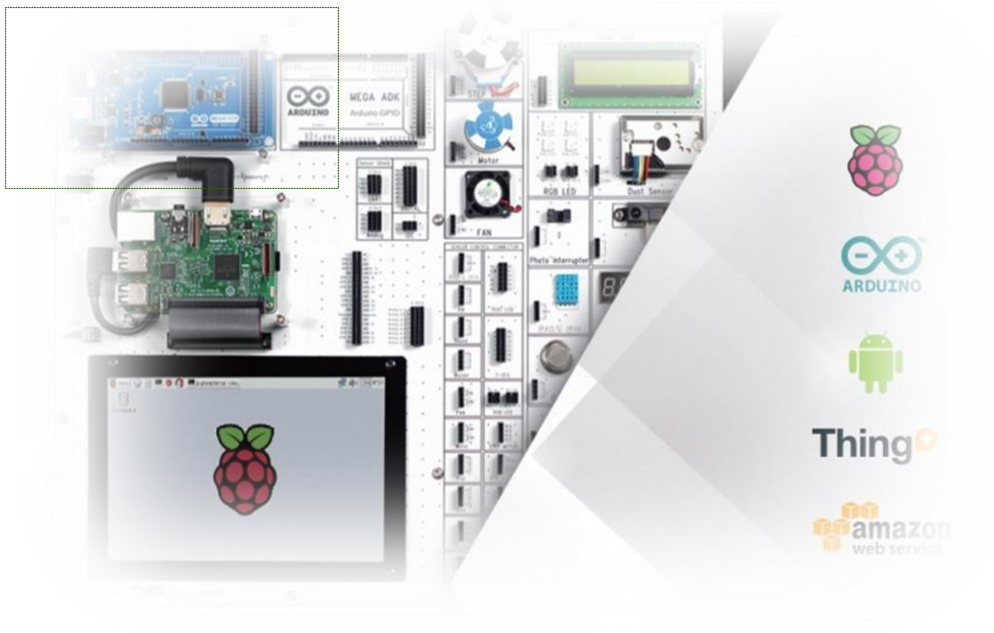
마이크로프로세서
(강의자료 #7)



교과목명 : 마이크로프로세서 (01)

담당교수 : 이 수 형

E-mail : soohyong@uu.ac.kr

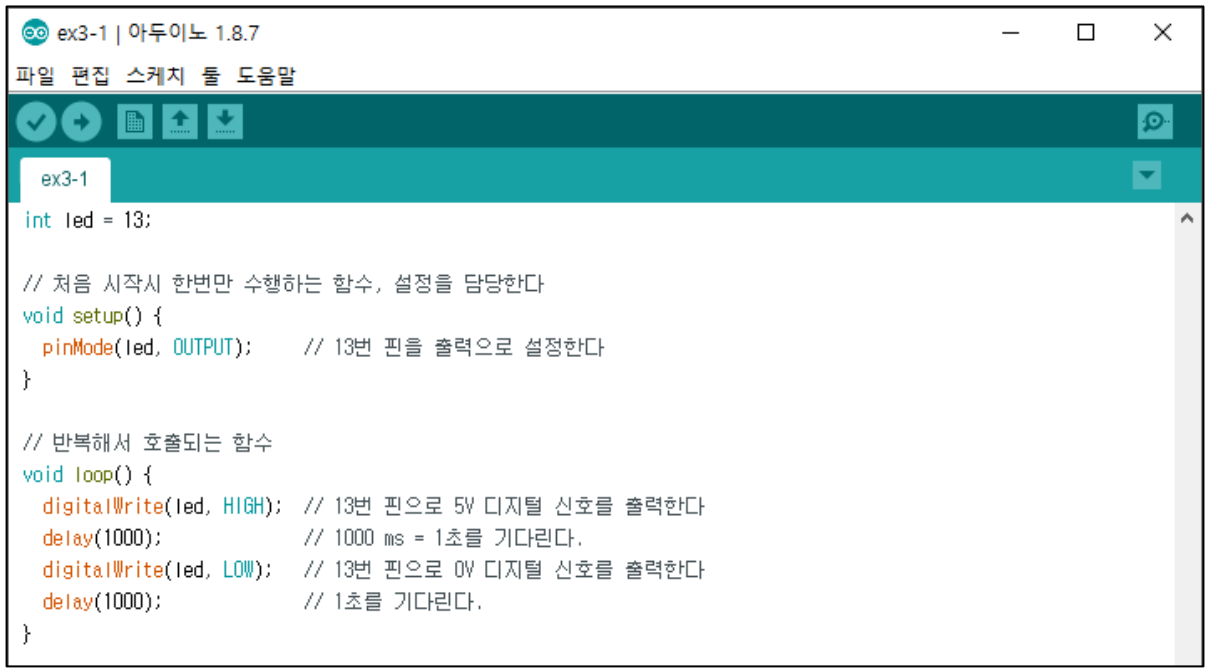
교재명 : 스마트기기 개발을 위한 아두이노, 이수형. (LINC+ 사업단 배포)

수강생 안내

- 본 교과목은 기본적으로 “대면”수업입니다.
- ‘코로나바이러스감염증-19’ 등의 이유로 대면수업을 받지 못하는 학생들을 위해서 ‘비대면 실시간 수업’을 병행합니다. 기본적으로 대면 수업이므로 대면 수업에 참석한 학생들은 교실에서 출석을 체크하면 되며, 참석하지 못하고 ‘비대면 수업’을 듣는 학생들은 ‘실시간 온라인 강의’에 참석하면 출석이 반영됩니다.
- 수강생 여러분들은 강의 자료를 온라인 배포 및 게재 등의 행위를 할 시 저작권 문제가 발생할 수 있사오니 이에 유념하여 강의 자료를 공유하는 행위는 삼가 바랍니다.
- 교재는 “스마트기기 개발을 위한 아두이노”이며, 나누어준 키트는 각자 잘 관리하면서 실습실(대면) 및 집(비대면)에서 실험/실습을 수행할 수 있도록 진행합니다.

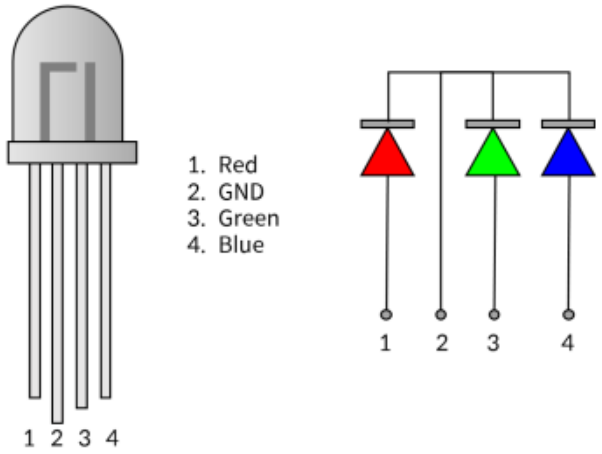
지난시간에는?

6. 출력장치

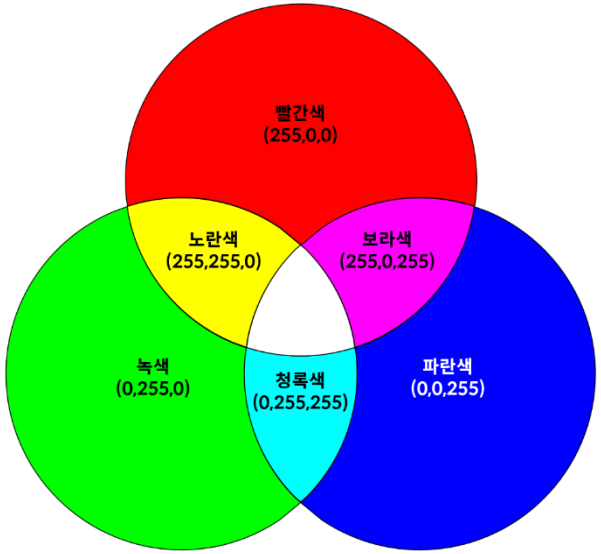

ex3-1 | 아두이노 1.8.7
파일 편집 스케치 툴 도움말
ex3-1
int led = 13;
// 처음 시작시 한번만 수행하는 함수, 설정을 담당한다
void setup() {
pinMode(led, OUTPUT); // 13번 핀을 출력으로 설정한다
}
// 반복해서 호출되는 함수
void loop() {
digitalWrite(led, HIGH); // 13번 핀으로 5V 디지털 신호를 출력한다
delay(1000); // 1000 ms = 1초를 기다린다.
digitalWrite(led, LOW); // 13번 핀으로 0V 디지털 신호를 출력한다
delay(1000); // 1초를 기다린다.
}

6.1 RGB LED

- RGB LED
  - LED내부에 3색의 LED가 들어가있는 소자
  - 각각의 색의 밝기를 조절함으로 모든 색을 표현

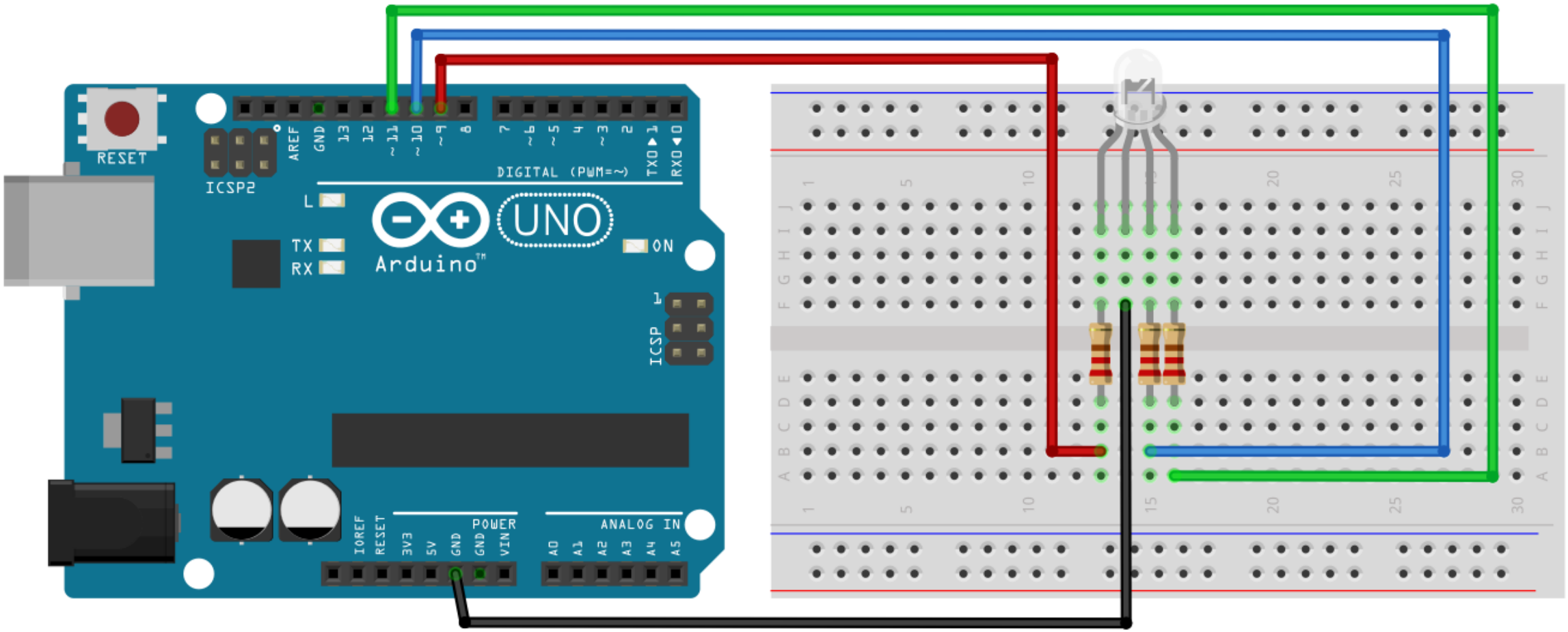


RGB LED의 구조
(common cathode)



빛의 삼원색

- 회로구성


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino™
ON
ICSP
POWER
IOREF
RESET
3V3
5V
GND
GND
VIN
ANALOG IN
A0
A1
A2
A3
A4
A5

예제 6-1: RGB LED 제어 스케치

```
// RGB 각 색에 해당하는 핀 번호 설정(PWM)
int red = 9;     // 빨간색
int blue = 10;   // 파란색
int green = 11; // 녹색

void setup()
{
  // 출력핀 설정
  pinMode(red, OUTPUT);
  pinMode(green, OUTPUT);
  pinMode(blue, OUTPUT);
}

void loop()
{
  // 빨간색 표시
  analogWrite(red, 255);    // 빨간색 켜기
  analogWrite(green, 0);    // 녹색 끄기
  analogWrite(blue, 0);     // 파란색 끄기
  delay(1000);              // 1초간 대기
…
```

예제 6-1: RGB LED 제어 스케치

```
  // 녹색 표시
  analogWrite(red, 0);      // 빨간색 끄기
  analogWrite(green, 255);  // 녹색 켜기
  analogWrite(blue, 0);     // 파란색 끄기
  delay(1000);              // 1초간 대기

  // 파란색 표시
  analogWrite(red, 0);      // 빨간색 끄기
  analogWrite(green, 0);    // 녹색 끄기
  analogWrite(blue, 255);   // 파란색 켜기
  delay(1000);              // 1초간 대기

  // 흰색 표시
  analogWrite(red, 255);    // 빨간색 켜기
  analogWrite(green, 255);  // 녹색 켜기
  analogWrite(blue, 255);   // 파란색 켜기
  delay(1000);              // 1초간 대기
}
```

예제 6-2: 색을 선택하는 함수사용

색을 선택하는 함수

```
// 주어진 red, green, blue 성분으로 LED 제어
void setcolor(int r, int g, int b)
{
  analogWrite(red, r);   // 빨간색
  analogWrite(green, g); // 녹색
  analogWrite(blue, b);  // 파란색
}
```

수정한 loop()함수

```
void loop()
{
  // 빨간색 표시
  setcolor(255, 0, 0);
  delay(1000);                  // 1초간 대기
  // 녹색 표시
  setcolor(0, 255, 0);
  delay(1000);                  // 1초간 대기
  // 파란색 표시
  setcolor(0, 0, 255);
  delay(1000);                  // 1초간 대기
  // 흰색 표시
  setcolor(255, 255, 255);
  delay(1000);                  // 1초간 대기
}
```

예제 6-3: 난수발생

- 난수 (random number)
  - 무작위로 만들어지는 수
  - C언어에서는 rand() 함수 사용 : 0 ~ RAND_MAX 사이의 숫자
    - `int r = rand() % 256;` ⇒ 0~255까지 난수를 발생시킴
  - 아두이노에서는 random() 함수 사용
    - `int ra = random(200);` ⇒ 0~199 사이의 난수 발생
    - `int rb = random(10, 20);` ⇒ 10~19 사이의 난수 발생

예제 6-3. 난수를 이용한 RGB LED 제어 프로그램 (loop 함수만)

```
void loop()
{
  // 0~255사이의 난수를 생성
  int r = rand() % 256;
  int g = rand() % 256;
  int b = rand() % 256;
  // 생성한 세개의 난수로 RGB성분을 조정
  setcolor(r, g, b);
  // 10ms 대기
  delay(10);
}
```

```
int r = random(256);
int g = random(256);
int b = random(256);
```

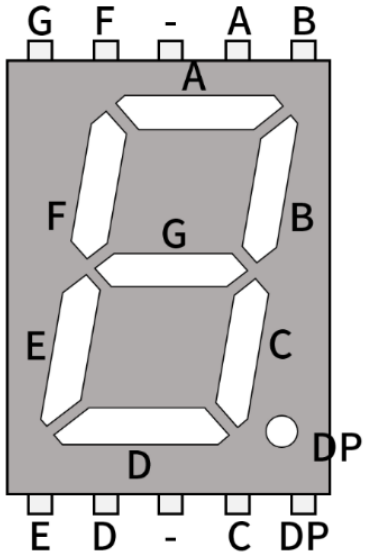
과제

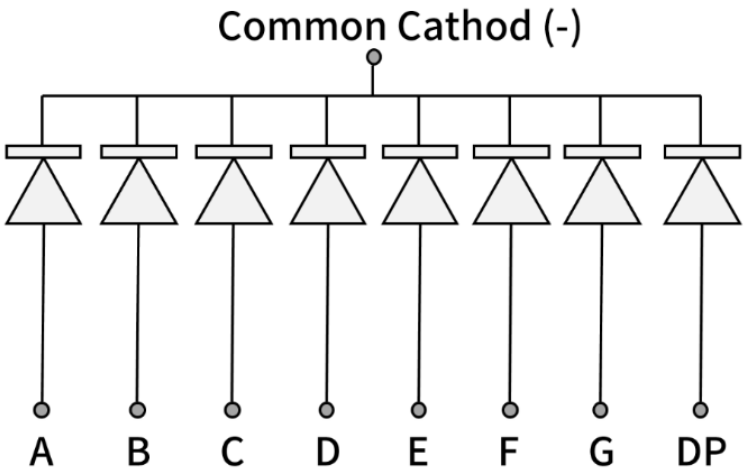
- 앞에서 수행한 실습 과제를 동일하게 수행하고 스케치 파일, 결과 파일을 제출하라.

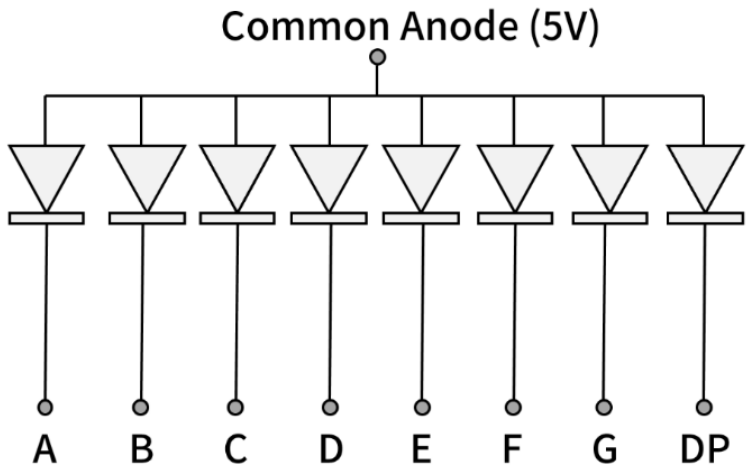
- 예제 6-2를 응용하여 시리얼 통신을 이용하여 전송하는 문자에 따라 색상이 변하는 스케치를 완성하라.
  - ‘r’ : red, ‘g’ : green, ‘b’ : blue, ‘m’ : magenta (보라색), ‘c’ : cyan (하늘색), ‘y’ : yellow, ‘w’ : white, ‘k’ : black → 빛의 3원색을 이용하는 7가지 색상

6.2 FND (Flexible Number Display)

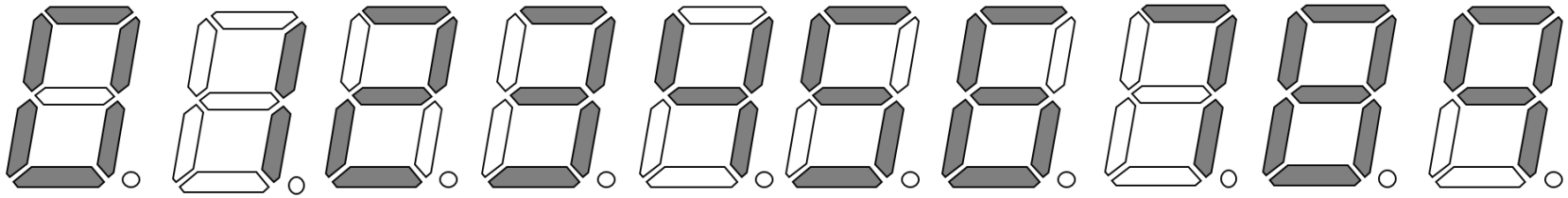
- 7개의 LED를 이용하여 숫자를 표시하는 LED
  - 7-Segment Display (+ 1 LED for dot)
  - 숫자 0을 표시 : A, B, C, D, E, F 의 LED들을 ON





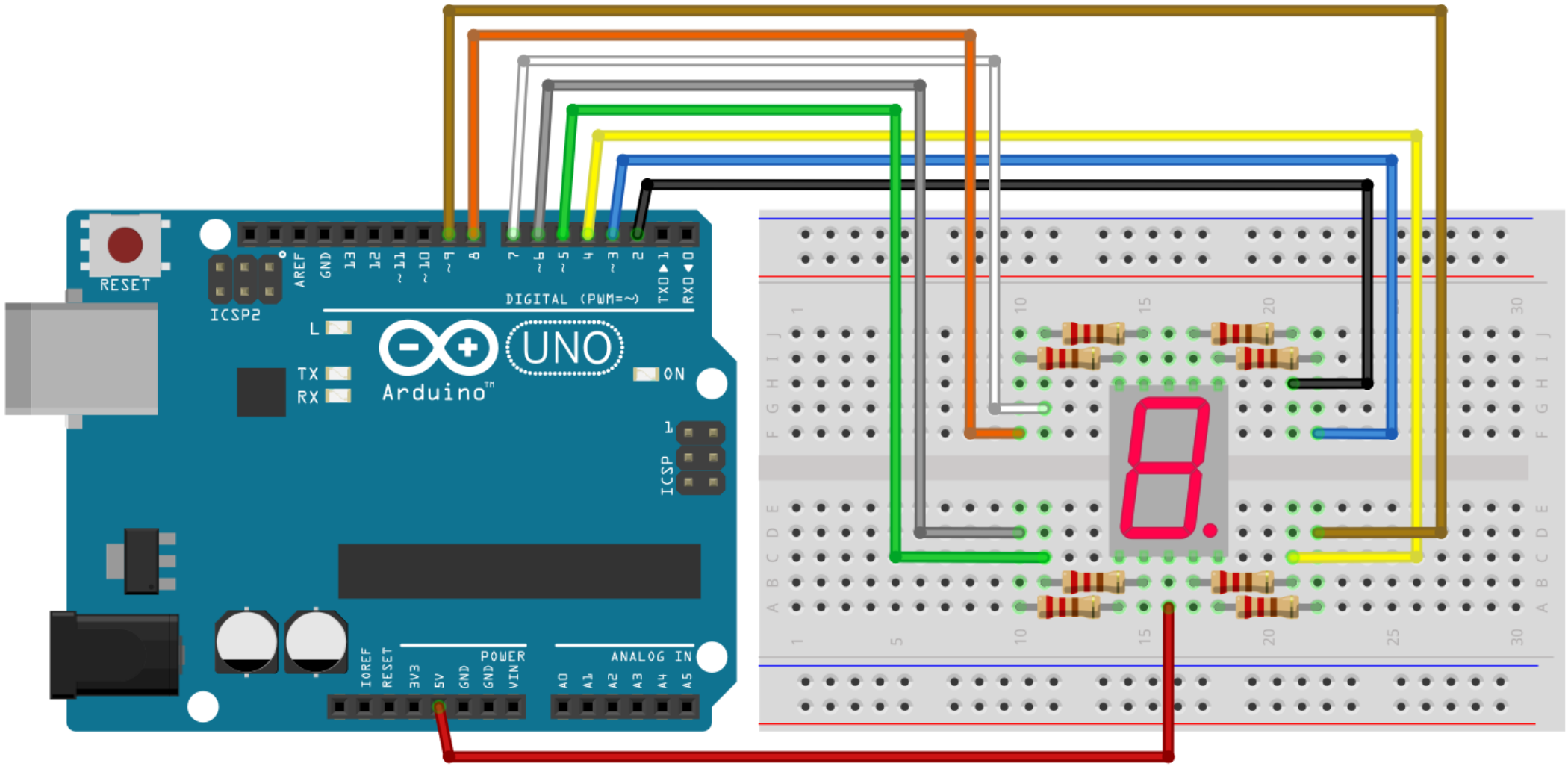


FND의 숫자 표시 테이블



	A	B	C	D	E	F	G	DP
0	1	1	1	1	1	1	0	0
1	0	1	1	0	0	0	0	0
2	1	1	0	1	1	0	1	0
3	1	1	1	1	0	0	1	0
4	0	1	1	0	0	1	1	0
5	1	0	1	1	0	1	1	0
6	1	0	1	1	1	1	1	0
7	1	1	1	0	0	0	0	0
8	1	1	1	1	1	1	1	0
9	1	1	1	1	0	1	1	0

예제 6-4 : 회로 구성



아두이노 우노 : 1개, 브레드보드 : 1개, 공통 애노드형 FND : 1개, 저항 : 220Ω 8개, 점퍼선 여러 개

예제 6-4: 스케치

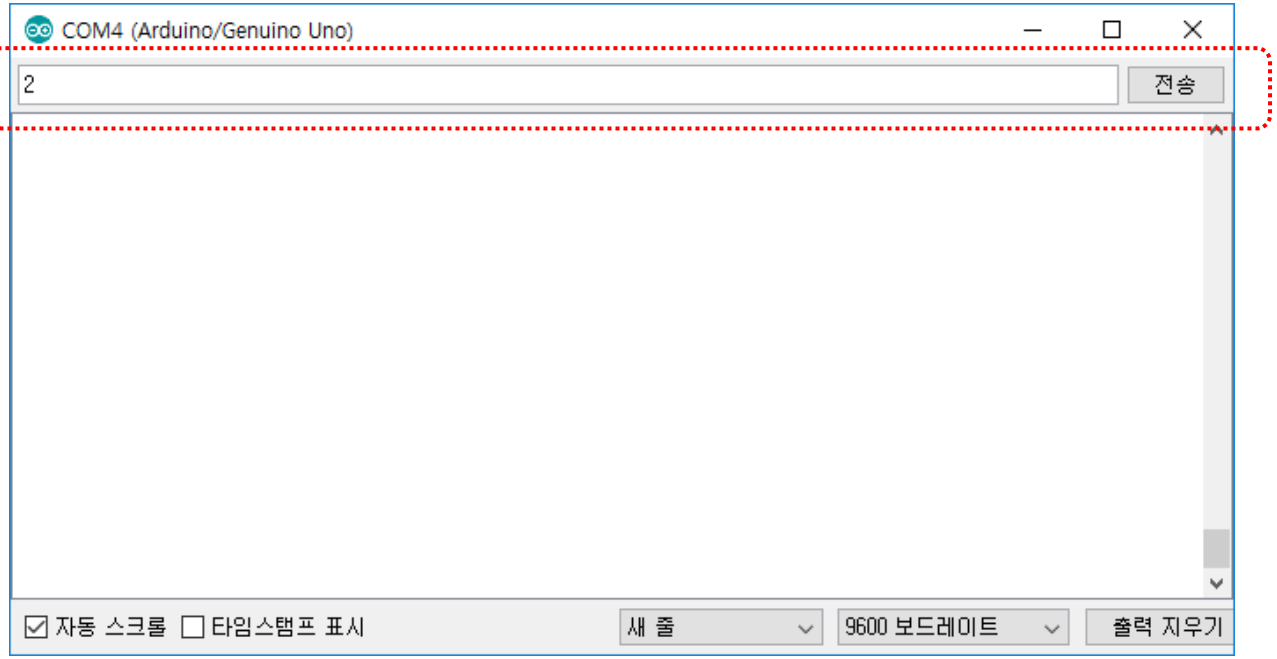
예제 6-4. 7 segment LED 표시 프로그램

```
// 7segment의 A, B, C, D, E, F, G, H핀들을 정의
int ledPins[] = {2, 3, 4, 5, 6, 7, 8, 9};

// 숫자별로 해당하는 LED를 설정한다
// 공통애노드(common anode)형의 경우
// 10개의 숫자에 대해서 8개의 LED에 해당하면 0, 아니면 1

char numLeds[10][8] = {
  {0, 0, 0, 0, 0, 0, 1, 1}, // 0
  {1, 0, 0, 1, 1, 1, 1, 1}, // 1
  {0, 0, 1, 0, 0, 1, 0, 1}, // 2
  {0, 0, 0, 0, 1, 1, 0, 1}, // 3
  {1, 0, 0, 1, 1, 0, 0, 1}, // 4
  {0, 1, 0, 0, 1, 0, 0, 1}, // 5
  {0, 1, 0, 0, 0, 0, 0, 1}, // 6
  {0, 0, 0, 1, 1, 1, 1, 1}, // 7
  {0, 0, 0, 0, 0, 0, 0, 1}, // 8
  {0, 0, 0, 0, 1, 0, 0, 1}  // 9
};
```

```
// 초기화 함수
void setup() {
  int i;
  // 8개의 LED핀들을 모두 출력용으로 설정
  for (i=0; i<8; i++) {
    pinMode(ledPins[i], OUTPUT);
  }
}

void loop() {
  int i, j;
  // 0부터 9까지 숫자 순서대로
  for (i=0; i<10; i++) {
    // 8개의 LED를 상황에 맞추어서 출력
    for(j=0; j<8; j++) {
      digitalWrite(ledPins[j], numLeds[i][j]);
    }
    // 1초 동안 대기
    delay(1000);
  }
}
```

시리얼 통신을 이용한 아두이노 제어

- PC에서 아두이노로 데이터 전송
  - Serial.available() 함수와 Serial.read() 함수 이용
    - Serial.available() : 읽어들일 데이터가 있으면 TRUE 반환
    - Serial.read() : 시리얼 통신을 통해서 데이터 읽기 (1 byte)
  - PC에서는 시리얼 모니터를 통해서 전송



예제 6-5: 스케치

예제 6-4. 7 PC에서 시리얼 통신으로 아두이노 제어

```
void loop() {
  if(Serial.available()) {
    // 전송된 문자를 한 글자 읽어들임
    int data = Serial.read();
    // 숫자가 전송되는 경우만
    if(data >= '0' && data <= '9') {
      // 문자를 숫자로 변경
      int num = data - '0';
      // 8개의 LED를 순서대로 표시
      for(int j=0; j<8; j++) {
        digitalWrite(ledPins[j], numLeds[num][j]);
      }
    }
  }
  delay(10);
}
```

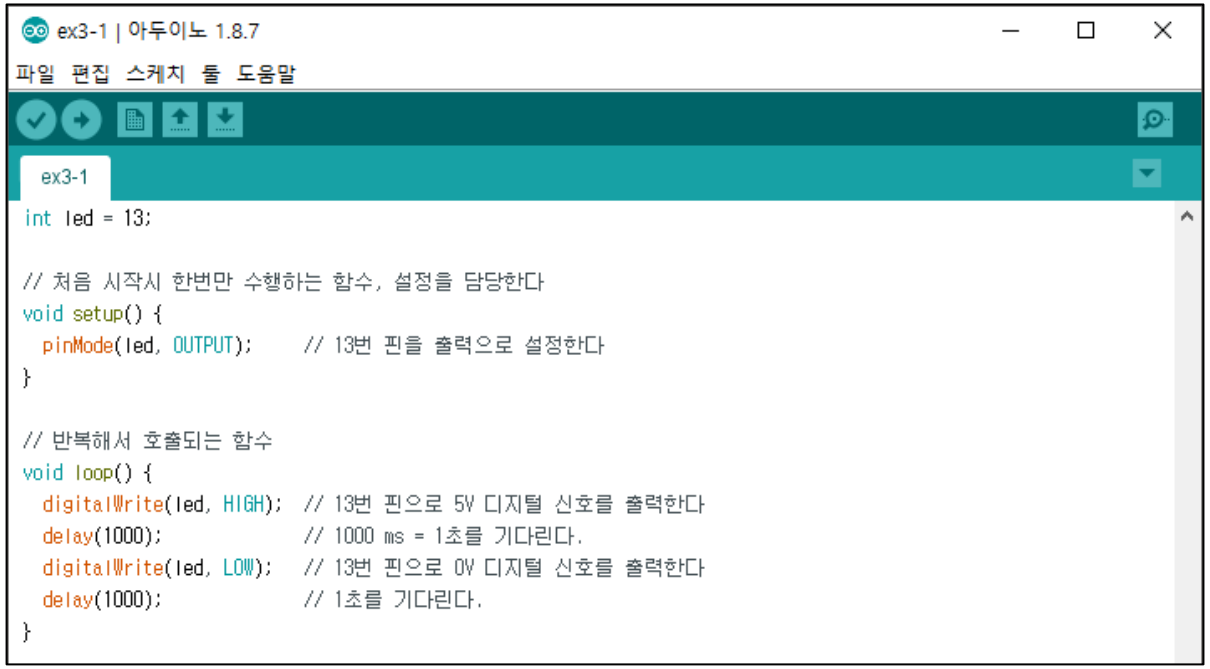
실습 과제

- 실습 과제
  - 앞에서 수행한 실습 과제를 동일하게 수행하고 스케치 파일, 결과 파일을 제출하라.

- 텀 프로젝트
  - Term Project의 주제를 정해서 계획서를 제출하시오. (기한 2주)
  - 계획서의 양식은 자유롭게 하되 [제목, 사용 모듈(센서 등), 간단한 기능] 등 HWP 1장 정도로 작성하여 제출하시오.

A. C++기초 #1

※ 참고 : 2019학년도 시스템응용프로그래밍 I,
교재 : 명품 C++ Programming, 황기태, 생능출판사



A.1 C++ 언어란?

- C++ 언어
  - C언어의 기능
    - 대부분의 기능이 호환됨, 좀 더 엄격하게 바뀐 부분이 일부 있음
  - 객체지향 프로그래밍 (Object Oriented Programming) ⇒ 클래스
    - 캡슐화(encapsulation), 상속성(inheritance), 다형성(polymorphism)
  - 더욱 편리하게 확장된 문법
    - 인라인 함수 : 함수 호출 대신 함수 코드의 삽입
    - 함수 중복 (function overloading) : 동일한 이름의 인수의 개수/타입이 다른 여러 개의 함수
    - 디폴트 매개변수(default parameter) : 미리 지정한 인수값 생략 가능
    - 참조 (reference) : 기존의 변수의 다른 이름 지정
    - 참조에 의한 호출 : 인수로 참조 전달 → 포인터 변수의 단점 해결 가능
    - new/delete 연산자 : 동적 메모리 할당/해제
    - 연산자 재정의 (operator overriding) : 임의의 객체 타입(클래스)에 연산자 사용 가능

A.1 C++ 언어란?

- 기타 C vs. C++
  - printf() 함수 대신 cout 객체 사용
    - `printf("Hello\n"); ⇒ cout << "Hello\n";`
  - 한줄 주석문 추가 (//)
    - // 이후의 문장이 모두 주석 처리됨
  - 함수를 사용하기 전에 반드시 함수의 선언이 있어야 함
    - C에서는 경고, C++에서는 오류 ⇒ 더 엄격해짐
  - 실행문 중간에 변수 정의 가능
    - C언어에서는 블록(보통 함수)의 제일 위부분에만 변수 정의/선언 가능
    - C++에서는 사용하는 위치에서 변수 정의 ⇒ 가독성 향상 및 타이핑 오류 감소
  - STL 과 같은 다양한 라이브러리 및 템플릿(template) 사용 가능

A.2 C++ 확장 문법

- 함수 중복 (다형성)
  - 하나의 이름, 여러 개의 함수, 인수의 개수/타입으로 구분

```
int sum(int a, int b, int c) {
  return a + b + c;
}

double sum(double a, double b) {
  return a + b;
}

int sum(int a, int b) {
  return a + b;
}
```

성공적으로 중복된 sum() 함수들

```
int main(){
  cout << sum(2, 5, 33);

  cout << sum(12.5, 33.6);

  cout << sum(2, 6);
}
```

중복된 sum() 함수 호출.
컴파일러가 구분

A.2 C++ 확장 문법

- 디폴트 매개 변수
  - 인수의 값을 지정하지 않은 경우, 미리 지정된 인수값으로 대체함

```
void star(int a=5); // a의 디폴트 값은 5
```

```
star(); // 매개 변수 a에 디폴트 값 5가 전달됨. star(5);와 동일
star(10); // 매개 변수 a에 10을 넘겨줌
```

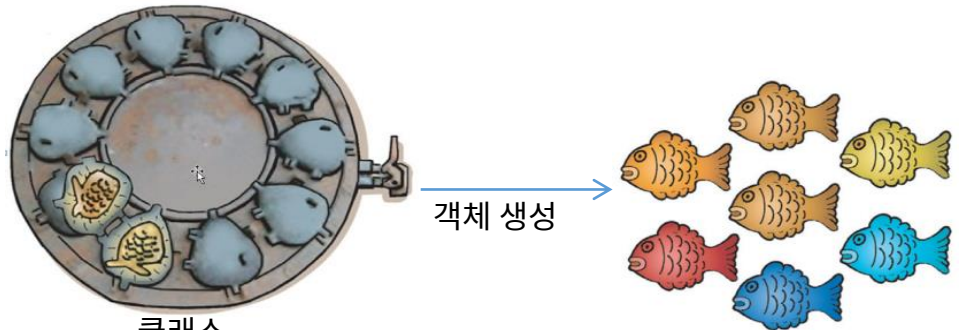
  - 디폴트 매개 변수는 뒤쪽의 인수에만 사용가능함 → 모호성 방지

컴파일 오류

```
void calc(int a, int b=5, int c, int d=0); // 컴파일 오류
void sum(int a=0, int b, int c); // 컴파일 오류

void calc(int a, int b=5, int c=0, int d=0); // 컴파일 성공
```

A.3 클래스

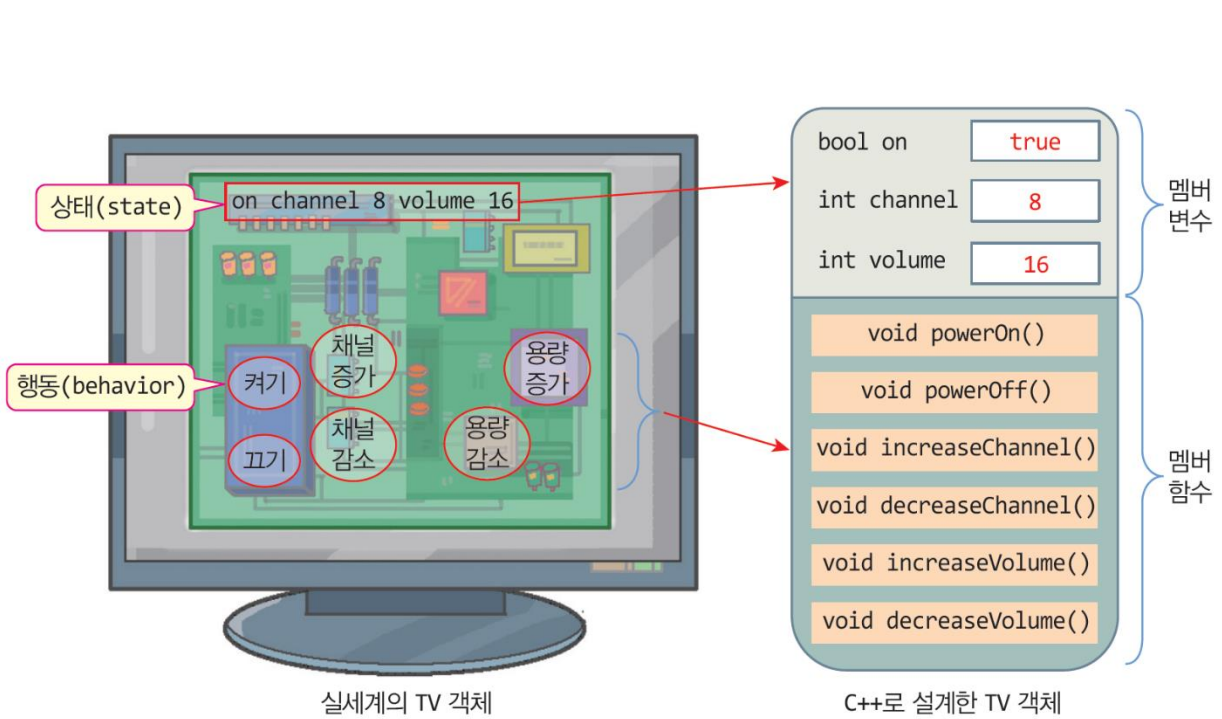
- 클래스 (class)
  - C 언어의 구조체(structure)의 확장
  - 객체를 만들어내기 위한 설계도, 틀 (실제 메모리를 점유하지 않음)
  - 멤버 변수와 멤버 함수로 구성
- 객체 (object, instance, …)
  - 클래스를 이용하여 실제 구현(만들어진) 것 : 변수와 비슷한 개념
  - 내부에 클래스에서 선언한 멤버 변수/함수들이 포함됨
  - 각 객체는 독립된 메모리를 점유함



※ 그림 출처 : 명품 C++ Programming, 황기태, 생능출판사

A.3 클래스 : OOP

- TV의 추상화 : C++ 에서 TV를 다루기 위한 프로그래밍 방법



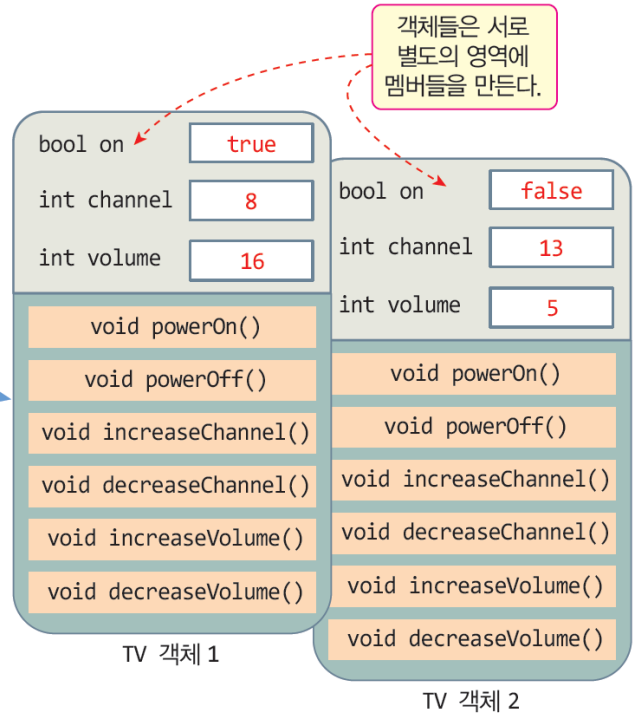
실세계의 TV 객체

C++로 설계한 TV 객체

```
class TV {
   bool on;
   int channel;
   int volume;
public:
   void powerOn() { ... }
   void powerOff() { ... }
   void increaseChannel() { ... }
   void decreaseChannel() { ... }
   void increaseVolume() { ... }
   void decreaseVolume() { ... }
};
```

TV 클래스

객체 생성



TV 객체 1

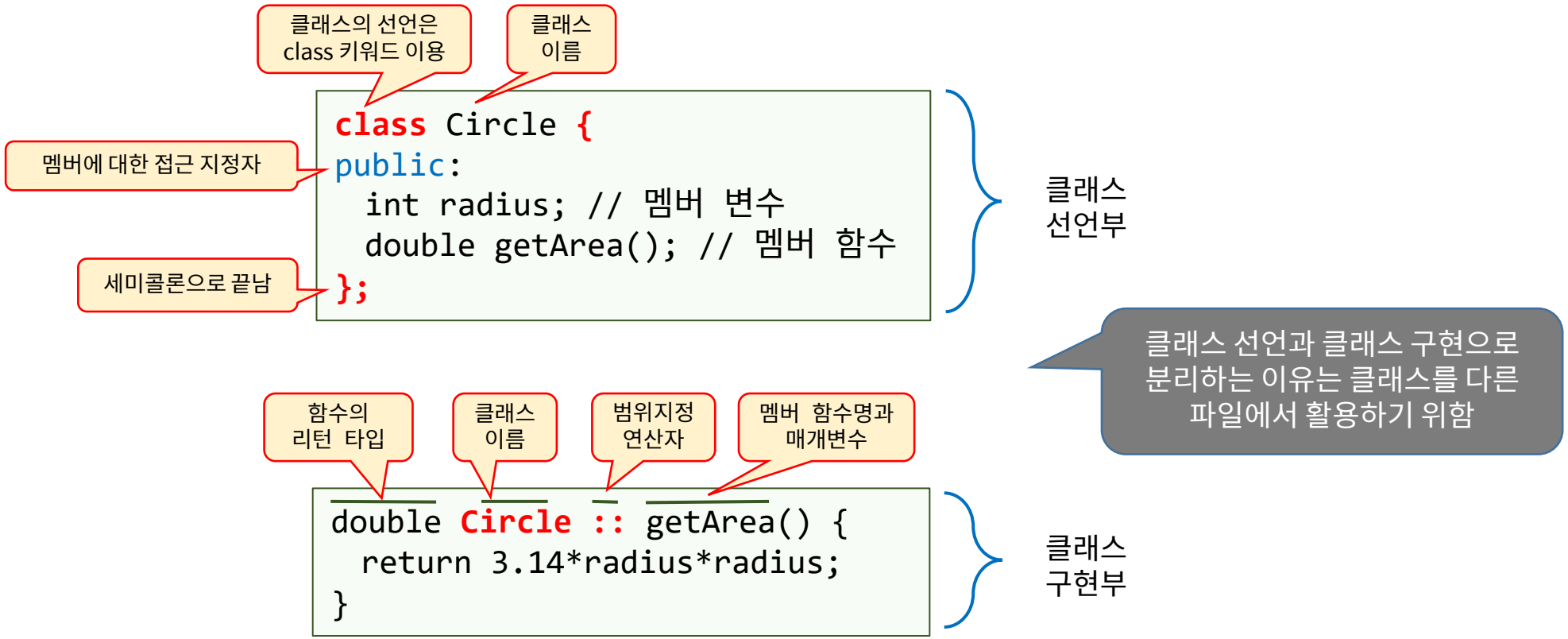
TV 객체 2

(b) C++로 표현한 TV 클래스와 TV 객체들

※ 그림 출처 : 명품 C++ Programming, 황기태, 생능출판사

A.3 클래스 정의

- 클래스의 정의



※ 참고 문헌 : 명품 C++ Programming, 황기태, 생능출판사

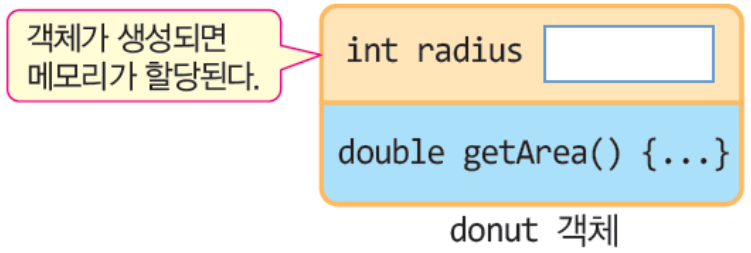
A.3 클래스 객체 정의 및 사용

- 객체의 생성 및 사용
  - 객체의 이름 및 객체 생성

```
Circle  donut; // 이름이 donut 인 Circle 타입의 객체 생성
```

객체의 타입. 클래스 이름

객체 이름



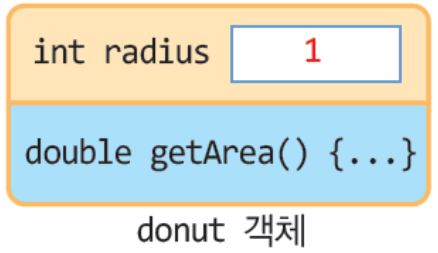
  - 객체의 멤버 변수 접근 : 멤버 연산자 (.) 사용

```
donut.radius = 1; // donut 객체의 radius 멤버 값을 1로 설정
```

객체 이름

객체 이름과 멤버 사이에 . 연산자

멤버 변수



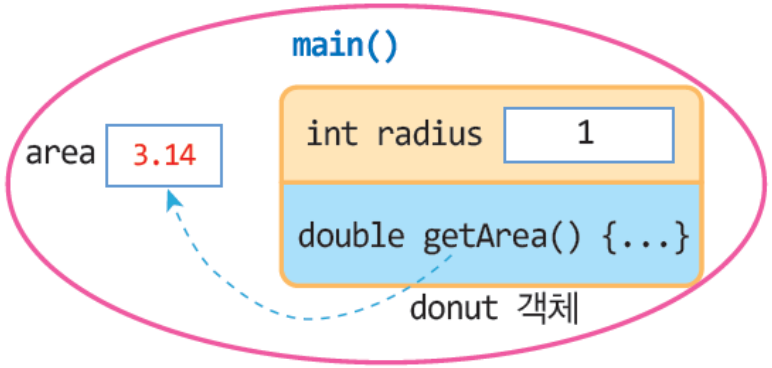
  - 객체의 멤버 함수의 접근 : 멤버 연산자 사용

```
double area = donut.getArea(); //donut 객체의 면적 알아내기
```

객체 이름

객체 이름과 멤버 사이에 . 연산자

멤버 함수 호출



※ 참고문헌 : 명품 C++ Programming, 황기태, 생능출판사

A.3 클래스 객체 정의 및 사용

```
#include <iostream>
using namespace std;

class Circle {
public:
    int radius;
    double getArea();
};

double Circle::getArea() {
    return 3.14*radius*radius;
}

int main() {
    Circle donut;
    donut.radius = 1; // donut 객체의 반지름을 1로 설정
    double area = donut.getArea(); // donut 객체의 면적 알아내기
    cout << "donut 면적은 " << area << endl;

    Circle pizza;
    pizza.radius = 30; // pizza 객체의 반지름을 30으로 설정
    area = pizza.getArea(); // pizza 객체의 면적 알아내기
    cout << "pizza 면적은 " << area << endl;
}
```

객체 donut 생성

donut의 멤버 변수 접근

donut의 멤버 함수 호출

```
donut 면적은 3.14
pizza 면적은 2826
```

A.3 클래스 : 생성자

- 생성자(constructor)
  - 객체가 생성되는 시점에 자동으로 호출되는 함수 : 주로 객체의 초기화 담당

```
#include <iostream>
using namespace std;
class Circle {
public:
    int radius;
    Circle(); // 매개 변수 없는 생성자
    Circle(int r); // 매개 변수 있는 생성자
    double getArea();
};
Circle::Circle() {
    radius = 1;
    cout << "반지름 " << radius << " 원 생성" << endl;
}
Circle::Circle(int r) {
    radius = r;
    cout << "반지름 " << radius << " 원 생성" << endl;
}
double Circle::getArea() {
    return 3.14*radius*radius;
}
```

Circle(); 자동 호출

30

Circle(30); 자동 호출

```
int main() {
    Circle donut; // 매개 변수 없는 생성자 호출
    double area = donut.getArea();
    cout << "donut 면적은 " << area << endl;

    Circle pizza(30); // 매개 변수 있는 생성자 호출
    area = pizza.getArea();
    cout << "pizza 면적은 " << area << endl;
}
```

```
반지름 1 원 생성
donut 면적은 3.14
반지름 30 원 생성
pizza 면적은 2826
```


※ 참고문헌 : 명품 C++ Programming, 황기태, 생능출판사