Analog Input/Output
part 1

부산대학교 공과대학 전기컴퓨터공학부
정보컴퓨터공학전공

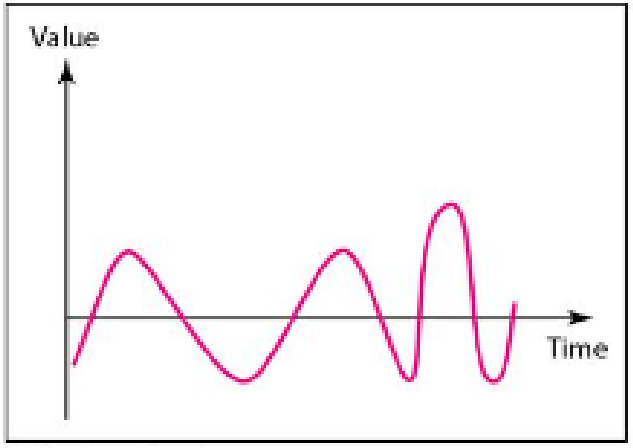
Analog vs. Digital

❖ Analog

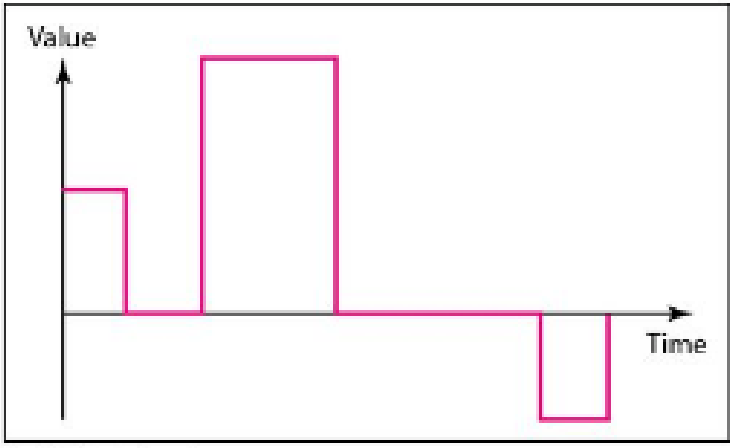
- 영어의 analogous(비슷한) 에서 유래함
- 수학적으로 정확하게 표현하면, 아날로그는 연속적 = continuous

❖ Digital

- 디지털의 어원은 손가락이라는 라틴어 digit에서 유래됨
- 디지털은 불연속적 = 이산적 = discrete



a. Analog signal



b. Digital signal

Analog vs. Digital

❖ 디지털의 장점

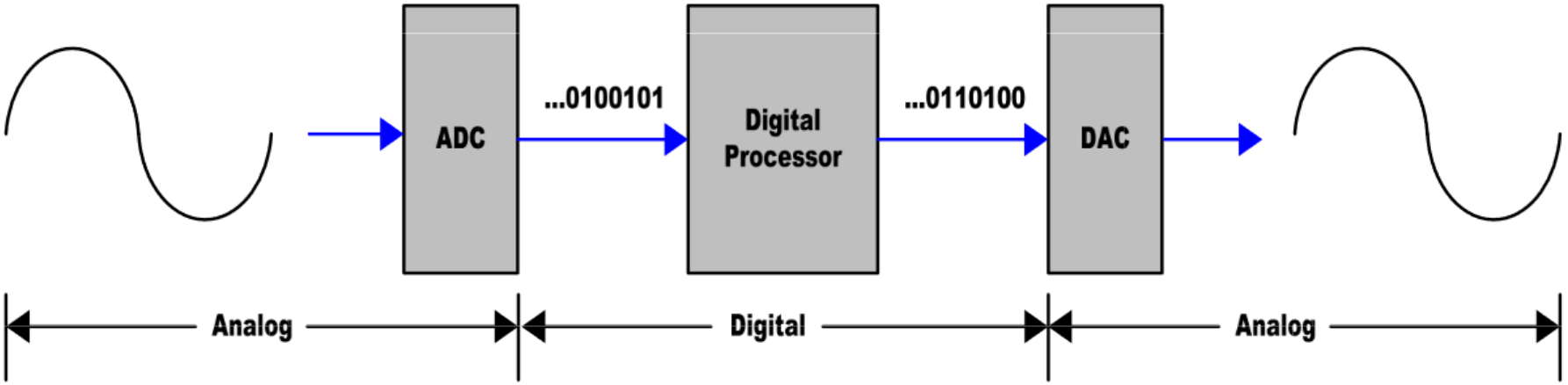
- 노이즈에 강하다
- 온도 특성이 좋다
- 데이터 처리가 편하다.

❖ 아날로그의 장점

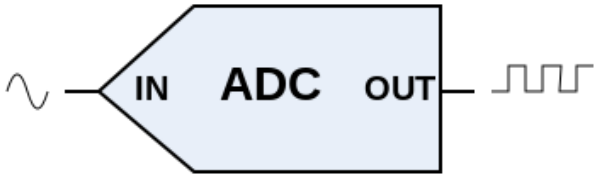
- 원본 데이터에 충실하다.

아날로그-디지털 변환 과정

- ADC (Analog to Digital Converter): 아날로그 데이터를 디지털 데이터로 변환
- DAC (Digital to Analog Converter): 디지털 데이터를 아날로그 데이터로 변환



ADC (Analog to Digital Converter)

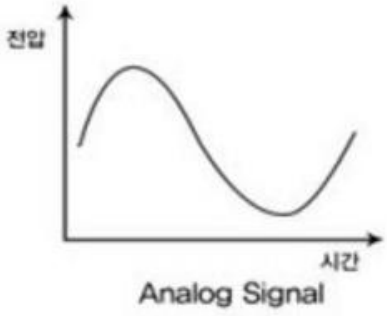


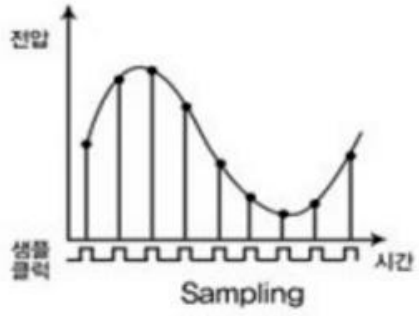
❖ ADC 1단계: 필터링

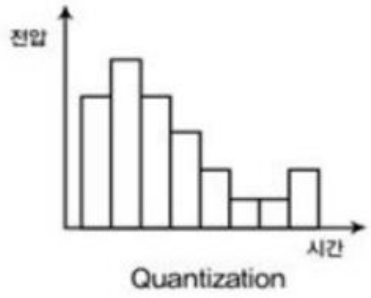
- 본래 신호를 정확히 "표본화(Sampling)"하기 위해 잡음 등의 신호를 차단하는 과정
- Ex) 음성 신호의 경우 원하는 대역폭에 대한 신호만 필터링한다.

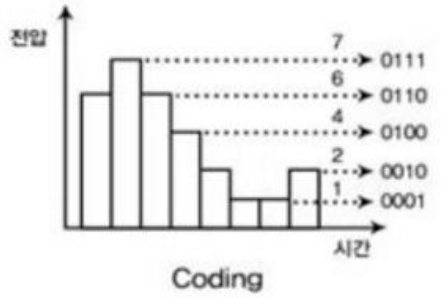
❖ ADC 2단계: 표본화 (Sampling)

- 아날로그 파형을 디지털 형태로 변환하기 위해 표본을 취하는 것을 의미
- 표본화율(Sampling Rate) : 1초 동안에 취한 표본수(디지털화하는 횟수)를 말하며, 단위는 주파수와 같은 Hz를 사용

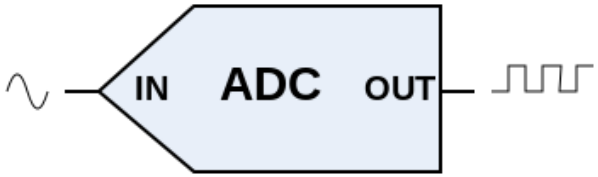






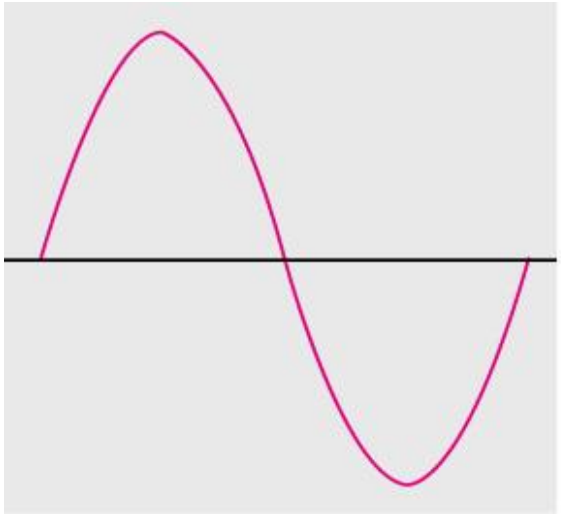


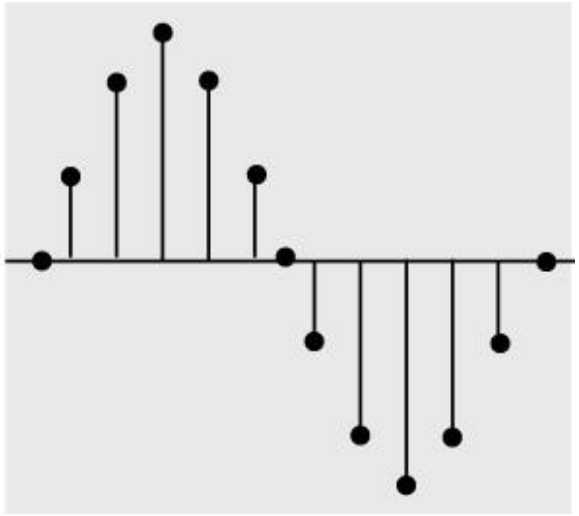
ADC (Analog to Digital Converter)



❖ ADC 2단계: 표본화 (Sampling)

- 아날로그 파형을 디지털 형태로 변환하기 위해 표본을 취하는 것을 의미
- 표본화율(Sampling Rate) : 1초 동안에 취한 표본수(디지털화하는 횟수)를 말하며, 단위는 주파수와 같은 Hz를 사용
  - 0.1초마다 샘플링을 수행하며, sampling rate는 10Hz

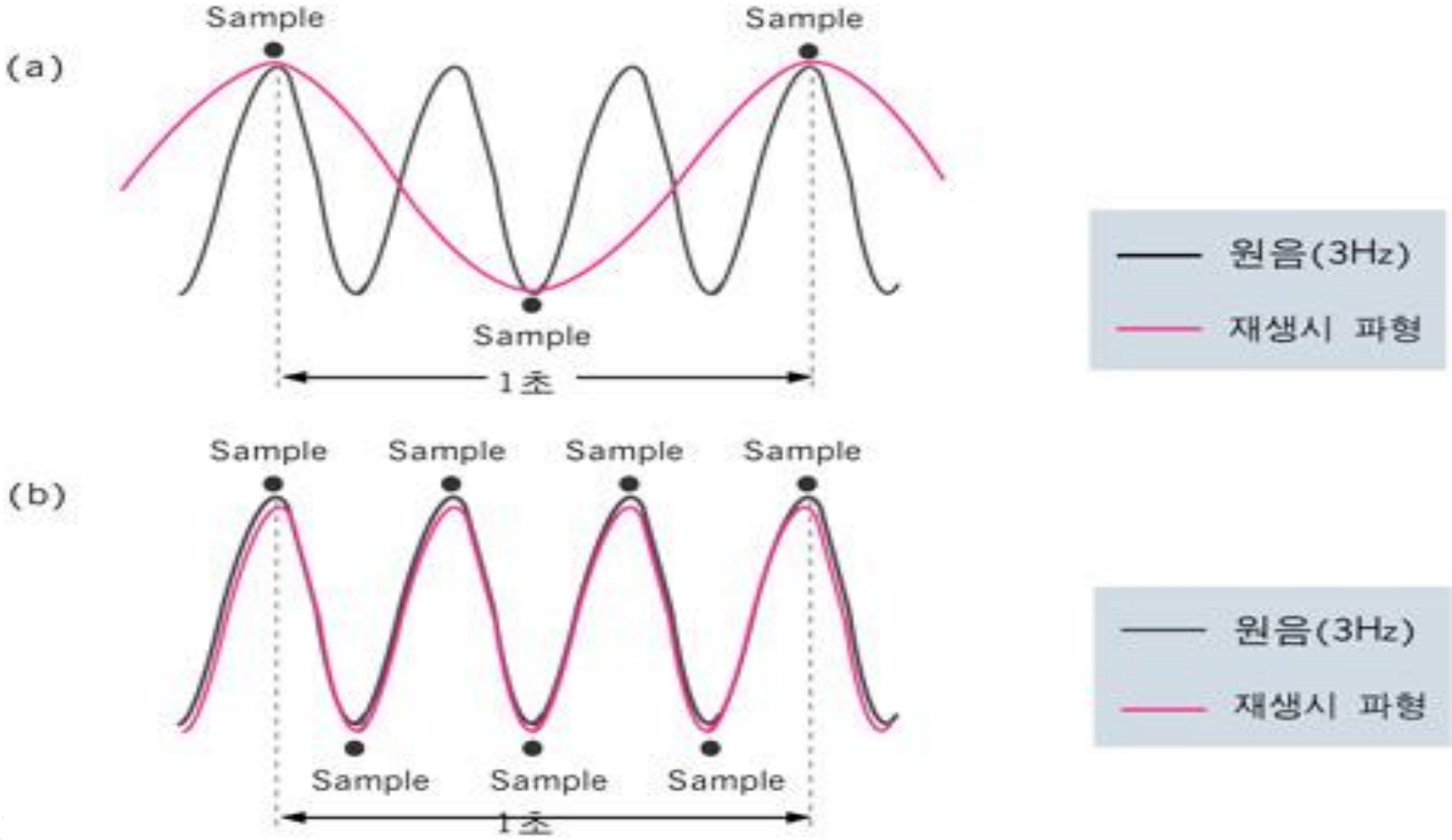




ADC (Analog to Digital Converter)

❖ ADC 2단계: 표본화 (Sampling)

- Sampling rate가 높아지면 본래 신호 특성을 더 잘 유지 할 수 있지만, 데이터양이 많아진다.



ADC (Analog to Digital Converter)

❖ ADC 2단계: 표본화 (Sampling)

- 나이퀴스트 정리(Nyquist theorem)
  - 최대 주파수 성분이 f_{max}인 아날로그 신호는 적어도 $2f_{max}$이상의 sampling rate로 샘플링할 경우 원 신호를 완전히 복원할 수 있다.
- 나이퀴스트 주파수 (Nyquist Rate) = 최소 샘플링 주파수
  - 표본화 정리에 따라 원래의 정보를 재생할 수 있도록 신호가 갖는 최고 주파수(f_{max})의 두 배가 되는 표본화 주파수(f_s)를 말함
  - $f_s = 2f_{max}$

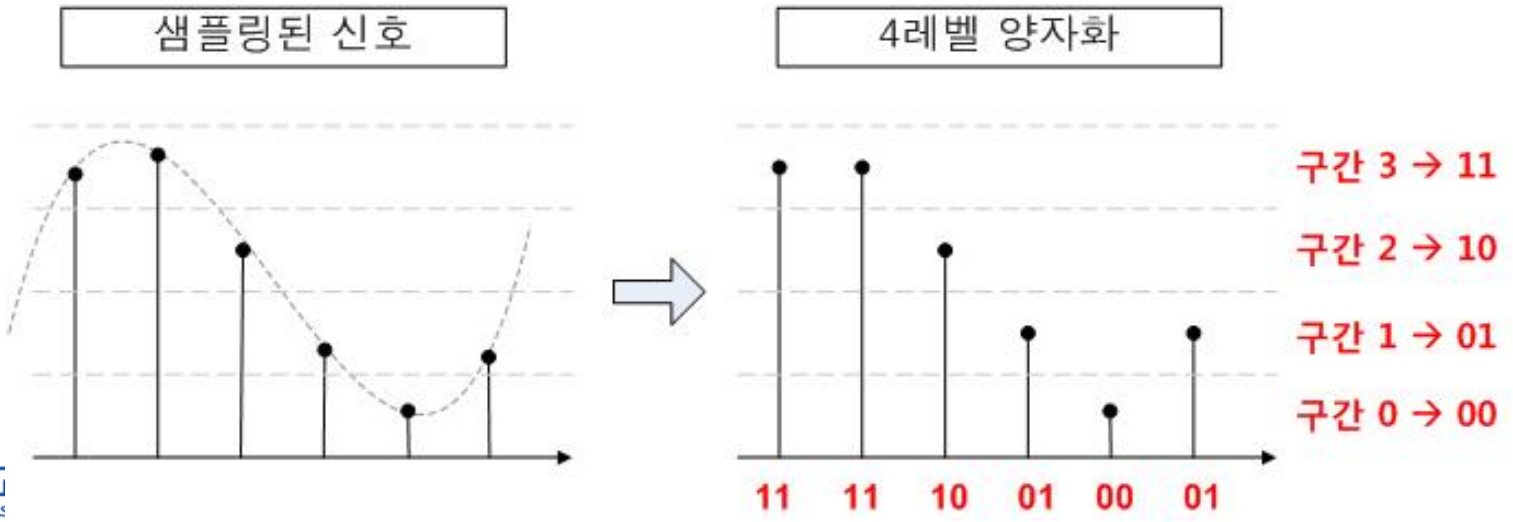
ADC (Analog to Digital Converter)

❖ ADC 3단계: 양자화 (Quantization)

- 표본화에서 얻어진 수치를 대표 값으로 n 개의 레벨로 분해하고, 샘플 값을 근사 시키는 과정
- 디지털 형태로 표현할 때 어느 정도의 정밀도를 가지고 표현할 것인지를 의미.
- 표본화된 각 점에서 값을 표현하기 위해 사용되는 비트 수

❖ ADC 4단계: 부호화 (Coding)

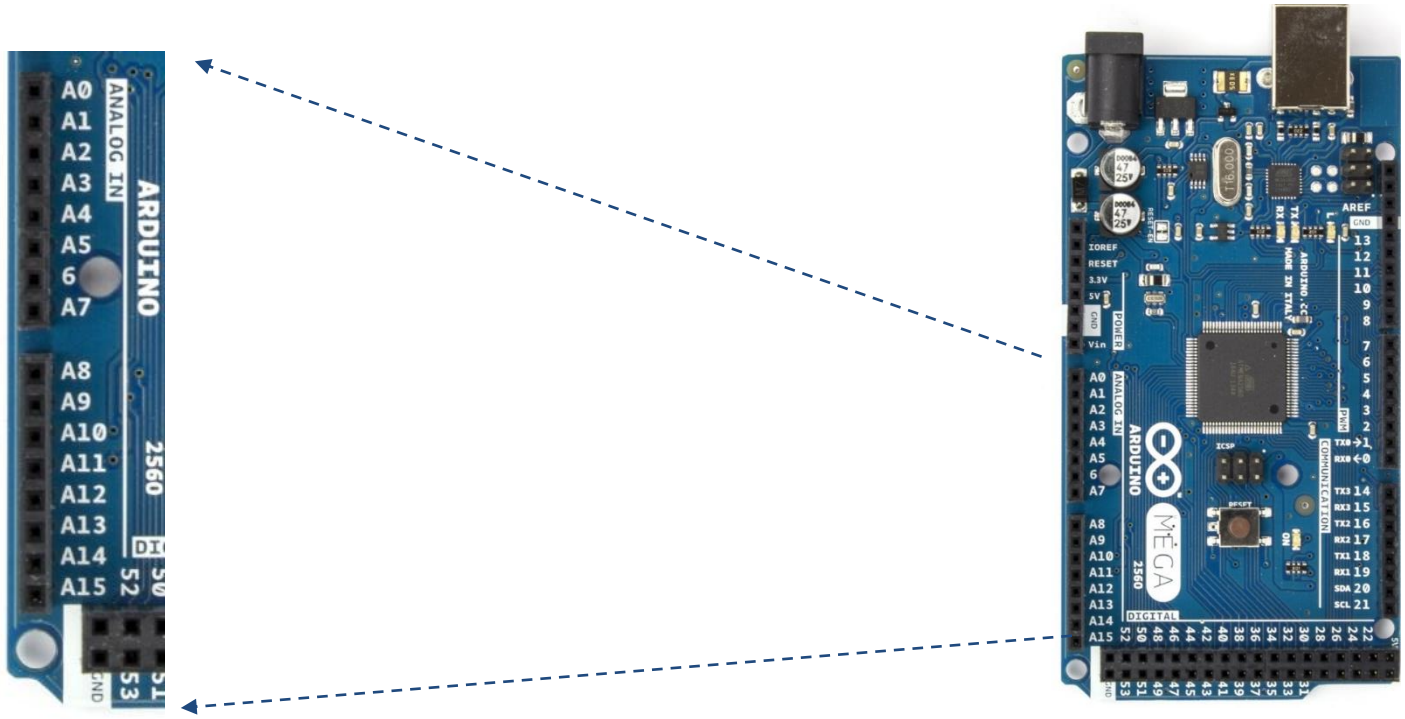
- 양자화된 값을 비트로 변환



아날로그 데이터 처리

❖ 아날로그 데이터 입력

- ATmega2560 마이크로컨트롤러에는 16채널의 10비트 해상도 아날로그-디지털 변환기(ADC)가 포함되어 있음
- 각 채널에는 'A0'에서 'A15'까지의 상수가 할당
  - A0는 디지털 54번, A15는 디지털 69번과 동일함



아날로그 데이터 입력

❖ 16 채널의 ADC

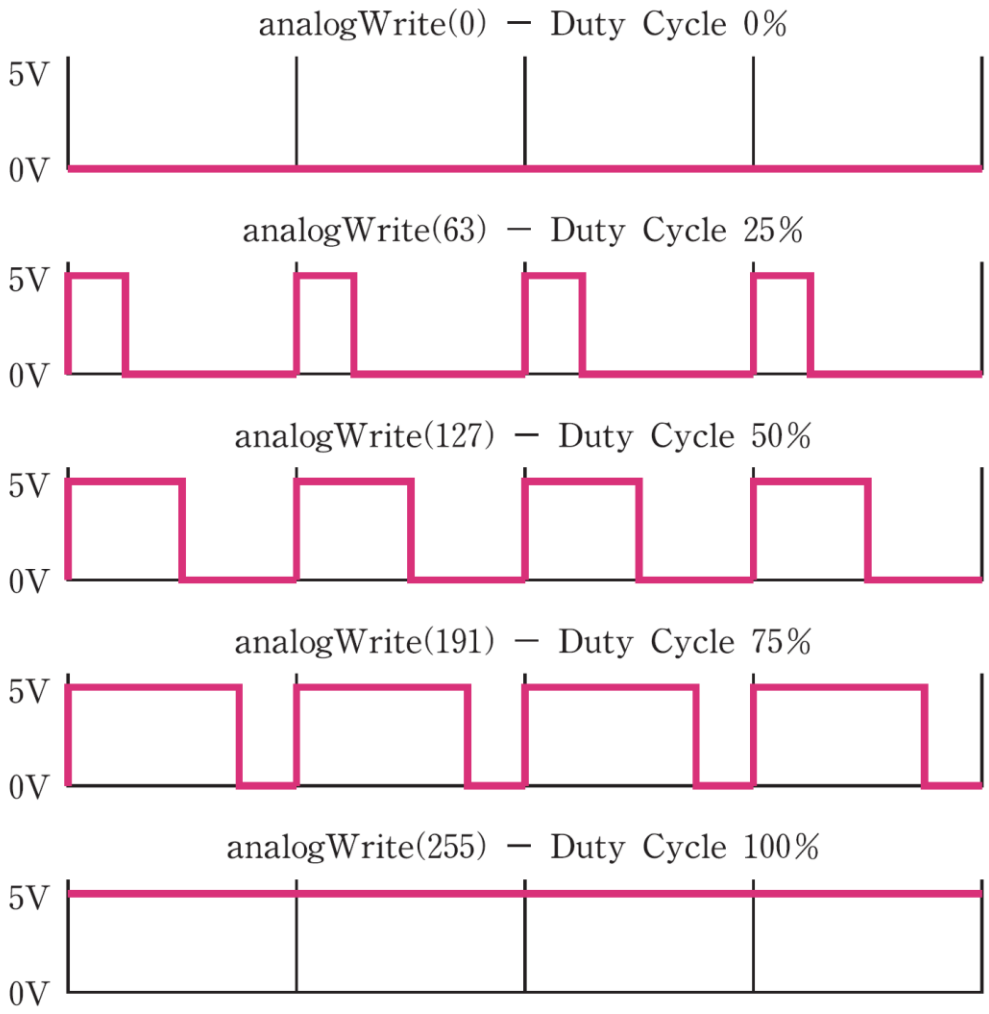
- 하나의 아날로그-디지털 변환기를 공유하므로 동시에 여러 채널 사용은 불가능

❖ 10비트 해상도

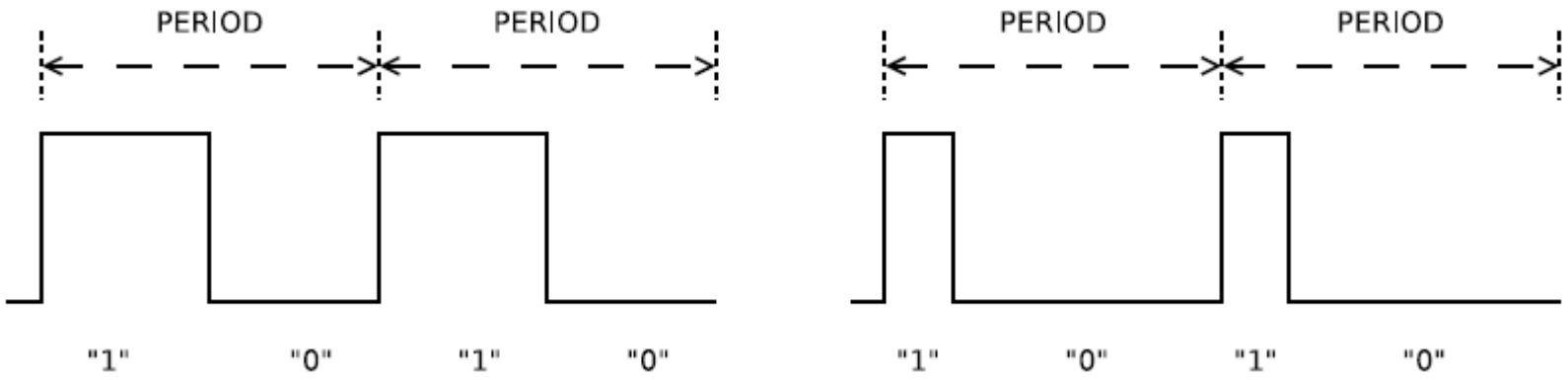
- 0에서 1023 사이의 정수값 반환
- 1023에 해당하는 기준 전압은 ATmega2560의 동작 전압인 5V가 일반적으로 사용됨
  - 기준 전압은 analogReference 함수로 변경 가능
- 5V / 1024 ≅ 4.9mV 전압 차이 인식 가능

Pulse-Width Modulation (PWM)

❖ 아날로그 신호를 디지털화 하여 인코딩 하는 방법

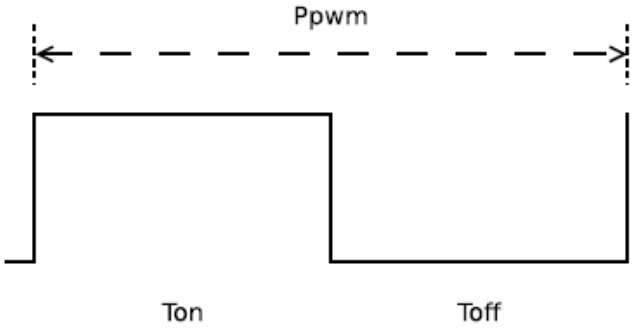


Pulse-Width Modulation (PWM)



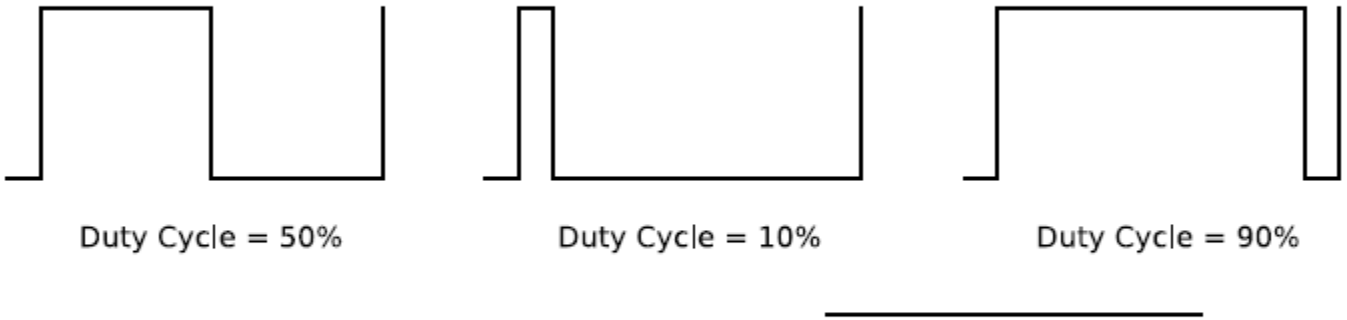
- Pulse Width Modulation (PWM) is a technique of modulation of a digital signal in order to obtain an analog value.
- It based on generating a square wave with a given frequency
- In the square wave, the “0” part and the “1” part have different duration
- The difference, in percentage, is called duty cycle

Pulse-Width Modulation (PWM)

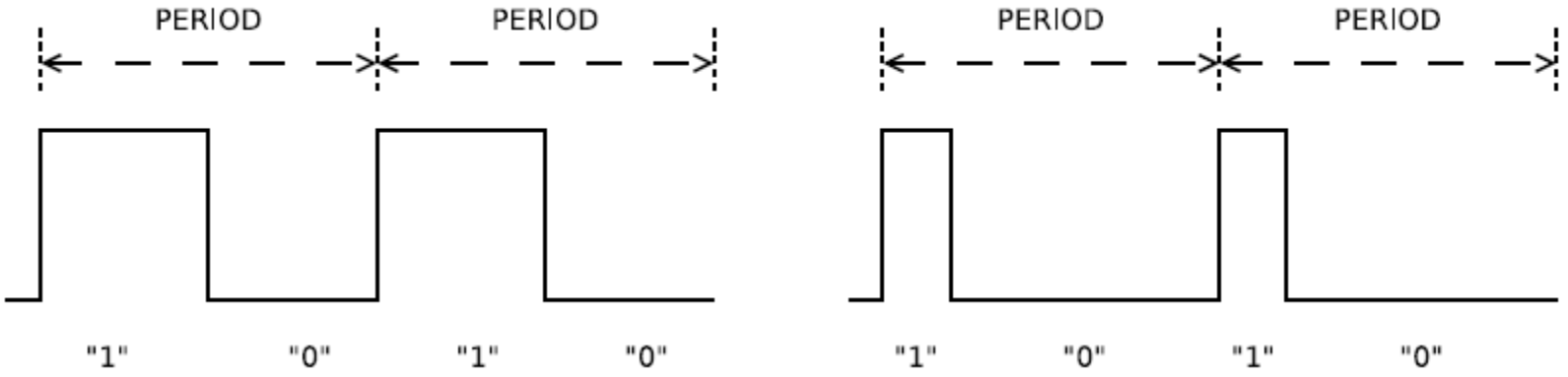


- The Duty Cycle is defined as the percentage of Ton with respect to the total period Ppwm of the signal:

$$Duty\ Cycle = \frac{T_{on}}{P_{pwm}} \cdot 100$$



Pulse-Width Modulation (PWM)



❖ PWM has multiple utilisations:

1. To simply transfer an analog value over a digital line; devices receiving a PWM signal can interpret the "analog value" by measuring the duration of the "1" part with respect to the total frequency;

2. To modulate a typical on/off system; e.g. to change the intensity of a light generated by a lamp, a LED, etc.;

3. To drive power systems without affecting performances; e.g. To drive a DC motor.

아날로그 데이터 처리

❖ 아날로그 데이터 출력

- ATmega2560에서 아날로그 데이터 출력은 불가능
- 펄스 폭 변조(PWM:Pulse Width Modulation) 신호를 통해 아날로그 데이터 출력과 유사한 효과를 얻을 수 있음
  - PWM 신호는 디지털 신호의 일종임
  - PWM 신호 출력 함수가 analogWrite이므로 흔히 아날로그 데이터 출력으로 불림
- 15개의 핀으로 PWM 신호 출력 가능
  - 디지털 2번 ~ 13번
  - 디지털 44번 ~ 46번
- 0과 255 사이의 값을 출력

아날로그 데이터 입출력 함수

int analogRead(uint8_t pin)

- **매개변수**
  - pin : 핀 번호
- **반환값** : 0에서 1023 사이의 정수값

void analogReference(uint8_t type)

- **매개변수**
  - type : DEFAULT, INTERNAL, INTERNAL1V1, INTERNAL2V56, EXTERNAL 중 한 가지
- **반환값** : 없음

void analogWrite(uint8_t pin, int value)

- **매개변수**
  - pin : 핀 번호
  - value : 듀티 사이클(duty cycle). 0(항상 OFF)에서 255(항상 ON) 사이의 값
- **반환값** : 없음

기준 전압 설정 옵션

옵션	설명
DEFAULT	μC의 동작 전압(5V)으로 설정
INTERNAL	내부 기준 전압(1.1V)으로 설정 ATmega2560에서는 사용 불가능
INTERNAL1V1	내부 1.1V로 설정 ATmega2560에서만 사용 가능
INTERNAL2V56	내부 2.56V로 설정 ATmega2560에서만 사용 가능
EXTERNAL	AREF 핀에 인가된 0~5V 사이 전압으로 설정

Lab.5-1

Analog Input with a Potentiometer

❖ Experimental Sequence

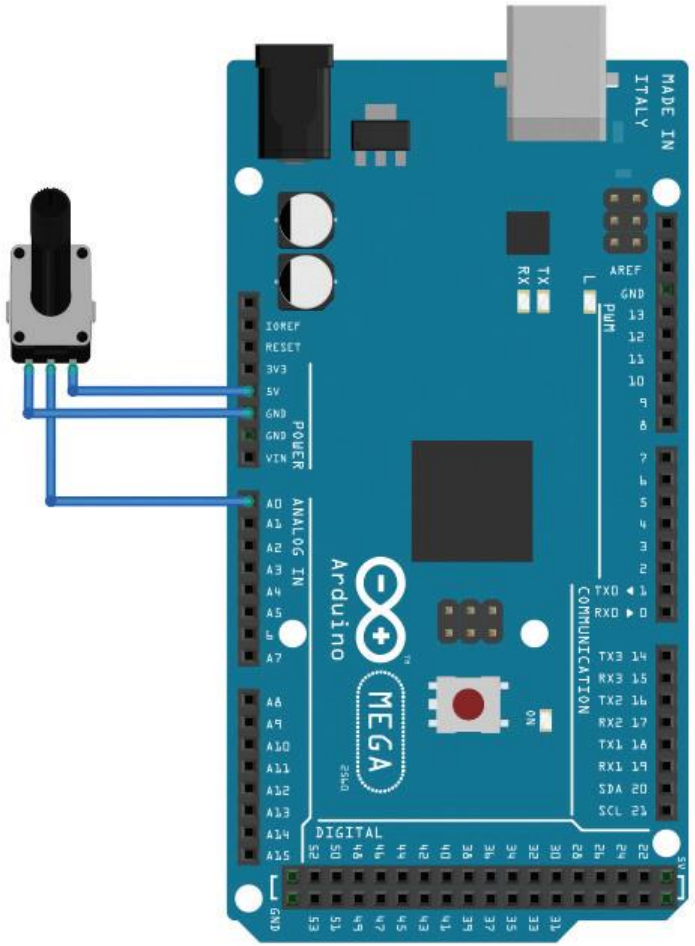
① Connecting a Potentiometer and printing Out the Resistance Value (Sketch 7-1)

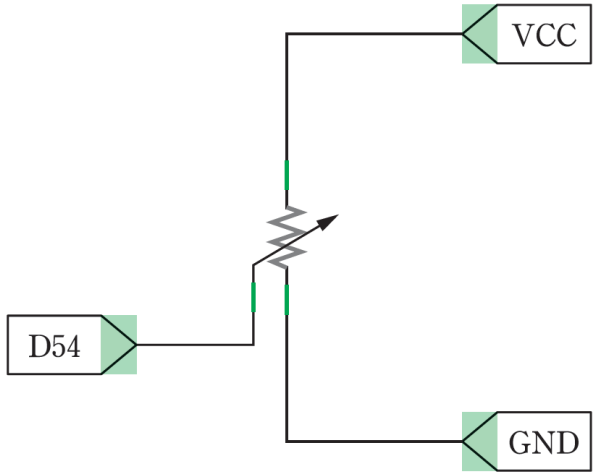
② Lighting 4 LEDs According to the Value of a Potentiometer (Sketch 7-2)

③ Check the Result

① Connecting a Potentiometer

❖ A potentiometer is a simple knob that provides a variable resistance





Lab.5-1

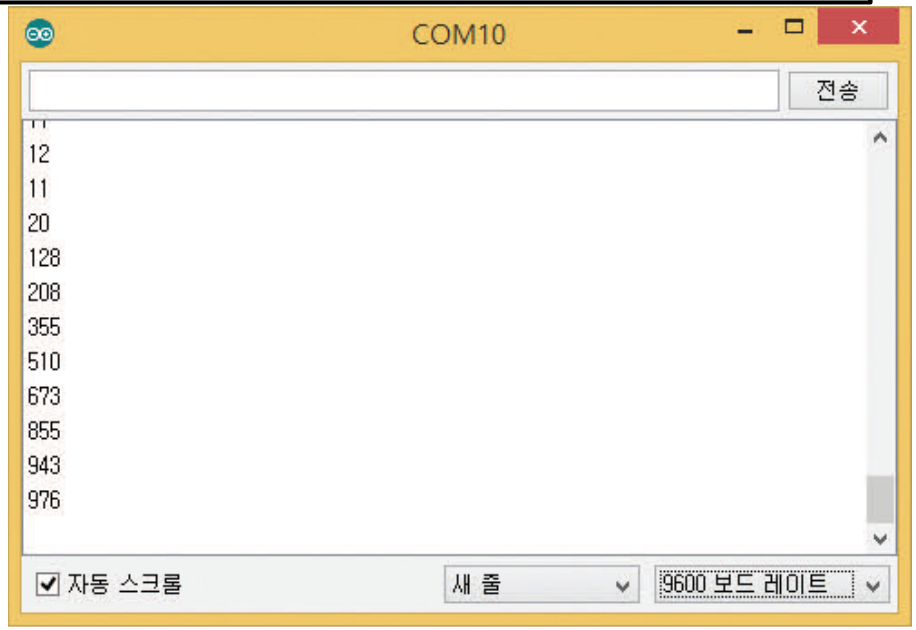
① Printing out the Resistance Value of a Potentiometer

Sketch 7-1

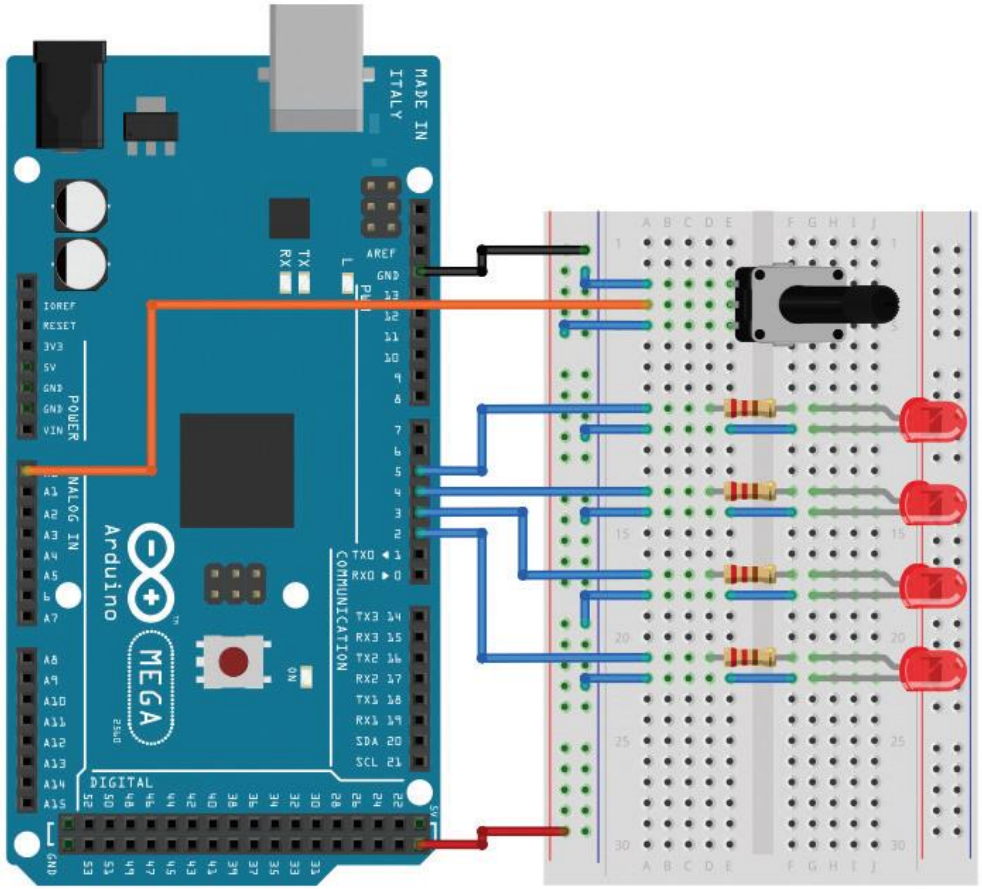
```
int vResistor = A0; // Pin A0 connected to a Potentiometer

void setup() {
  Serial.begin(9600); // Open the serial port at 9600 bps
  pinMode(vResistor, INPUT);
}

void loop() {
 // Read the value form a potentiometer and print it
  Serial.println(analogRead(vResistor));
  delay(1000); // Wait for a second (1000ms)
}
```



② Controlling a LED Lighting using a Potentiometer



Reuse the potentiometer circuit of experiment ①

Control the number of LEDs turned on, according to the resistance value of the potentiometer

② Controlling a LED Lighting using a Potentiometer

Sketch 7-2

```
int vResistor = A0; // Pin A0 connected to a Potentiometer
int pins_LED[] = {2, 3, 4, 5}; // Pins connected to LEDs

void setup() {
  Serial.begin(9600); // Open the serial port at 9600 bps
  pinMode(vResistor, INPUT);
  for (int i = 0; i < 4; i++) {
    pinMode(pins_LED[i], OUTPUT);
    digitalWrite(pins_LED[i], LOW);
  }
}

void loop() {
  int adc = analogRead(vResistor); // Read the value form a potentiometer
  int count_led = (adc >> 8) + 1; // Calculate the number of LEDs turned on
  for (int i = 0; i < 4; i++) { // LEDs output
    if (i < count_led)
      digitalWrite(pins_LED[i], HIGH);
    else
      digitalWrite(pins_LED[i], LOW);
  }
  // Printing out the ADC value and the number of LEDs turned on
  Serial.println(String("ADC : ") + adc + ", LED count : " + count_led);
  delay(1000); // Wait for a second (1000ms)
}
```

Conversion of 10 bit value into 2 bit value through bit shifting

③ Check the Result

- Show your Sketch code
- Show the execution of Sketch 7-2
- Answer the TA's questions and insert the score result to PLMS LAB 5

Lab.5-2

PWM Analog Output

❖ Experimental Sequence

① Adjusting the Brightness of RGB LED (Sketch 7-3)

Keep the circuit for 4 LEDs

② Adjusting the Brightness of LEDs using a Potentiometer (1) (Sketch 7-4)

③ Adjusting the Brightness of LEDs using a Potentiometer (2) (Sketch 7-5)

④ Check the Result

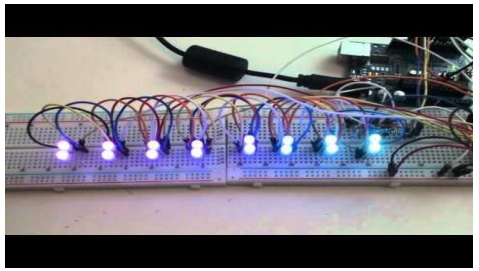
① RGB LED

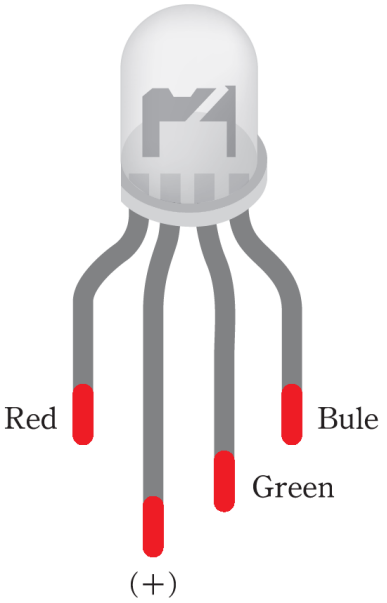
❖ RGB LED

- Consists of one common pin and three R,G,B control pin
- In the case of common anode type, GND output for a control pin make the light turn on (cf. Common cathode)

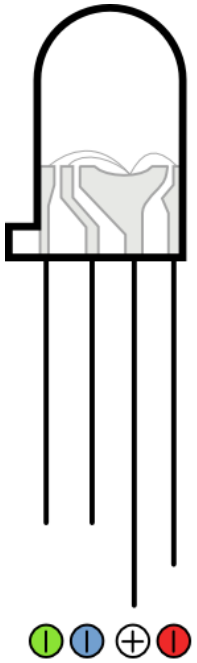
❖ Brightness Control : Common anode type

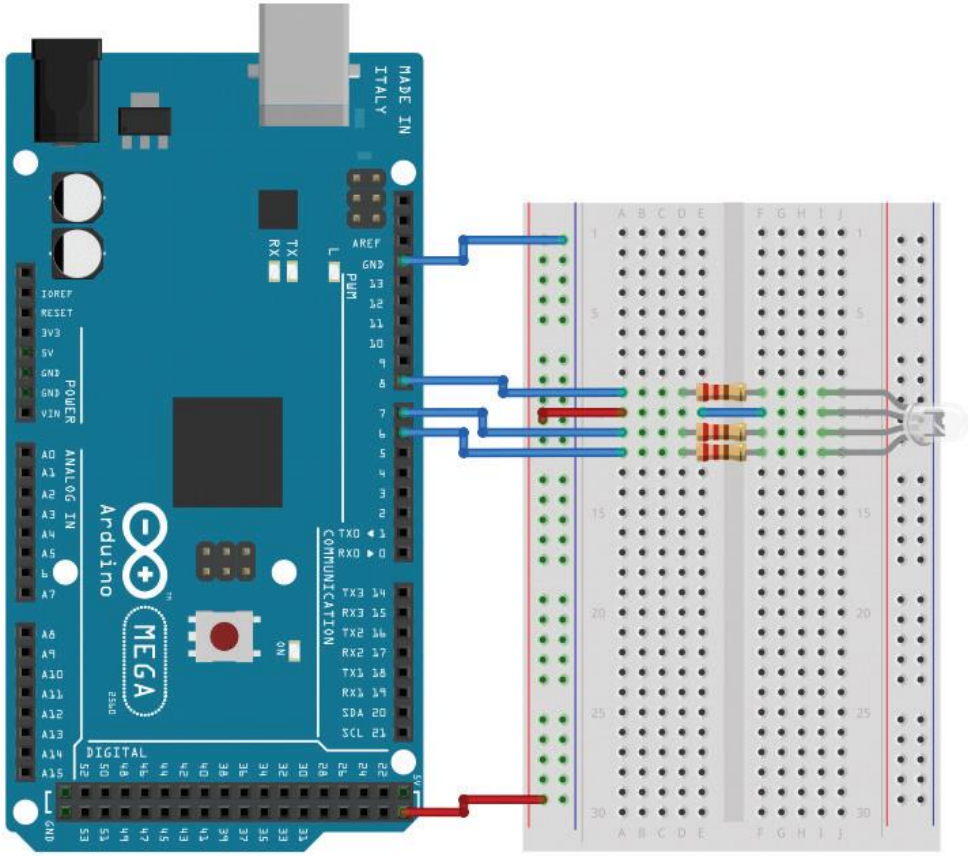
- Maximum brightness : analogWrite(0)
- Minimum brightness : analogWrite(255)

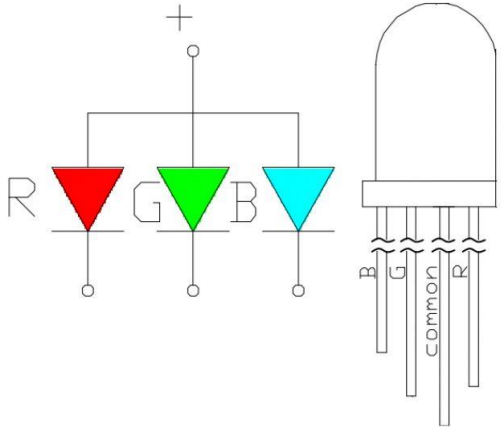


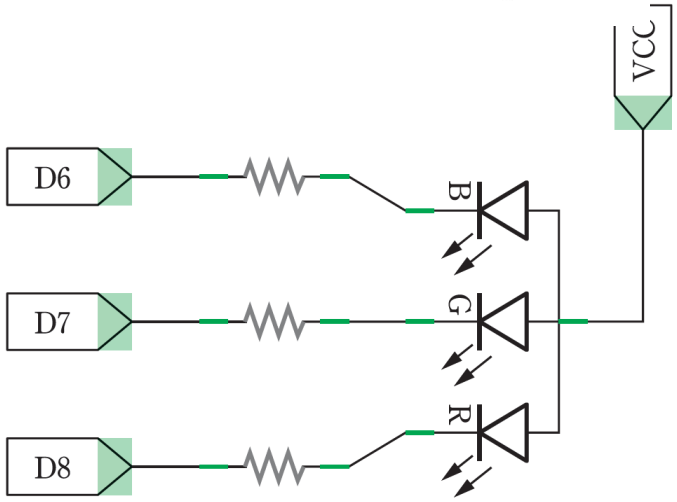


① Connecting an RGB LED









Lab.5-2

① Adjusting the Brightness of an RGB LED

Sketch 7-3

```
// Pins connected to an RGB LED

int RGB_LED[] = {6, 7, 8};
void setup() {
  for (int i = 0; i < 3; i++) {
    pinMode(RGB_LED[i], OUTPUT);
  }
}

void loop() {
  // Adjusting Blue color.
  // Turn off Green and Red color
  digitalWrite(RGB_LED[1], HIGH);
  digitalWrite(RGB_LED[2], HIGH);
  for (int i = 255; i >= 0; i--) {
    analogWrite(RGB_LED[0], i);
    delay(10);
  }

// Adjusting Green color.
// Turn off Blue and Red color
  digitalWrite(RGB_LED[0], HIGH);
  digitalWrite(RGB_LED[2], HIGH);
  for (int i = 255; i >= 0; i--) {
    analogWrite(RGB_LED[1], i);
    delay(10);
  }

// Adjusting Red color.
// Turn off Green and Blue color
  digitalWrite(RGB_LED[0], HIGH);
  digitalWrite(RGB_LED[1], HIGH);
  for (int i = 255; i >= 0; i--) {
    analogWrite(RGB_LED[2], i);
    delay(10);
  }
}
```

Lab.5-2

② Adjusting the Brightness of LEDs using a Potentiometer (1)

Sketch 7-4

```
int pins_LED[] = {2, 3, 4, 5}; // Pins connected to LEDs

void setup() {
  Serial.begin(9600);
  for (int i = 0; i < 4; i++) {
    pinMode(pins_LED[i], OUTPUT);
  }
  pinMode(A0, INPUT); // Pin A0 connected to a Potentiometer
}

void loop() {
  int ADC_value = analogRead(A0); // ADC value
  int PWM_value = ADC_value >> 2; // PWM value
  Serial.print(String("ADC value : ") + ADC_value);
  Serial.println(String(", PWM value : ") + PWM_value);
  for (int i = 0; i < 4; i++) { // Adjusting the brightness of LEDs
    analogWrite(pins_LED[i], PWM_value);
  }
  delay(1000);
}
```

Lab.5-2

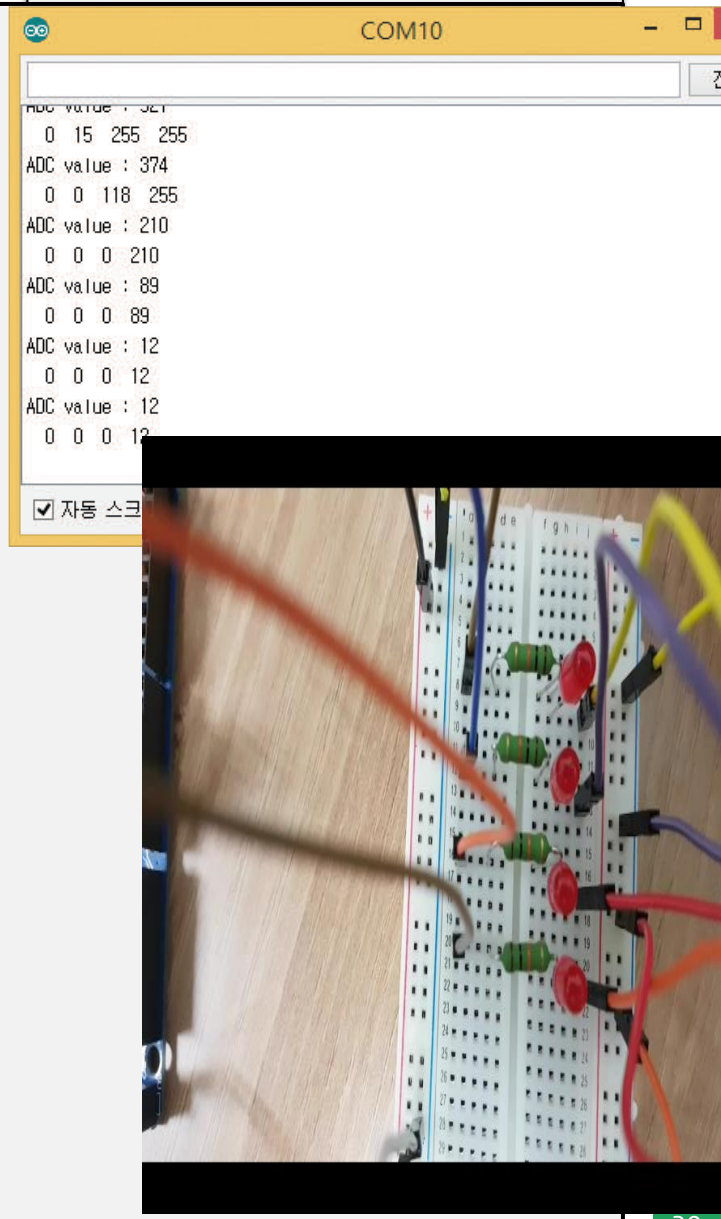
③ Adjusting the Brightness of LEDs using a Potentiometer (2)

Sketch 7-5

```
int pins_LED[] = {2, 3, 4, 5}; // Pins connected to LEDs

void setup() {
  Serial.begin(9600);
  for (int i = 0; i < 4; i++) {
    pinMode(pins_LED[i], OUTPUT);
  }
  // Pin A0 connected to a Potentiometer
  pinMode(A0, INPUT);
}

void loop() {
  int ADC_value = analogRead(A0); // ADC value
  int PWM_value[4] = {0, };
  Serial.println(String("ADC value : ") + ADC_value);
  for (int i = 3; i >= 0; i--) {
  // Calculating the brightness of each LED
    if (ADC_value >= 256 * i) {
      PWM_value[i] = ADC_value - 256 * i;
      ADC_value -= (PWM_value[i] + 1);
    }
    // Adjusting the brightness of LEDs
    analogWrite(pins_LED[i], PWM_value[i]);
    Serial.print(" ");
    Serial.print(PWM_value[i]);
  }
  Serial.println();
  delay(500);
}
```



④ Check the Result

- Show your Sketch code
- Show the execution of Sketch 7-3 and 7-5
- Answer the TA's questions and insert the score result to PLMS LAB 5

맺는말

❖ 아날로그 데이터 입력

- 10비트 해상도의 ADC를 통해 0~1023 사이의 양자화된 디지털 값 입력
- 16 채널
- 1023에 해당하는 기준 전압을 설정하여야 함

❖ 아날로그 데이터 출력

- ATmega2560에는 DAC가 없으므로 아날로그 데이터 출력은 불가능
- 디지털 신호의 일종인 펄스 폭 변조 신호를 통해 아날로그 데이터와 유사한 효과를 얻을 수 있음
- LED 밝기 제어, 모터 속도 제어 등에 PWM 신호가 사용

HW.5

Homework

❖ Write a Sketch code performs the following operation

- Turns on 4 LEDs in order
  - Each LED is connected to digital pin 2,3,4,5, respectively
- Turns on a LED with adjusting the brightness from 0% to 100% using PWM
  - For example, the brightness of the LED connected to digital pin 2 is adjusted from 0(0%) to 255(100%), then initialized to 0(0%)

Digital pin	Initial Brightness	Initial analogWrite value
2	0%	0
3	25%	63
4	50%	127
5	75%	191

- Please refer to the working movie

❖ Submission

1. Insert your Sketch code to PLMS HW 5
2. Submit a link to the video recording the result of your work.