

5. 디지털 출력

마이크로컨트롤러는 디지털 데이터만을 처리할 수 있다. 아날로그 데이터 역시 입력이 가능하지만 실제 처리는 아날로그-디지털 변환기(Analog-to-Digital Converter, ADC)를 거쳐 디지털 형태로 변환된 데이터만이 가능하며 출력 역시 처리된 디지털 데이터를 디지털-아날로그 변환기(Digital-to-Analog Converter, DAC)를 거쳐 출력한다. 표 1은 아두이노 우노에서 사용할 수 있는 아날로그 및 디지털 데이터 입출력 방법을 요약한 것으로 이 장에서는 발광 다이오드를 사용하여 디지털 데이터를 출력하는 방법에 대해 알아본다.

	입력	출력
디지털	<pre>int pinNo = 13; pinMode(pinNo, INPUT); int value = digitalRead(pinNo);</pre>	<pre>int pinNo = 13; int value = HIGH; pinMode(pinNo, OUTPUT); digitalWrite(pinNo, value);</pre>
아날로그	<pre>int pinNo = A0; pinMode(pinNo, INPUT); int value = analogRead(pinNo);</pre>	<pre>int pinNo = 3; int dutyCycle = 128; pinMode(pinNo, OUTPUT); analogWrite(pinNo, dutyCycle);</pre>

표 1. 디지털 및 아날로그 데이터 입출력 함수

발광 다이오드(Light Emitting Diode, LED)는 순방향으로 전원이 연결되었을 때 빛을 발산하는 다이오드의 일종으로 간단한 정보 표시를 위한 장치에 많이 사용된다. LED는 첨가된 화학 물질에 따라 다양한 색의 빛을 낼 수 있으며 적색과 녹색 LED를 흔히 볼 수 있다. 이외에도 리모컨에 사용되는 적외선 LED, 살균 및 소독용의 자외선 LED 등도 어렵지 않게 찾아볼 수 있다. LED는 두 개의 다리를 가지며 긴 쪽이 양극(anode)이다.

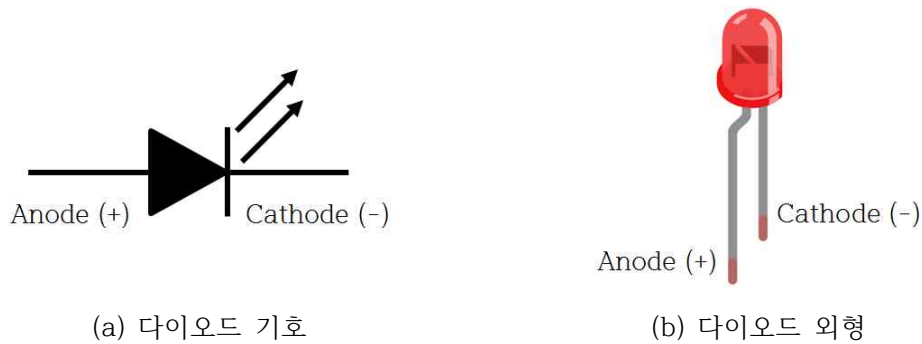


그림 1. 발광 다이오드 (LED)

1. Blink

디지털 입출력 핀을 출력으로 사용하기 위해서는 입력으로 사용하는 경우에서와 마찬가지로 먼저 해당 핀을 pinMode 함수를 통해 출력으로 설정하여야 한다.

```
void pinMode(uint8_t pin, uint8_t mode)
    pin : 설정하고자 하는 핀 번호
    mode : INPUT, OUTPUT, INPUT_PULLUP 중 하나
    반환값 : 없음
```

아두이노의 핀은 디폴트값으로 입력 상태로 설정되어 있으므로 출력으로 사용하기 위해서는 반드시 pinMode 함수를 사용하여야 한다. 핀 모드가 설정되면 digitalWrite 함수를 통해 해당 핀으로 디지털 데이터를 출력할 수 있으며 출력값은 미리 정의된 상수인 HIGH 또는 LOW를 사용할 수 있다. 핀의 모드가 입력으로 설정된 상태에서 digitalWrite 함수는 내장 풀업 저항의 사용 여부를 설정되므로 주의하여야 한다.

```
void digitalWrite(uint8_t pin, uint8_t value)
    pin : 핀 번호
    value : HIGH 또는 LOW
    반환값 : 없음
```

아두이노 스케치에서 가장 먼저 접하는 스케치는 Blink로 아두이노 프로그램에 포함되어 있으며 이 책의 1장에서도 사용하였다. 스케치 1은 아두이노 프로그램에서 제공하는 Blink 예제에서 주석을 삭제하고 출력으로 사용한 핀 번호를 10번으로 변경한 스케치이다.

스케치 1. 디지털 10번 핀 Blink

```
int led = 10;                // 10번 핀에 LED 연결

void setup() {
  pinMode(led, OUTPUT);      // 10번 핀을 출력 모드로 설정
}

void loop() {
  digitalWrite(led, HIGH);    // 10번 핀으로 HIGH (1 또는 ON) 출력
  delay(1000);                // 1초 (1000ms) 대기
  digitalWrite(led, LOW);     // 10번 핀으로 LOW (0 또는 OFF) 출력
  delay(1000);                // 1초 (1000ms) 대기
}
```

아두이노 우노 보드에서 10번 디지털 핀에는 LED가 연결되어 있어 Blink 스케치의 경우 컴퓨터와 USB 연결 이외의 연결은 필요하지 않다. DM-UStyle EX 보드에도 10번 디지털 핀에 연결된 LED가 포함되어 있으며 디지털 10번 핀에도 LED가 연결되어 있다. 디지털 10번 핀에 연결된 LED는 펄스 폭 변조 (Pulse Width Modulation, PWM) 신호 출력 테스트를 위해서도 사용된다. 디지털 10번 핀에 그림 2와 같이 LED를 연결하고 스케치 1을 실행시켜 보자. LED가 1초 간격으로 깜빡거리는가?

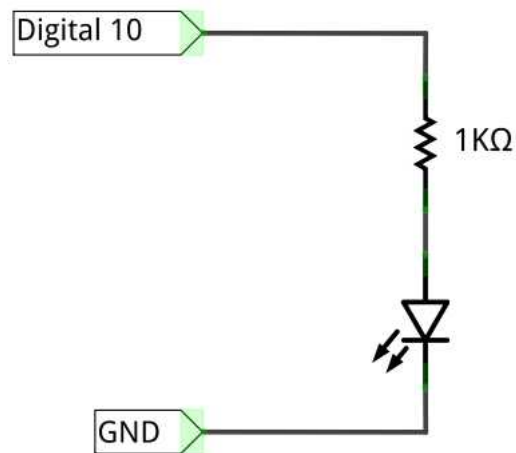


그림 2. 디지털 10번 핀에 LED 연결을 위한 회로도

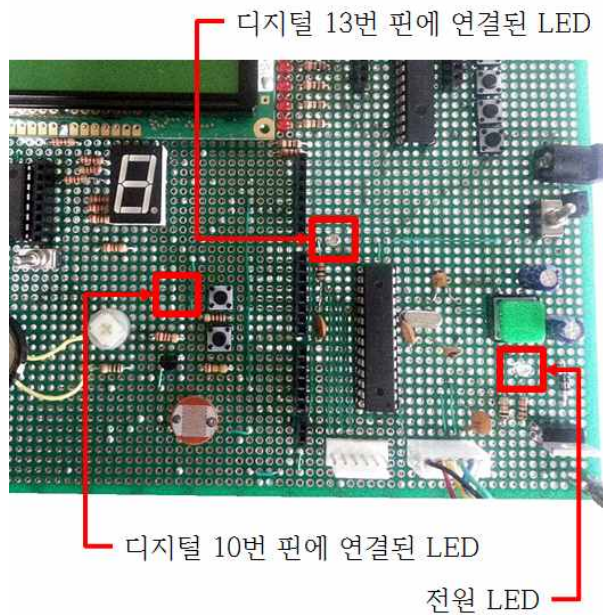


그림 3. DM-UStyle EX 보드의 LED

아날로그 입력 핀을 디지털 입력을 위해 버튼을 연결하여 사용할 수 있는 것과 마찬가지로 LED를 연결하여 디지털 출력으로 사용하는 것 역시 가능하다. 아날로그 0번 핀 A0에 그림 2에서와 유사하게 LED를 연결해 보자. DM-UStyle EX 보드에도 A0 핀에 연결된 LED는 포함되어 있지 않다. 스케치 1에서 'int led = 10;' 문장을 'int led = A0;'로 변경하고 실행시켜 보자. LED가 스케치 1에서와 마찬가지로 1초 간격으로 점멸하는 것을 확인할 수 있다.

2. 버튼 상태 나타내기

마이크로컨트롤러는 외부의 데이터를 받아들이고 이를 처리하여 결과를 낸다. 입력으로는 버튼을 출력으로는 LED를 사용할 수 있으므로 버튼이 눌러진 상태에서 LED가 켜지고 눌러지지 않은 상태에서 LED가 꺼지도록 해보자. 이는 간단한 초인종으로 활용할 수 있다. 먼저 그림 4와 같이 디지털 9번 핀에 풀다운 저항을 사용하여 버튼을 연결하고 디지털 10번 핀에는 LED를 연결한다. DM-UStyle EX 보드에는 디지털 9번과 10번 핀에 연결된 버튼과 LED가 포함되어 있다.

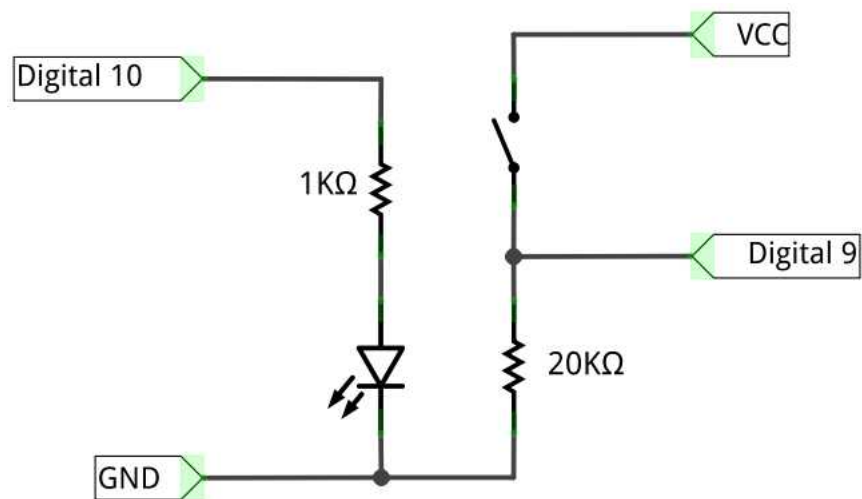


그림 4. 버튼 및 LED 연결 회로도

스케치 2는 버튼의 상태를 읽고 이를 LED로 출력하는 스케치의 예이다. 스케치 2의 loop 함수에는 디바운스를 위해 delay 함수가 포함되어 있다.

스케치 2. 버튼 상태를 LED로 나타내기

```
int button = 9;           // 9번 핀에 버튼 연결
int led = 10;             // 10번 핀에 LED 연결

void setup() {
  pinMode(button, INPUT);  // 9번 핀을 입력 모드로 설정
  pinMode(led, OUTPUT);    // 10번 핀을 출력 모드로 설정
}
```

```

void loop() {
    boolean value = digitalRead(button);    // 버튼 상태 읽기

    if(value){                              // 버튼이 눌려진 경우
        digitalWrite(led, HIGH);
    }
    else{                                    // 버튼이 눌러지지 않은 경우
        digitalWrite(led, LOW);
    }

    delay(100);                             // 디바운스
}

```

3. 스피커 출력

아두이노 우노로 아날로그 신호 출력은 불가능하지만 펄스 폭 변조 (PWM) 신호를 사용하여 아날로그 신호와 유사한 효과를 얻을 수 있으며 tone 함수를 통해 단음을 재생하는 것이 PWM 출력을 사용하는 예 중 하나이다. tone 함수는 50% 듀티 사이클¹⁾과 지정된 주파수를 가지는 구형파(square wave)를 사용하여 단음을 재생하며 피에조(piezo) 스피커나 다이내믹 스피커를 연결하여 재생되는 소리를 들을 수 있다.

```

void tone(uint8_t pin, unsigned int frequency, unsigned long duration = 0)
    pin : 핀 번호
    frequency : 출력 주파수 (unsigned int)
    duration : 출력 지속시간, 밀리초 단위 (unsigned long)
    반환값 : 없음

```

톤을 재생할 때 특정 시간에는 하나의 톤만을 재생할 수 있다. 다른 핀을 통해 이미 톤이 재생 중이면 tone 함수를 호출하여도 아무런 소리를 들을 수 없으며 동일한 핀을 통해 tone 함수를 다시 호출하면 새롭게 지정된 주파수의 톤이 재생된다.

tone 함수에는 지속시간을 지정할 수 있으며 지속시간이 지정되지 않으면 디폴트값인 0이 사

1) PWM 신호의 듀티 사이클은 다음 장에서 자세히 다룬다.

용되어 noTone 함수가 호출될 때까지 음이 재생된다. noTone 함수는 tone 함수에 의한 구형파 출력을 멈추어 음의 재생을 멈춘다.

```
void noTone(uint8_t pin)
    pin : 핀 번호
    반환값 : 없음
```

tone 함수를 사용하여 음을 재생하기 위해서는 각 음높이에 해당하는 주파수를 알고 있어야 한다. 각 음 높이에 해당하는 주파수는 부록 3의 'pitches.h'에 정의되어 있으며 인터넷에서 다운 받을 수 있다.²⁾ 다운받은 pitches.h 파일은 스케치북 디렉터리 아래 'libraries' 디렉터리에 'pitches' 이름의 디렉터를 만들고 그 아래에 복사해 놓는다. 스케치북 디렉터리는 '파일 → 환경 설정' 메뉴에서 확인할 수 있다.

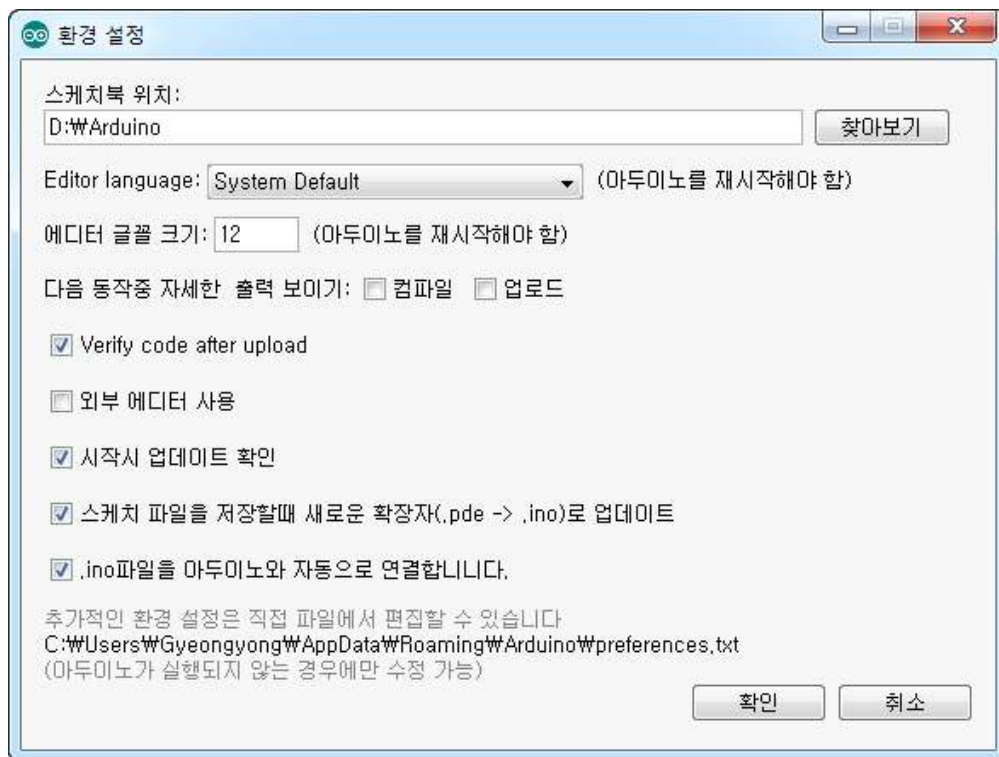


그림 5. 스케치북 디렉터리

2)

<http://code.google.com/p/arduino/source/browse/trunk/build/shared/examples/Digital/toneMultiple/pitches.h?r=943>

그림 5와 같이 스케치북 디렉터리가 'D:\Arduino'로 설정된 경우 pitches.h 파일은 'D:\Arduino\libraries\pitches' 디렉터리에 복사해 놓는다. 파일 복사가 완료되면 아두이노 프로그램에서 '스케치 → 라이브러리 가져오기...' 메뉴에서 'pitches' 항목을 확인할 수 있다. pitches 항목을 선택하면 스케치에 자동으로 해당 라이브러리를 사용하기 위해 필요한 헤더 파일이 포함된다. pitches 라이브러리의 경우 '#include <pitches.h>'가 추가되는 것을 확인할 수 있다.

라이브러리 설치가 완료되었으면 그림 6과 같이 스피커를 연결한다. DM-UStyle EX 보드에는 스피커가 포함되어 있다.

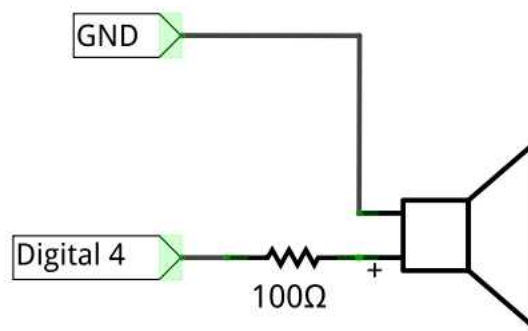


그림 6. 스피커 연결 회로도

먼저 '스케치 → 라이브러리 가져오기... → pitches' 메뉴를 선택하여 'pitches.h' 파일을 포함시키고 스케치 3을 업로드 하여 실행시켜 보자. 익숙한 멜로디가 들리지 않는가?

스케치 3. tone 함수를 이용한 멜로디 재생

```
#include <pitches.h>

int speakerPin = 4;           // 스피커 연결 핀

int melody[] = {              // 학교 종 멜로디. 0은 침표
    NOTE_G4, NOTE_G4, NOTE_A4, NOTE_A4, NOTE_G4, NOTE_G4, NOTE_E4, 0,
    NOTE_G4, NOTE_G4, NOTE_E4, NOTE_E4, NOTE_D4, 0,
    NOTE_G4, NOTE_G4, NOTE_A4, NOTE_A4, NOTE_G4, NOTE_G4, NOTE_E4, 0,
    NOTE_G4, NOTE_E4, NOTE_D4, NOTE_E4, NOTE_C4, 0
}
```

```
};

int noteDuration = 4;           // 4분 음표

void setup() {
}

void loop() {
    for (int thisNote = 0; thisNote < sizeof(melody) / sizeof(int); thisNote++) {
        int noteLength = 1000 / noteDuration;           // 음표 길이를 시간으로 변환
        tone(speakerPin, melody[thisNote], noteLength); // 멜로디 재생

        delay(noteLength);
        noTone(speakerPin);           // 재생 중지
    }

    while(true);           // 1회 재생 후 대기
}
```

4. millis 함수

Blink 예제에서 점멸 시간 조절을 위해 delay 함수를 사용하였다. delay 함수는 지정한 시간만큼 프로그램의 실행을 일시 중지시키는 함수이다.

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

delay 함수와 비슷하면서도 보다 정밀한 지연 시간을 지정할 수 있는 함수가 delayMicrosecond 함수로 마이크로초 단위로 지연 시간을 지정할 수 있다.

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

delay 함수가 사용하면 간편하게 시간 지연을 설정할 수 있지만 delay 함수가 실행 중인 동안 다른 대부분의 작업 역시 진행될 수 없다. 즉, delay 함수로 1초 지연을 하는 경우 1초 동안은 센서의 값을 입력받을 수 없으며 출력 역시 낼 수 없다. 따라서 입력 즉시 즉각적인 반응을 요하는 등 타이밍이 중요한 스케치에서는 delay 함수를 사용하지 않는 것이 좋다.

delay 함수를 대신하여 시간 조절을 위해 사용할 수 있는 함수로 millis 함수와 micros 함수가 있다. millis 함수는 현재 실행 중인 프로그램이 시작된 이후의 경과 시간을 밀리초 단위로 반환한다. millis 함수의 반환값은 unsigned long 형이므로 2^{32} 밀리초 $\approx$ 50일의 시간이 경과하면 오버플로우(overflow)로 인해 다시 0이 된다.

```
unsigned long millis(void)
    반환값 : 프로그램이 시작된 이후의 밀리초(millisecond) 단위 경과 시간
```

micros 함수는 millis 함수와 유사하지만 마이크로초 단위의 시간을 반환한다. 2^{32} 마이크로초 $\approx$ 70분의 시간이 경과하면 오버플로우가 발생하고 다시 0이 된다.

```
unsigned long micros(void)
    반환값 : 프로그램이 시작된 이후의 마이크로초(microsecond) 단위 경과 시간
```

스케치 4는 millis 함수를 이용하여 Blink 스케치를 수정한 예이다. delay 함수를 사용하지 않고도 디지털 10번 핀에 연결된 LED가 1초 간격으로 점멸하는 것을 확인할 수 있다.

스케치 4. millis 함수를 이용한 Blink

```
int led = 10; // 10번 핀에 LED 연결
```

```
unsigned long time_previous, time_current;      // 이전 및 현재까지의 실행 시간
boolean state;                                  // LED 점멸 상태

void setup() {
  pinMode(led, OUTPUT);                        // 10번 핀을 출력 모드로 설정
  time_previous = millis();                    // 이전 LED 상태 변경 시간
  state = false;                               // 현재 LED 상태
}

void loop() {
  time_current = millis();                     // 현재까지의 시간

  // LED 상태를 변경한 시간이 1000밀리초 이상 경과한 경우
  if(time_current - time_previous > 1000){
    time_previous = time_current;              // LED 상태 변경 시간 저장
    state = !state;                            // LED 상태 반전

    digitalWrite(led, state);                  // LED 점멸
  }
}
```

스케치 4와 같이 간단한 스케치의 경우에는 굳이 millis 함수를 사용하지 않아도 된다. 하지만 delay 함수가 실행 중인 동안 버튼 입력을 받아야 하는 경우를 생각해 보자. 그림 7과 같이 두 개의 LED와 하나의 스위치를 연결한다. DM-UStyle EX 보드에는 추가로 연결해야 하는 부품은 없다. 디지털 10번 핀에 연결된 LED는 1초 간격으로 점멸하고 디지털 13번 핀에 연결된 LED는 버튼이 눌러진 경우에만 켜지도록 스케치를 작성해 보자.

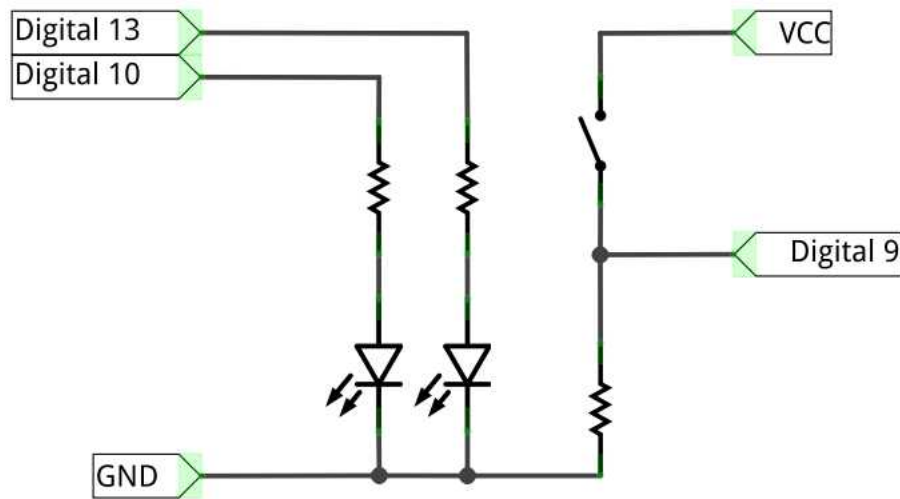


그림 7. LED 두 개와 버튼 연결 회로도

스케치 5는 그림 7의 회로도를 테스트하기 위해 delay 함수를 이용하여 작성한 스케치의 예이다.

스케치 5. delay 함수를 이용한 LED 제어

```
// 10번 핀에 LED 연결. 1초 간격으로 LED 점멸
int led1 = 10;
// 13번 핀에 LED 연결. 버튼 상태 표시
int led2 = 13;
int button = 9; // 9번 핀에 버튼 연결
int led_state; // 10번 핀 LED 상태

void setup() {
  pinMode(led1, OUTPUT); // 10번 핀을 출력 모드로 설정
  pinMode(led2, OUTPUT); // 13번 핀을 출력 모드로 설정
  pinMode(button, INPUT); // 9번 핀을 입력 모드로 설정
  led_state = false; // 1초 간격으로 점멸
}

void loop() {
  boolean b_state = digitalRead(button); // 버튼 상태 검사
  digitalWrite(led2, b_state); // 13번 핀으로 버튼 상태 출력
```

```

    led_state = !led_state;           // 10번 핀 LED 반전
    digitalWrite(led1, led_state);

    delay(1000);                      // 1초 대기
}

```

스케치 5를 업로드 하고 디지털 9번 핀에 연결된 버튼을 눌러보자. 디지털 10번 핀에 연결된 LED는 1초 간격으로 점멸한다. 하지만 디지털 13번 핀에 연결된 LED는 버튼의 상태를 즉각적으로 나타내주지 못한다. 이는 delay 함수가 실행 중인 도중 버튼을 눌렀다 때는 경우 버튼 상태의 변화를 검사할 수 없기 때문이다. delay 함수의 지연 시간을 더 길게 설정하면 13번 핀에 연결된 LED 오동작을 보다 명확하게 알아차릴 수 있다.

스케치 5에서 delay 함수를 millis 함수도 대체하여 버튼 상태 변화를 즉각적으로 반영할 수 있도록 수정해 보자. 이를 위해서는 스케치 4에서와 같이 이전 상태 변화 시간을 저장하는 변수를 두고 현재 시간과 비교하여 지정한 시간 경과를 계속적으로 검사하여 비교하여야 한다. 스케치 6은 스케치 5를 millis 함수를 이용하여 수정한 예이다. 스케치 5와 달리 스케치 6은 디지털 10번 핀에 연결된 LED가 1초 간격으로 점멸함과 동시에 디지털 13번 핀에 연결된 LED로 버튼의 상태를 즉각적으로 나타내 줌을 확인할 수 있다.

스케치 6. millis 함수를 이용한 LED 제어

```

// 10번 핀에 LED 연결. 1초 간격으로 LED 점멸
int led1 = 10;
// 13번 핀에 LED 연결. 버튼 상태 표시
int led2 = 13;
int button = 9;           // 9번 핀에 버튼 연결
int led_state;            // 10번 핀 LED 상태

unsigned long time_previous; // 10번 핀 LED 상태 변경 시간
unsigned long time_current;  // 현재 시간

void setup() {
    pinMode(led1, OUTPUT);    // 10번 핀을 출력 모드로 설정
}

```

```

pinMode(led2, OUTPUT);           // 13번 핀을 출력 모드로 설정
pinMode(button, INPUT);         // 9번 핀을 입력 모드로 설정
led_state = false;               // 1초 간격으로 점멸

time_previous = millis();
}

void loop() {
  boolean b_state = digitalRead(button); // 버튼 상태 검사
  digitalWrite(led2, b_state);           // 13번 핀으로 버튼 상태 출력

  time_current = millis();               // 현재 시간
  if(time_current - time_previous > 1000){ // 이전 상태 변경 이후 1초 경과
    time_previous = time_current;         // 이전 상태 변경 시간 갱신
    led_state = !led_state;               // 10번 핀 LED 반전
    digitalWrite(led1, led_state);
  }
}

```

5. Blink 속도 조절

Blink 예제에서는 LED의 점멸 속도가 고정되어 있다. millis 함수를 이용하여 스케치 실행 중간에 LED의 점멸 속도를 바꾸는 스케치를 작성해 보자. 먼저 그림 4와 같이 회로를 구성한다. LED 점멸 시간은 0.5초와 1초 두 가지로 하고 버튼이 눌리지면 두 가지 속도를 번갈아 선택하도록 해 보자. 스케치 7은 LED의 점멸 속도 조절을 위한 스케치의 예이다. 버튼이 눌려진 경우 디바운스를 위해 delay 함수를 사용하였으며 점멸 시간이 바뀌는 경우 시리얼 모니터로 그 결과를 출력하도록 하였다..

스케치 7. 점멸 속도 변경 Blink I

```

int led = 10;           // 10번 핀에 버튼 연결
int button = 9;         // 9번 핀에 버튼 연결
int led_state;          // LED 점멸 상태
unsigned long interval = 500; // 점멸 간격 (밀리초)

```

```
unsigned long time_previous;
unsigned long time_current;

void setup() {
    Serial.begin(9600);
    pinMode(led, OUTPUT);           // 10번 핀을 출력 모드로 설정
    pinMode(button, INPUT);         // 9번 핀을 입력 모드로 설정
    led_state = false;              // LED 점멸 상태

    time_previous = millis();
}

void loop() {
    boolean b_state = digitalRead(button); // 버튼 상태 검사
    // 버튼이 눌러진 경우 0.5초와 1초 간격으로 번갈아 설정
    if(b_state){
        if(interval == 500){
            interval = 1000;
        }
        else{
            interval = 500;
        }
        delay(500);                 // 디바운스
        Serial.println("Blink speed : " + String(interval)); // 현재 상태 출력
    }

    time_current = millis();
    if(time_current - time_previous > 1000){ // 이전 상태 변경 이후 1초 경과
        time_previous = time_current;       // 이전 상태 변경 시간 갱신
        led_state = !led_state;              // 10번 핀 LED 반전
        digitalWrite(led, led_state);
    }
}
```

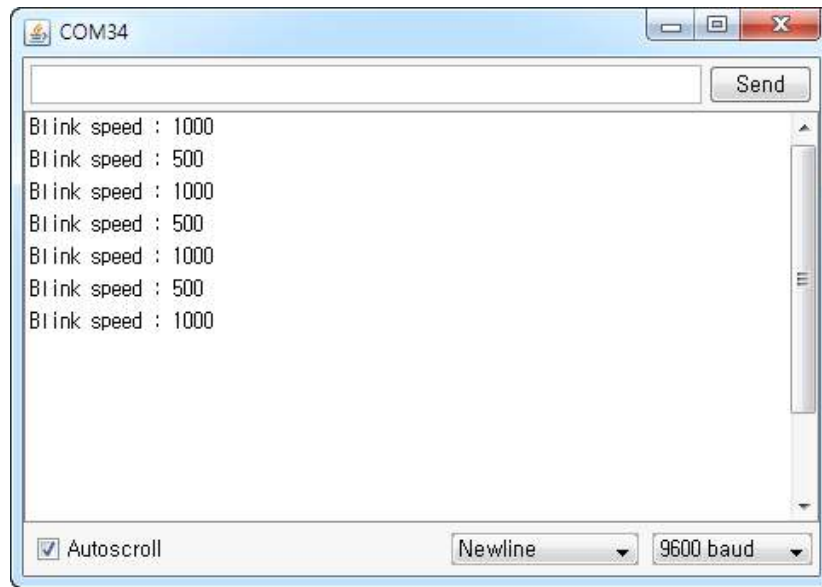


그림 8. 스케치 7 실행 결과

스케치 7에서 초기값은 0.5초 간격으로 LED가 점멸하도록 설정되어 있다. 스케치 7의 경우 점멸 시간을 두 가지를 번갈아 사용하도록 되어 있지만 점멸 시간을 세 가지 이상으로 설정할 수도 있다. 스케치 8은 세 가지 이상의 점멸 시간을 설정하고 선택할 수 있도록 코드 7을 수정한 예이다. 단, 스케치 8의 동작은 스케치 7과 동일하다.

스케치 8. 점멸 속도 변경 Blink II

```
int led = 10;                // 10번 핀에 버튼 연결
int button = 9;              // 9번 핀에 버튼 연결
int led_state;               // LED 점멸 상태
unsigned long intervals[] = {500, 1000}; // 점멸 간격 (밀리초)
unsigned long interval = 500;
int index = 0;

unsigned long time_previous;
unsigned long time_current;

void setup() {
  Serial.begin(9600);
  pinMode(led, OUTPUT);      // 10번 핀을 출력 모드로 설정
```

```
pinMode(button, INPUT);           // 9번 핀을 입력 모드로 설정
led_state = false;                 // 1초 간격으로 점멸

time_previous = millis();
}

void loop() {
  boolean b_state = digitalRead(button);           // 버튼 상태 검사
  // 버튼이 눌러진 경우 모드 변경
  if(b_state){
    // intervals에 지정된 시간 간격의 개수를 자동으로 계산하여
    // 현재 점멸 시간 간격을 선택
    index = (index + 1) % (sizeof(intervals) / sizeof(unsigned long));

    interval = intervals[index];
    delay(500);                               // 디바운스
    Serial.println("Mode " + String(index) + " : " + interval);
  }

  time_current = millis();
  if(time_current - time_previous > interval){    // 이전 상태 변경 이후 1초 경과
    time_previous = time_current;                // 이전 상태 변경 시간 갱신
    led_state = !led_state;                      // 10번 핀 LED 반전
    digitalWrite(led, led_state);
  }
}
```

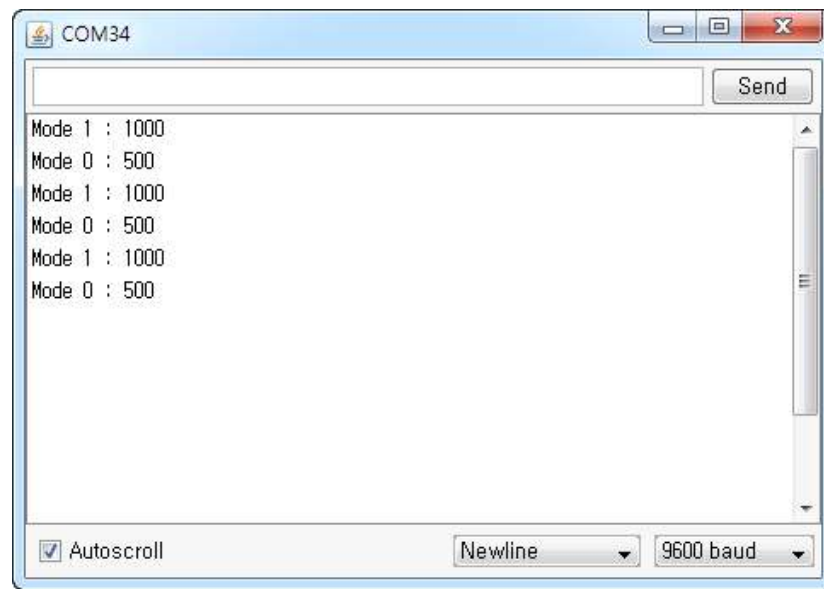


그림 9. 스케치 8 실행 결과