02주차

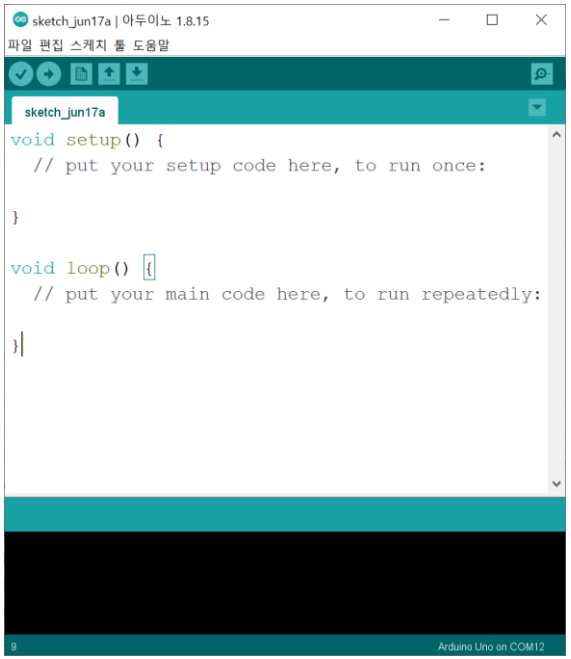
아두이노와 친해지기

목차

01 아두이노 생태계 탐험

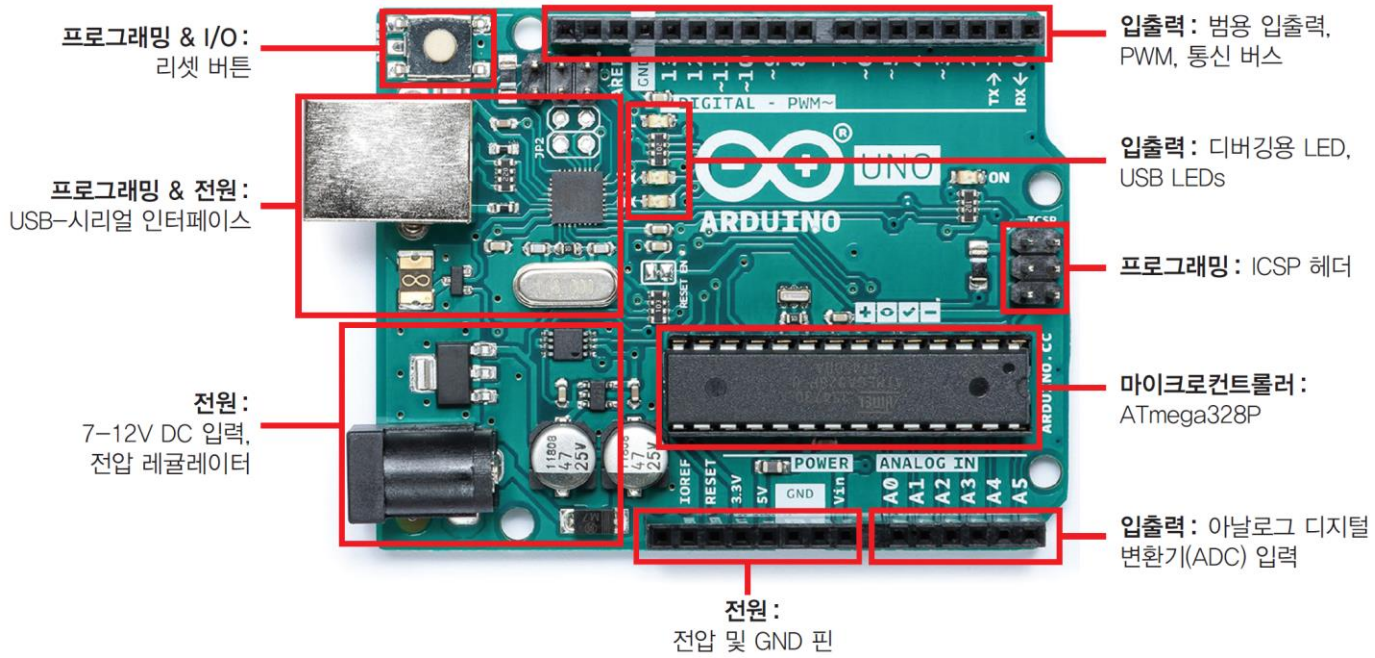
아두이노 프로젝트 구성 요소

- 아두이노(Arduino)
  - 마이크로컨트롤러 플랫폼
  - 마이크로컨트롤러 보드+ 소프트웨어 개발 환경(IDE)
  - 오픈 소스 기반



아두이노 보드 : 구성요소

- 마이크로 컨트롤러 : 아두이노의 두뇌
- 프로그래밍 : 스케치 또는 펌웨어 업로드 지원
- 입출력 : 센서 등 입출력 장치 연결
- 전원 : 마이크로컨트롤러 및 보드의 동작 전압 공급



아두이노 보드 : 구성요소

- PWM은 Pulse Width Modulation (펄스 폭 변조)의 약자
- 전압을 연속적으로 바꿀 수 없는 디지털 핀에서 듀티 사이클(duty cycle, ON 되는 시간의 비율)을 조절하여 원하는 평균 전압을 생성
- 예를 들어 아두이노 우노 기준:
  - HIGH = 5V
  - LOW = 0V
  - 만약 50%의 듀티 사이클로 ON/OFF를 빠르게 반복하면 → 평균 약 2.5V처럼 동작
  - LED나 모터의 밝기나 세기를 변경 가능

아두이노 보드: 마이크로컨트롤러

- 초기 아두이노 보드는 아트멜(Atmel)에서 제작하는 Atmega 시리즈 마이크로컨트롤러 사용
- 스케치 또는 펌웨어의 컴파일 결과인 기계어 파일을 저장 및 실행
- 아두이노 우노 : 8비트 16MHz ATmega328P 사용

아두이노 보드: 프로그래밍 인터페이스

- 프로그램 업로드를 위해 사용
- 일반적인 방법
- 아두이노 우노의 경우
  - USB 연결만으로 USART 통신을 통해 업로드 가능
    → 부트로더가 있어 가능
  - 아두이노 우노, 메가2560에는 USB 연결을 위한 ATmega16U2 마이크로컨트롤러가 포함되어 있음
  - 아두이노 레오나르도, Cortex-M0 기반 아두이노 보드는 마이크로컨트롤러 자체에서 USB 연결 지원

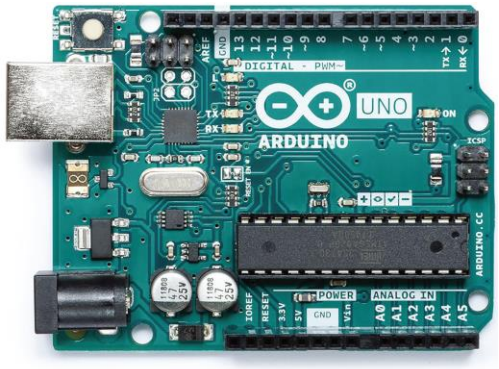
* USART(**U**niversal **As**ynchronous **R**eceiver **T**ransmitter) 통신: 별도의 클락으로 데이터를 타이머에 따라 송수신

아두이노 보드: 입력과 출력 / 전원

- 입력과 출력
  - 범용 입출력(GPIO) : 디지털 데이터 입출력을 위해 사용
  - ADC : 아날로그 데이터 입력을 위해 사용
  - 이외에 시리얼 통신, 펄스 폭 변조 신호 출력, 외부 인터럽트 등 지원
- 전원
  - 전원 공급 방법
    - USB 연결을 통한 5V 전원 공급
    - 배럴 잭이나 VIN 핀을 통한 6~20V 전원 공급 : 레귤레이터를 거쳐 5V 공급
  - 일부 아두이노 보드 및 확장 보드는 3.3V 전원을 사용하기도 함
    - 레귤레이터를 통해 3.3V 전원 역시 공급 가능

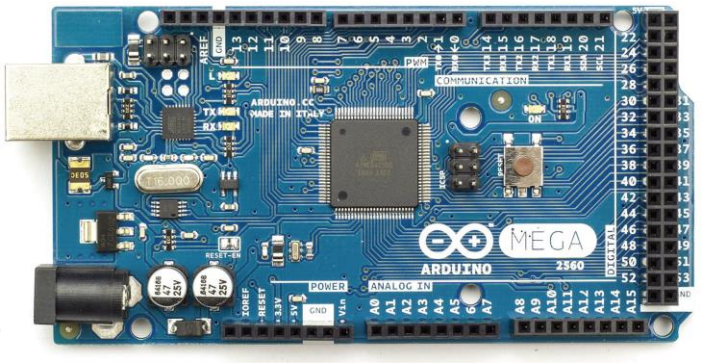
아두이노 보드 종류

- 아두이노 우노
  - 아두이노의 대표 보드이며 이 책에서 주로 사용
  - ATmega328P 마이크로컨트롤러 사용



아두이노 우노

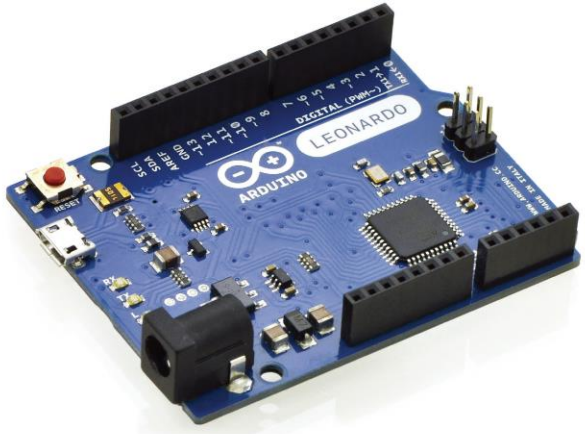
- 아두이노 메가2560
  - 아두이노 우노의 업그레이드 버전
  - 아두이노 우노보다 많은 수의 범용 입출력 핀, 큰 메모리, 많은 ADC 채널, 4개의 하드웨어 시리얼 인터페이스 등 지



아두이노 메가2560

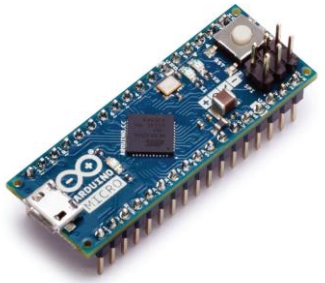
아두이노 보드 종류

- 아두이노 레오나르도
  - USB 인터페이스를 지원하는 ATmega32U4 마이크로컨트롤러 사용
  - 별도의 MCU 없이 USB 연결 가능
  - 조이스틱, 키보드 등 USB 장치를 만들기 위해 사용 가능



아두이노 레오나르도

- 아두이노 마이크로
  - 아두이노 레오나르도와 기능면에서 동일
  - 브레드보드에 연결하여 사용 가능한 소형 보드



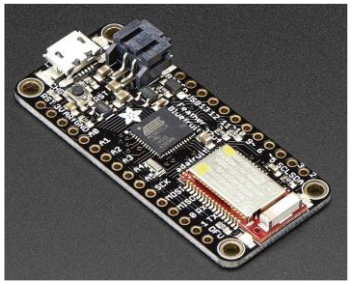
아두이노 마이크로

아두이노 보드 종류

- 아두이노 듀에
  - Arm Cortex-M3 기반 SAM3X 마이크로컨트롤러 사용
  - 높은 해상도의 ADC, DAC, USB 호스트, 84Mhz 클록, 32비트 CPU 등의 고성능 아두이노 보드

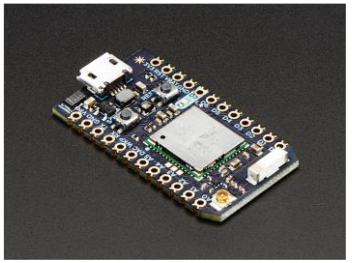
아두이노 듀에

- Adafruit Feather 32U4 Bluefruit
  - 아두이노 레오나르도와 같은 ATmega32U4 마이크로컨트롤러 사용
  - 블루투스 통신 기능 포함



Adafruit Feather 32U4 Bluefruit

- Particle Photon
  - Cortex-M3 기반 STM32F205 마이크로컨트롤러 사용
  - 와이파이 통신 기능 포함



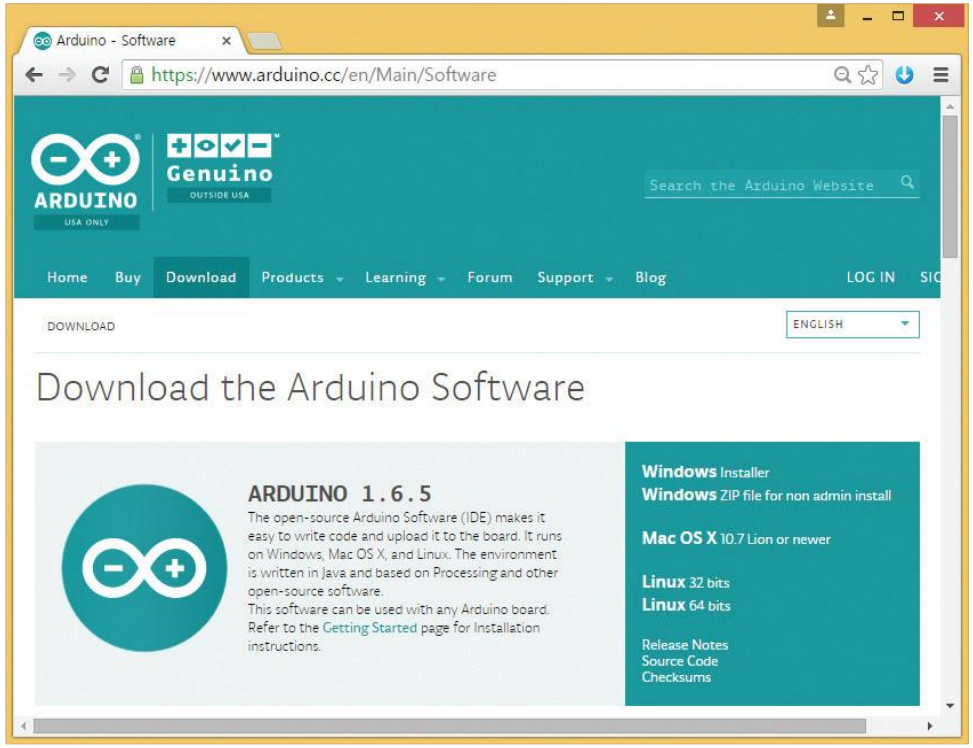
Particle Photon

아두이노 보드 vs 라즈베리파이

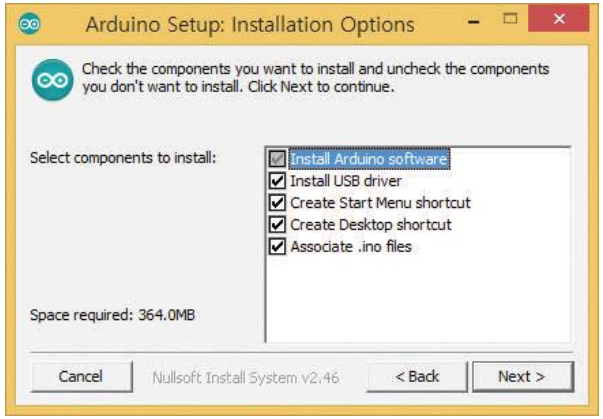
구분	아두이노	라즈베리파이
모양		
Spec	CPU : ATMega328 CPU Speed : 16Mhz RAM : 2K	CPU : ARM Cortex-A53 CPU Speed : 1.2Ghz RAM : 1G
OS	없음	라즈베리안(리눅스 계열), 리눅스, 윈도우
개발 언어	c언어	파이썬 ,웹프로그램,C언어등등
개발 방법	IDE(통합개발환경)에서 코딩 후 아두이노 보드로 업로드	os가 설치되어 있으므로 개발툴을 이용하여 개발 후 컴파일된 실행파일 수행

02 아두이노 IDE 환경 설치 및 설정

아두이노 IDE 설치 – 1

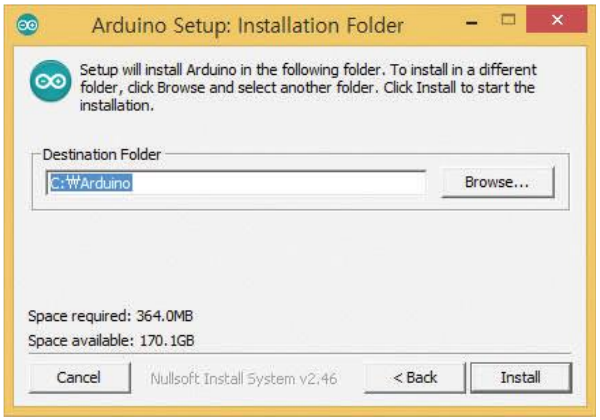


(1) IDE 다운로드 : www.arduino.cc

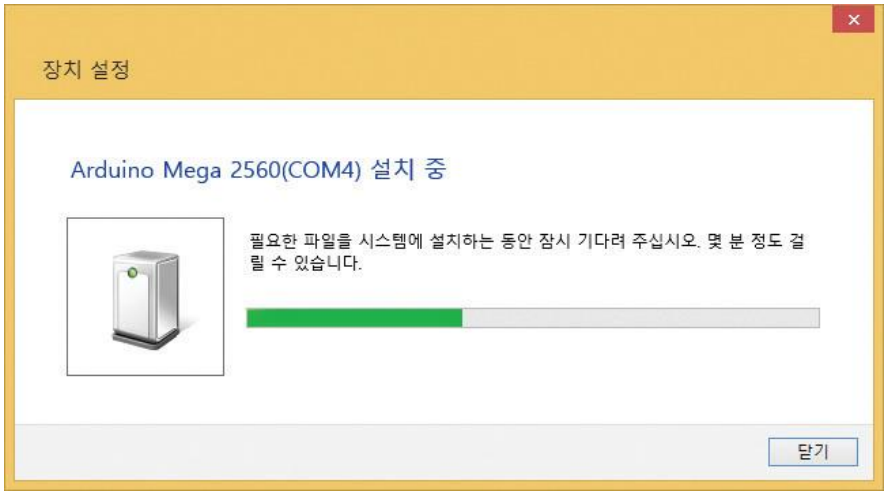


(2) 인스톨러 실행

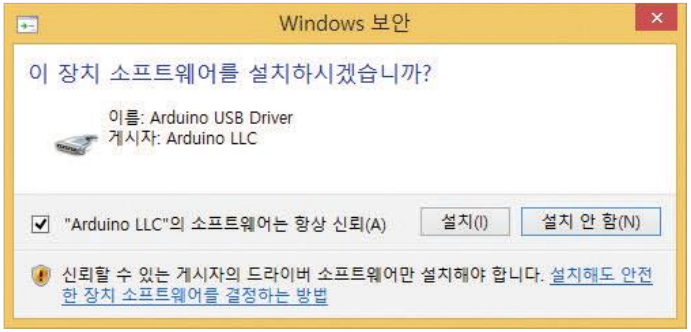
아두이노 IDE 설치 – 2



(3) 설치 디렉터리 지정



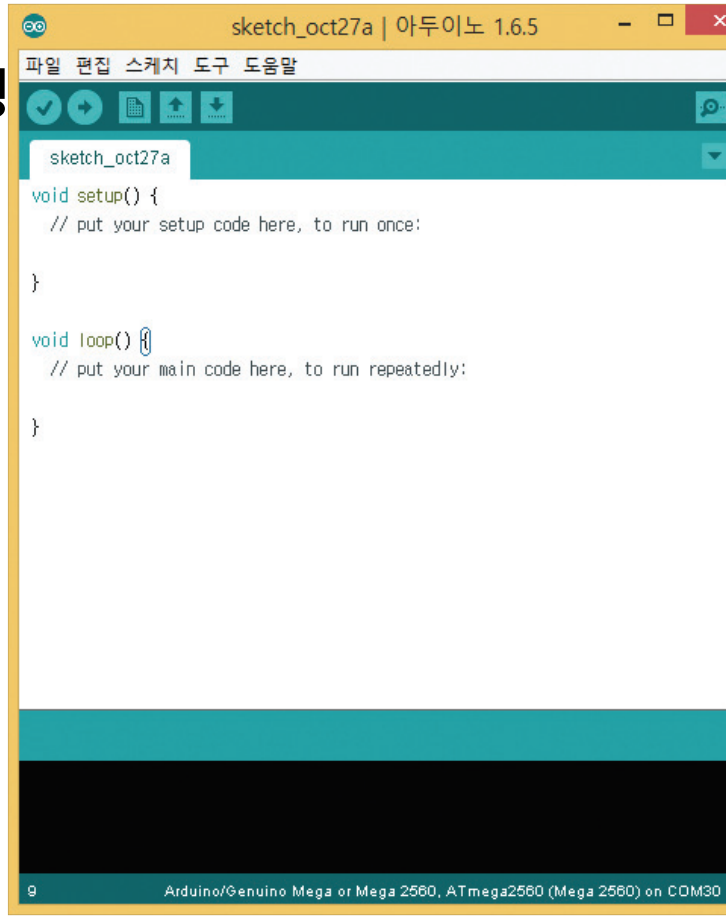
(5) 아두이노 보드 연결
: 가상 COM 포트 자동 설정



(4) 아두이노 보드 드라이버 자동 설치

아두이노 IDE

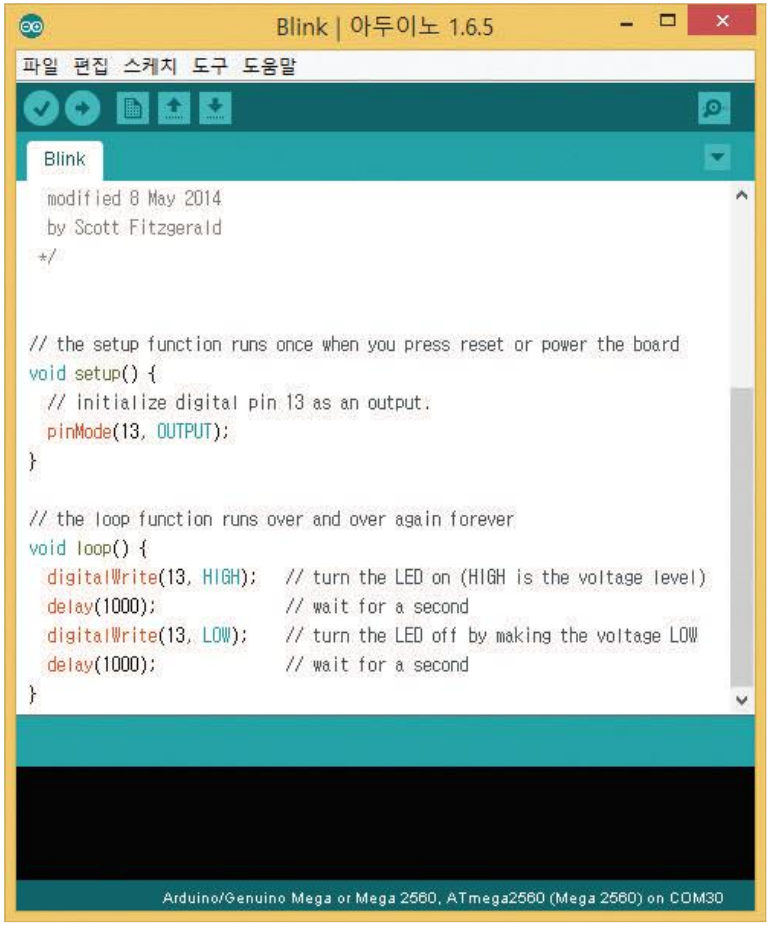
- 작고 간단한 통합개발환경
- 초보자를 위해 꼭 필요한 기능들만으로 구성
- 한 번의 클릭으로 컴파일에서 업로드까지 진행
- C로 구현되어 OS 간 이식성이 뛰어남
- 디버깅 기능은 제공하지 않음



첫 번째 아두이노 프로그램

- **아두이노에서 프로그램은 그림을 그리듯 쉽게 작성할 수 있다는 의미에서 '스케치(sketch)'라 불림**

- **블링크 (Blink) 스케치**
  - 13번 핀의 LED를 1초 간격으로 점멸하는 스케치
  - C 언어에서의 'Hello World'에 해당
  - '파일 → 예제 → 01.Basics → Blink' 선택



첫 번째 프로그램 자세히 살펴보기

- 줄 1~23 : 주석
  - '/*'로 시작하고 '*/'로 끝나는 여러 줄로 이루어진 주석
- 줄 25 : 한 줄 주석
  - 같은 줄에서 '//' 뒤의 내용은 주석으로 처리
- 줄 26~29 : 초기화 setup 함수
  - 프로그램 실행이 시작될 때 한 번만 실행
  - 핀의 입출력 설정, 통신 인터페이스 초기화 등의 작업 수행
- 줄 28 : 핀의 입출력 상태 설정 pinMode 함수
  - 아두이노의 디지털 핀은 입력 또는 출력으로 사용 가능
  - 입력과 출력으로 동시에 사용은 불가능하므로 pinMode 함수로 입력 또는 출력으로 사용하도록 지정
  - 디폴트 상태는 입력 상태

첫 번째 프로그램 자세히 살펴보기

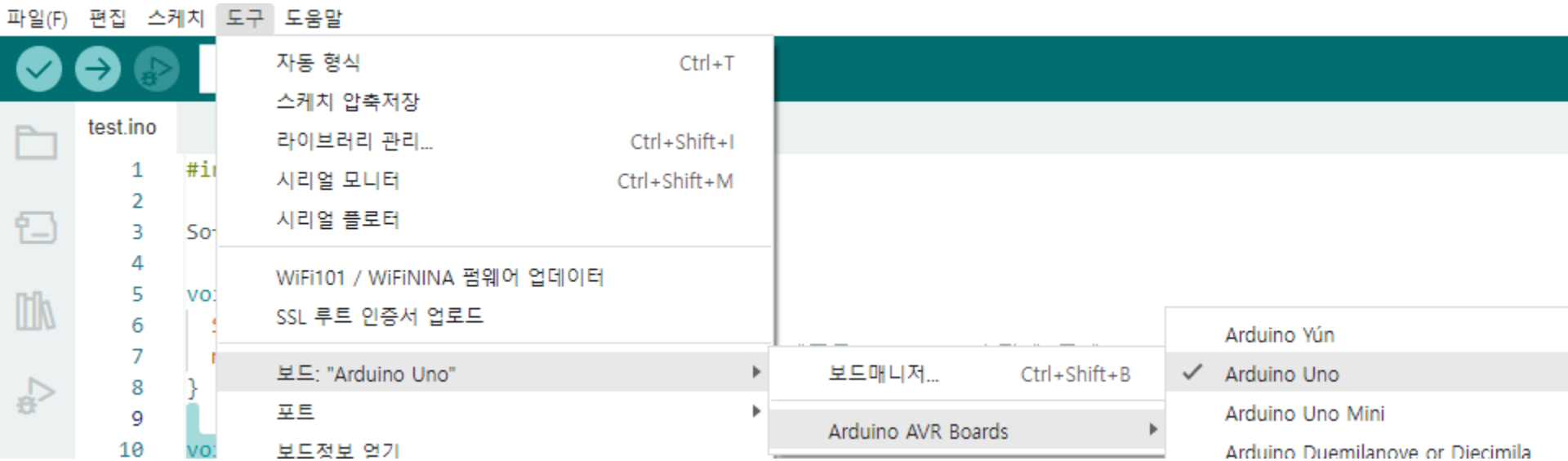
- 줄 28 : pinMode 함수
  - '/*'로 시작하고 '*/'로 끝나는 여러 줄로 이루어진 주석
  - 첫 번째는 핀 번호를 지정
    - 아두이노 우노에서 내장 LED는 13번 핀에 연결되어 있음
    - 아두이노 보드에 따라 내장 LED가 연결된 번호는 다를 수 있음
    - 13 대신 내장 LED가 연결된 핀 번호를 나타내는 LED_BUILTIN 사용 가능
  - 두 번째는 핀의 상태로 입력(INPUT) 또는 출력(OUTPUT)을 지정
- 줄 32~37 : 반복 수행 loop 함수
  - 초기화를 위한 setup 함수 수행 이후 반복해서 loop 함수가 호출됨
  - setup과 loop 함수는 아두이노의 기본 함수로 수행할 작업이 없어도 반드시 포함해야 함

첫 번째 프로그램 자세히 살펴보기

- 줄 33 : 디지털 데이터 출력 digitalWrite 함수
  - 2개의 매개변수를 가짐
  - 첫 번째는 핀 번호를 지정
  - 두 번째 매개변수는 핀으로 출력될 값 지정
    - HIGH(5V) 또는 LOW(0V) 지정
- 줄 34 : 시간 지연 delay 함수
  - 밀리초 단위의 지연 시간을 매개변수로 가짐
  - 블링크 예제에서는 1초, 즉, 1000밀리초 간격으로 점멸 상태가 바뀜

프로그램 업로드를 위한 설정 1

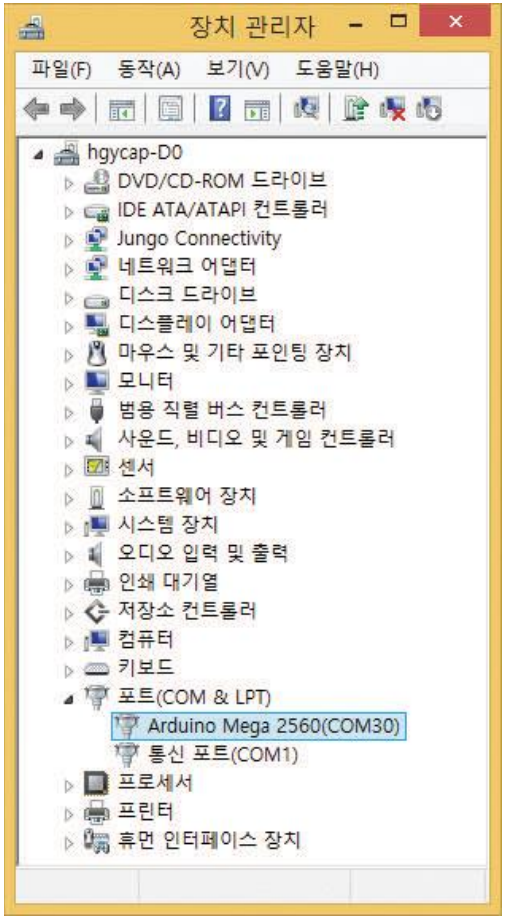
- **아두이노 보드 선택**
  - 사용하고자 하는 보드를 선택
  - '도구 → 보드' 메뉴에서 'Arduino/Arduino Uno' 선택



프로그램 업로드를 위한 설정 3

포트 선택

- 아두이노 보드에 할당된 가상 COM 포트 선택
- '도구 → 포트' 메뉴에서 해당 포트 선택
- 아두이노 보드에 할당된 포트는 장치관리자에서 확인



* 이 과목에서는 "Arduino Uno"선택

프로그램 업로드

- **툴바 사용**
  - 확인 : 스케치 컴파일
  - **업로드** : 스케치 컴파일 및 생성된 기계어 파일을 아두이노로 업로드
    → 한 번의 클릭으로 업로드까지 진행
  - 시리얼 모니터 : 컴퓨터와의 데이터 송수신을 위한 프로그램 실행

스케치의 구조

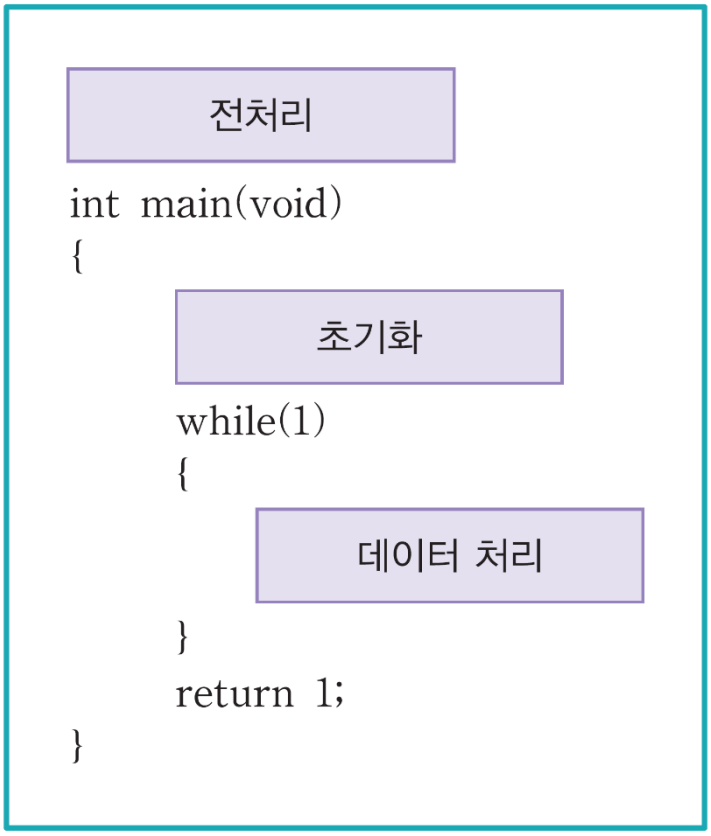
- **스케치는 C/C++을 기반으로 함**
- **main 함수는 존재하지 않음**
  - main 함수는 숨겨져 있으므로 신경 쓰지 않아도 됨
- **2개의 기본 함수로 구성**
  - setup 함수

    - 초기화 함수
    - 스케치 실행이 시작될 때 한 번만 실행

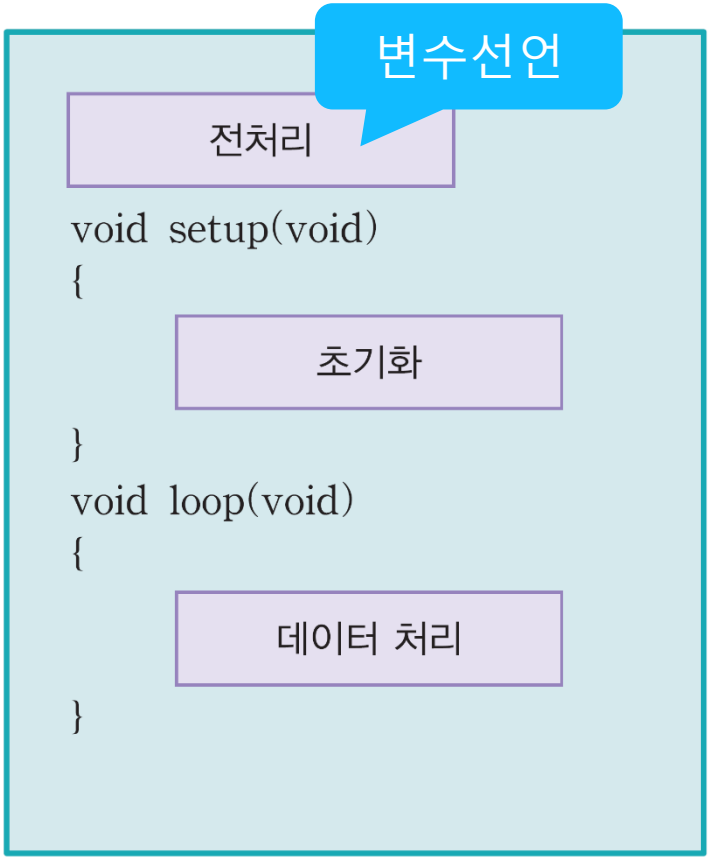
  - loop 함수

    - 반복 실행 함수
    - 아두이노를 위한 프로그램에서 메인/이벤트 루프에 해당

스케치의 구조 비교

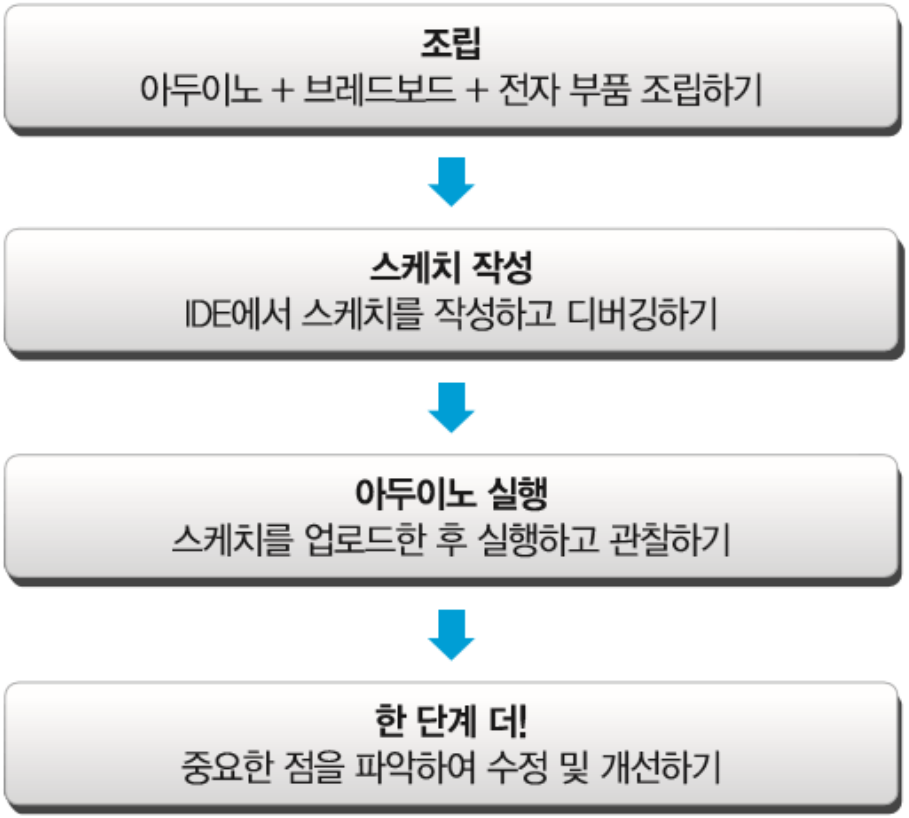


기존 C 프로그램의 구조



아두이노를 위한 스케치의 구조

아두이노 실행 단계


조립
아두이노 + 브레드보드 + 전자 부품 조립하기
스케치 작성
IDE에서 스케치를 작성하고 디버깅하기
아두이노 실행
스케치를 업로드한 후 실행하고 관찰하기
한 단계 더!
중요한 점을 파악하여 수정 및 개선하기

03 아두이노 기본 클래스

아두이노 라이브러리

- **AVR 마이크로컨트롤러의 특정 기능이나, 특정 주변장치 제어를 위한 기능을 라이브러리로 구현**
- **C++ 기반의 객체지향 방식으로 작성**
- **종류**
  - 기본 라이브러리 : 아두이노에서 제공
    - 아두이노 설치 디렉터리 아래 'libraries' 디렉터리
  - 확장 라이브러리 : 써드 파티에서 제공
    - 스케치북 디렉터리 아래 'libraries' 디렉터리

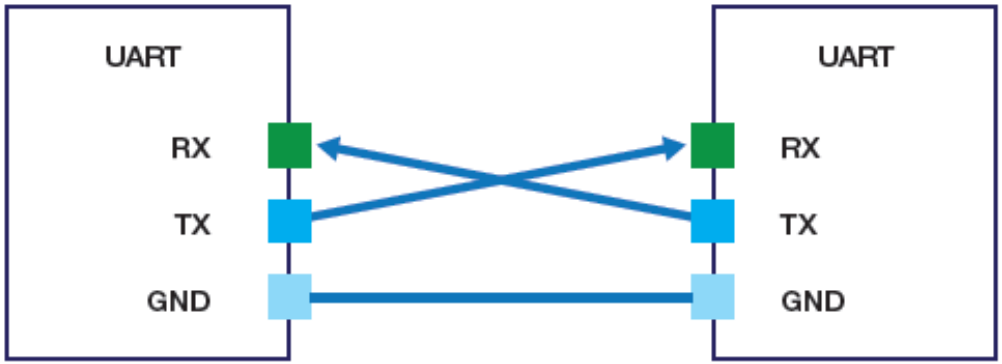
* AVR 마이크로컨트롤러: ATMEL사에서 제작된 축소 명령어 집합 컴퓨터 (RISC, Reduced instruction set computer)구조의 MCU(Micro Controller Unit)

아두이노 라이브러리

- **라이브러리와 별도로 아두이노는 2개의 기본 클래스 제공**
  - UART 시리얼 통신을 위한 Serial 클래스와 문자열 처리를 위한 String 클래스
    * UART (Universal Asynchronous serial Receiver and Transmitter)
  - 별도의 헤더 파일(*.h) 없이 사용 가능

- **시리얼 통신이란?**

정보처리기기는 기본으로 시리얼 통신을 지원한다. 두 기기 간에 통신 속도를 약속하고 송수신 회선을 각각 지정하여 데이터를 주고받는다.



RX: 수신단 / TX: 송신단/ GND: 그라운드 접지(전류)

- UART에는 TX(transmission)와 RX(reception)이 존재한다.
- TX는 **다른** 기기의 RX와 연결하고 RX는 다른 기기의 TX와 연결한다.
- GND는 서로 연결한다.

시리얼 통신

클럭(Clock) 속도란?

- **클럭 신호**는 마이크로컨트롤러(CPU, MCU) 내부에서 **연산이나 신호 처리를 동기화**하는 기준 신호
- 흔히 **Hz(헤르츠)** 단위로 표현: 1Hz = 1초에 1번 신호 변화
- 예: 아두이노 우노(ATmega328P)
  - 기본 클럭 = 16MHz → 1초에 16,000,000번 동작 가능한 기준 신호가 있음

클럭 속도가 높으면?

- 1초 동안 더 많은 연산을 처리할 수 있음 → CPU가 빠름
- 단, 전력 소모 ↑, 발열 ↑ 가능

시리얼 통신

동기식(Synchronous) 통신/작동

- 동기식: 데이터 전송이나 동작이 클럭 신호와 정확히 맞물려서 수행됨
- 장점: 빠르고 타이밍이 정확
- 단점: 클럭 신호를 공유해야 해서 배선/설정 복잡

Ex) 마이크로컨트롤러 내부 연산 → 모든 레지스터와 연산이 클럭 펄스마다 동작

SPI, I2C 통신 → 마스터 클럭과 동기화된 데이터 전송

비동기식(Asynchronous) 통신/작동

- 비동기식: 클럭 신호 없이, 시작/종료 신호(Start/Stop)로 동작을 맞춤
- 장점: 배선 단순, 클럭 공유 필요 없음
- 단점: 동기식보다 속도가 느리고, 타이밍 오차 가능성 있음

Ex) UART/Serial 통신 → 시작 비트, 데이터, 정지 비트로 데이터를 주고받음

마이크로컨트롤러 내부에서도 인터럽트 기반 이벤트 처리는 비동기적

Serial 클래스

- **UART 시리얼 통신을 위한 클래스 UART(Universal Asynchronous Receiver/Transmitter)**
- **실제 클래스 이름은 Serial_이며, 그 객체가 Serial임**
- **우노는 하나의 UART 시리얼 통신 포트만을 제공**

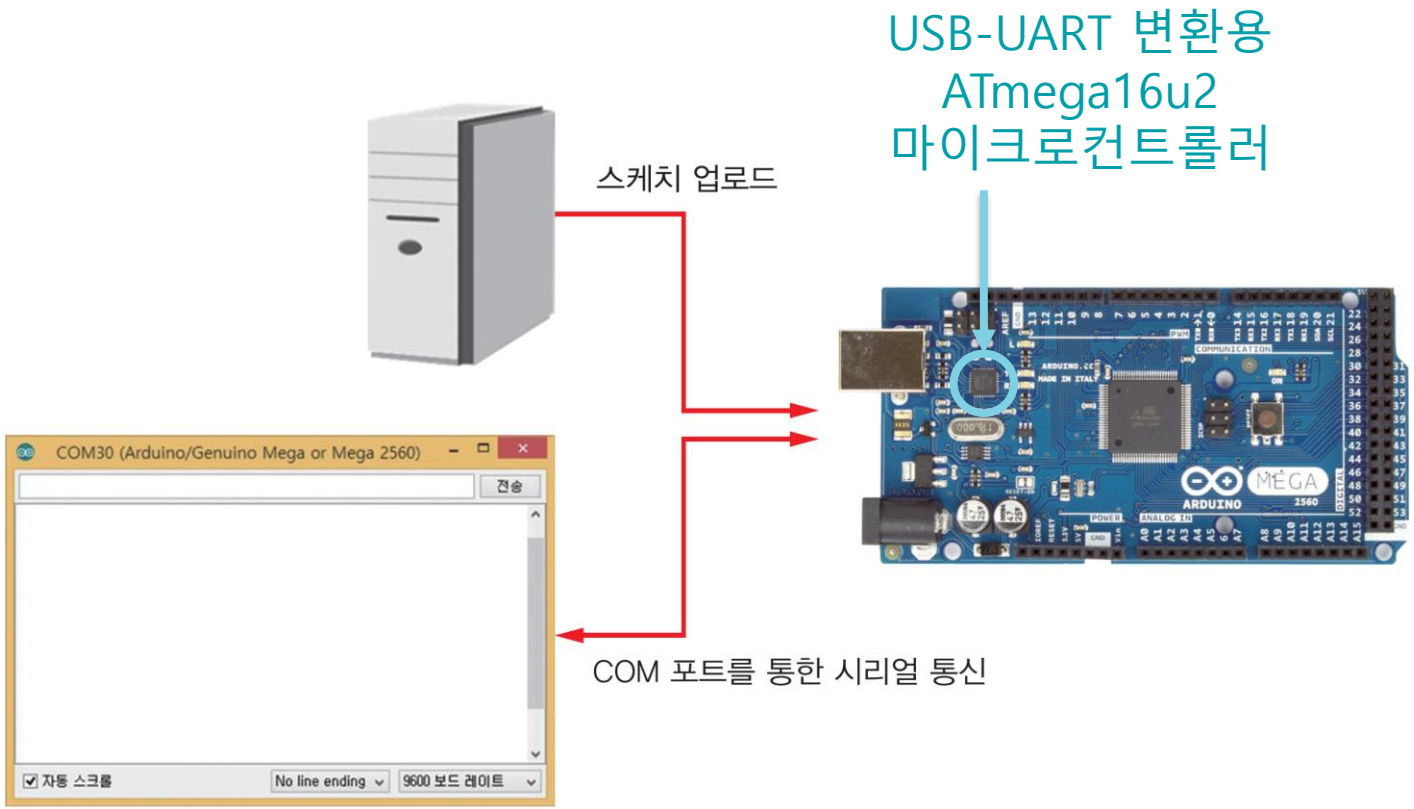
UART 시리얼 통신

- RS-232C 시리얼 케이블에서 정의된 RX, TX, GND 핀을 사용하여 통신
- UART 시리얼 통신은 시리얼 통신의 한 종류이지만, 흔히 시리얼 통신은 UART 시리얼 통신을 가리킴
- 아두이노 보드와 컴퓨터의 통신은 USB로 이루어짐
  - 아두이노 보드의 ATmega16u2 컨트롤러가 USB와 UART 사이의 변환을 담당
  - 0번 UART 채널에 해당하는 Serial 객체를 통해 USB를 통한 컴퓨터와의 시리얼 통신 수행

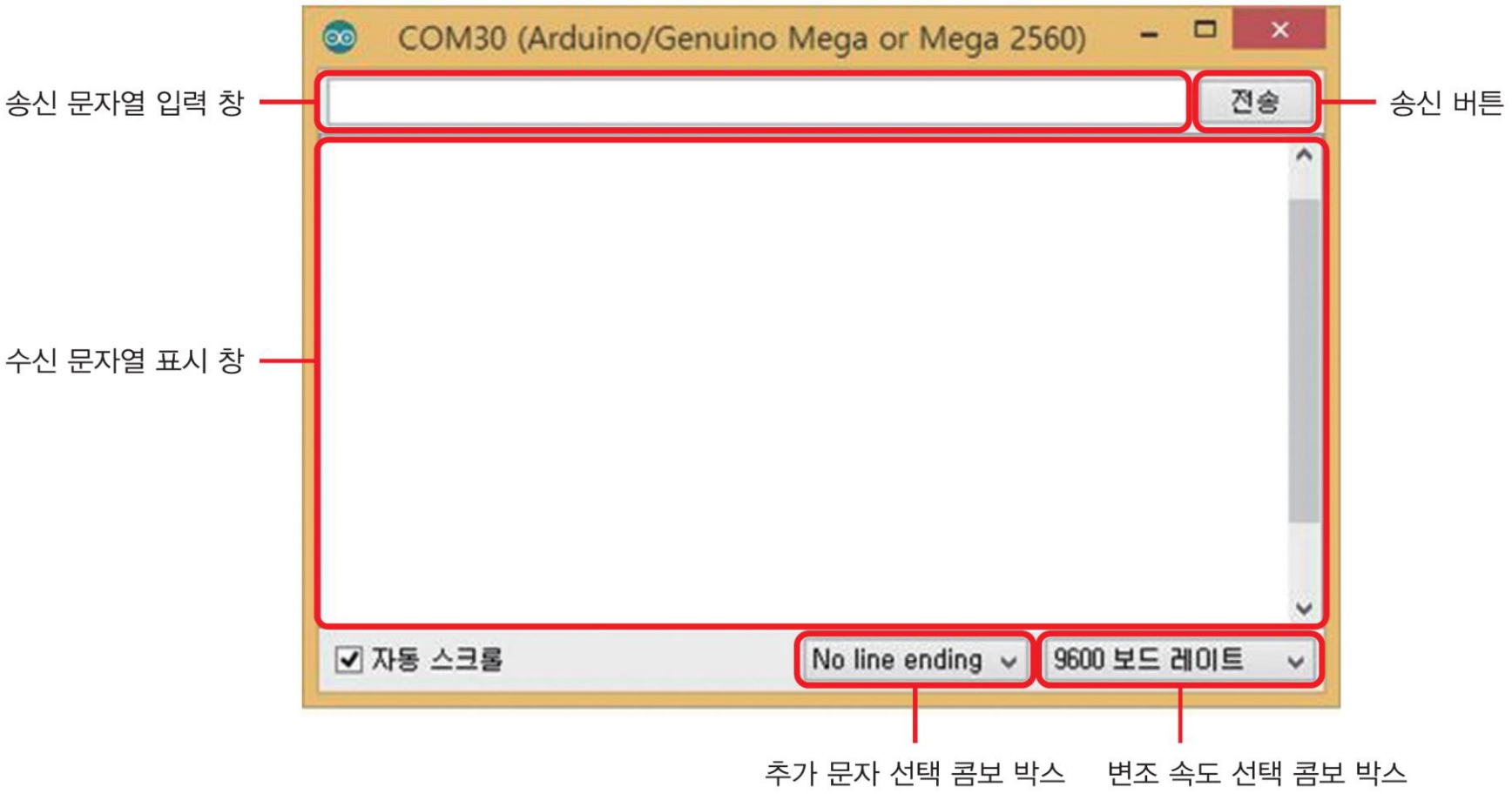
컴퓨터-아두이노 USB 연결

① 스케치 업로드를 위해 사용

② 시리얼 통신을 위해 사용



시리얼 모니터


COM30 (Arduino/Genuino Mega or Mega 2560)
송신 문자열 입력 창
전송
송신 버튼
수신 문자열 표시 창
자동 스크롤
No line ending
9600 보드 레이트
추가 문자 선택 콤보 박스
변조 속도 선택 콤보 박스

Serial 클래스 – 데이터 출력 함수

size_t print(value, format)

- **매개변수**
  - value : 출력값(char, char 배열, String, 정수, 실수 등)
  - format : 출력 형식
- **반환값** : 시리얼 포트로 출력된 바이트 수

size_t wirte(uint8_t ch)

- **매개변수**
  - ch : 출력할 바이트 단위 데이터
- **반환값** : 시리얼 포트로 출력된 바이트 수

uno_ex2-1.ino: print와 write

Serial.print() : 숫자를 "사람이 읽을 수 있는 문자열"로 바꿔서 출력
예: **Serial.print(65);** → **"65"** (문자 **'6'**과 **'5'** 전송)

Serial.write() : 숫자 값을 그대로 **1**바이트(또는 지정한 크기)로 전송
예: **Serial.write(65);** → 아스키 코드 **65** → **'A'**

즉, Serial.write()는 통신 장치끼리 원시 데이터를 주고받을 때 주로 쓰이고, 사람이 확인하려고 할 때는 **Serial.print()/println()**을 더 많이 사용

ex2_1.ino | Arduino IDE 2.3.2

파일 편집 스케치 도구 도움말(H)

Arduino Uno

ex2_1.ino.ino

```
1  int data = 65;//변수 선언
2
3  void setup()
4  {
5
6    Serial.begin(9600);
7
8    Serial.print("data is ");
9    Serial.println(data);//숫자표현
10
11   Serial.print("data is ");
12   Serial.write(data);//아스키코드 변환하여 문자 표현
13
14 }
15
16 void loop()
17 {
18
19 }
```

통신속도 일치 확인!

Serial.print()
•데이터를 **그대로 출력**
•출력 후 **줄바꿈 없음**

Serial.println()
•데이터를 출력한 뒤 **자동으로 개행 문자(\r\n)** 추가
•즉, 출력 후 줄을 바꿔 줌

보드레이트**(9600bps)**의 의미
•**9600bps** = 1초에 9600 비트(bit)를 전송
•UART(Universal Asynchronous Receiver/Transmitter) **비동기 통신 표준**에서, 송신기와 수신기가 **같은 속도로 비트를 처리**하도록 맞추는 값임.

출력 시리얼 모니터

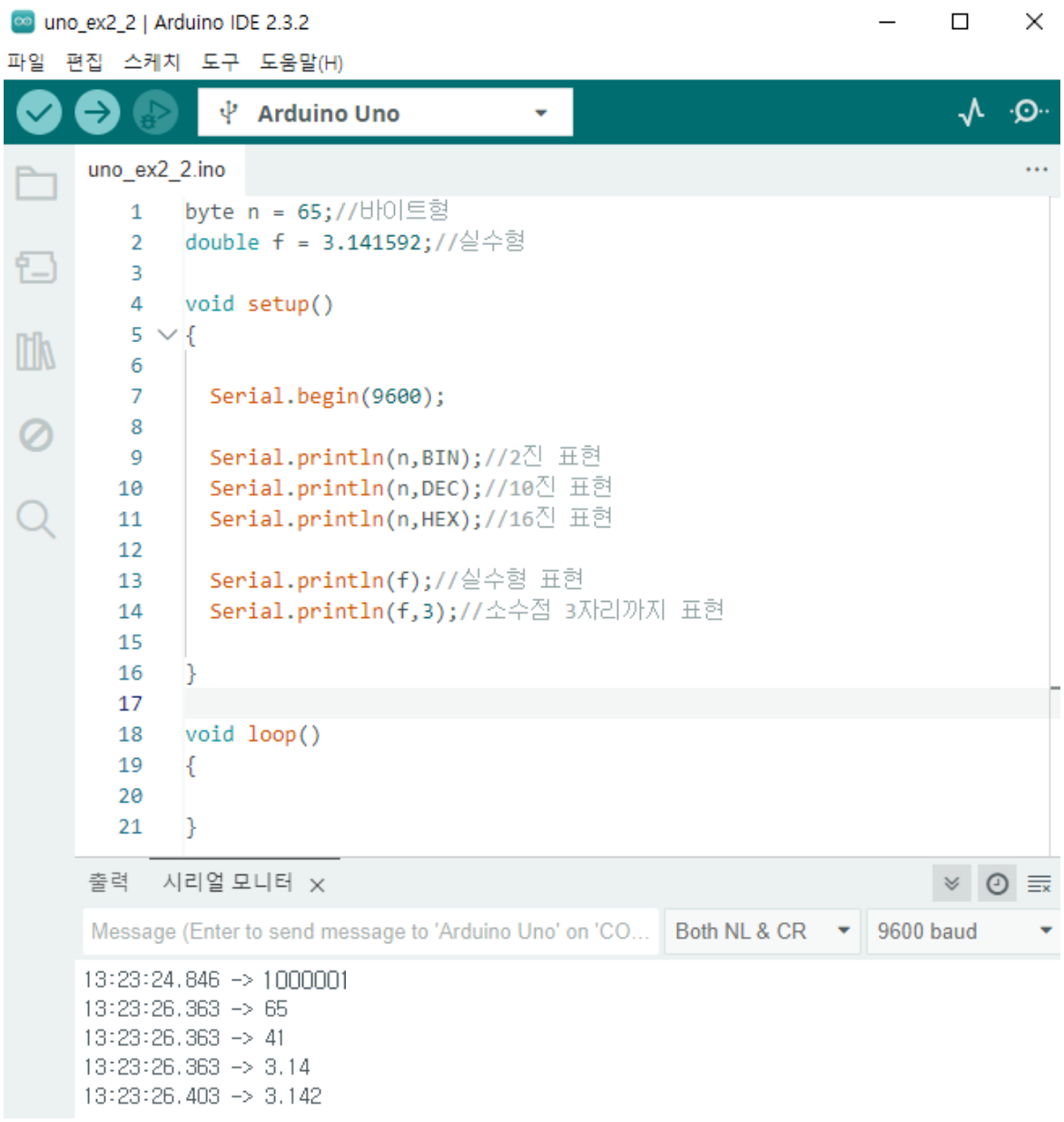
Message (Enter to send message to 'Arduino Uno' on 'CO...

Both NL & CR

9600 baud

```
13:19:31.832 -> data is 65
13:19:33.362 -> data is A
```

uno_ex2-2.ino : 진법과 소수점 이하 자릿수



uno_ex2-3.ino : 에코 백

uno_ex2_3.ino

```
1   byte data = 0;
2
3   void setup()
4   {
5     Serial.begin(9600);
6   }
7
8   void loop()
9   {
10    if(Serial.available()>0){//수신된 존재 여부 확인
11
12      data = Serial.read();//실제 데이터 읽기
13
14      Serial.print("Echo back:");
15      Serial.println(data);
16    }
17  }
```

Serial.available() → 수신 버퍼에 남아 있는 바이트 수를 반환
> 0이면 읽을 데이터가 있다는 뜻

출력 시리얼 모니터 ×

단어("apple") 입력후 엔터

Message (Enter to send message to 'Arduino

```
20:05:59.057 -> Echo back:a 97
20:05:59.057 -> Echo back:p 112
20:05:59.090 -> Echo back:p 112
20:05:59.090 -> Echo back:l 108
20:05:59.129 -> Echo back:e 101
20:05:59.129 -> Echo back:
20:05:59.129 -> 10
```

시리얼 통신으로 받은 데이터를 다시 보내주는(에코) 예제 : PC 시리얼 모니터에서 데이터 보내면 → 아두이노가 그대로 다시 출력(에코).

제어 문자		공백 문자		구두점		숫자		알파벳			
10진	16진	문자	10진	16진	문자	10진	16진	문자	10진	16진	문자
0	0x00	NUL	32	0x20	SP	64	0x40	@	96	0x60	`
1	0x01	SOH	33	0x21	!	65	0x41	A	97	0x61	a
2	0x02	STX	34	0x22	"	66	0x42	B	98	0x62	b
3	0x03	ETX	35	0x23	#	67	0x43	C	99	0x63	c
4	0x04	EOT	36	0x24	$	68	0x44	D	100	0x64	d
5	0x05	ENQ	37	0x25	%	69	0x45	E	101	0x65	e
6	0x06	ACK	38	0x26	&	70	0x46	F	102	0x66	f
7	0x07	BEL	39	0x27	'	71	0x47	G	103	0x67	g
8	0x08	BS	40	0x28	(	72	0x48	H	104	0x68	h
9	0x09	HT	41	0x29	)	73	0x49	I	105	0x69	i
10	0x0A	LF	42	0x2A	*	74	0x4A	J	106	0x6A	j
11	0x0B	VT	43	0x2B	+	75	0x4B	K	107	0x6B	k
12	0x0C	FF	44	0x2C	,	76	0x4C	L	108	0x6C	l
13	0x0D	CR	45	0x2D	-	77	0x4D	M	109	0x6D	m
14	0x0E	SO	46	0x2E	.	78	0x4E	N	110	0x6E	n
15	0x0F	SI	47	0x2F	/	79	0x4F	O	111	0x6F	o
16	0x10	DLE	48	0x30	0	80	0x50	P	112	0x70	p
17	0x11	DC1	49	0x31	1	81	0x51	Q	113	0x71	q
18	0x12	DC2	50	0x32	2	82	0x52	R	114	0x72	r
19	0x13	DC3	51	0x33	3	83	0x53	S	115	0x73	s
20	0x14	DC4	52	0x34	4	84	0x54	T	116	0x74	t
21	0x15	NAK	53	0x35	5	85	0x55	U	117	0x75	u
22	0x16	SYN	54	0x36	6	86	0x56	V	118	0x76	v
23	0x17	ETB	55	0x37	7	87	0x57	W	119	0x77	w
24	0x18	CAN	56	0x38	8	88	0x58	X	120	0x78	x
25	0x19	EM	57	0x39	9	89	0x59	Y	121	0x79	y
26	0x1A	SUB	58	0x3A	:	90	0x5A	Z	122	0x7A	z
27	0x1B	ESC	59	0x3B	;	91	0x5B	[	123	0x7B	{
28	0x1C	FS	60	0x3C	<	92	0x5C	₩	124	0x7C	|
29	0x1D	GS	61	0x3D	=	93	0x5D	]	125	0x7D	}
30	0x1E	RS	62	0x3E	>	94	0x5E	^	126	0x7E	~
31	0x1F	US	63	0x3F	?	95	0x5F	_	127	0x7F	DEL

String 클래스

- **문자열 처리를 위한 클래스**
- **C++에서의 문자열 처리를 위한 클래스와 유사**
- **문자, 문자열, 숫자 등으로부터 String 객체 생성 가능**
- **String 객체와 다른 타입의 데이터 결합으로 새로운 String 객체 생성**

```
String( " abc " ) + 123;          // String 객체 + 정수
```

- **큰따옴표로 표시되는 문자열은 String 클래스 객체가 아닌 문자 배열로 처리**

```
" abc "  + 123;                   // 문자열 배열 + 정수 (×)
```

uno_ex2-4.ino : 문자열 생성 방법

uno_ex2_4.ino

```
1   String str1="Echo:";
2   String str2="";//입력받은 문자 조합
3   byte data = 0;//키보드 입력
4
5   void setup()
6   {
7     Serial.begin(9600);
8   }
9
10  void loop()
11  {
12    if(Serial.available()>0){//수신된 존재 여부 확인
13
14      data = Serial.read();//실제 데이터 읽기
15
16      if(data==10){//엔터 입력시
17
18        //실제 출력
19        Serial.println(str1+" "+str2);
20        str2="";//문자열 초기화
21
22      }else{//엔터 입력 아닌경우 모두 문자
23        str2+=(char)data;//기존 문자열 조합에 추가
24      }//if-else end
25    }//if end
26  }//loop end
```

출력 시리얼 모니터 ×

Message (Enter to send message to 'Arduino Uno' on 'COM4')

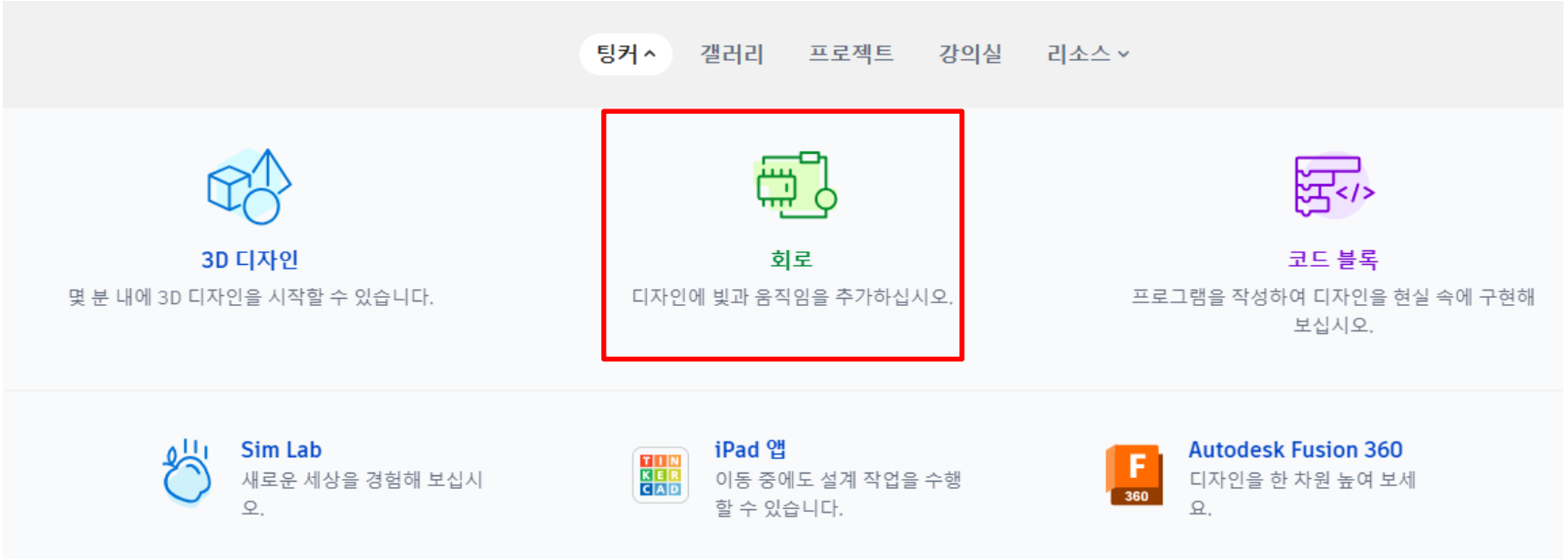
13:36:43.873 -> Echo: apple

* 앞으로 입력 데이터는 □ 로 표시

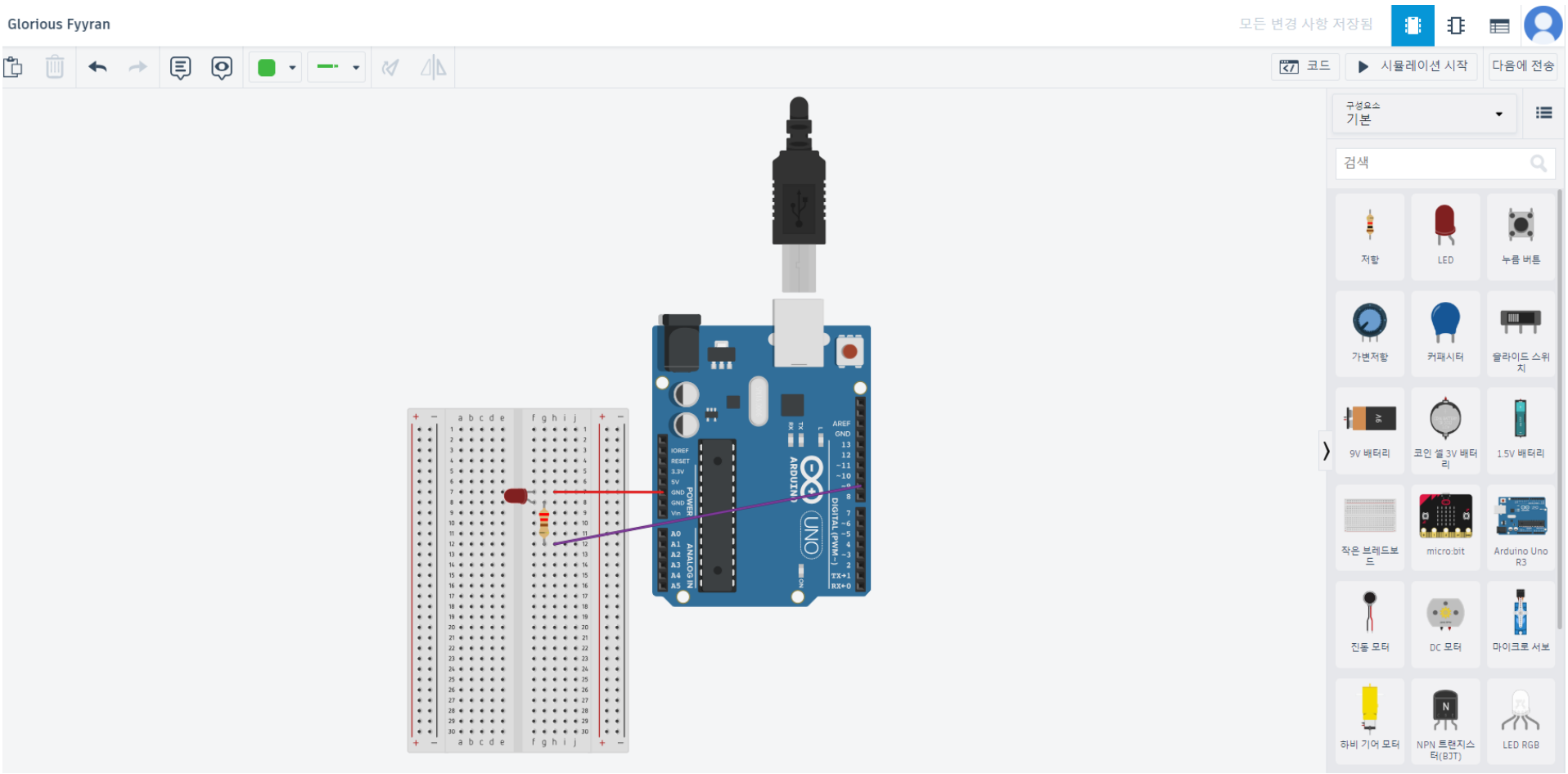
04 아두이노 온라인 시뮬레이션

팅커캐드

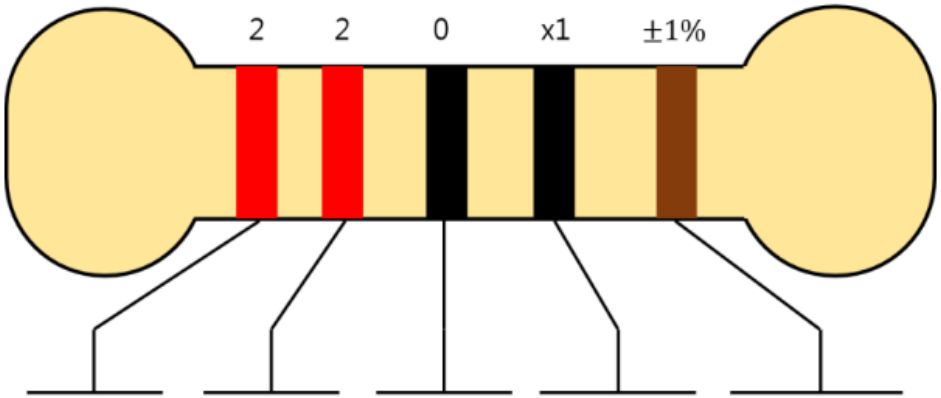
- https://www.tinkercad.com 회원가입



팅커캐드


Glorious Fyyran
모든 변경 사항 저장됨
코드
시뮬레이션 시작
다음에 전송
구성요소
기본
검색
저항
LED
누름 버튼
가변저항
커패시터
슬라이드 스위치
9V 배터리
코인 셀 3V 배터리
1.5V 배터리
작은 브레드보드
micro:bit
Arduino Uno R3
진동 모터
DC 모터
마이크로 서보
하비 기어 모터
NPN 트랜지스터(BJT)
LED RGB

4. 저항 읽는 법



첫 번째	두 번째	세 번째	단위	오차
0	0	0	x0.01(은)	±10% (은)
1	1	1	x0.1(금)	±5%(금)
2	2	2	x1	
3	3	3	x10	±1%
4	4	4	x100	±2%
5	5	5	x1K	
6	6	6	x10K	
7	7	7	x100K	±0.5%
8	8	8	x1M	±0.25%
9	9	9	x10M	±0.1%

< 220 Ω 저항 >

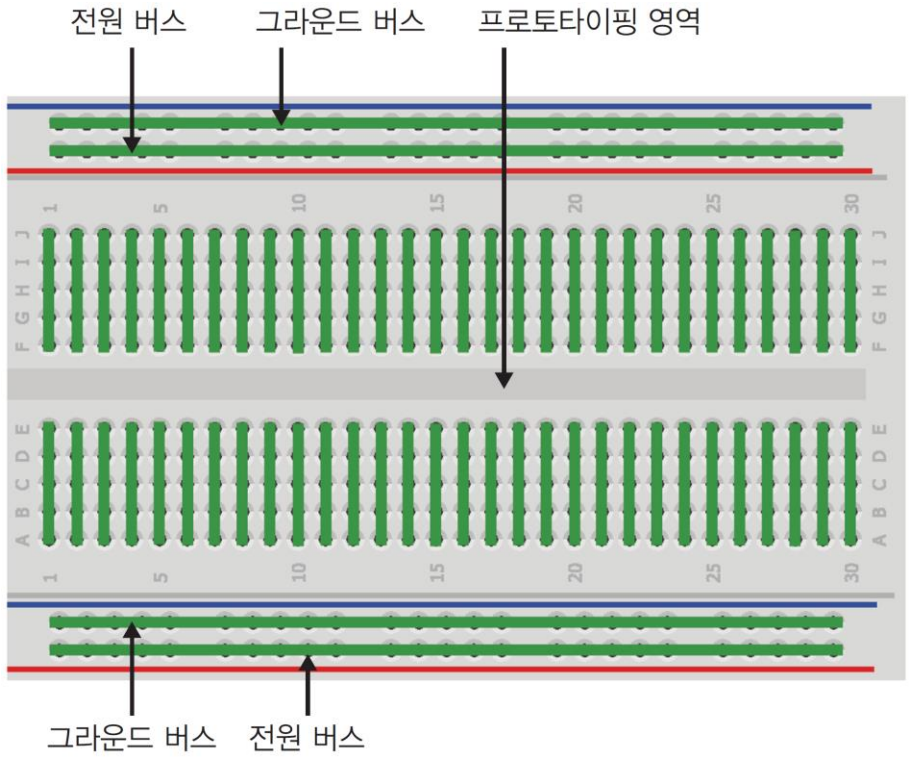
요약

- **아두이노 보드 구성 : 전원, 입출력핀**
- **아두이노 보드 종류 : 아누이노 우노, 메가 등 (이외 라즈베리파이)**
- **아두이노 IDE환경 설치 및 설정**
  - 보드 선택 / 포트 반드시 확인
- **아두이노 프로그램 실행단계**
- **기본 클래스**
  - Serial
    - 통신 초기화: begin(속도)
    - 출력함수: print(), println(), write()
    - 입력함수: available(), read()
  - String

LED 모듈 연결 실습

필요한 도구 : 브레드 보드

- 부품을 납땜하지 않고도 회로 구성을 가능하게 하는 프로토타이핑 도구
- 양쪽에 파란색(그라운드)과 빨간색(전원 버스) 홀은 연결된 상태
- 수직 홀은 5개 그룹으로 연결된 상태
- 각 핀이 내부적으로 연결되어 있어 각 전기/전자 부품을 연결하기 편리함

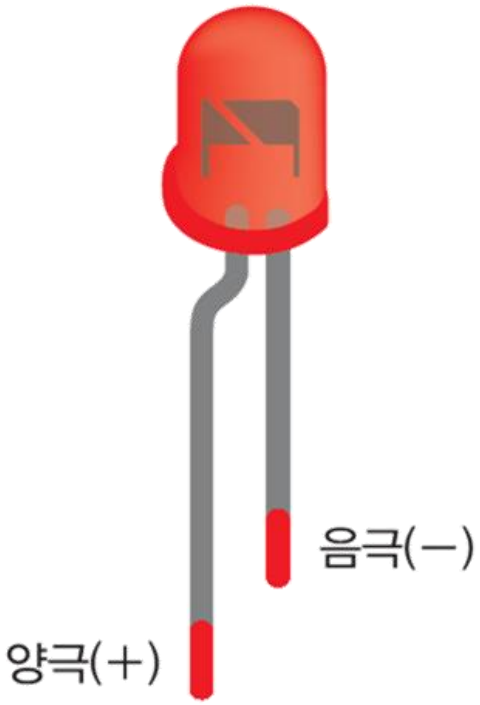


필요한 도구 : LED(Light Emitting Diode, 발광 다이오드)

- 양극(+, 애노드)에서 음극(-, 캐소드)방향으로만 전류가 흐름
  - 따라서 +, - 방향을 고려하여 연결해야 한다.

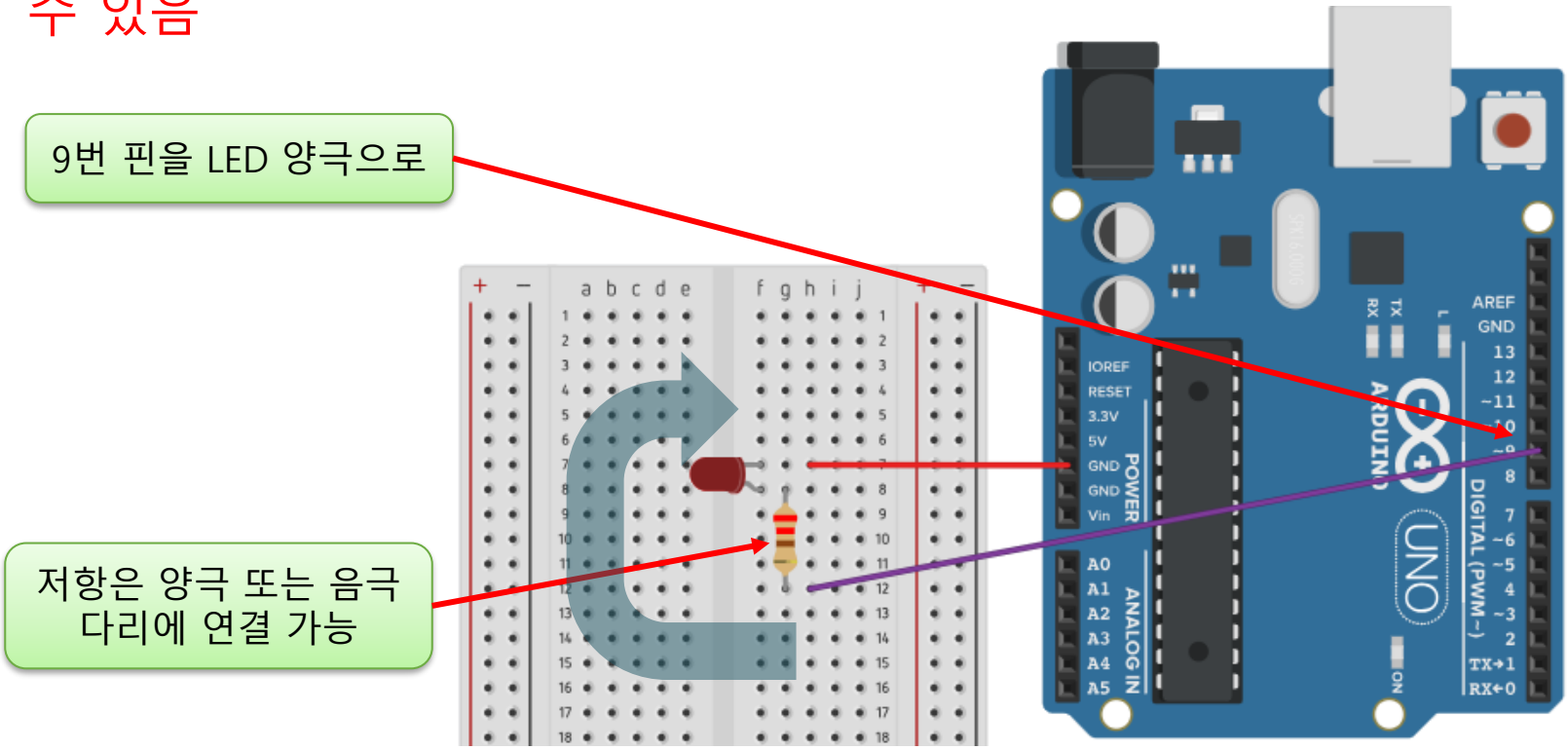
- 양극과 음극의 2개 다리를 가짐
  - 다리 길이가 긴 쪽이 양극

- 과도한 전류가 흐르지 않도록 전류 제한 필요
  - 200Ω 전후 저항이 일반적으로 사용됨



아두이노에 LED 연결하기

- 아래 그림과 같이 LED를 아두이노에 연결한다.
  - 일반적인 LED는 2V의 전압과 2mA의 전류를 소모하기 때문에 5V의 아두이노 전압을 낮추기 위해 저항을 사용해야함
  - 220Ω 저항을 LED에 직렬로 연결
- 저항을 사용하지 않고 LED를 직접 연결하면 LED가 타버리거나 손상될 수 있음



LED 예제 함수

void pinMode(uint8_t pin, uint8_t mode)

- **매개변수**
  - pin : 설정하고자 하는 핀 번호
  - mode : INPUT, OUTPUT, INPUT_PULLUP 중 하나
- **반환값** : 없음

void digitalWrite(uint8_t pin, uint8_t value)

- **매개변수**
  - pin : 핀 번호
  - value : HIGH(1) 또는 LOW(0)
- **반환값** : 없음

=>디지털 출력: 1또는 0으로만 표현

void delay(unsigned long ms)

- **매개변수**
  - ms : 밀리초 단위의 지연 시간
- **반환값** : 없음

uno_ex3-1: LED 모듈 예제

- 아두이노의 9번 핀에 연결된 LED를 켜기 위해, 9번 핀을 출력 모드로 설정

```
1   const int LED=9; //9번 핀을 사용하는 LED 상수 정의
2
3   void setup()
4   {
5       pinMode(LED, OUTPUT); //LED(9번 핀)를 출력으로 지정
6       digitalWrite(LED, HIGH); //LED를 HIGH로 지정
7   }
8
9   void loop()
10  {
11      //이 실습에서는 loop문을 비움
12  }
```

➔ 9번 핀을 5V로 설정 → LED 켜짐

디지털 9번 핀 (5V) → 저항 (220Ω) → LED (+극 → -극) → GND → 아두이노 내부 회로 → 다시 전원 공급 회로로 반환

```
const int LED = 9;

void setup() {
  // put your setup code here, to run once:
  pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW);
}

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

```
void setup() {
  // put your setup code here, to run once:
  pinMode(9, OUTPUT);
  digitalWrite(9, HIGH);
  delay(1000);
  digitalWrite(9, LOW);
  delay(1000);
}
void loop() {
  // put your main code here, to run repeatedly:
}
```

Thank You