

Chapter 9. 주기적인 데이터 처리 - delay, millis

마이크로컨트롤러는 아날로그 및 디지털 신호를 입력 받거나 출력할 수 있다. 아날로그와 디지털 신호를 사용하여 제어 장치를 구성하는 경우 흔히 접하게 되는 문제 중 하나가 '일정 시간 간격으로 특정 동작을 반복하는' 문제이다. 1초 간격으로 LED가 점멸하도록 하는 문제, 1초 간격으로 증가하는 카운터를 만드는 문제, 10초 간격으로 온도 센서의 값을 읽어 텍스트 LCD에 표시하는 문제 등이 이러한 예에 속한다. 이처럼 일정 시간 간격으로 특정 동작을 반복할 필요가 있을 때 사용할 수 있는 아두이노 함수에 delay와 millis가 있다.

1. delay

delay는 지정한 시간만큼 실행을 일시 중지하기 위해 사용한다.

```
void delay(unsigned long ms)
- 매개변수
  ms : 밀리초 단위의 지연 시간
- 반환값 : 없음
```

delay 함수에 사용되는 매개변수는 밀리초 단위의 지연 시간을 지정하며, 최대 약 50일의 지연 시간을 지정할 수 있다. delay가 밀리초 단위의 지연 시간을 지정한다면 delayMicroseconds는 마이크로초 단위의 지연 시간을 지정할 수 있다.

```
void delayMicroseconds(unsigned int us)
  us : 마이크로초 단위의 지연 시간
반환값 : 없음
```

블링크 스케치를 기억하는가? 1초 간격으로 LED를 점멸하기 위해 delay 함수를 사용하였다. 블링크 스케치를 수정하여 디지털 13번 핀에 연결된 LED를 1초 단위로 점멸하면서, 디지털 14번 핀에 연결된 버튼이 눌러질 때마다 디지털 2번 핀에 연결된 LED 상태를 반전시키는 스

케치를 delay 함수를 사용하여 작성해 보자.

스케치 1. 버튼과 LED 제어 - delay 함수 사용

```
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 눌려진 경우
    LED_state2 = !LED_state2; // 2번 LED 반전
    digitalWrite(pin_LED2, LED_state2);
  }
}
```

스케치 1을 업로드 해보자. 디지털 13번 핀에 연결된 LED가 1초 간격으로 점멸하는 것을 확인할 수 있다. 하지만 디지털 14번 핀에 연결된 버튼을 눌러보면 디지털 2번 핀에 연결된 LED의 상태가 바뀌지 않는 경우가 있음을 발견할 수 있다. delay 함수는 프로그램의 실행을 일시적으로 중단하는 함수이므로 delay 함수가 실행 중인 동안 프로그램은 더 이상 진행되지 않고 delay 함수에서 멈추게 된다. 따라서 버튼을 누르는 순간 즉시 LED의 상태가 바뀔 것을

기대하겠지만 delay 함수가 실행 중인 동안 버튼이 눌러지면 버튼이 눌러진 것을 확인할 수 없다. 따라서 즉각적인 반응이 필요한 스케치에서는 delay 함수를 사용할 수 없으며, 대신 millis 함수를 사용하여야 한다.

2. millis

millis 함수는 현재 실행 중인 스케치가 시작된 이후의 경과 시간을 밀리초 단위로 반환한다.

```
unsigned long millis(void)
```

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

delay 함수를 사용하여 1초를 대기하면 1초 동안은 다른 작업을 할 수 없다. 이와 달리 millis 함수는 현재까지의 실행시간을 즉시 반환하므로 즉시 다른 작업을 수행할 수 있다. 하지만 millis 함수를 사용하는 경우에는 1초가 경과되었는지의 여부를 계속적으로 검사하여야 하는 단점도 있다. 블링크 스케치로 돌아가 보자. 스케치 2는 delay 함수 없이 millis 함수를 사용하여 구현한 블링크의 예이다.

스케치 2. 블링크 - millis 함수 사용

```
int pin_LED = 13;                // LED 연결 핀
boolean LED_state = false;       // LED 상태
unsigned long time_previous, time_current;
unsigned long count = 0;         // loop 함수 실행 횟수

void setup() {
    pinMode(pin_LED, OUTPUT);
    digitalWrite(pin_LED, LED_state);

    Serial.begin(9600);
    time_previous = millis();     // 시작 시간
}
```

```
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;  
    }  
}
```

스케치 2는 블링크 스케치와 동작을 보여주지만 작성해야 하는 코드도 길고 복잡하다. 하지만 스케치 2에서 delay 함수는 사용하지 않았다. 시리얼 모니터를 실행시켜 보자.

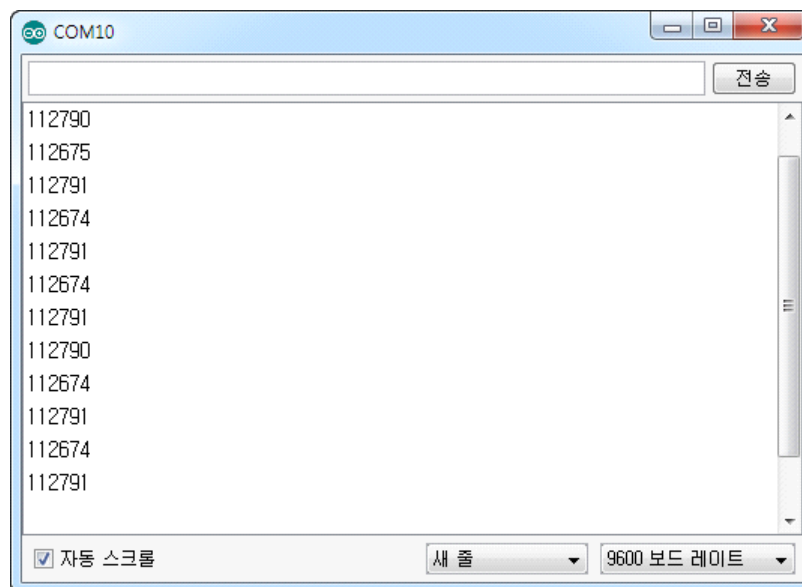


그림 1. 스케치 2 실행 결과

아두이노에서 loop 함수는 스케치가 실행 중인 동안 반복적으로 호출되는 함수이다. 스케치 1의 경우 loop 함수 내에 delay(1000) 문장이 존재하므로 1초에 한 번 loop 함수가 실행된다. 하지만 스케치 2의 경우에는 1초에 무려 10만 번이 넘게 실행되고 있음을 확인할 수 있다. 따라서 스케치 2과 같이 delay 함수 없이 millis 함수를 사용하여 작성한다면 10만 번이 넘게 버튼이 눌러졌는지 여부를 확인할 수 있고 버튼이 눌러지는 즉시 LED 상태를 반전시킬 수 있다. 스케치 1에서 delay 함수를 제거하고 millis 함수로 작성한 스케치의 예가 스케치 3이다.

스케치 3. 버튼과 LED 제어 - millis 함수 사용

```
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
}

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);
    }
}
```

스케치 3을 업로드 하고 버튼을 눌러보자. 버튼을 누르는 즉시 LED의 상태가 바뀌는 것을 확인할 수 있을 것이다. 스케치 3에서도 delay 함수가 사용되어 의아해할 수 있겠지만 loop 함수는 1초에 10만 번 이상 실행되므로 버튼을 아주 짧은 시간 동안 누른다고 하더라도 여러 번 버튼 상태를 검사하기 때문에 LED가 아주 빠른 속도로 깜빡거리는 것을 방지하기 위해 사용된 것이다.