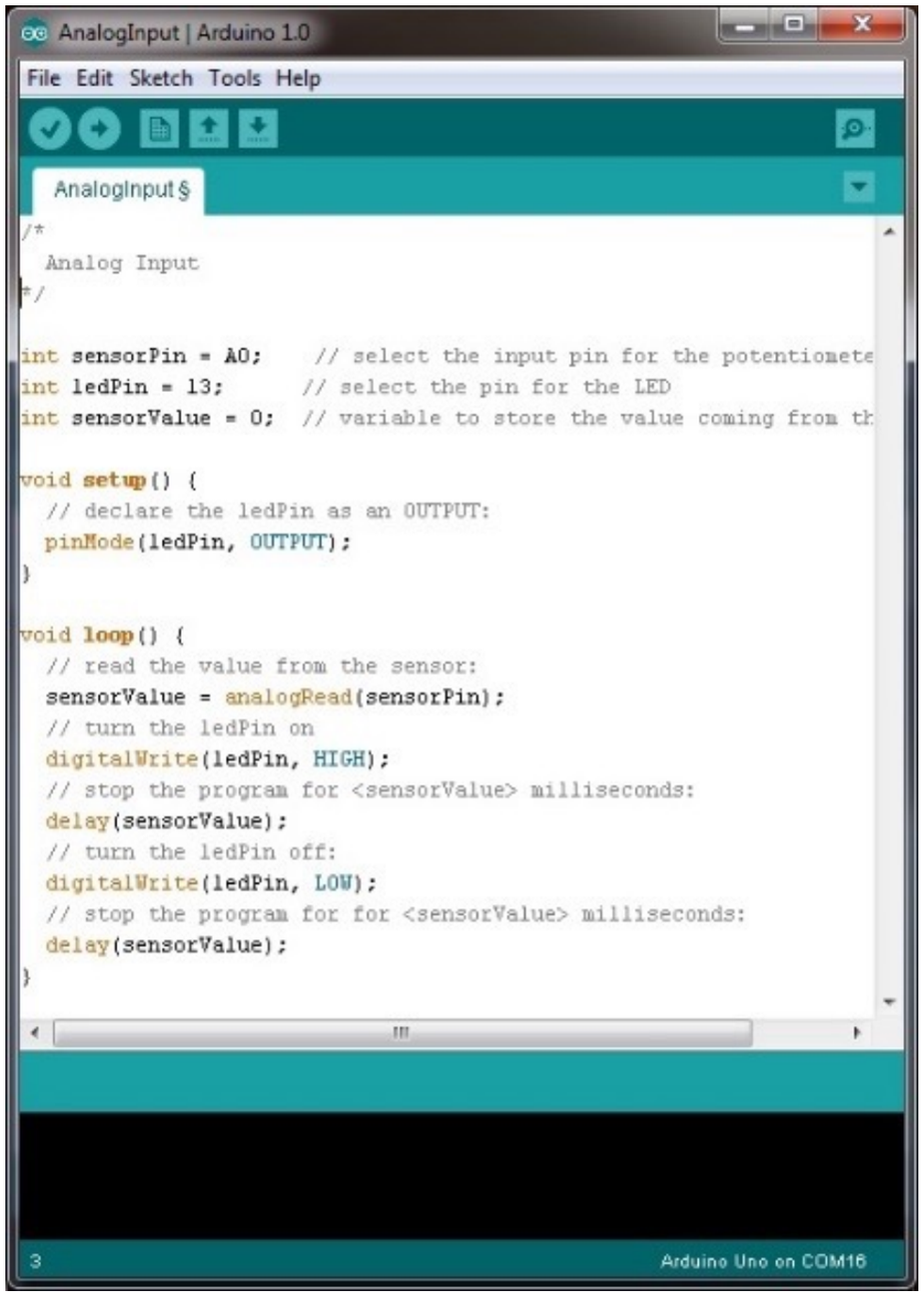
아두이노 쉴드(Shield)

- **아두이노 보드와 결합하여 특정 기능을 수행할 수 있도록 만들어진 부품**
  - 이더넷 쉴드, ZigBee 쉴드, Bluetooth 쉴드 …



쉴드를 적층한 예

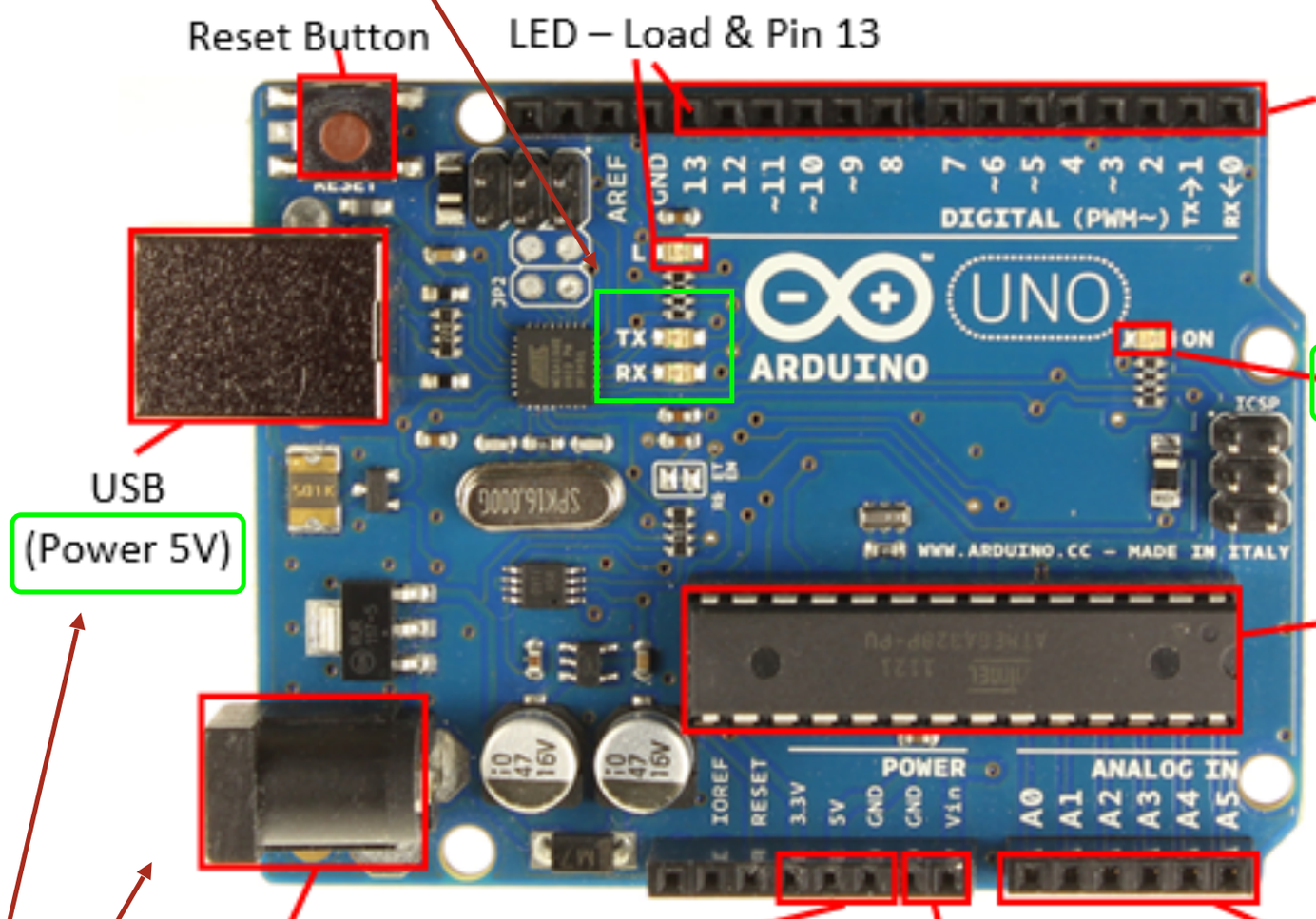
Sketch IDE



- **Cross-Platform 소프트웨어**
  - PC에서 소스 작성 및 컴파일
  - 최종 생성되는 프로그램(Binary File)은 아두이노에서 실행
  - 생성된 프로그램은 아두이노의 Flash memory에 다운로드 되어 실행된다
- **C++를 사용하여 응용프로그램 코드를 작성**

Arduino UNO R3 보드

시리얼 케이블을 이용한 통신상태 표시:
- TX(아두이노->컴퓨터)
- RX(컴퓨터->아두이노)



아두이노 보드에 전원이 인가되어 있는 경우 켜짐

아두이노 보드에 전원을 공급할 수 있는 장치들

아두이노 UNO 입출력 핀

- **디지털 입출력 핀**
  - 5V 디지털 입출력(INPUT/OUTPUT) 핀
    - HIGH (5V, ON) / LOW (0V, OFF)
  - 14개의 핀 사용 가능 : 0~13
    - ~가 표시된 핀은 PWM(Pulse Width Modulation) 기능 제공
  - GPIO (General Purpose Input Output) 용도와 special function 용도
- **아나로그 입력 핀**
  - 6개의 입력(INPUT) 전용 핀: A0~A5
  - 아나로그 입력 전압(0V~5V)을 디지털 값으로 읽을 수 있음

아두이노 UNO 핀의 종류와 기능

- **전원 (Power) 핀**
  - GND (그라운드, Ground): 0V를 유지, 음극 (-)으로 볼 수 있음
  - Vin: 외부에서 아두이노에 전원 공급하는 경우 사용할 수 있음(9V)
  - 5V: 아두이노에서 외부로 5V 전원을 공급하기 위한 핀
  - 3.3V: 아두이노에서 외부로 3.3V 전원을 공급하기 위한 핀
- **기타**
  - AREF: 아나로그 입력시에 기준 전압을 제공하기 위한 핀
  - RESET: 이 핀에 LOW 전압을 가하면 아두이노가 리셋됨
  - IOREF: 쉴드와의 연동시 기준 전압을 제공하는 핀

아두이노 응용 개발 과정

1. 아두이노와 소자들을 연결

- 반드시 전원이 인가되지 않은 상태(USB를 끊은 상태)에서만 아두이노 보드 및 소자를 연결/변경 할 것

2. 아두이노 IDE를 실행

3. 소스코드(Sketch) 작성

4. 컴파일하고 업로드

5. 동작 확인

- 의도한 동작이 수행되지 않으면
  - 회로 연결 상태를 확인
  - 연결 상태에 문제가 없는 경우 ③ - ⑤를 반복

아두이노 IDE 설치

- 아두이노 홈페이지 방문후 'Download' 메뉴를 선택하고 IDE가 설치될 OS를 선택
  - 홈페이지: http://www.arduino.cc
- 설치 폴더를 꼭 확인할 것
  - C:\Program Files\Arduino

아두이노와 PC와의 연결

- **Arduino 보드와 PC를 USB 케이블을 이용하여 연결**
- **PC의 '장치관리자'에서 'ArduinoUNO(COM?)'가 표시되었는지 확인**
  - 설치되지 않았으면 Arduino 설치 폴더의 'drivers' 폴더 내에 있는 'arduino.inf' 파일을 설치
  - COM (시리얼통신) 번호 확인
- **아두이노 IDE에서 설정 완료**
  - '도구->보드 ' 에서 'Arduino UNO' 선택
  - '도구->시리얼 포트'에서 연결된 COM 포트 번호 선택

아두이노 IDE 실행 화면

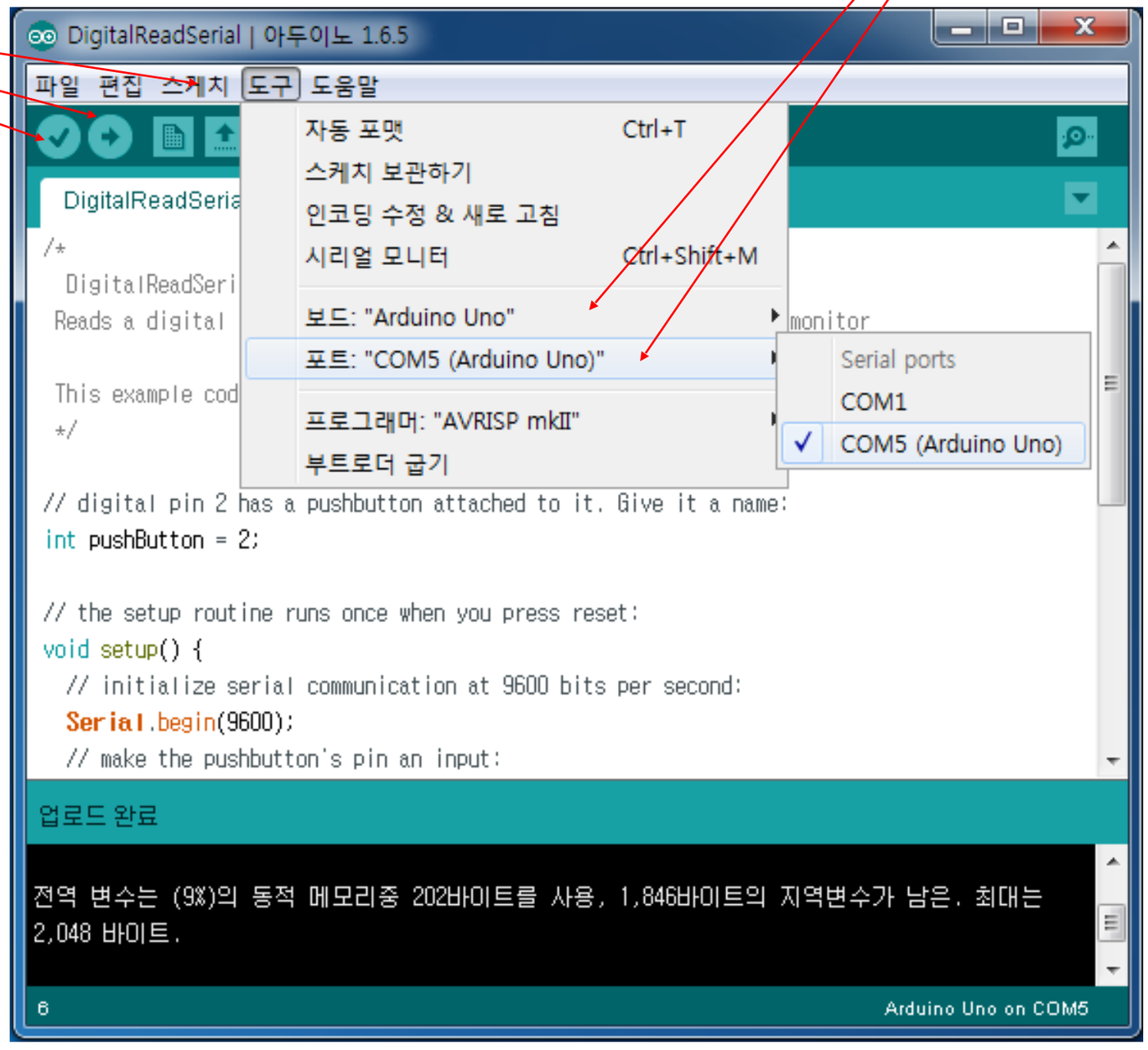
'확인'/'컴파일'
Arduino 보드로 프로그램 업로드

Arduino 보드 및
COM 포트 선택

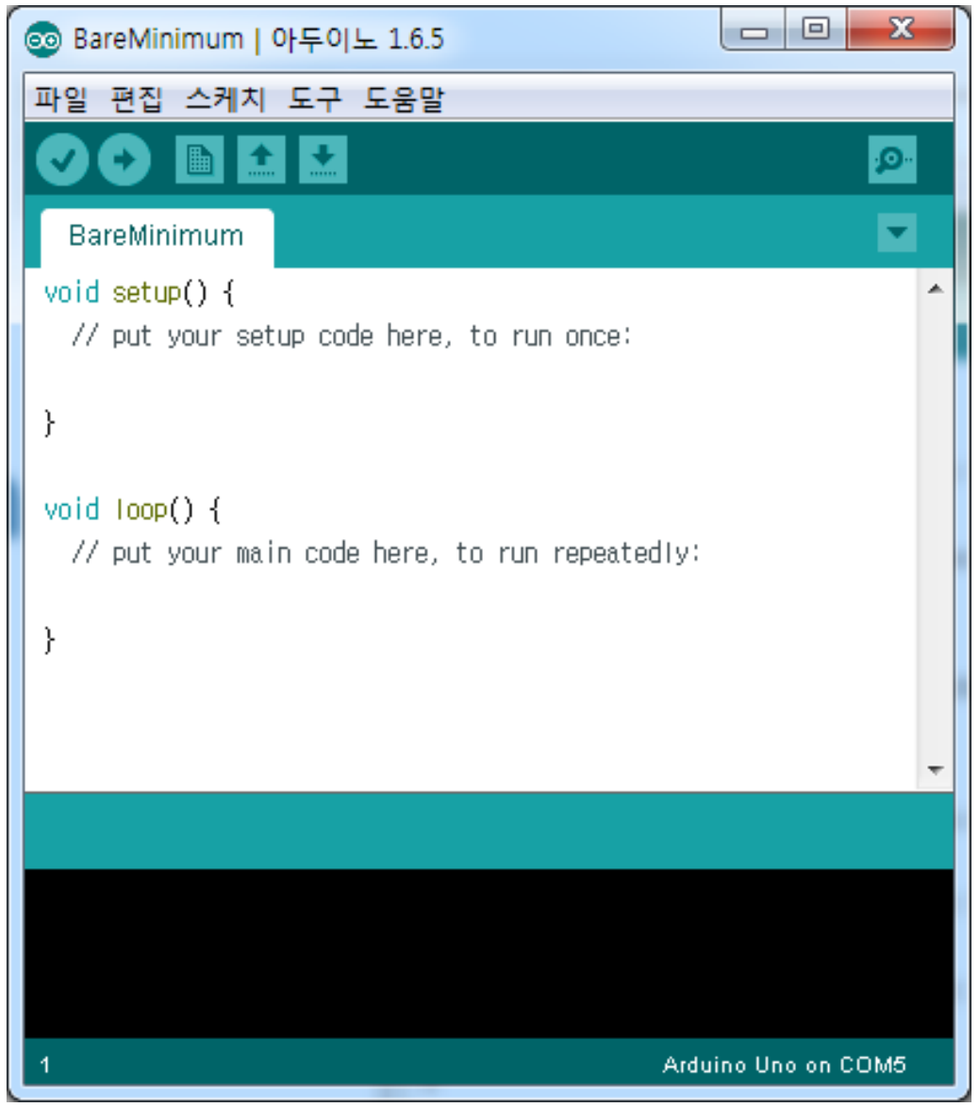
Sketch 코드
작성 영역

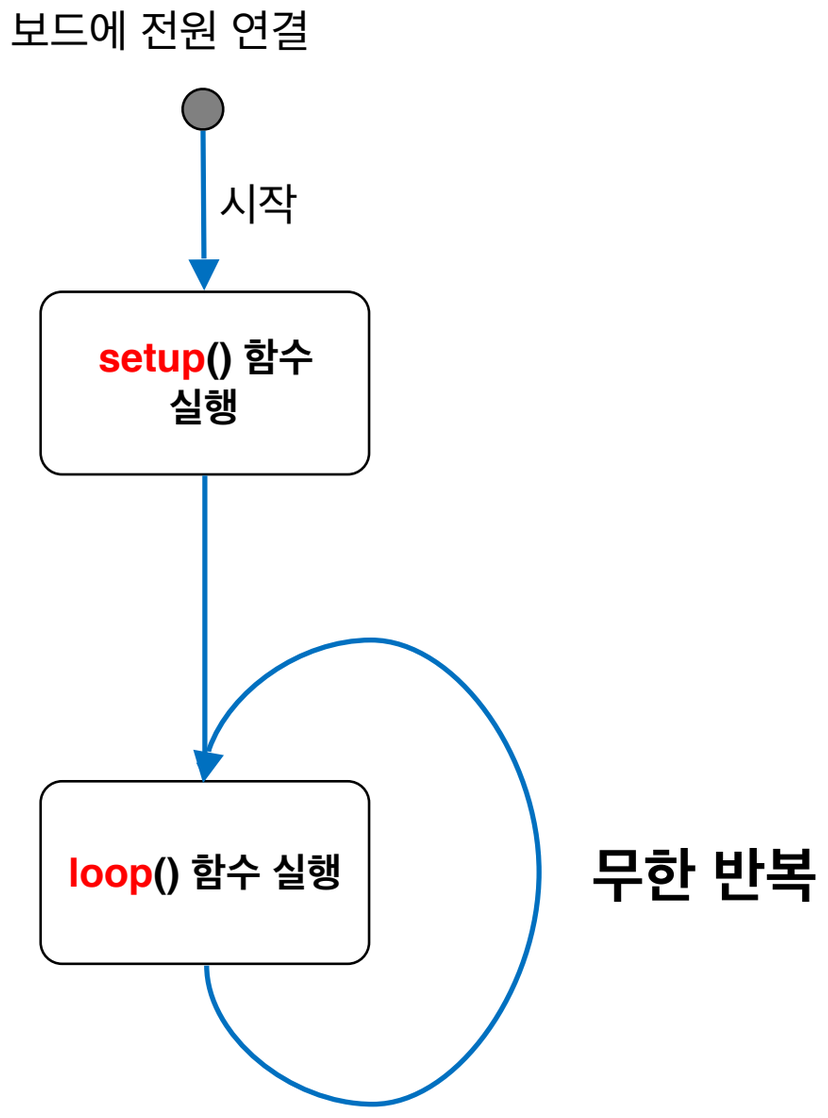
업로드 결과 메시지

컴파일 결과 메시지



Sketch 코드의 구조와 동작





Arduino Uno: Memory

- **32K Flash memory**
  - 부트로더(2K)와 사용자 프로그램(sketch)이 저장
- **2K SRAM (static random access memory)**
  - 프로그램 실행 중에 데이터가 저장
- **1K EEPROM (비휘발성)**
  - 전원이 꺼진 후에도 보존되어야 하는 데이터의 저장

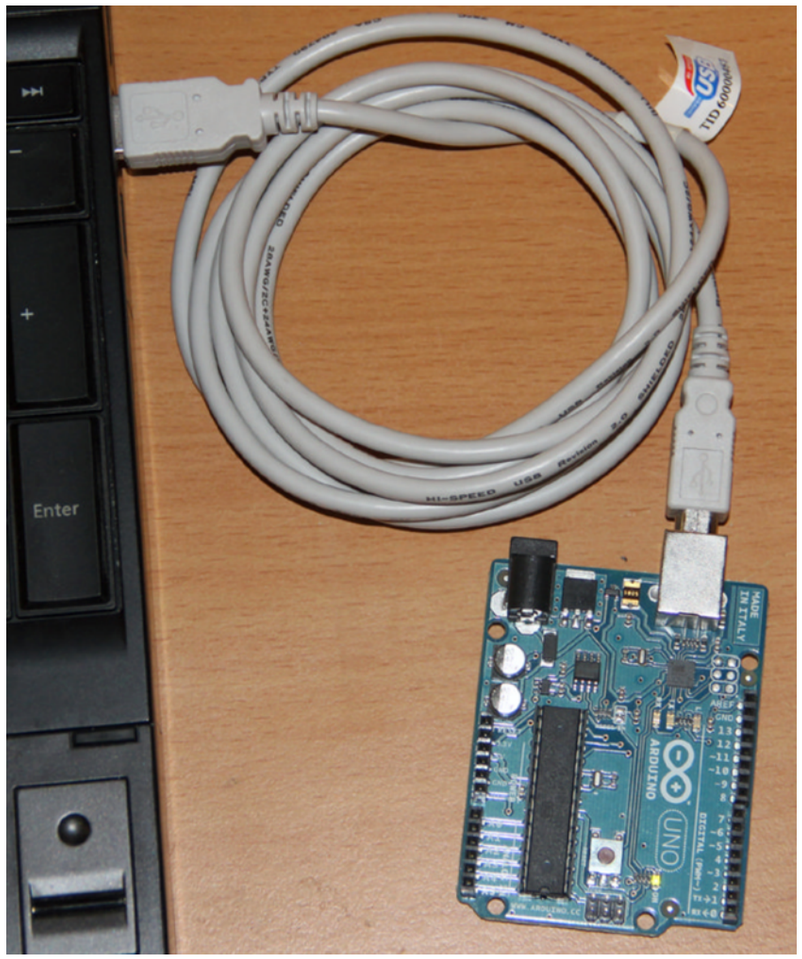
Bootloader

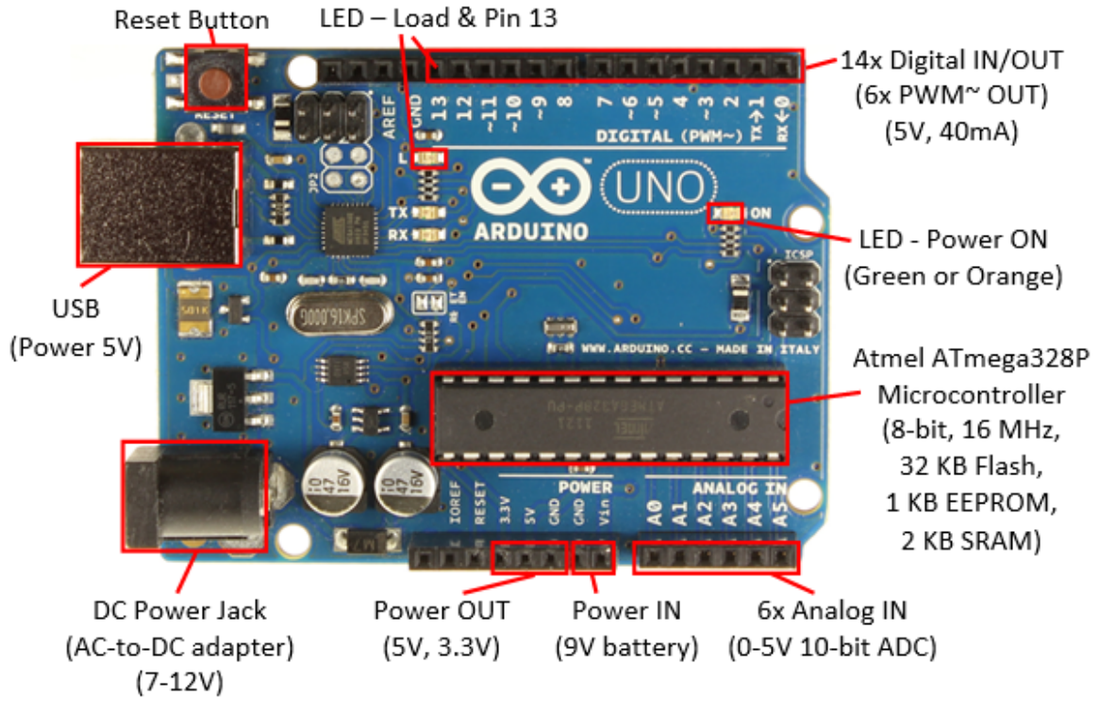
- 아두이노는 임베디드보드로는 드물게 부트로더가 내장되어 있음
- 전원을 켜면 부트로더가 실행됨
- 부트로더는 몇 초간 **UART(**시리얼 포트)를 통해 **IDE**로 부터의 **upload** 명령을 대기
- **IDE**로 부터 **upload** 명령이 있으면 전송되는 프로그램을 **Flash**에 저장하고 실행함
- 명령이 없으면 현재 **Flash**에 저장되어 있는 프로그램을 실행

Hello World ! from Arduino

Blink

- 아두이노를 **PC의 USB**에 연결한다.
- **13번 pin**과 연결된 자체 **LED**가 깜빡거리는 것을 확인한다. **왜 깜빡일까?**

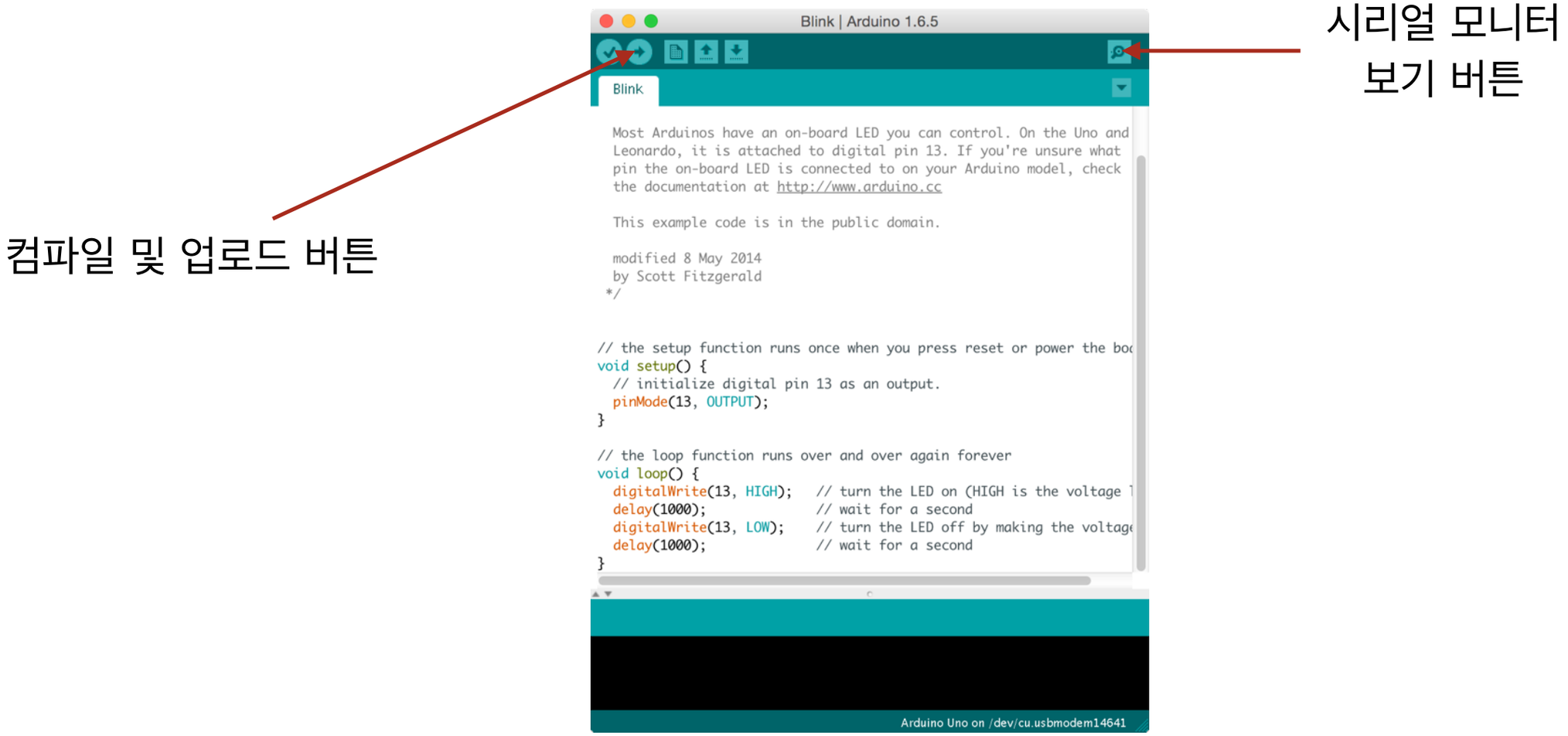




Blink

- **IDE를 실행하고 File->Examples->Basics->Blink 예제를 선택하고 실행해본다.**

  1. Tools 메뉴에서 board와 port를 먼저 설정한다.
  2. 화살표 버튼을 눌러 컴파일 및 업로드한다.



Sketch01: Blink

setup 함수는 처음에 한 번 실행된다.

```
void setup() {
    pinMode(13, OUTPUT);
}
```

아두이노에서는 프로그램 소스를 sketch라고 부른다. 스케치는 기본적으로 setup 함수와 loop함수로 구성된다. 물론 다른 함수를 추가할 수도 있다.

```
void loop() {
    digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
    delay(1000);              // wait for a second
    digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
    delay(1000);              // wait for a second
}
```

loop 함수는 아두이노의 전원이 켜져있는 동안 계속해서 호출/실행된다.

Sketch01: Blink

LED는 13번 핀과 연결되어 있다. 13번 핀을 이렇게 출력모드로 설정하였다. 이렇게 입력용도로 사용할 것인지 출력용도로 사용할 것인지를 선택할 수 있는 포트 혹은 핀을 GPIO 포트/핀이라고 부른다.

HIGH와 LOW는 그냥 1과 0의 값을 가지는 매크로(macro)이다. digitalWrite함수는 지정된 핀(이 예에서는 13번 핀)에 LOW (0V)혹은 HIGH (5V)의 출력을 내보낸다.

```
void setup() {
    pinMode(13, OUTPUT);
}

void loop() {
    digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
    delay(1000);              // wait for a second
    digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
    delay(1000);              // wait for a second
}
```

delay함수는 지정된 milisecond 동안 아무일도 하지 않고 기다리게하는 함수이다.

Sketch02: Serial write

File->Examples->Basics->AnalogSerial 예제를 선택한 후 아래와 같이 코드를 약간 수정하여 실행하고, serial monitor를 띄워서 결과를 확인한다.

```
void setup() {
    // initialize serial communication at 9600 bits per second:
    Serial.begin(9600);
    pinMode(13, OUTPUT);
}
```

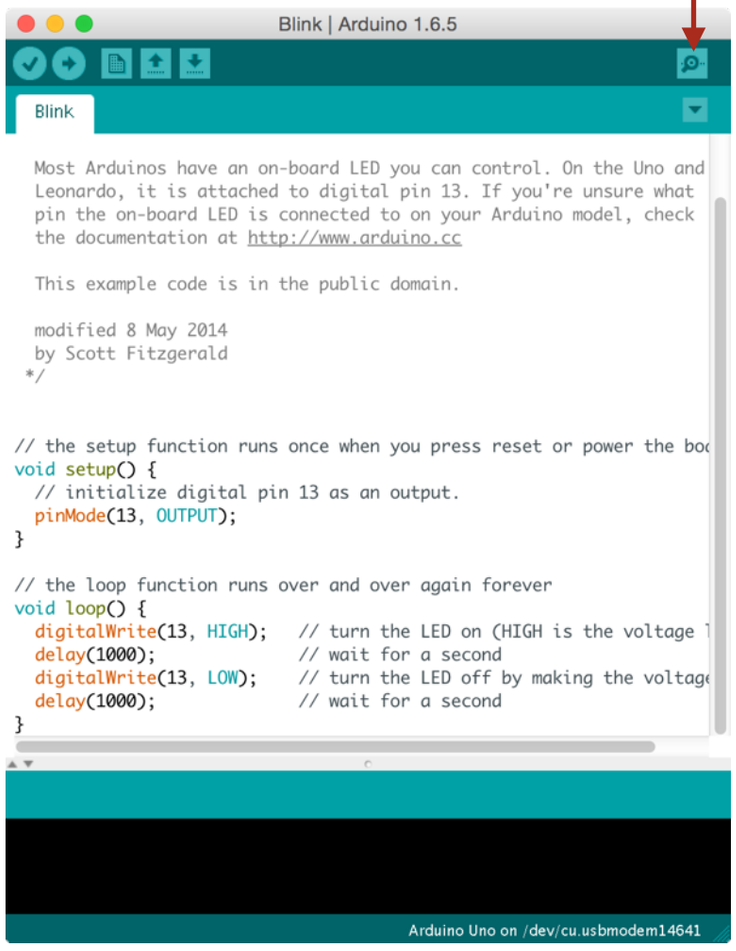
시리얼 통신의 bit rate를 9600 baud로 설정하고 초기화한다.

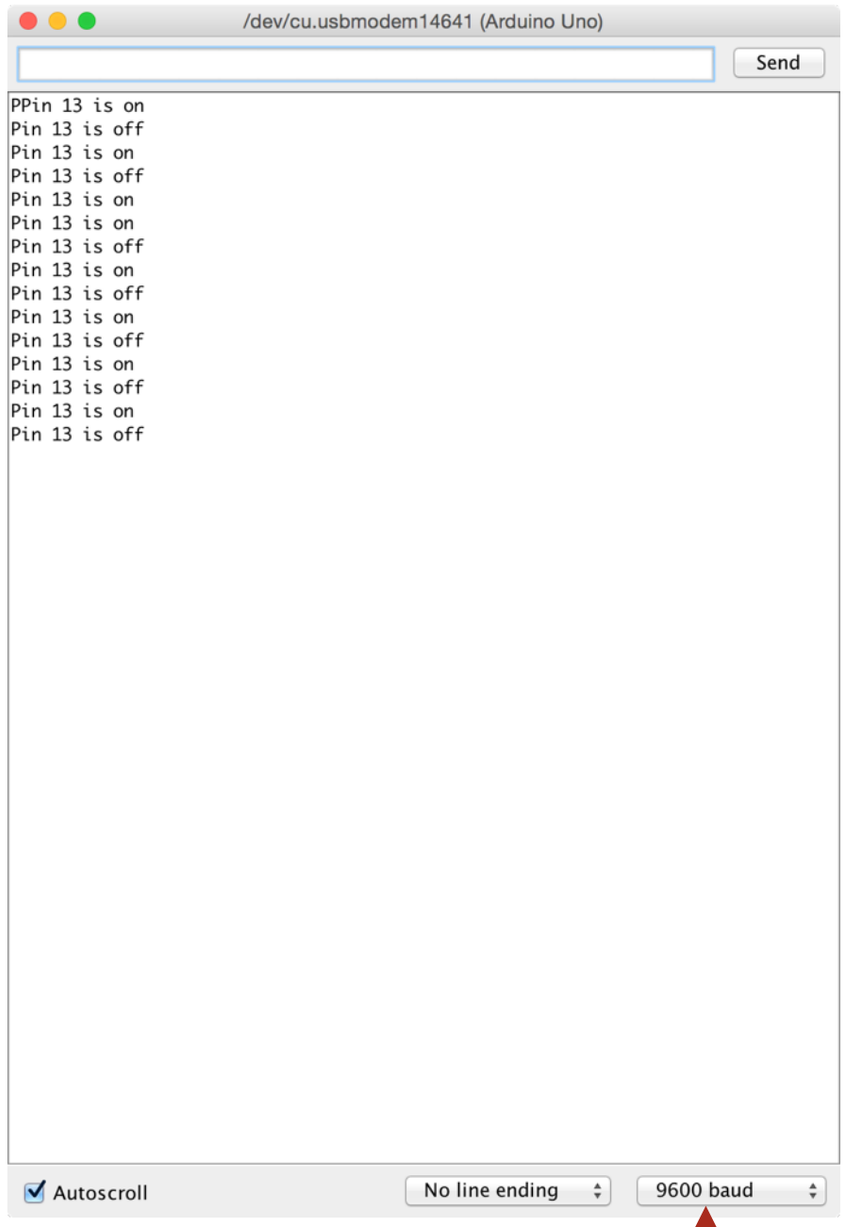
적절한 메시지를 PC로 전송한다.

```
void loop() {
    Serial.println("Pin 13 is on");
    digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
    delay(1000);              // wait for a second
    Serial.println("Pin 13 is off");
    digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
    delay(1000);
}
```

Sketch02: Serial write

시리얼 모니터
보기 버튼을 클릭한다.





스케치에서 설정해준 값인
9600 baud로 맞춘다.

Sketch03: Serial Read

```
char data; //Holds incoming character

void setup()
{
  Serial.begin(9600); //Serial Port at 9600 baud
}

void loop() {
  //Only print when data is received
  if (Serial.available() > 0)
  {
    data = Serial.read(); //Read byte of data
    Serial.print(data);   //Print byte of data
  }
}
```

데이터가 수신되었는지 검사한다.

Where is `main` ?

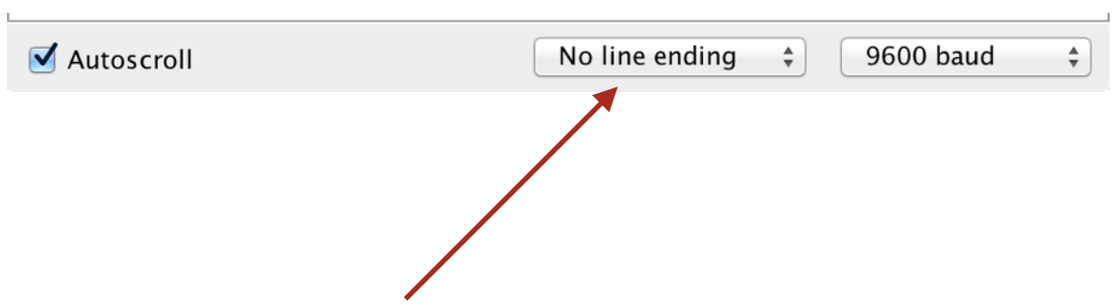
```
#include <Arduino.h>

// Declared weak in Arduino.h to allow user redefinitions.
int atexit(void (* /*func*/ )()) { return 0; }

// Weak empty variant initialization function.
// May be redefined by variant files.
void initVariant() __attribute__((weak));
void initVariant() { }

int main(void)
{
    init();

    initVariant();

    #if defined(USBCON)
    USBDevice.attach();
    #endif

    setup();

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

    return 0;
}
```

Arduino설치폴더/hardware/arduino/cores/arduino/main.cpp

실습 1

- 시리얼 통신의 **bit rate**를 바꿔본다.
- 좀 더 긴 문자열을 전송해본다.
- **print**와 **println**의 차이를 확인해본다.
- 정수나 실수 혹은 숫자와 문자가 섞인 메시지를 전송해본다. **(1부터 10까지 돌아가면서 전송해본다.)**

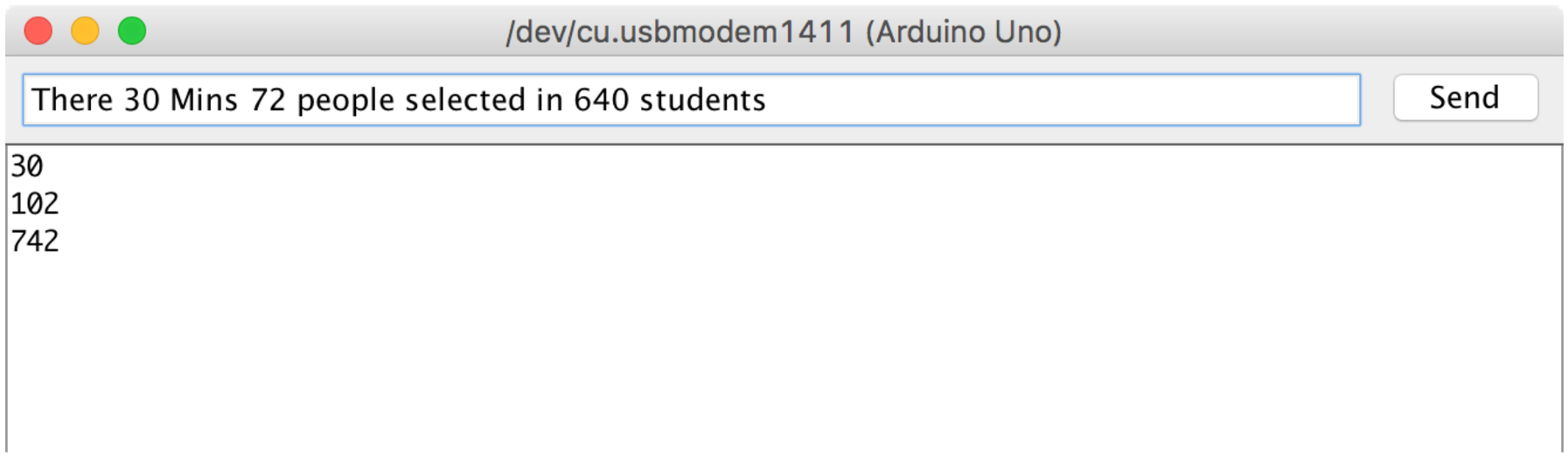
```
Message 1
Message 2
Message 3
…
Message 10
Message 1
…
```



- **PC**에서 **Arduino**로 전송할 때 **라인 끝 처리방법**의 차이를 테스트해본다.

실습 2

- **다음과 같은 일을 하는 스케치를 작성하라.**
  - PC에서 Arduino로 정수와 문자가 뒤섞인 한 라인의 메시지를 전송한다.
  - 전송되는 메시지에 포함된 모든 정수의 누적합을 다시 PC로 전송한다.
  - 이 일을 계속 반복한다.



Arduino Language Reference를 참고하라.