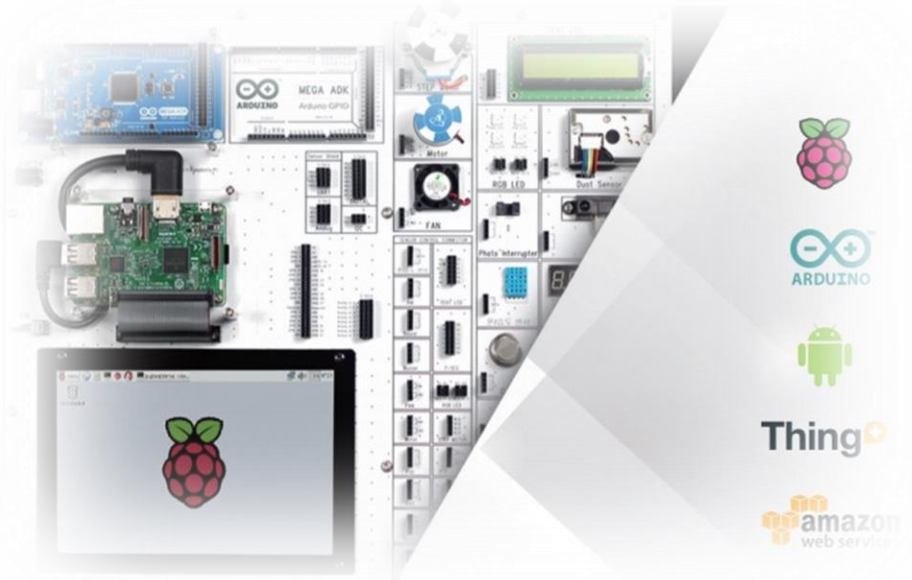


2021년도 2학기 휴먼스마트기기설계

강의자료 #3



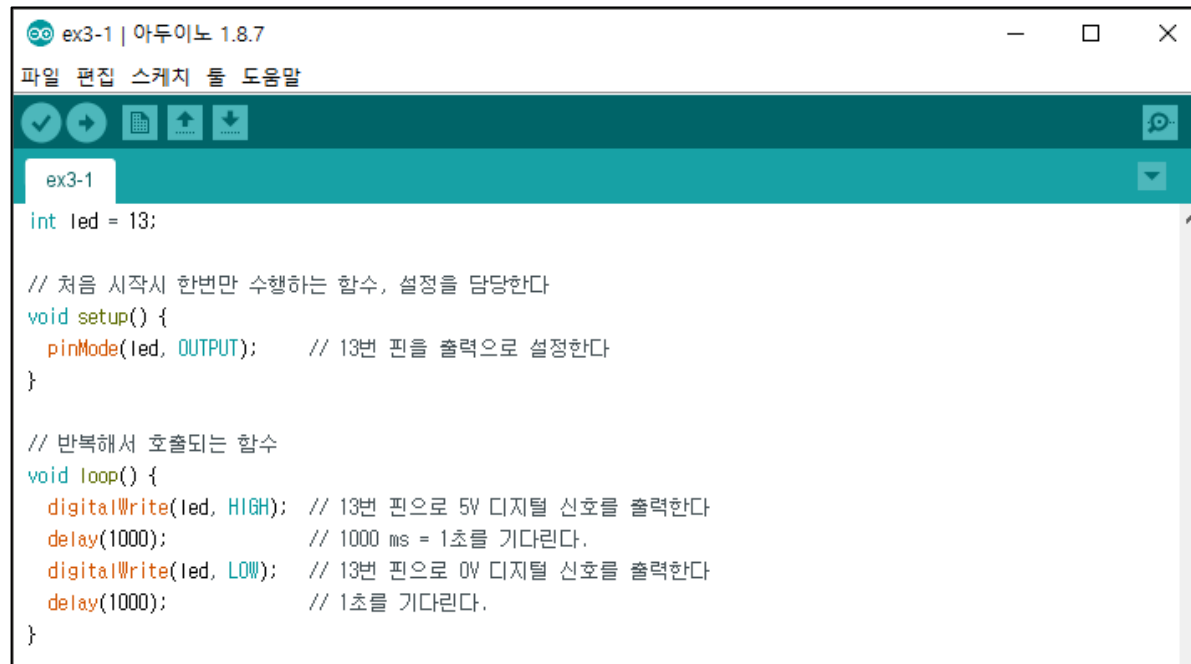
교과목명 : 휴먼스마트기기설계 (01)

담당교수 : 이 수 형

E-mail : soohyong@uu.ac.kr

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

1학기 마이크로프로세서 복습 (Arduino 개발)



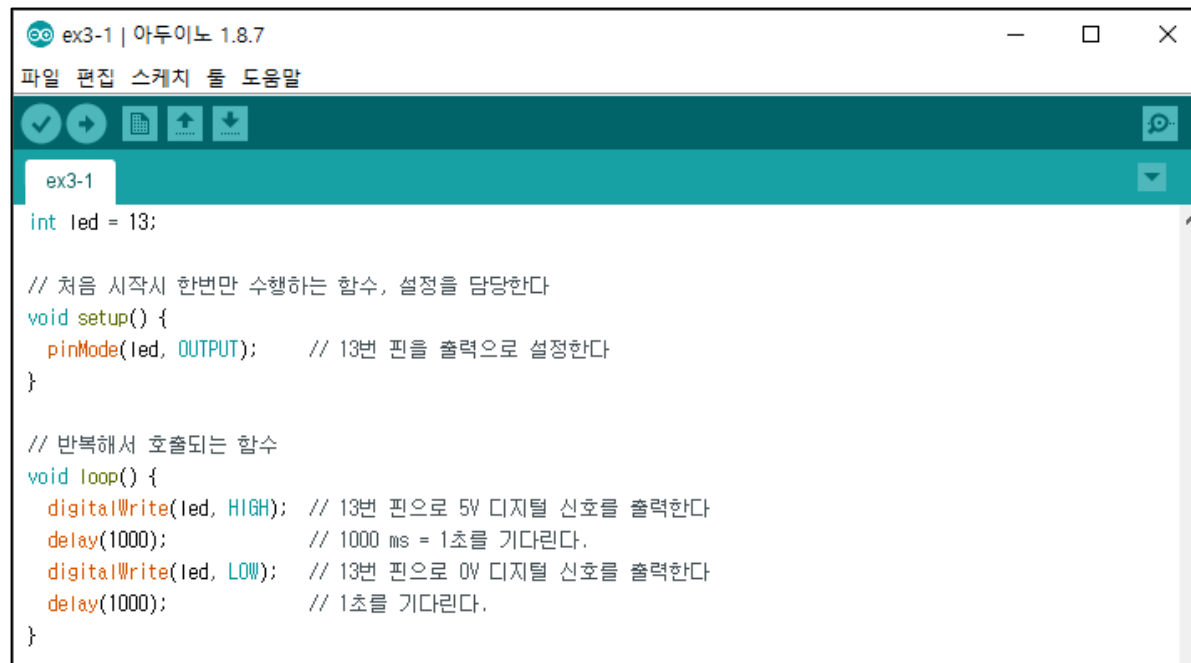
```
ex3-1 | 아두이노 1.8.7
파일 편집 스케치 툴 도움말
ex3-1
int led = 13;

// 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.
}
```

5. 아날로그 입출력의 기초 (1학기 마이크로프로세서 : Arduino)

※ “스마트기기개발을 위한 아두이노” 교재의 5장 참고, NodeMCU 부분은 교재에 없음



```
ex3-1 | 아두이노 1.8.7
파일 편집 스케치 툴 도움말
ex3-1
int led = 13;

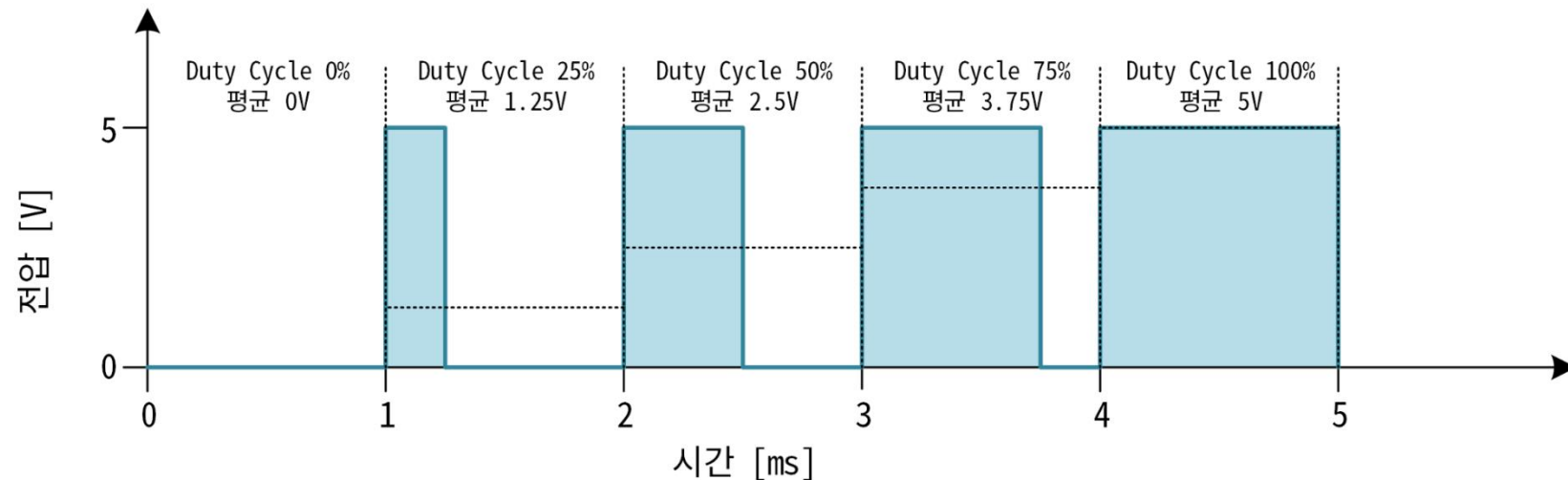
// 처음 시작시 한번만 수행하는 함수, 설정을 담당한다
void setup() {
  pinMode(led, OUTPUT); // 13번 핀을 출력으로 설정한다
}

// 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.
}
```

- 아두이노 우노의 입/출력 기능
 - 14개의 디지털 입/출력 기능
 - 6개의 아날로그 입력 기능
 - 디지털 출력 : HIGH → 5V, LOW → 0V
 - 아날로그 출력 기능은 없음
 - PWM(Pulse Width Modulation) 출력으로 해결
- NodeMCU의 기능
 - 17개의 입/출력 기능 (다른 기능과 공용으로 사용)
 - 1개의 아날로그 입력 기능
 - 디지털 출력 : HIGH → 3.3V, LOW → 0V
 - PWM 출력 : S/W 방식으로 모든 핀에서 사용 가능

• PWM(Pulse Width Modulation)

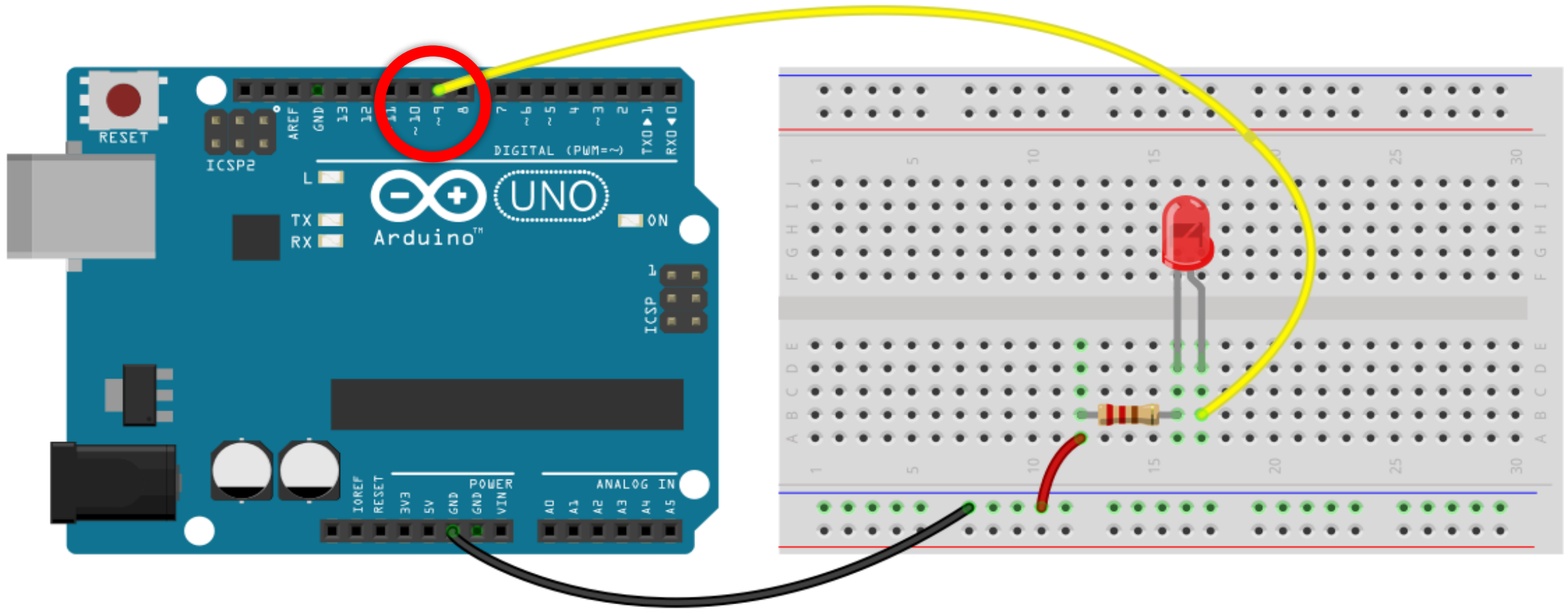
- 사각형 펄스를 주기적으로 발생시키면서 5V, 0V의 발생 시간을 조절하면서 평균 전압을 제어
- 5V, 0V의 발생 시간의 비 : 듀티 사이클 (duty cycle)
- 아두이노 : 6개의 디지털 출력핀이 가능 (~기호로 표시)
- `analogWrite(9, 127);` → 9번 핀으로 50%듀티 사이클 출력



- PWM(Pulse Width Modulation) in NodeMCU

- 사각형 펄스를 주기적으로 발생시키면서 3.3V, 0V의 발생 시간을 조절하면서 평균전압을 제어
- 3.3V, 0V의 발생 시간의 비 : 듀티 사이클 (duty cycle)
- NodeMCU : 모든 디지털 출력핀이 가능하며 S/W 방식으로 처리함
- `analogWrite(9, 512);` → 9번 핀으로 50%듀티 사이클 출력
- Arduino : 0~256, NodeMCU (ESP8266) : 0~1023

- 회로 구성



- 스케치

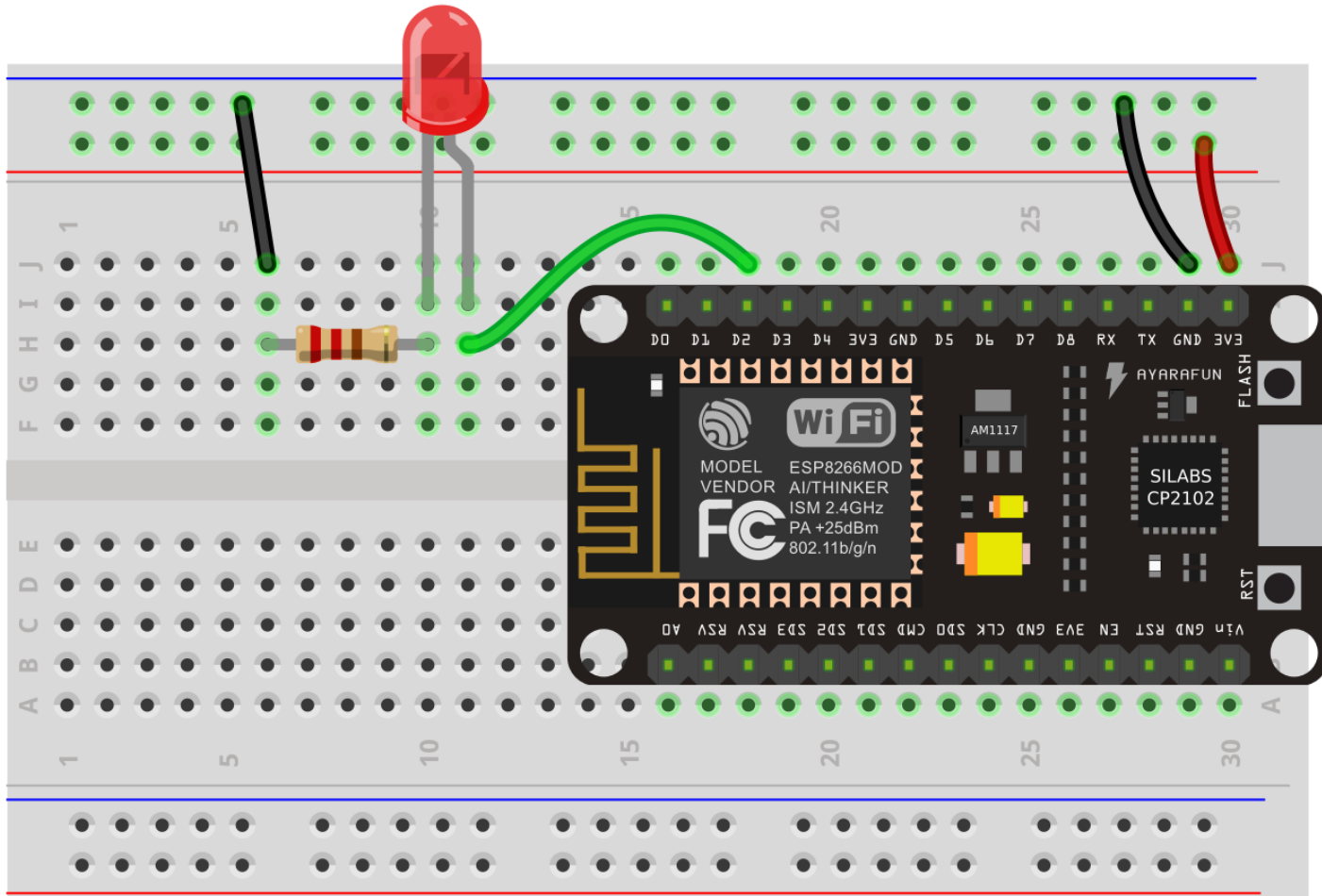
예제 5-1. 아날로그 출력 예제

```
void setup()
{
  // PWM 신호를 사용하기 위하여 9번 핀을 출력으로 설정
  pinMode(9, OUTPUT);
}

void loop()
{
  // 변수 정의 : 밝기를 저장함 (0 ~ 255)
  int br;
  // for문을 사용하여 변수 br의 값을 0에서 255까지 5씩 증가
  for (br = 0; br <= 255; br += 5) {
    analogWrite(9, br);
    delay(30); // 30 마이크로초 대기
  }
  // for문을 사용하여 변수 br의 값을 255에서 0까지 5씩 감소
  for (br = 255; br >= 0; br -= 5) {
    analogWrite(9, br);
    delay(30); // 30 마이크로초 대기
  }
}
```


NodeMCU를 이용한 PWM 출력

- 회로 구성 : 예제 4.1과 동일



- 스케치

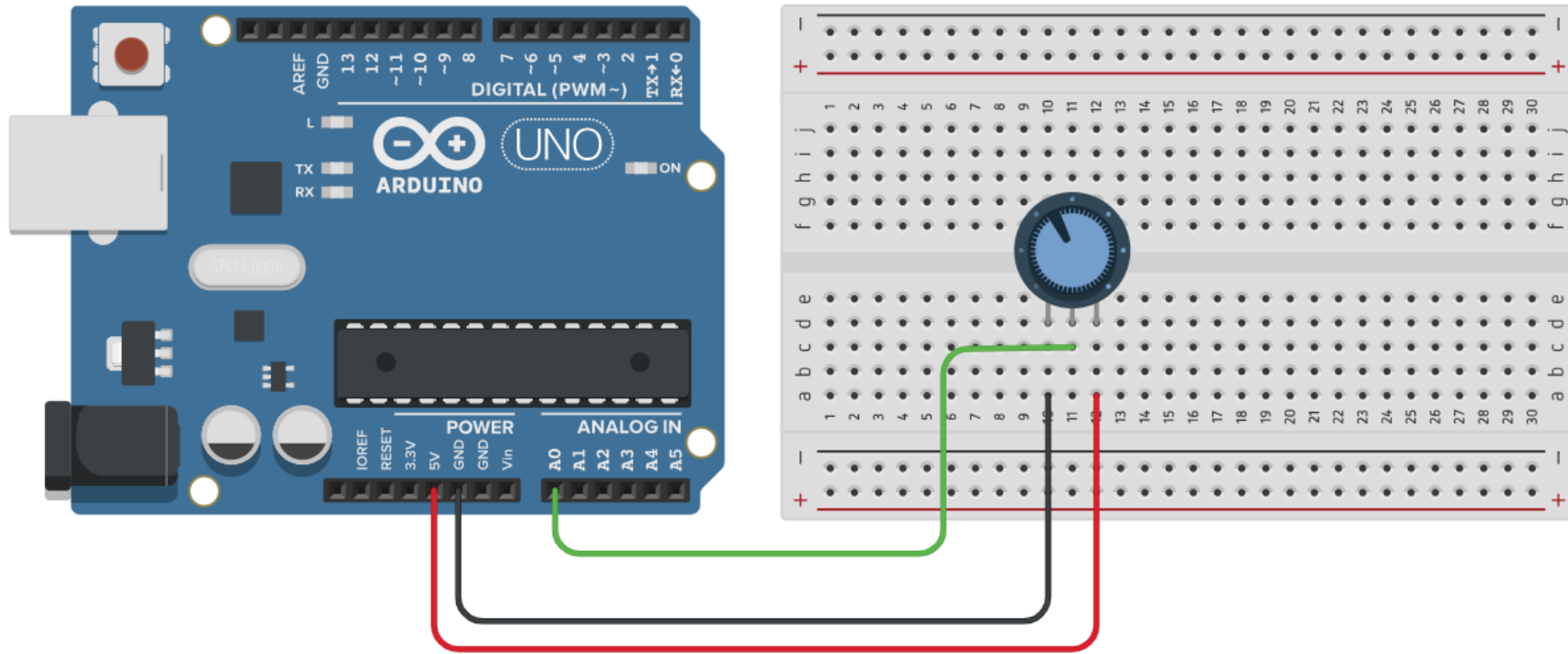
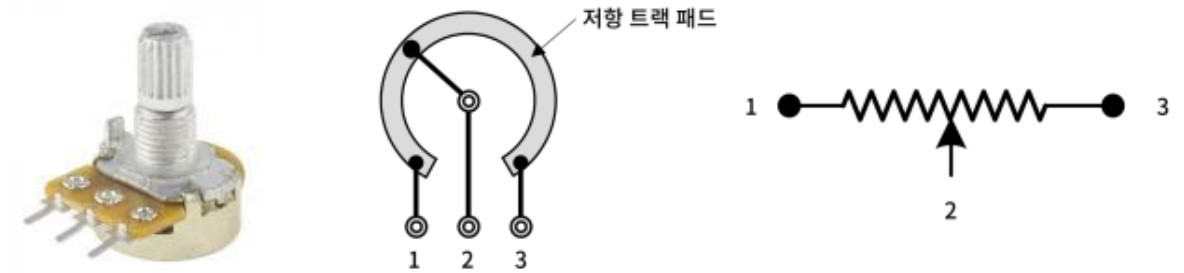
NodeMCU용 아날로그 출력 예제

```
void setup()
{
  // PWM 신호를 사용하기 위하여 D2 핀을 출력으로 설정
  pinMode(D2, OUTPUT);
}

void loop()
{
  // 변수 정의 : 밝기를 저장함 (0 ~ 1023)
  int br;
  // for문을 사용하여 변수 br의 값을 0에서 1023까지 5씩 증가
  for (br = 0; br <= 1023; br += 10) {
    analogWrite(9, br);
    delay(30); // 30 마이크로초 대기
  }
  // for문을 사용하여 변수 br의 값을 1023에서 0까지 5씩 감소
  for (br = 1023; br >= 0; br -= 10) {
    analogWrite(9, br);
    delay(30); // 30 마이크로초 대기
  }
}
```

5.2 포텐쇼미터를 이용한 아날로그 입력

- 포텐쇼미터(potentiometer)
 - 가변저항
- 회로구성



- 스케치

예제 5-1. 포텐쇼미터를 이용한 아날로그 입력

```
// 아날로그 입력 핀(A0~A5) 중에서 A0를 입력으로 사용
int sensor = A0;

void setup()
{
  // 시리얼 통신 초기화
  Serial.begin(9600);
  // 아날로그 센서 포트를 입력으로 사용
  pinMode(sensor, INPUT);
}

void loop()
{
  // 아날로그 센서를 통해서 값을 읽어들이기
  int value = analogRead(sensor);
  // 시리얼 통신을 이용하여 PC로 전송
  Serial.println(value);
  // 100ms 대기
  delay(100);
}
```

- Arduino : 아날로그 센서 읽기 (예: A0)
 - 초기화 : `setup()` 함수에서 `pinMode(A0, INPUT);`
 - 값 읽어들이기 : `int value = analogRead(A0);`
 - 입력 : 0 ~ 5V 전압 측정
 - 값의 범위 : 0 ~ 1023 (아두이노에서는 10 bit ADC를 사용)
 $0000000000_2 (0_{10}) \sim 1111111111_2 (1023_{10})$
 - 입력전압의 계산 : $\text{입력전압} = \frac{\text{아날로그입력값}}{1024} \times 5 \text{ [V]}$
- NodeMCU
 - 0 ~ 3.3V의 전압 측정
 - 입력전압의 계산 : $\text{입력전압} = \frac{\text{아날로그입력값}}{1024} \times 3.3 \text{ [V]}$

- 스케치

예제 5-2. 포텐쇼미터를 이용한 아날로그 입력

```
// 아날로그 입력 핀(A0~A5) 중에서 A0를 입력으로 사용
int sensor = A0;

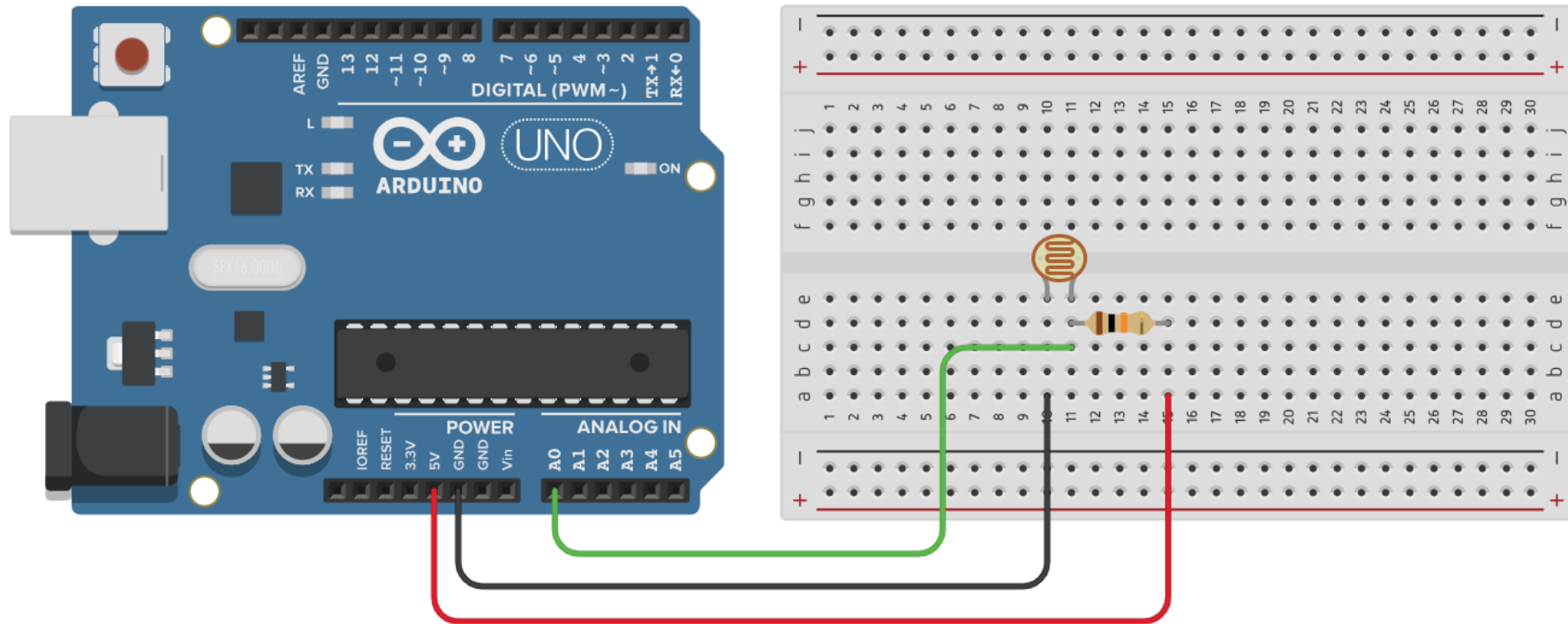
void setup()
{
  // 시리얼 통신 초기화
  Serial.begin(9600);
  // 아날로그 센서 포트를 입력으로 사용
  pinMode(sensor, INPUT);
}

void loop()
{
  // 아날로그 센서를 통해서 값을 읽어들이기
  int value = analogRead(sensor);
  // 시리얼 통신을 이용하여 PC로 전송
  Serial.println(value);
  // 100ms 대기
  delay(100);
}
```

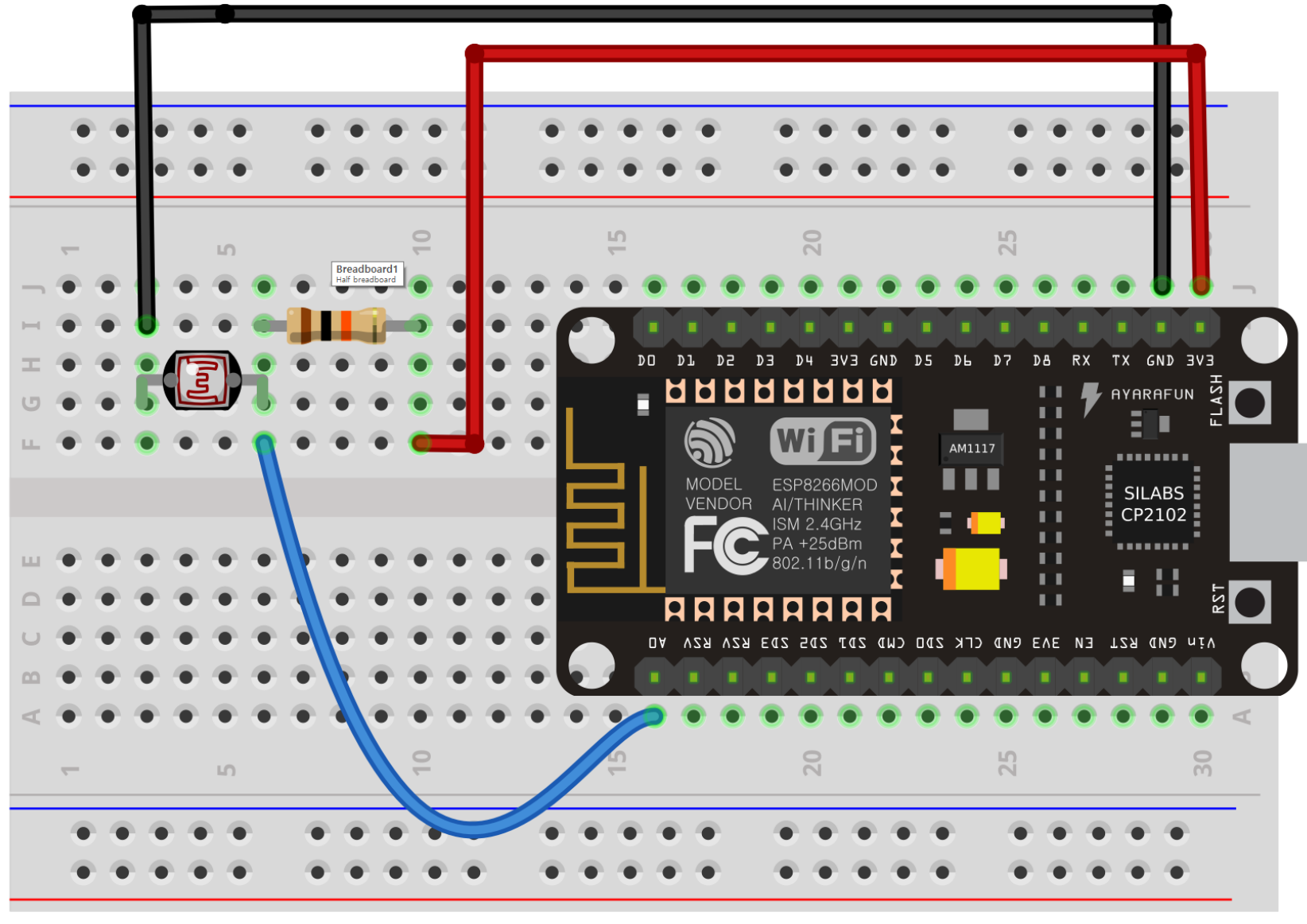
- 센서 : 조도, 온도, 유량, 음향, 가스 등의 물리적인 양을 측정하여 전기 성분으로 변환시켜주는 소자
- CdS (황화카드뮴) : 포토레지스터, 조도센서 → 주위의 밝기에 반응하여 소자의 저항값이 수십 Ω ~수백k Ω 으로 변하는 소자



- 회로 구성 : 스케치는 동일하게 사용

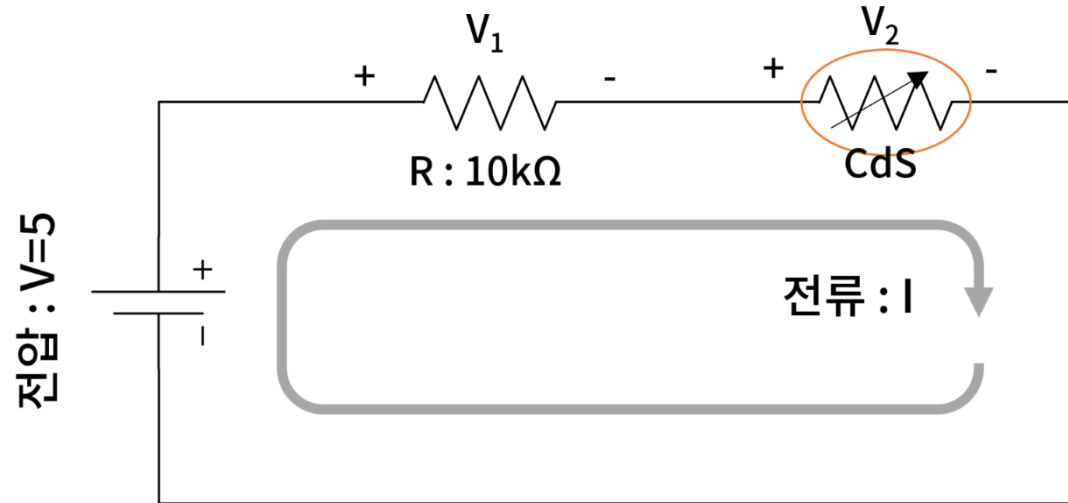


NodeMCU



3

- 회로도



- KVL : $V_1 + V_2 = 5$, CdS의 저항 : $R_C \rightarrow V_1 : V_2 = R : R_C$

$$- V_2 = \frac{R_C}{R} V_1 = \frac{R_C}{R} (5 - V_2) \Rightarrow V_2 = \frac{R_C}{R + R_C} \times 5 \text{ [V]}$$

$$- A0 \text{의 입력값} : A0 = \frac{R_C}{R + R_C} \times 1024$$

- 스케치

예제 5-3. 포텐쇼미터를 이용한 아날로그 입력 (예제 5-2와 동일)

```
// 아날로그 입력 핀(A0~A5) 중에서 A0를 입력으로 사용
int sensor = A0;

void setup()
{
  // 시리얼 통신 초기화
  Serial.begin(9600);
  // 아날로그 센서 포트를 입력으로 사용
  pinMode(sensor, INPUT);
}

void loop()
{
  // 아날로그 센서를 통해서 값을 읽어들이
  int value = analogRead(sensor);
  // 시리얼 통신을 이용하여 PC로 전송
  Serial.println(value);
  // 100ms 대기
  delay(100);
}
```

시리얼 모니터 & 시리얼 플로터

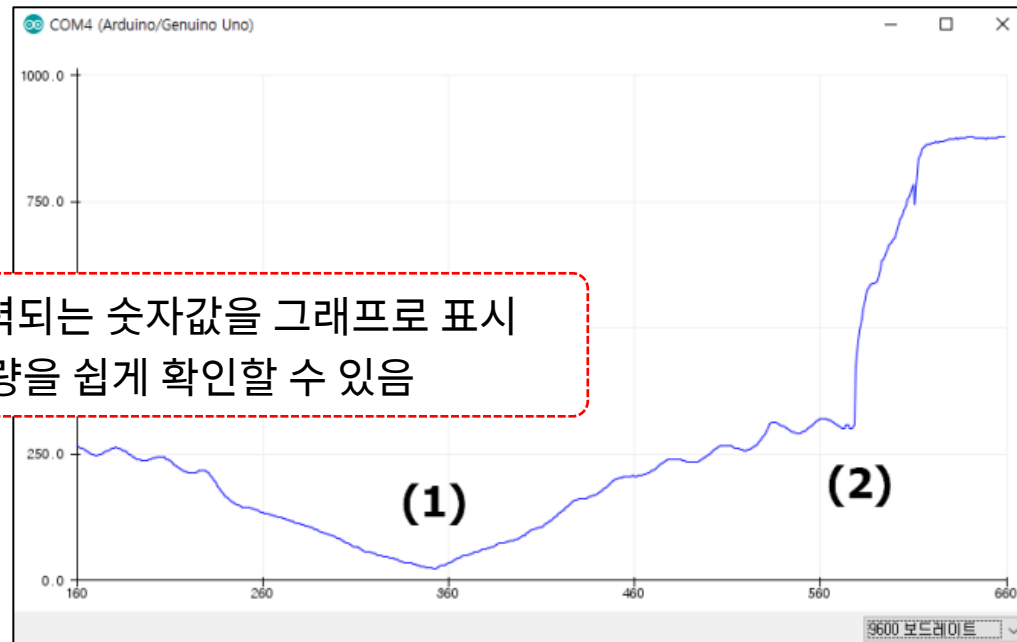
```
196
170
99
30
31
20
20
21
21
24
26
23
304
305
320
660
667
664
661
659
```

ex5-3 | 아두이노 1.8.8

파일 편집 스케치 툴 도움말

- 자동 포맷 Ctrl+T
- 스케치 보관하기
- 인코딩 수정 & 새로 고침
- 라이브러리 관리... Ctrl+Shift+I
- 시리얼 모니터 Ctrl+Shift+M
- 시리얼 플로터 Ctrl+Shift+L**
- WiFi101 / WiFiNINA Firmware Updater

보ards: "Arduino/Genuino Uno" >
ports: "COM4 (Arduino/Genuino Uno)" >
boards: 보드 정보 열기



시리얼 플로터 : 시리얼 통신으로 입력되는 숫자값을 그래프로 표시
⇒ 시간에 따른 센서값의 변화량을 쉽게 확인할 수 있음

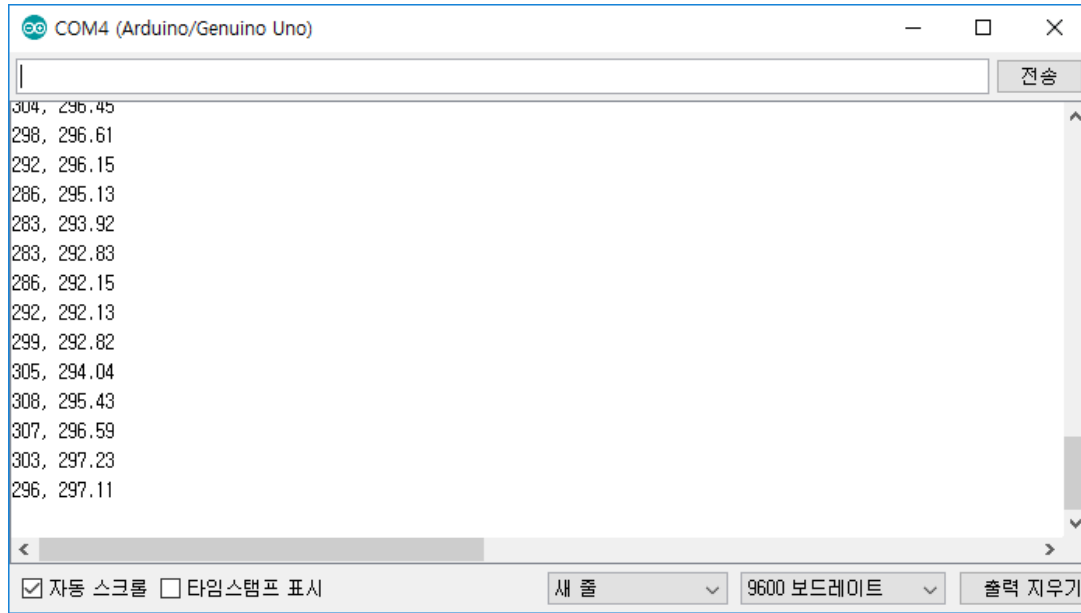
- 지수평균(exponential average) 구하기
 - 센서의 값이 외부 잡음 등의 영향으로 불안정할 때의 보상
 - 과거의 센서값과 현재의 센서값의 평균값 계산
 - 가중평균 : 각 값들의 가중치를 주어 평균 계산
 - 예] `value_ave = value_ave * (1.0 - 0.5) + value * 0.5;`
 - 현재값 가중치 : 0.5

예제 5-4. 조도 센서값의 지수평균 구하기

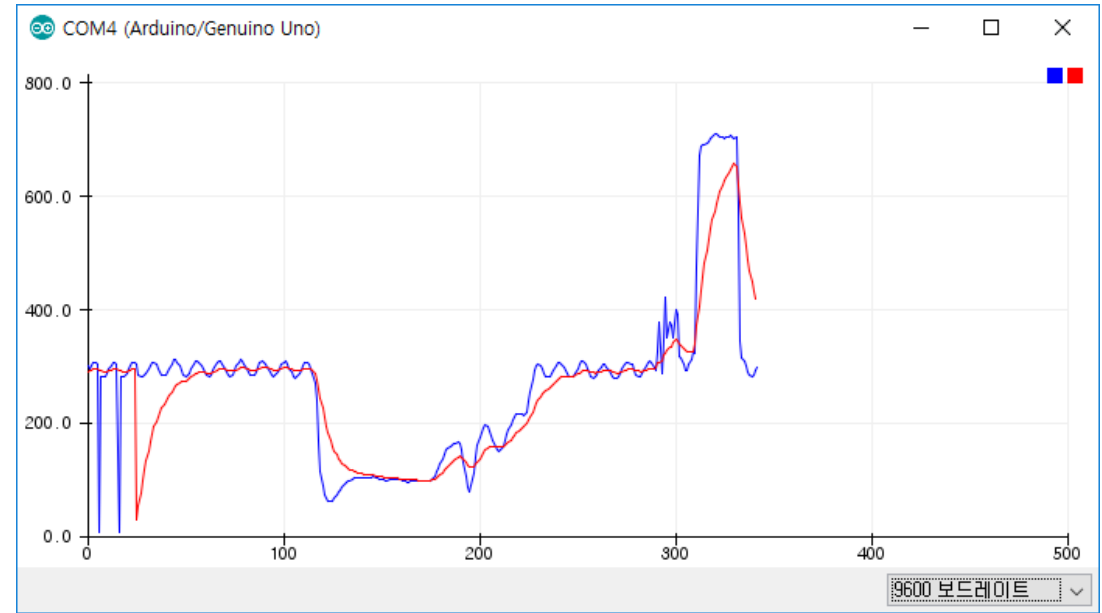
```
int sensor = A0;      // 아날로그 입력 핀(A0~A5) 중에서 A0를 입력으로 사용
double value_ave = 0; // 센서값의 지수평균을 저장하는 변수

void setup()
{
  Serial.begin(9600); // 시리얼 통신 초기화
  pinMode(sensor, INPUT); // 아날로그 센서 포트를 입력으로 사용
}

void loop()
{
  int value = analogRead(sensor); // 아날로그 센서를 통해서 값을 읽어들이м
  value_ave = value_ave * 0.9 + value * 0.1; // 지수평균 구하기, 계수 : 0.1
  // 시리얼 통신을 이용하여 PC로 전송
  Serial.print(value); // 현재 측정값 전송, 줄바꿈 안함
  Serial.print(", "); // 쉼표 및 공백
  Serial.println(value_ave); // 평균값 전송, 줄바꿈 하기
  // 100ms 대기
  delay(100);
}
```



시리얼통신으로 한 줄에 2개의 값을 전송하면
시리얼 플로터에서는 두가지 색으로 나타남



- 파란색 선 : 센서로부터 입력받은 값
- 빨간색 선 : 지수평균값