아날로그 2
오실로스코프와 함수발생기
주기적인 데이터 처리

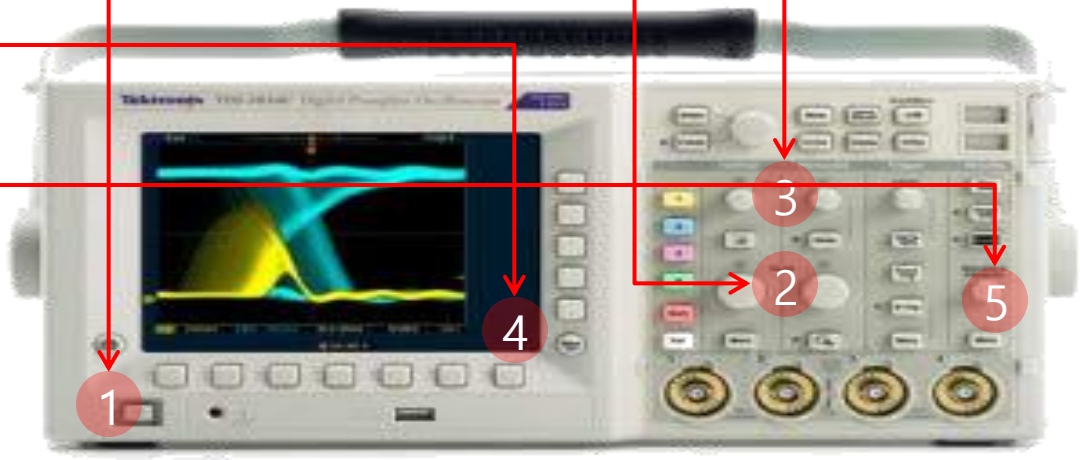
부산대학교 공과대학 전기컴퓨터공학부
정보컴퓨터공학전공

오실로스코프(Oscilloscope,OSC)

- 특정 시간 간격의 전압 변화를 볼 수 있는 장치
- 시간에 따라 변화하는 신호를 주기적이고 반복적인 하나의 전압 형태로 파악할 수 있다

오실로스코프의 기능

- 스위치
  - 전원 스위치
  - 수직 및 수평 스케일 조절기
  - 수직 및 수평 위치 조절기
  - 측정값 설정 스위치
  - Intensity 조절기

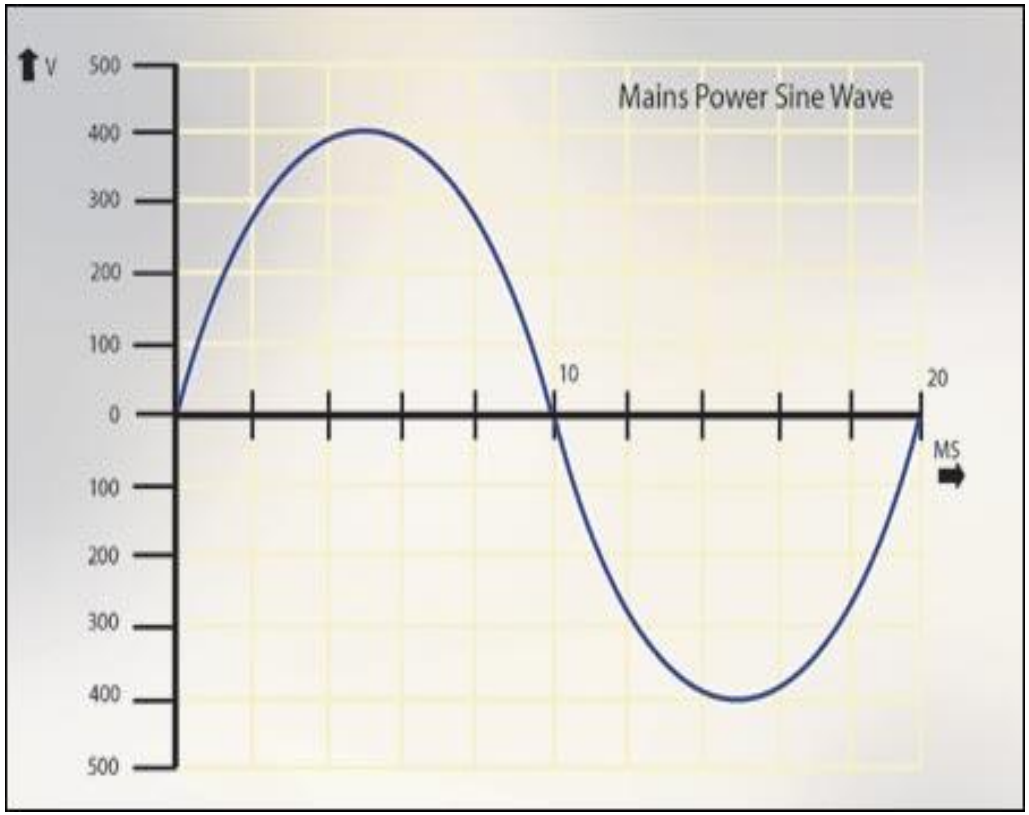


측정스위치->측정기능선택
(주기, rms, peak to peak...)

오실로스코프의 특징

❖ X축

- sec/div 표시
- 파형의 주기 = (sec / div) * 눈금 수
- 예 : 2msec/div * 10div = 20msec
- 주파수 = 1/ 주기
- 예 : f = 1/T = 1 / 20msec = 50Hz



❖ Y축

- volt/div 표시
- 전압 = (v/div) * 눈금 수
- 예 : 100mv/div * 4 div = 400mV
- Vp=400 mV, Vpp=800 mV
- 실제는 측정값 설정 버튼으로 기능 설정 가능
  - *Vp : peak voltage*
  - *Vpp : peak-to-peak voltage*

오실로스코프의 특징 (con't)

❖ 파형의 표시

- $F(t) = A \cdot \sin(\omega t + \theta) = A \cdot \sin(2\pi f t + \theta)$
- 여기서 A : 진폭(v), f : 주파수(Hz), θ : 위상 (rad)

❖ 기타

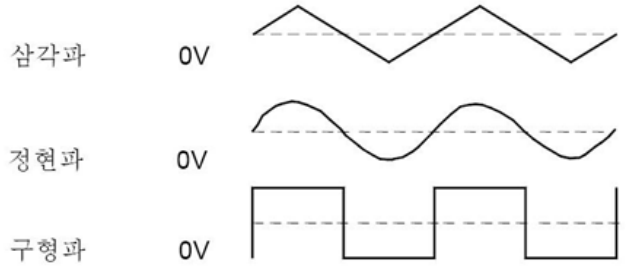
- 입력 임피던스가 높으므로 멀티미터보다 정확한 값을 획득 가능
- OSC를 잘 다룰 수 있어야 한다(익숙할 때까지)
- OSC는 제품마다 조금씩 다르지만 기본적 기능은 비슷

❖ 정현파의 실효값 (RMS 값)

$$V_{rms} = \sqrt{\frac{1}{T} \cdot \int_{t_0}^{t_0+T} V^2(t)\, dt} = \frac{V_p}{\sqrt{2}}$$

함수발생기(Function Generator)

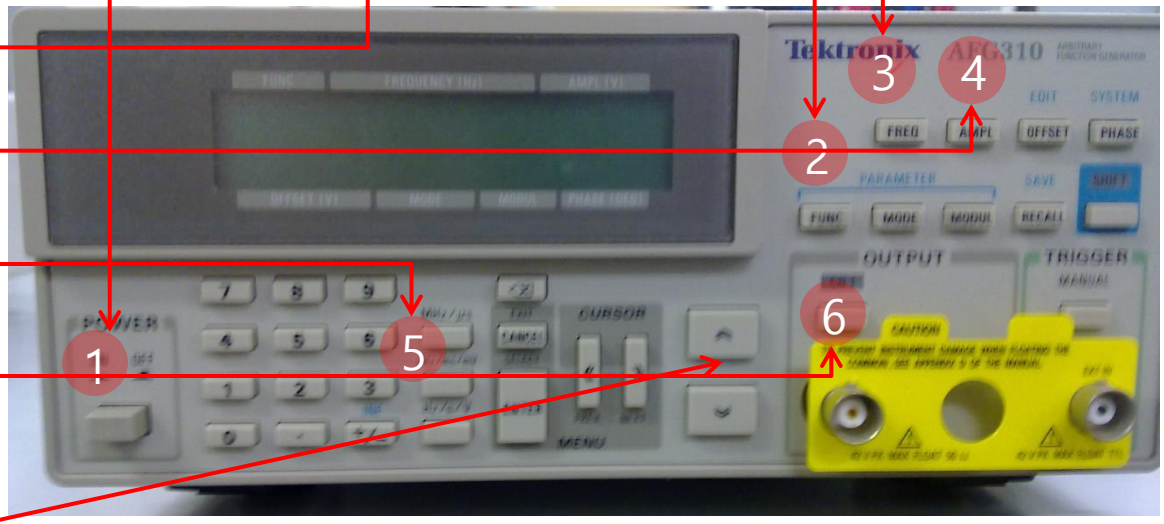
- 여러 가지 형태의 함수를 발생시켜 전압 형태로 출력하는 장치
- 정현파, 구형파, 삼각파 등 출력 가능
- 진폭조절기로 출력조정 가능
- 스위치



- 전원 스위치
- 주파수 조절 스위치
- 파형 조절 스위치
- 진폭 조절 스위치
- 숫자 입력 스위치
- 출력 enable 스위치

FREQ -> 숫자입력(5) -> 단위입력(5) -> ENTER

AMPL -> 숫자입력(5) -> 단위입력(5) -> ENTER



파형 -> 버튼 -> ENTER

Lab.6-1

OSC 및 함수발생기

❖ 실험 순서

① 준비 및 기본 실험

- 수평감도, 수직감도

② 파형 측정 실험

③ 실효값 및 오차 측정

① 준비 및 기본 실험

❖ OSC에 관련된 용어 이해

- 초점 (Focus), 세기 (Intensity), 수직 및 수평 위치 조절 (Horizontal and Vertical Position), AC-GND-DC 스위치(Switch), 신호 입력 (Signal Input), 모드 스위치(Mode Switch), 트리거 단자(Trigger control)

1. 준비작업

① 오실로스코프를 켜고, 선명도, 밝기, 스크린 상의 수평선과 일치하게 조절

② 오실로스코프의 입력채널에 함수 발생기를 연결하고, 함수 발생기의 출력이 1kHz의 정현파가 되도록 설정

③ 오실로스코프의 수직감도를 1 V/div로 설정, 함수발생기의 전압을 2v로 설정하여 4Vpp 정현파 조정

① 준비 및 기본 실험

2. 수평감도

④ 식 T = 1 / f를 사용하여 ms 단위로 1kHz 정현파의 주기를 <결과표 6-1>에 기록한다.

⑤ 오실로스코프의 수평감도를 0.2 ms/div에 설정하라. 순서 4의 계산 결과를 사용하여, 1kHz 신호의 한 사이클이 적절하게 표시되도록 필요한 수평눈금 (div) 수를 측정하여 <결과표 6-1>에 기록한다.

⑥ 오실로스코프의 수평감도를 0.5 ms/div, 1 ms/div로 각각 바꾸면서, 1kHz 신호의 한 사이클이 적절하게 표시되도록 필요한 수평눈금(div) 수를 측정하여 <결과표 6-1>에 기록한다

⑦ 수평감도를 0.2 ms/div에서 0.5 ms/div과 1 ms/div로 바꾸었을때 정현파의 **모양**에 나타나는 효과는 무엇인가?

<결과표 6-1>

수평감도 측정(수직감도 1V/div)	1kHz, 4Vpp, 정현파
주기(ms)	1ms
수평감도 0.2ms/div, 1cycle 수평눈금(div)수	25
수평감도 0.5ms/div, 1cycle 수평눈금(div)수	10
수평감도 1ms/div, 1cycle 수평눈금(div)수	5

① 준비 및 기본 실험

3. 수직감도

⑧ 함수발생기에 손대지 말고, 수평감도를 다시 0.2 ms/div로 하고, 수직감도를 2 V/div로 바꾸어라. 이 감도를 이용하여 피크 대 피크값 사이의 수직눈금(div) 수를 먼저 계산한 다음, 감도에 곱하여 스크린 상의 정현파의 피크 대 피크값을 계산하여 <결과표 6-2>에 기록한다.

⑨ 오실로스코프의 수직감도를 0.5 V/div로 바꾸고, 순서 8을 반복한다.

⑩ 수직감도를 2 V/div에서 0.5 V/div로 바꾸었을 때 정현파 **모양**에 나타나는 효과는 무엇인가?

<결과표 6-2>

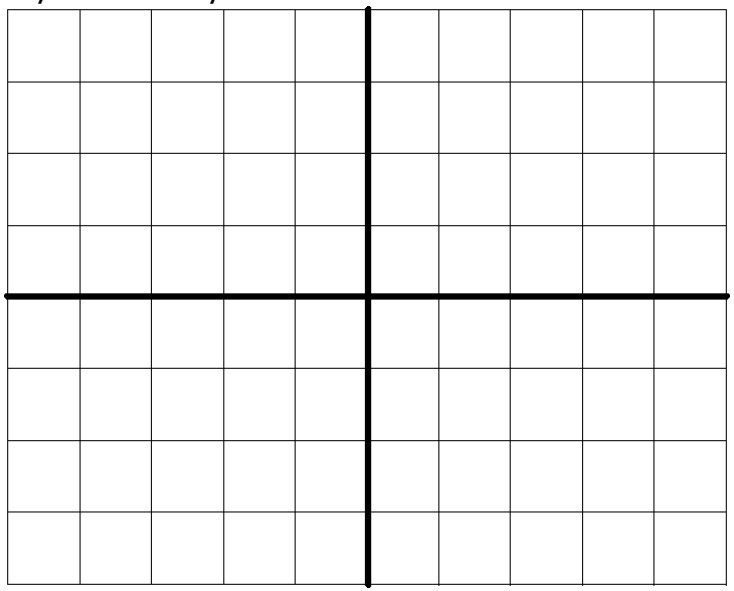
수직감도 측정(수평감도 0.2ms/div)	1kHz, 4Vpp,정현파
수직감도 2V/div, peak-to-peak 수직눈금(div)수	10
수직감도 2V/div, peak-to-peak voltage	4
수직감도 0.5V/div, peak-to-peak 수직눈금(div)수	4
수직감도 0.5V/div, peak-to-peak voltage	4

② 파형 측정

① 오실로스코프에 5kHz의 6 V_{pp} 정현파를 명확하게 표시되도록 필요한 모든 조정을 맞추어라. 스크린의 중앙에 0V를 설정하라. 선택한 감도(ms/div, v/div)를 기록한다. 필요한 수직, 수평눈금(div) 수를 주의깊게 기록하면서 <결과그림 6-a>에 파형을 그려라.

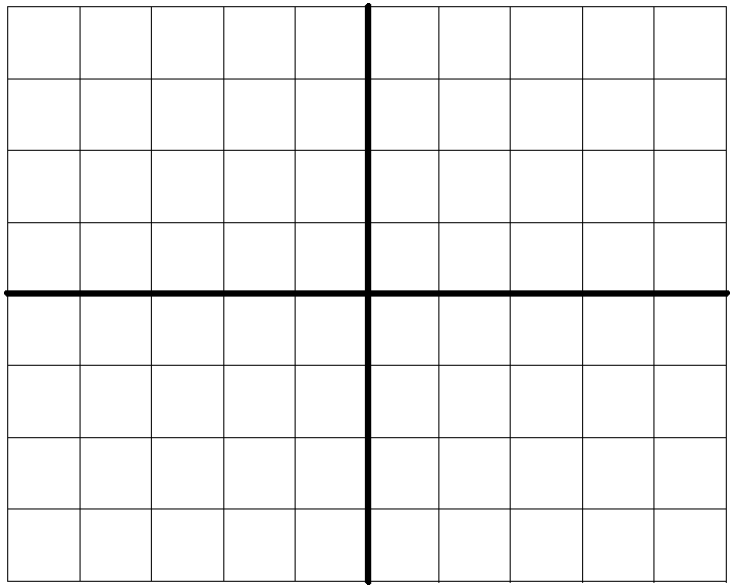
② <결과그림 6-b>에 100kHz 4 V_{pp} 구형파에 대해 순서 1를 반복한다.

5kHz, 6Vpp, 정현파
ms/div :0.2ms/div
v/div : 0.5V/div



<결과그림 6-a>

100kHz, 4Vpp, 구형파
ms/div :
v/div :



<결과그림 6-b>

③ 실효값 및 오차 측정

① 스크린 상에 1kHz, 4 Vpp정현파로 재조정하라. 정현파의 실효값을 계산하여 <결과표 6-3>에 기록한다.

② 오실로스코프에서 함수 발생기를 분리하고 디지털 멀티미터를 사용하여 함수 발생기의 출력의 실효값를 측정하여 <결과표 6-3>에 기록

③ 계산된 것과 측정된 레벨의 차를 다음 식을 사용하여 %오차의 크기를 계산하여 <결과표 6-3>에 기록

- V_A=계산치, V_M=측정치

$$\%\text{오차} = \left|\frac{V_A - V_M}{V_A}\right| \times 100\%$$

<결과표 6-3>

	1kHz, 4Vpp, 정현파
실효값 계산치	
실효값 측정치	
% 오차	

④ 결과확인

- 작성한 결과표와 결과 그림을 보여라.
- 조교의 물음에 답하고 채점 결과를 PLMS LAB 6에 직접 입력하라

디지털 및 아날로그 데이터 처리

	입력	출력

디지털	`int pinNo = 13;`
	
	`pinMode(pinNo, INPUT);`
	`boolean value =`
	`digitalRead(pinNo);`
		`int pinNo = 13;`
		`boolean value = HIGH;`
		
		`pinMode(pinNo, OUTPUT);`
		`digitalWrite(pinNo,`
		`value);`

아날로그	`int pinNo = A0;`
	
	`pinMode(pinNo, INPUT);`
	`int value =`
	`analogRead(pinNo);`
		`int pinNo = 3;`
		`int dutyCycle = 128;`
		
		`pinMode(pinNo, OUTPUT);`
		`analogWrite(`
		`pinNo, dutyCycle);`

주기적인 처리를 위한 함수

void delay(unsigned long ms)

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

void delayMicroseconds(unsigned int us)

  - us : 마이크로초 단위의 지연 시간
- **반환값** : 없음

unsigned long millis(void)

- **매개변수** : 없음
- **반환값** : 프로그램이 시작된 이후의 밀리초(millisecond) 단위 경과 시간

Lab.6-2

주기적인 처리

❖ 실험 순서

① 버튼 및 LED 연결

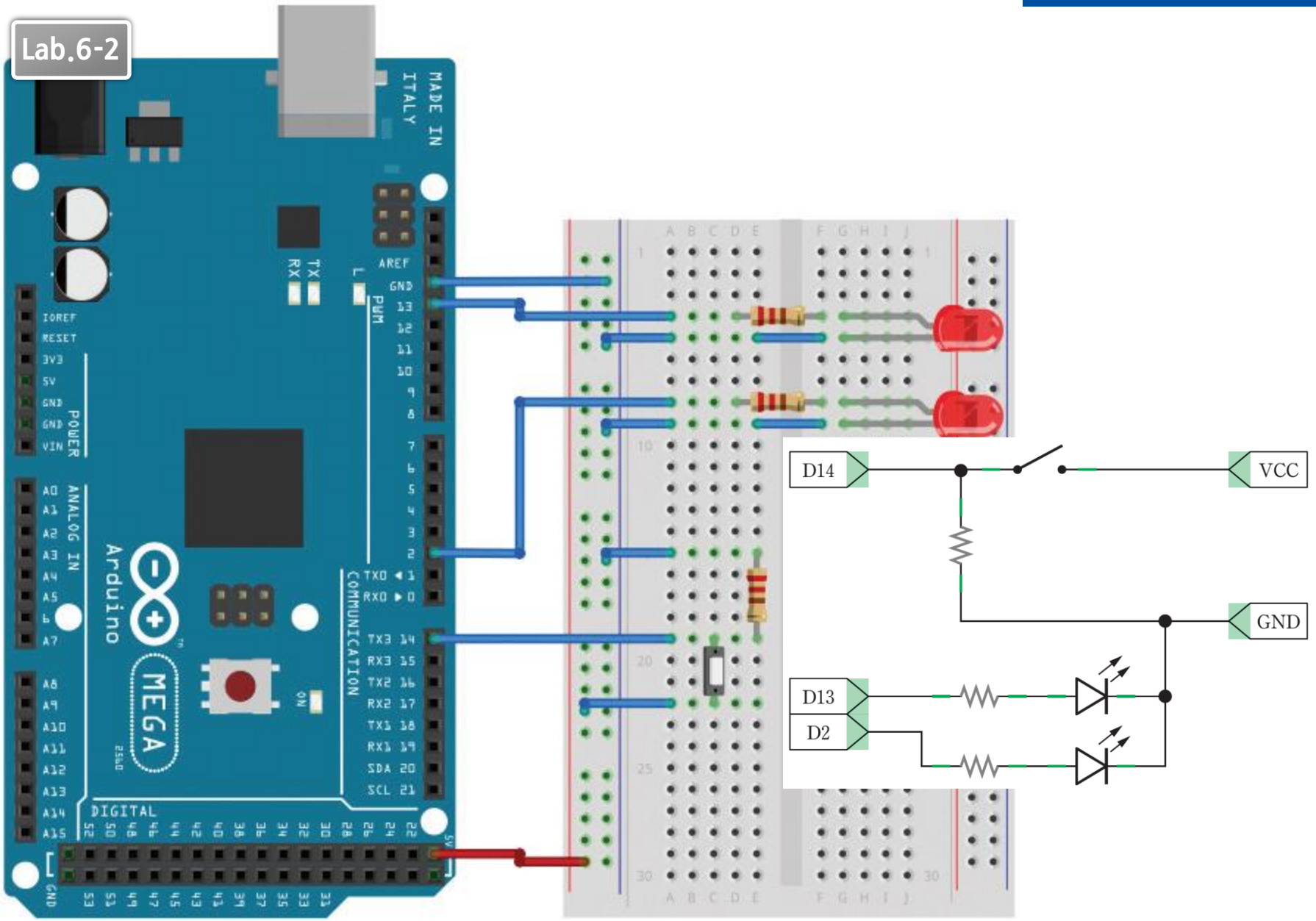
② Millis함수를 이용한 Blink (Sketch 8-3)

③ 주기적인 LED 제어 – delay 함수 이용 (Sketch 8-1)

④ 주기적인 LED 제어 – millis 함수 이용 (Sketch 8-4)

⑤ 가변저항을 이용한 주기제어(Sketch 8-6)

Lab.6-2


MADE IN ITALY
AREF
GND
13
12
11
10
9
8
7
6
5
4
3
2
TX
RX
L
PWM
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
A6
A7
ANALOG IN
Arduino
MEGA
ON
TX0 1
RX0 0
TX3 14
RX3 15
TX2 16
RX2 17
TX1 18
RX1 19
SDA 20
SCL 21
COMMUNICATION
DIGITAL
D14
VCC
GND
D13
D2

② delay 함수 vs millis 함수

❖ delay 함수를 통해 LED를 반전시킨 후 1초 동안 대기

```
void loop() {
  LED_state = !LED_state; // LED 상태 반전
  digitalWrite(pin_LED, LED_state); // LED 출력
  delay(1000);
}
```

Sketch 8-2

- delay 함수가 실행 중인 동안에는 대부분의 마이크로컨트롤러 동작이 중지됨

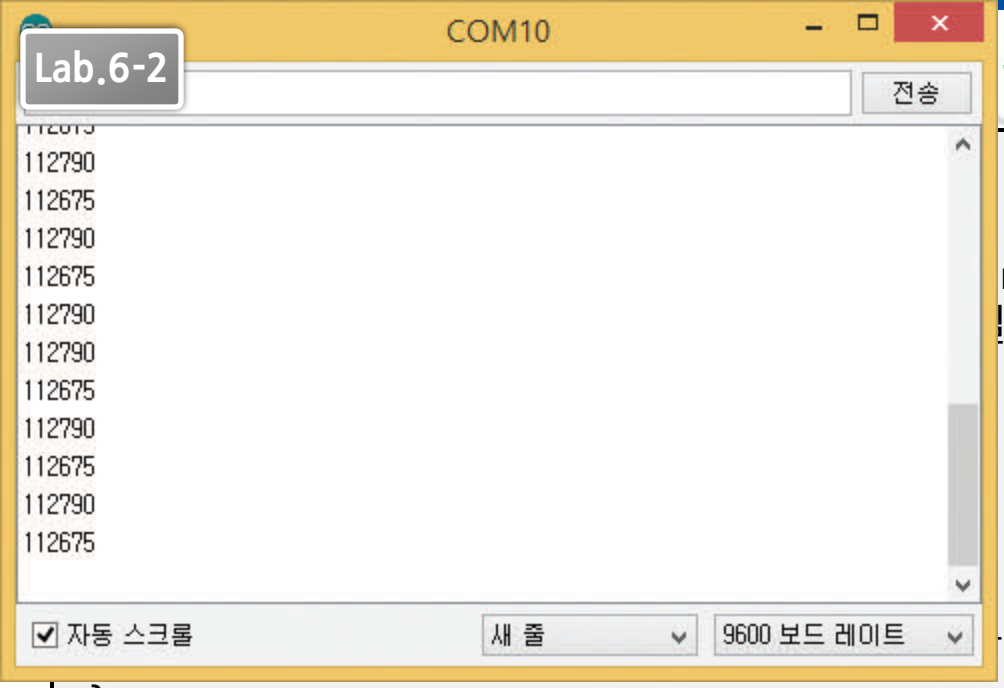
❖ millis 함수는 프로그램 시작 후의 실행 시간을 반환함

- **delay 함수와 달리 실행 시간을 바로 반환하므로 다른 작업이 가능**
- 일정 시간 간격을 설정하기 위해서는 경과 시간을 계속 검사하여야 함

unsigned long millis(void)

- **매개변수** : 없음
- **반환값** : 프로그램이 시작된 이후의 밀리초(millisecond) 단위 경과 시간

Lab.6-2



link

Sketch 8-3

```
nt;
실행 횟수
```

```
}

void loop() {
  time_current = millis(); // 현재 시간
  count++; // loop 함수 실행 횟수
  // 1초 이상 시간이 경과한 경우
  if (time_current - time_previous >= 1000) {
    time_previous = time_current; // 시작 시간 갱신
    LED_state = !LED_state; // LED 반전
    digitalWrite(pin_LED, LED_state);
    Serial.println(count); // 1초 동안 loop 함수가 실행된 횟수 출력
    count = 0;
  }
}
```

1초에 100,000회 이상
경과 시간을 검사

Lab.6-2

③ 주기적인 LED 제어 - delay

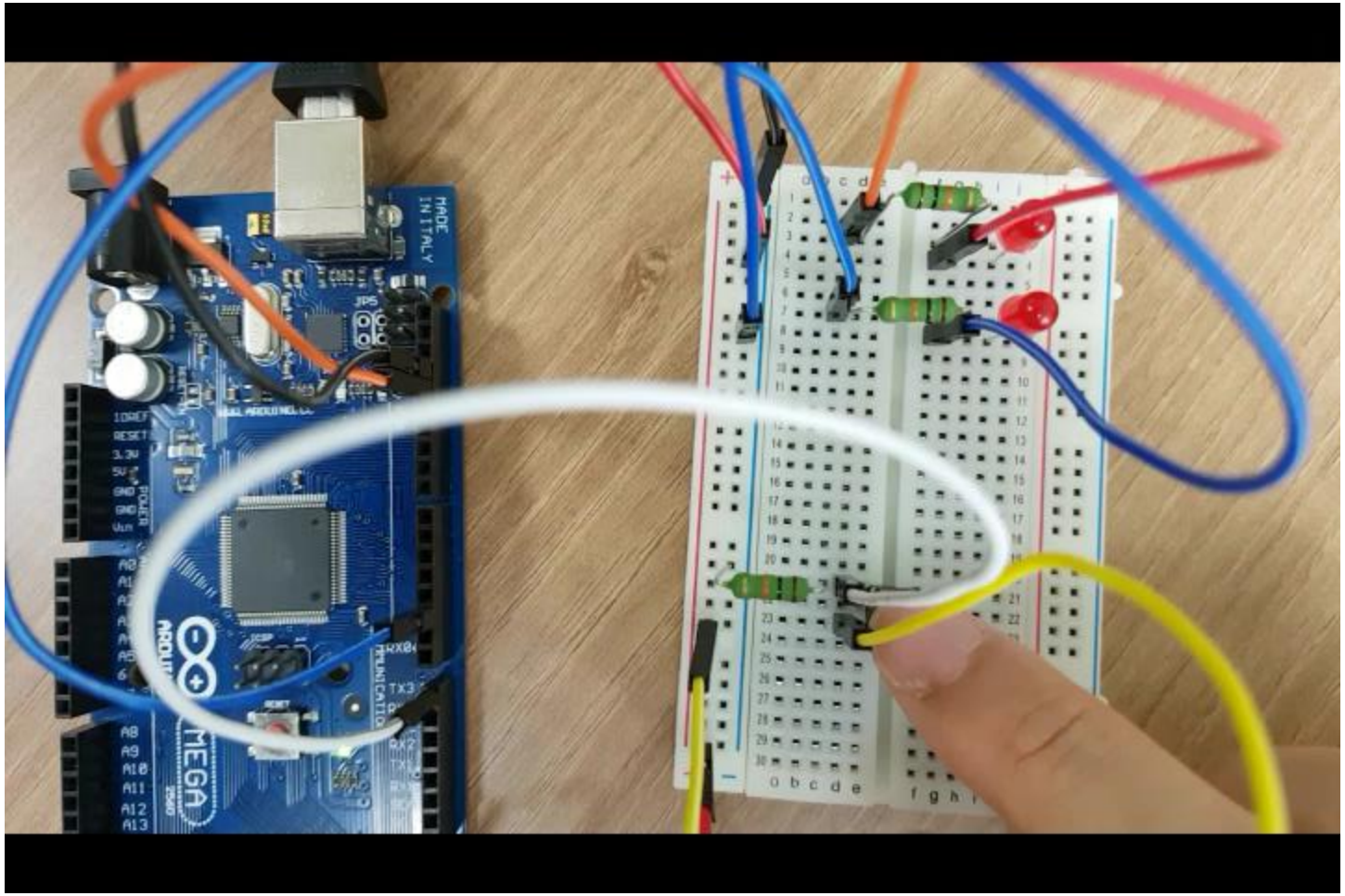
Sketch 8-1

```
int pin_button = 14; // 버튼 연결 핀
int pin_LED1 = 13, pin_LED2 = 2; // LED 연결 핀
boolean LED_state1 = false; // LED 상태
boolean LED_state2 = false;

void setup() {
  pinMode(pin_button, INPUT);
  pinMode(pin_LED1, OUTPUT);
  digitalWrite(pin_LED1, LED_state1);
  pinMode(pin_LED2, OUTPUT);
  digitalWrite(pin_LED2, LED_state2);
}

void loop() {
  digitalWrite(pin_LED1, LED_state1);
  delay(1000); // 1초 대기
  LED_state1 = !LED_state1; // 13번 LED 반전
  if (digitalRead(pin_button)) { // 버튼이 눌러진 경우
    LED_state2 = !LED_state2; // 2번 LED 반전
    digitalWrite(pin_LED2, LED_state2);
  }
}
```

2번째 LED가 버튼에 즉각적인 반응을 하는가?

왜 이런 현상이 발생하는가?

③ 주기적인 LED 제어 - delay



Lab.6-2

④ 주기적인 LED 제어 - millis

Sketch 8-4

```
int pin_button = 14; // 버튼 연결 핀
int pin_LED1 = 13, pin_LED2 = 2; // LED 연결 핀
unsigned long time_previous, time_current;
boolean LED_state1 = false; // LED 상태
boolean LED_state2 = false;

void setup() {
  pinMode(pin_button, INPUT);
  pinMode(pin_LED1, OUTPUT);
  digitalWrite(pin_LED1, LED_state1);
  pinMode(pin_LED2, OUTPUT);
  digitalWrite(pin_LED2, LED_state2);
  time_previous = millis(); // 현재 시간
}
```

Continued on next slide

Lab.6-2

④ 주기적인 LED 제어 - millis

Sketch 8-4

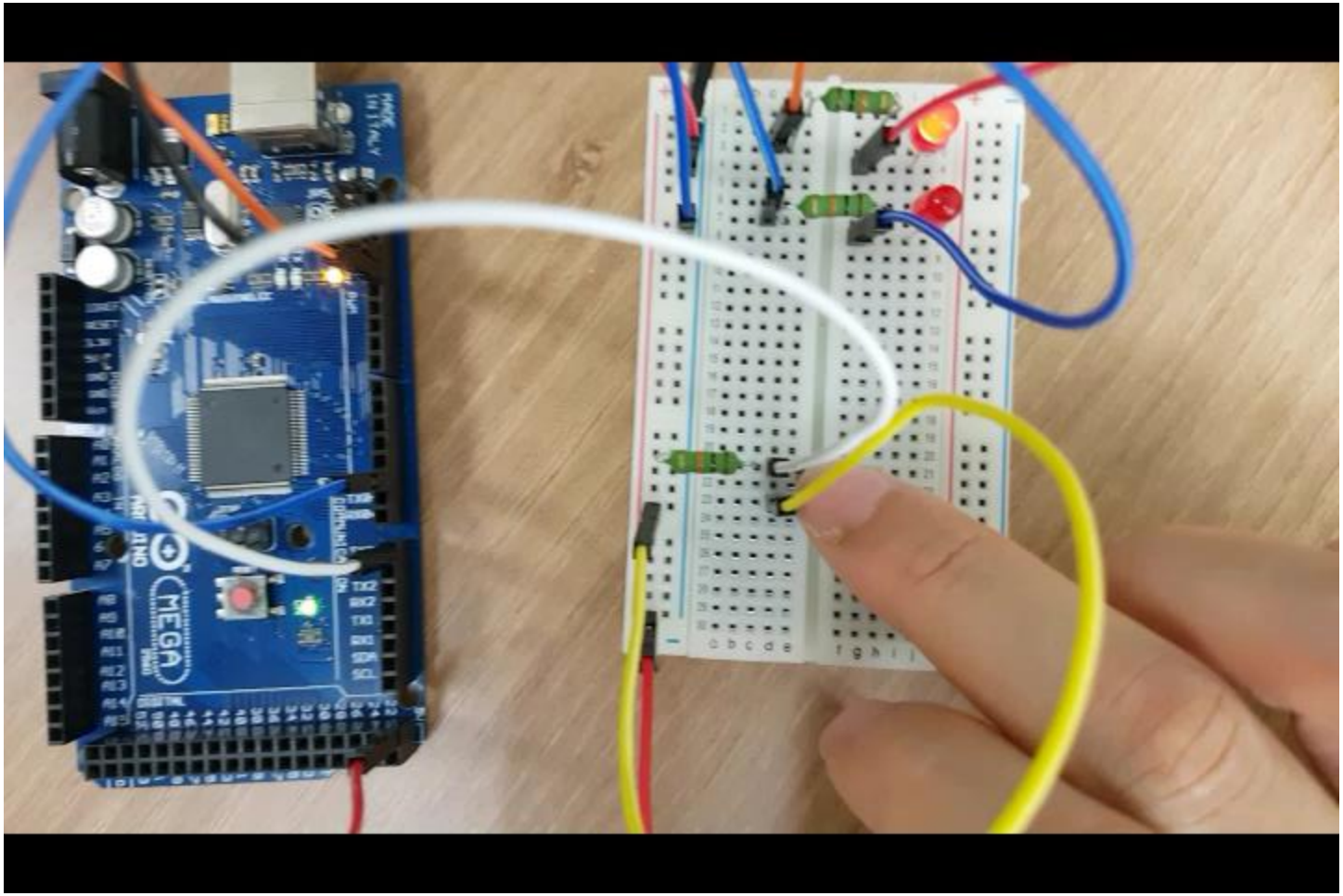
```
void loop() {
  time_current = millis();
  if (time_current - time_previous >= 1000) {
    time_previous = time_current; // 시작 시간 갱신
    LED_state1 = !LED_state1; // LED 반전
    digitalWrite(pin_LED1, LED_state1);
  }

  if (digitalRead(pin_button)) { // 버튼이 눌러진 경우
    LED_state2 = !LED_state2; // 2번 LED 반전
    digitalWrite(pin_LED2, LED_state2);
    delay(100);  // 왜 필요할까?
  }
}
```

첫 번째 LED의 blink

버튼 입력 처리

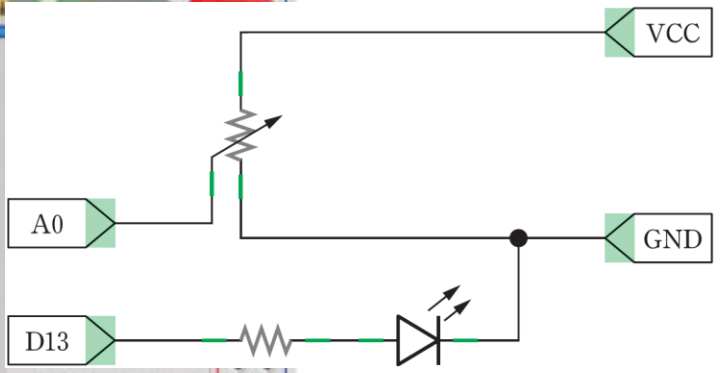
버튼을 누르고 있으면 LED는 어떻게 되는가?
해결 방법은 ? *(sketch 6-5 참고)*

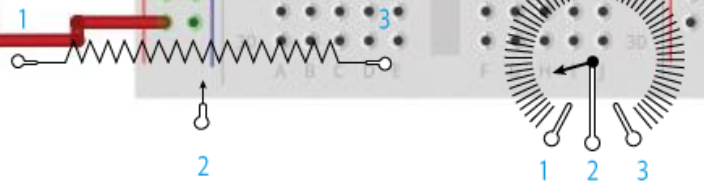
④ 주기적인 LED 제어 - millis



Lab.6-2

준비





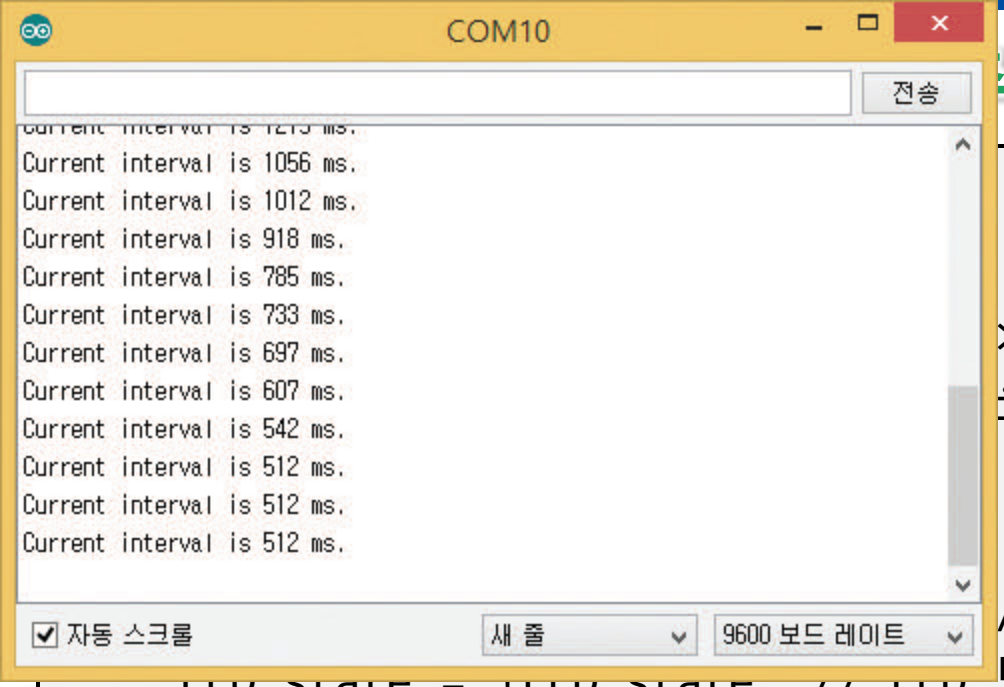
가변저항의 동작방식

⑤ 가변저항을 이용한 주기제어

Sketch 8-6

```
int pin_LED = 13; // LED 연결 핀
unsigned long time_previous, time_current;
unsigned long interval = 1000; // 점멸 간격
boolean LED_state = false; // LED 상태

void setup() {
  pinMode(A0, INPUT);
  pinMode(pin_LED, OUTPUT);
  digitalWrite(pin_LED, LED_state);
  Serial.begin(9600);
  time_previous = millis(); // 현재 시간
}
```

Continued on next slide



기제어

Sketch 8-6

```
>= interval) {
is ");


/ 시작 시간 갱신
반전
    digitalWrite(pin_LED, LED_state);
  }
  int adc = analogRead(A0); // 가변저항 읽기
  interval = map(adc, 0, 1023, 500, 1500); // 점멸 간격으로 변환
}
```

0.5 ~ 1.5초 사이의 시간으로 매핑

참고 : map() 함수

long map(long value, long fromLow, long fromHigh, long toLow, long toHigh)

Re-maps a number from one range to another. That is, a value of fromLow would get mapped to toLow, a value of fromHigh to toHigh, values in-between to values in-between, etc.

value: the number to map

fromLow: the lower bound of the value's current range

fromHigh: the upper bound of the value's current range

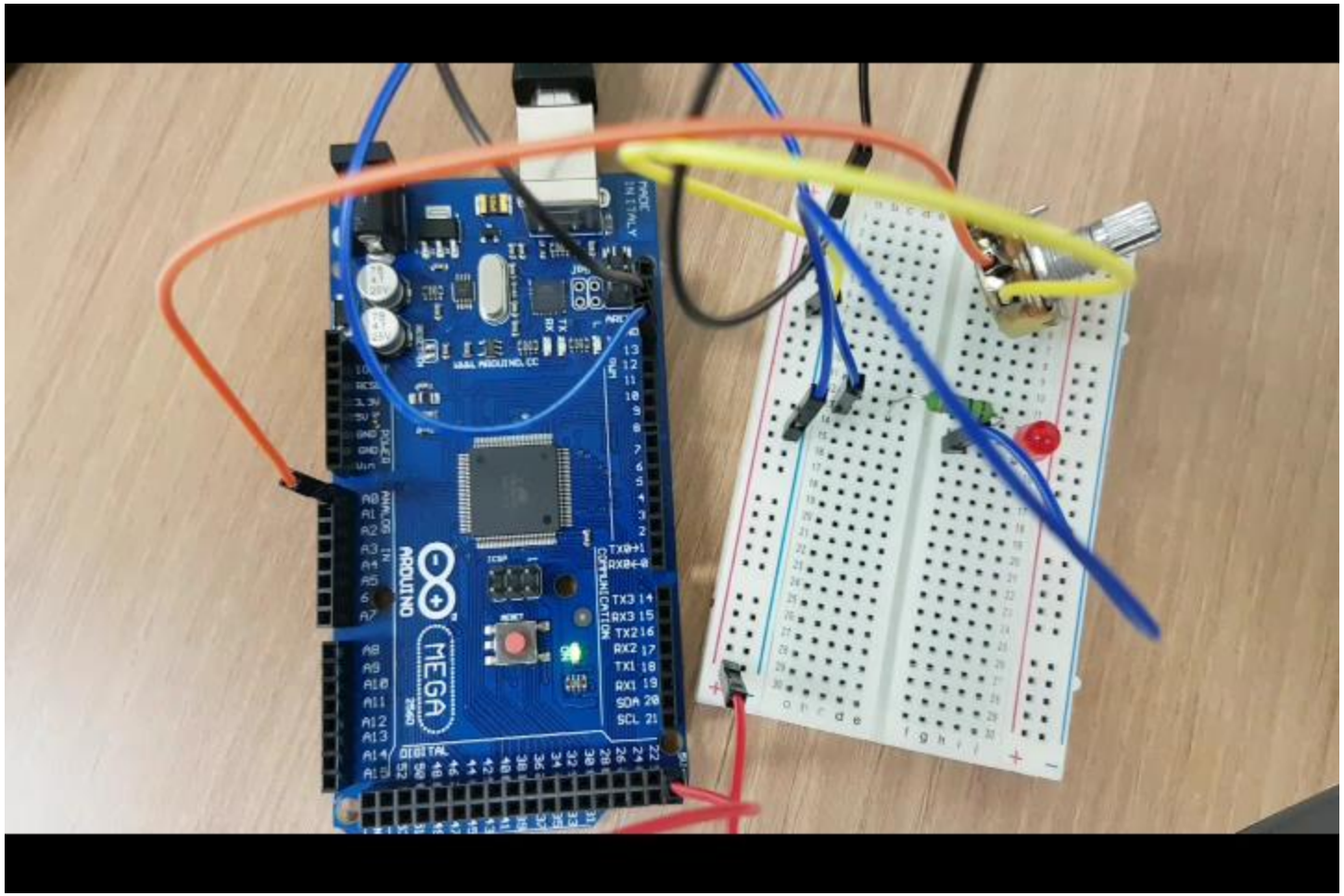
toLow: the lower bound of the value's target range

toHigh: the upper bound of the value's target range

```
new_value = map(value, 0, 10, 1, 100)
```

1 -> 10
2 -> 20
10 -> 100

⑤ 가변저항을 이용한 주기제어



⑥ 결과확인

- 작성한 Sketch 8-3, 8-5, 8-6 편집창을 보여라
- Sketch 8-3, 8-5, 8-6의 실행을 보여라
- 조교의 물음에 답하고 채점 결과를 PLMS LAB 6에 직접 입력하라

HW.6

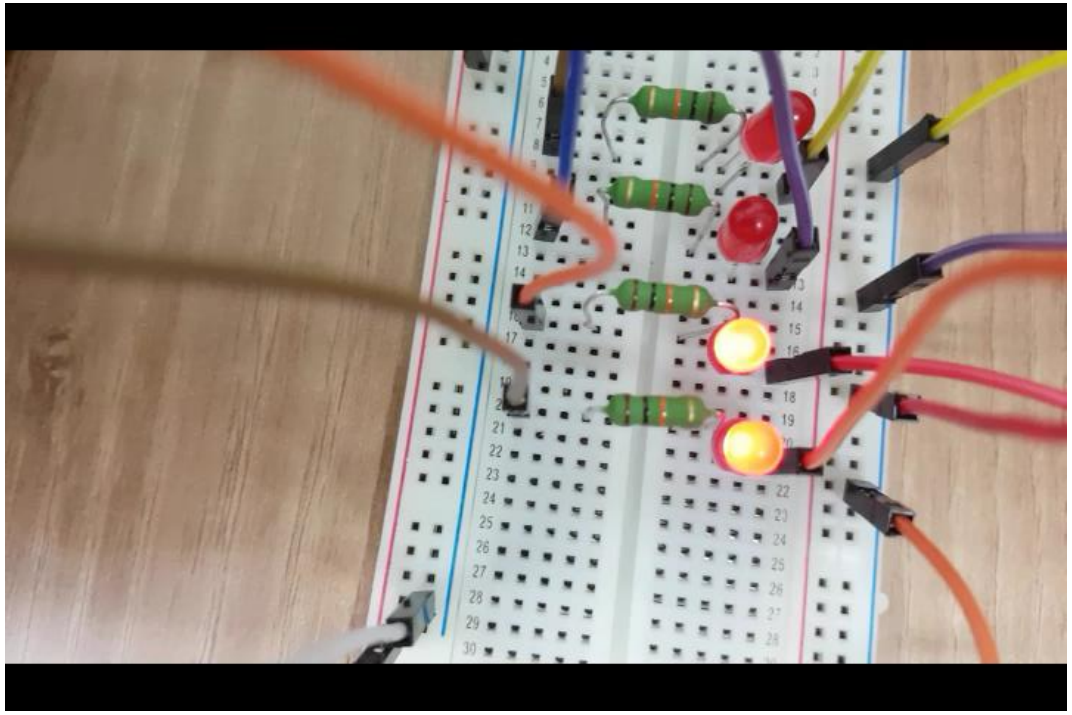
실습과제

❖ 아래 동작을 하는 Sketch를 작성하라

- Digital 2번,3번 핀에 LED연결
- 가변저항으로 2번, 3번 LED의 점멸 속도를 동시에 제어하라
- 가변저항으로부터의 입력이 커지면 2번핀 LED는 빨라지고 3번핀 LED는 느려지도록 구현 (가변저항 입력이 작아지면 반대로 동작)
- 점멸 주기는 (0.5초~ 1.5초)
- Millis 함수를 사용하여 구현
- 동작 동영상을 참고할 것

❖ 제출 방법

1. PLMS의 HW 6에 작성한 Sketch 코드를 입력하라
2. 또한 실행결과를 촬영한 동영상 링크를 포함 하여 제출하라



맺는말

❖ 일정 시간 간격으로 특정 동작을 반복하는 방법

- delay 함수 사용 방법
  - delay 함수가 실행 중인 동안은 대부분의 다른 작업을 수행하지 못함
  - 즉각적인 반응을 얻기 어려울 수 있음
- millis 함수 사용 방법
  - 현재까지의 실행 시간을 바탕으로 시간 경과를 계산
  - 코드가 복잡해지는 단점은 있지만 즉각적인 반응을 얻을 수 있음