

4. 디지털 입력

마이크로컨트롤러는 주변 환경으로부터 다양한 정보를 획득하고 이를 처리하여 그 결과를 출력하기 위해 사용된다. 이 때 마이크로컨트롤러가 받아들일 수 있는 데이터는 아날로그 및 디지털 형식이 가능하다. 하지만 마이크로컨트롤러가 처리할 수 있는 데이터는 디지털 데이터이므로 아날로그 데이터를 입력받는 경우 이를 아날로그-디지털 변환기(Analog-to-Digital Converter, ADC)를 거쳐 디지털 형식으로 변환하여 받아들인다. 아두이노 우노에 사용되는 ATmega328 마이크로컨트롤러 역시 ADC가 포함되어 있다. 출력 데이터 역시 아날로그 및 디지털 형식이 가능하지만 디지털-아날로그 변환기(Digital-to-Analog Converter, DAC)는 일부 마이크로컨트롤러에만 포함되어 있다. 아두이노 우노에 사용되는 ATmega328 마이크로컨트롤러의 경우 DAC는 포함되어 있지 않으므로 아날로그 데이터 출력은 불가능하지만 Arduino Due 보드에 사용된 AT91SAM3X8E 마이크로컨트롤러의 경우에는 DAC를 포함하고 있어 아날로그 데이터 출력이 가능하다. 아날로그 데이터 출력이 불가능한 경우에도 디지털 데이터의 일종인 펄스 폭 변조(Pulse Width Modulation, PWM) 신호를 사용함으로써 아날로그 출력과 유사한 효과를 얻을 수 있다. 표 1은 아두이노 우노에서 기본이 되는 디지털 및 아날로그 입출력 함수를 요약한 것이다.

	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

아두이노 우노에서 아날로그 출력은 불가능하지만 PWM 데이터를 출력하는 analogWrite 함수가 아날로그 데이터를 출력한 경우와 유사한 효과를 내므로 아날로그 출력으로 분류하였다.

이 장에서는 표 1에서 디지털 데이터를 입력하는 방법에 대해 알아본다.

1. 디지털 데이터 입력

아두이노 우노에는 14개의 디지털 입출력 핀과 6개의 아날로그 입력 핀이 준비되어 있다. 디지털 핀의 경우 0번에서 13번까지 번호가 지정되어 있고 아날로그 핀의 경우 A0에서 A5까지 상수가 정의되어 있다. 또한 아날로그 핀은 디지털 핀 이후 14번에서 19번까지의 번호를 사용할 수도 있다. 아날로그 핀은 ADC와 연결되어 있어 입력되는 아날로그 값을 디지털로 변환하여 읽어 들이기 위해 사용되지만 디지털 데이터의 입출력을 위해서 디지털 핀과 동일하게 사용하는 것도 가능하다. 따라서 아두이노 우노에서 사용 가능한 디지털 입출력 핀의 개수는 최대 20개까지 가능하다.

디지털 입출력 핀은 입력과 출력으로 모두 사용이 가능하지만 입력과 출력으로 동시에 사용할 수는 없다. 따라서 핀을 통해 데이터를 입출력하기 이전에는 핀을 입력 모드로 사용할 것인지 출력 모드로 사용할 것인지 지정하여야 하며 핀의 모드 지정을 위해서는 pinMode 함수를 사용하면 된다. pinMode 함수에서 지정할 수 있는 핀의 모드 중에는 INPUT_PULLUP 모드가 있다. ATmega328 마이크로컨트롤러의 입력 핀에는 내장 풀업 저항이 포함되어 있어서 풀업 저항이 필요한 경우 사용할 수 있다.

```
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 상태를 가진다. 따라서 입력으로 사용하기 위해서는 명시적으로 pinMode 함수를 사용하여 입출력 상태를 설정하지 않아도 입력으로 사용이 가능하다. 하지만 혼돈을 피하기 위해 입력으로 사용하는 경우에도 명시적으로 pinMode 함수를 사용하여 입력 상태로 설정한 이후 사용할 것을 추천한다.

핀의 모드가 설정되면 비트 단위의 디지털 데이터 입력을 위해 digitalRead 함수를 사용한다. digitalRead 함수는 0이나 1의 디지털 값을 미리 정의된 상수인 LOW 또는 HIGH로 반환한다.

```
int digitalRead(uint8_t pin)
    pin : 핀 번호
    반환값 : HIGH 또는 LOW
```

2. 풀업 및 풀다운 저항

입력 상태로 설정된 핀은 하이 임피던스 상태(high impedance state)에 있다고 말한다. 디지털 논리 회로에서는 일반적으로 5V(Vcc)는 1로 0V(Ground, GND)는 0으로 간주한다. 하이 임피던스 상태는 단어 의미 그대로 아주 큰 저항이 연결된 상태로 아주 적은 전류가 흐르는 상태를 말한다. 하이 임피던스 상태에 있는 핀은 1이나 0의 상태가 아닌 제 3의 상태로 플로팅(floating)되어 있어 외부 잡음이나 이웃한 핀의 상태에 따라 의미 없는 값을 반환할 수 있다. 이처럼 핀에 외부 회로가 연결되지 않은 경우 하이 임피던스 상태가 되어 입력값을 결정할 수 없는 경우를 방지하기 위해 (Vcc로) 풀업 저항 또는 (GND로) 풀다운 저항을 연결하여 1이나 0의 상태로 설정하여 사용하는 것이 일반적이다. 풀업이나 풀다운 저항은 일반적으로 수십 키로옴(k Ω)의 저항을 사용한다. 그림 1은 풀업 저항을 연결한 예를 보인 것이다. 그림 1-(a)의 경우 스위치가 눌러지면 0V 입력이 가해지지만 스위치가 눌러지지 않은 경우 하이 임피던스 상태가 되어 그 값을 예측할 수 없다. 그림 1-(b)와 같이 풀업 저항을 연결하면 스위치가 눌러지면 0V 입력이 가해지는 것은 동일하지만 스위치가 눌러지지 않은 경우에는 Vcc가 입력되어 논리 1이 보장된다.

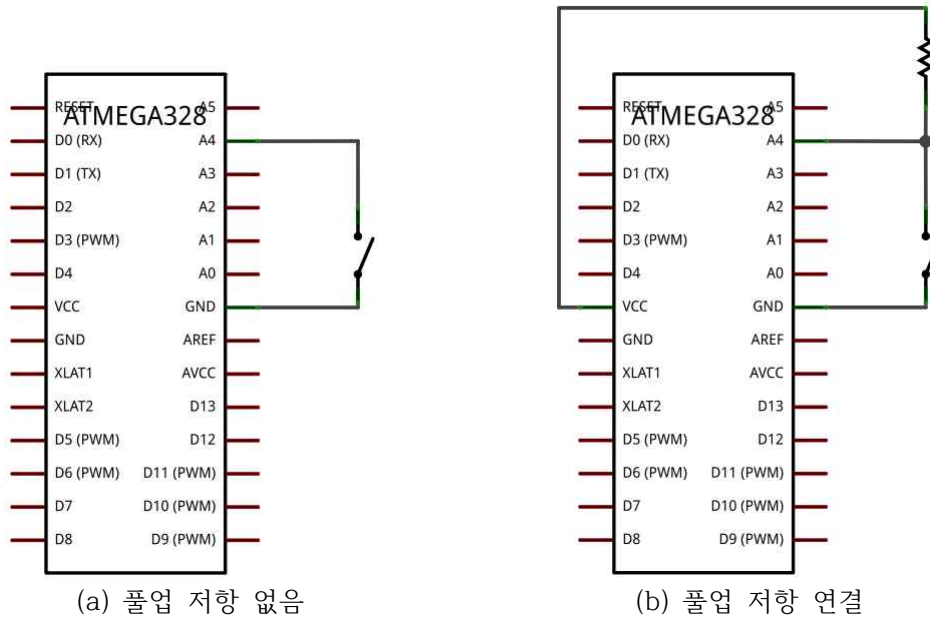


그림 1. 풀업 저항

그림	ON	OFF
1-(a)	0V (low)	floating
1-(b)	0V (low)	Vcc (high)

표 2. 풀업 저항에 의한 입력 상태

그림 2는 풀다운 저항을 연결한 예이다. 풀업 저항의 경우와 달리 스위치가 눌려진 경우 Vcc 입력이 가해지며 스위치가 눌러지지 않은 경우 0V 입력을 보장하기 위해 풀다운 저항이 사용된다.

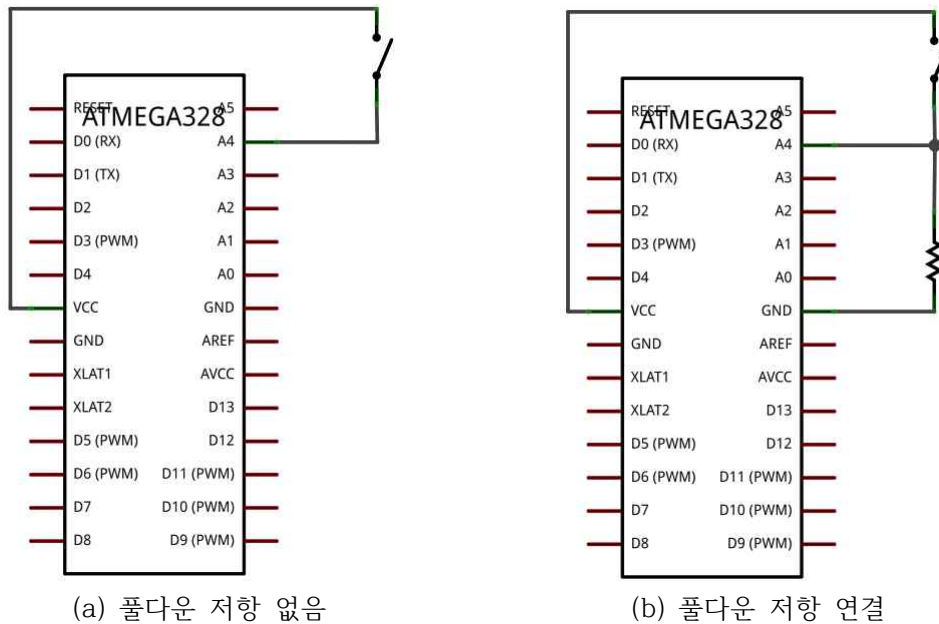


그림 2. 풀다운 저항

그림	ON	OFF
2-(a)	5V (high)	floating
2-(b)	5V (high)	0V (low)

표 3. 풀다운 저항에 의한 입력 상태

그림 3과 같이 디지털 9번 핀에 20 k Ω 풀다운 저항을 사용하여 버튼을 연결 해 보자. DM-UStyle EX 보드에는 디지털 9번 핀에 풀다운 저항을 통해 연결된 버튼과 아날로그 A0 핀에 풀업 저항을 통해 연결된 버튼이 포함되어 있다.

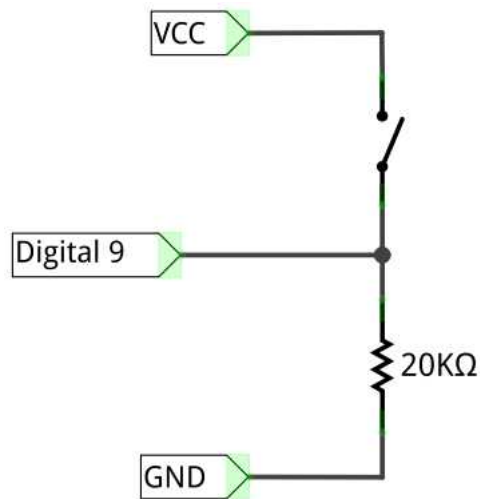


그림 3. 풀다운 저항 연결

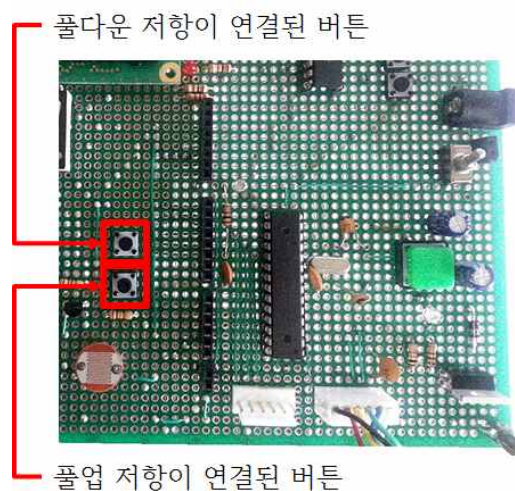


그림 4. DM-UStyle EX 보드의 풀다운 저항이 연결된 버튼

스케치 1은 버튼 입력 상태를 검사하여 해당 메시지를 시리얼 모니터로 출력하는 스케치이다. 버튼이 눌려진 경우에는 HIGH 값이 입력되며 버튼이 눌러지지 않은 경우에는 LOW 값이 입력된다. delay 함수는 밀리초 단위로 지정한 시간만큼 대기하는 함수로 스케치 1의 경우 약 1초에 한 번 버튼의 상태를 검사한다.

스케치 1. 풀다운 저항이 연결된 버튼

```
int buttonPin = 9;
```

```
void setup() {  
  Serial.begin(9600);  
  pinMode(buttonPin, INPUT);          // 9번 핀을 입력으로 사용  
}  
  
void loop() {  
  boolean value = digitalRead(buttonPin); // 버튼 상태 읽기  
  
  if(value == HIGH){                  // 버튼이 눌려진 경우  
    Serial.println("Button is ON.");  
  }  
  else{                               // 버튼이 눌려지지 않은 경우  
    Serial.println("Button is OFF.");  
  }  
  
  delay(1000);                        // 1초 대기  
}
```

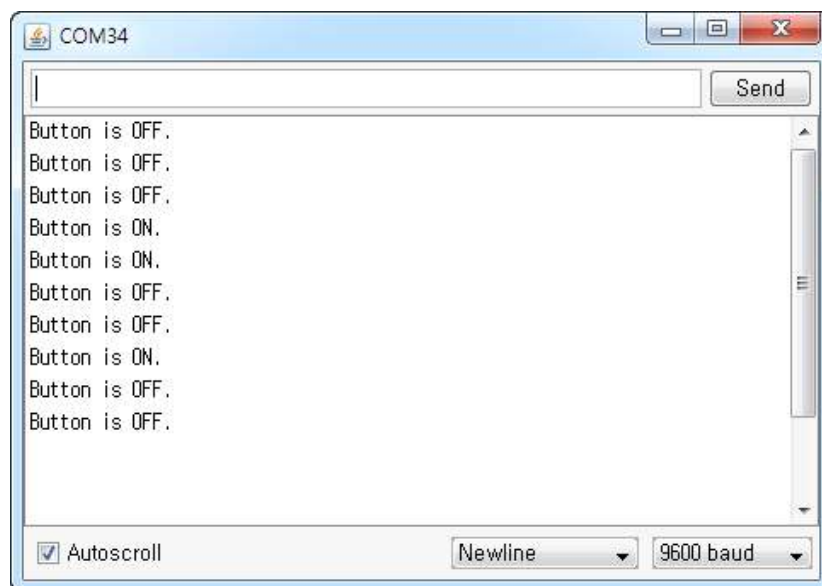


그림 5. 스케치 1 실행 결과

그림 6과 같이 풀업 저항을 사용하여 아날로그 0번 A0 핀에 버튼을 연결해 보자.

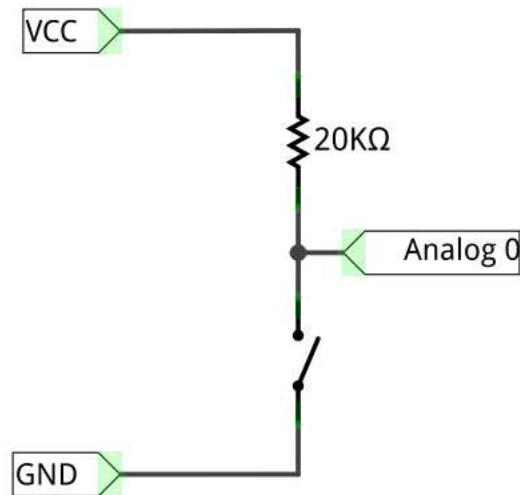


그림 6. 풀업 저항 연결

DM-UStyle EX 보드에서 A0에 연결된 버튼은 그림 4에서 확인할 수 있다. 스케치 2는 풀업 저항이 연결된 버튼으로부터 입력을 받아 그 결과를 출력하는 스케치로 스케치 1과 비교하였을 때 버튼이 연결된 핀의 번호가 바뀌었다. 풀업 저항이 연결되어 있는 경우 버튼이 눌려진 경우 0의 값이 입력되므로 시리얼 모니터로 출력하는 메시지가 스케치 1과는 반대로 설정되어 있음에 유의하여야 한다. 마지막 한 가지 유의하여야 할 점은 버튼이 아날로그 입력 핀에 연결되어 있다는 점이다. 아날로그 입력 핀은 아날로그 입력을 위해 사용되지만 디지털 입출력 핀으로도 사용이 가능하다고 앞에서 설명하였으며 디지털 입출력 핀을 사용하는 것과 동일한 방법으로 사용이 가능하다. 실행 결과는 그림 5와 동일하다.

스케치 2. 풀업 저항이 연결된 버튼

```
int buttonPin = A0;

void setup() {
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);          // A0번 핀을 입력으로 사용
}

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

```

if(value == LOW){                                // 버튼이 눌러진 경우
    Serial.println("Button is ON.");
}
else{                                             // 버튼이 눌러지지 않은 경우
    Serial.println("Button is OFF.");
}

delay(1000);                                    // 1초 대기
}

```

3. 내장 풀업 저항

아두이노의 입출력 핀에는 내장 풀업 저항이 연결되어 있으며 이를 사용하기 위해서는 pinMode 함수에서 INPUT_PULLUP 모드로 설정함으로써 가능하다. 그림 7과 같이 저항 없이 버튼만을 아날로그 A0번 핀에 연결해 보자. DM-UStyle EX 보드에도 내장 풀업 저항 테스트를 위한 버튼은 포함되어 있지 않으므로 핀 헤더를 통해 버튼을 연결하여야 한다.

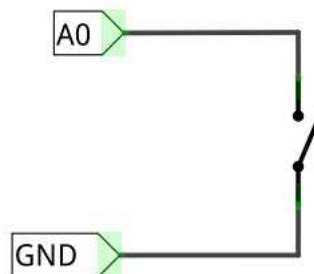


그림 7. 내장 풀업 저항 사용을 위한 버튼 연결 회로도

스케치 3은 내장 풀업 저항을 테스트하기 위한 스케치이며 실행 결과는 그림 5와 같다.

스케치 3. 내장 풀업 저항이 연결된 버튼

```
int buttonPin = A0;
```

```

void setup() {
  Serial.begin(9600);
  pinMode(buttonPin, INPUT_PULLUP);    // A0번 핀을 입력으로 사용
}

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

  if(value == LOW){                      // 버튼이 눌려진 경우
    Serial.println("Button is ON.");
  }
  else{                                  // 버튼이 눌려지지 않은 경우
    Serial.println("Button is OFF.");
  }

  delay(1000);                           // 1초 대기
}

```

스케치 3에서는 setup 함수에서 pinMode 함수를 통해 내장 풀업 저항을 사용하도록 설정하였다. 필요한 경우 프로그램 실행 중에도 내장 풀업 저항의 사용 여부를 설정할 수 있다. 내장 풀업 저항을 사용하기 위해서는 먼저 핀의 상태를 입력 상태로 설정한 상태에서 digitalWrite 함수를 사용하여 HIGH 값을 출력하면 된다. LOW 값을 출력하면 풀업 저항 사용은 해제된다.

```

pinMode(pin, INPUT);           // 핀을 입력 상태로 설정
digitalWrite(pin, HIGH);       // 풀업 저항 연결

```

4. 여러 개 버튼 입력

그림 8은 여러 개의 버튼 입력을 위한 버튼 모듈의 회로도이다. 버튼 모듈은 4개의 풀다운 저항이 연결된 버튼과 4개의 풀업 저항이 연결된 버튼으로 구성된다.

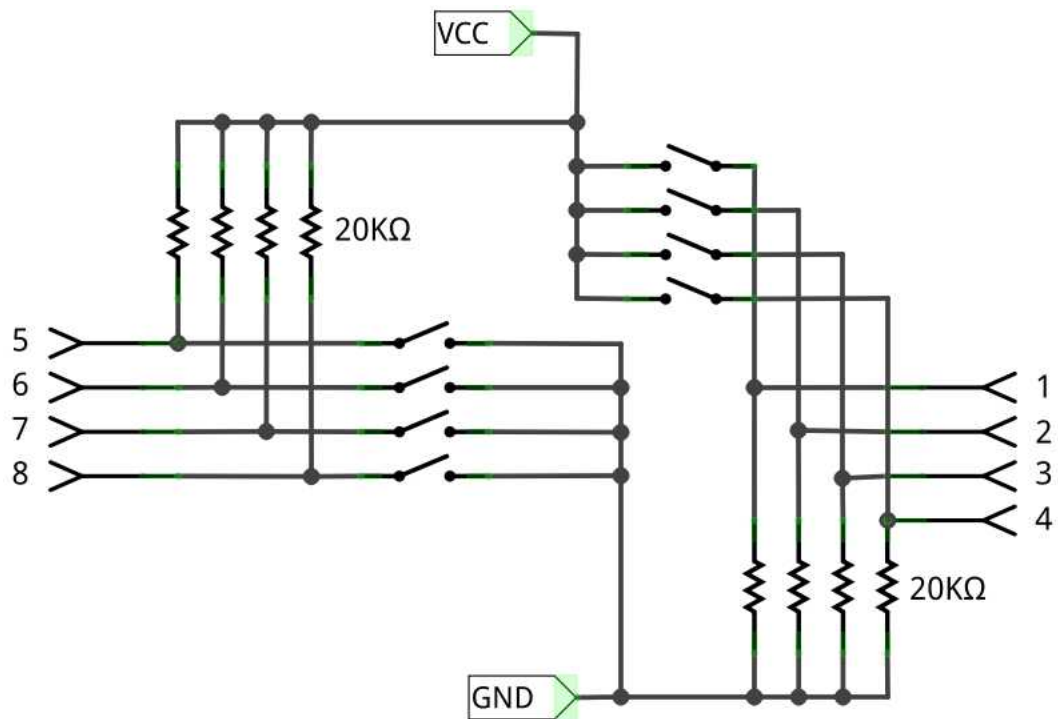


그림 8. 버튼 모듈 회로도

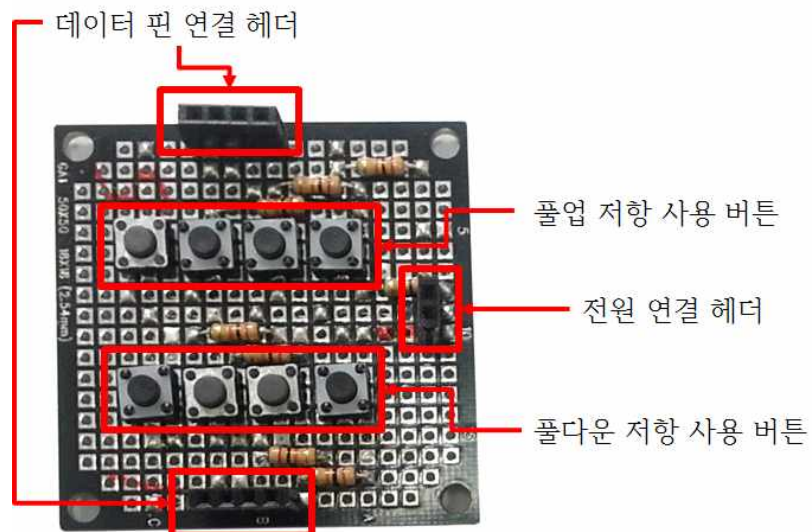


그림 9. 버튼 모듈

버튼 모듈을 디지털 2, 3, 4, 5, 6, 7, 8, 11번에 연결한다. DM-UStyle EX 보드에서 디지털 9번 핀에는 버튼이 풀다운 저항을 통해 연결되어 있고 10번 핀 역시 LED가 저항을 통해 연결되어 있으므로 버튼 모듈을 연결하는 경우 정상적으로 동작하지 않는다.

스케치 4는 8개 버튼의 상태를 읽어서 HIGH 입력이 가해지는 경우 'O'를, LOW 입력이 가해지는 경우 'X'를 출력하도록 한다. 1에서 4번 버튼은 풀다운 저항을 사용하고 있고 5에서 8번 버튼은 풀업 저항을 사용하고 있으므로 버튼을 누르지 않은 상태에서의 출력값은 'X X X X O O O O'가 된다.

스케치 4. 버튼 모듈

```
// 버튼 모듈 연결 핀
int pins[] = {2, 3, 4, 5, 6, 7, 8, 11};

void setup() {
    Serial.begin(9600);                // 시리얼 통신 초기화
    for(int i = 0; i < 8; i++)        // 버튼 연결 핀들을 입력으로 설정
        pinMode(pins[i], INPUT);
}

void loop() {
    for(int i = 0; i < 8; i++){
        boolean value = digitalRead(pins[i]); // 버튼 상태 읽기

        if(value){                        // 버튼이 HIGH 상태인 경우
            Serial.print("O ");
        }
        else{                             // 버튼이 LOW 상태인 경우
            Serial.print("X ");
        }
    }
    Serial.println();

    delay(1000);                        // 1초 대기
}
```

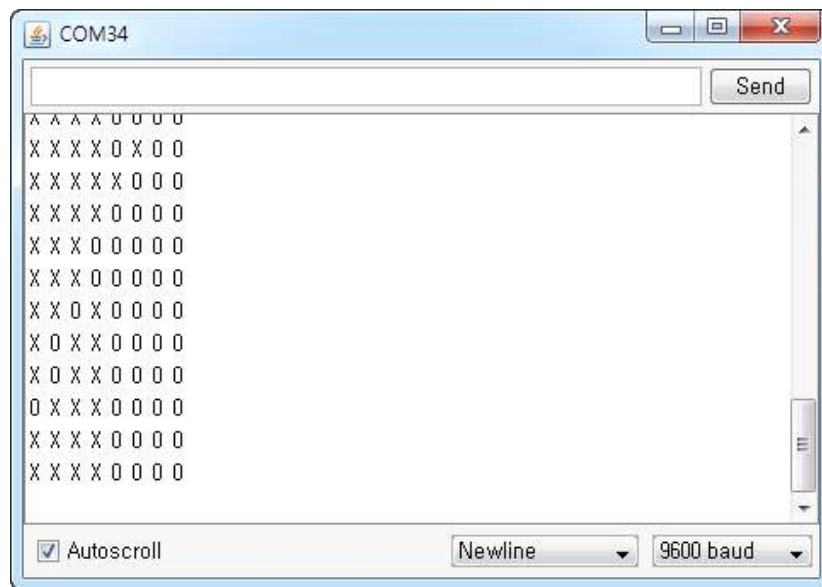


그림 10. 스케치 4 실행 결과

5. 채터링 (chattering)

지금까지의 스케치는 버튼 입력을 1초에 한 번만 검사하도록 하였다. 이전 버튼 상태와 현재 버튼 상태가 다른 경우에만 상태 변화를 출력하도록 스케치를 수정해 보자. 상태 비교를 위해서는 이전 상태를 저장하기 위해 별도의 변수가 필요하다. 스케치 5는 버튼의 상태가 바뀔 때만 출력하는 스케치의 예이다. 스케치 5의 테스트를 위해서는 그림 3의 풀다운 저항이 연결된 버튼을 사용한다.

스케치 5. 버튼 상태 변화 감지

```
int buttonPin = 9;
// 이전 버튼 상태와 현재 버튼 상태 저장을 위한 변수
boolean button_previous, button_current;

void setup() {
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);           // 9번 핀을 입력으로 사용

  // 버튼이 눌러지지 않은 상태로 초기화
  button_previous = false;
```

```
    button_current = false;
}

void loop() {
    button_current = digitalRead(buttonPin);           // 현재 버튼 상태 읽기

    if(button_current != button_previous){             // 이전 버튼 상태와 다른 경우
        button_previous = button_current;             // 이전 버튼 상태 갱신

        if(button_current == HIGH){                   // 버튼이 눌려진 경우
            Serial.println("X -> O");
        }
        else{                                          // 버튼이 눌러지지 않은 경우
            Serial.println("O -> X");
        }
    }
}
```

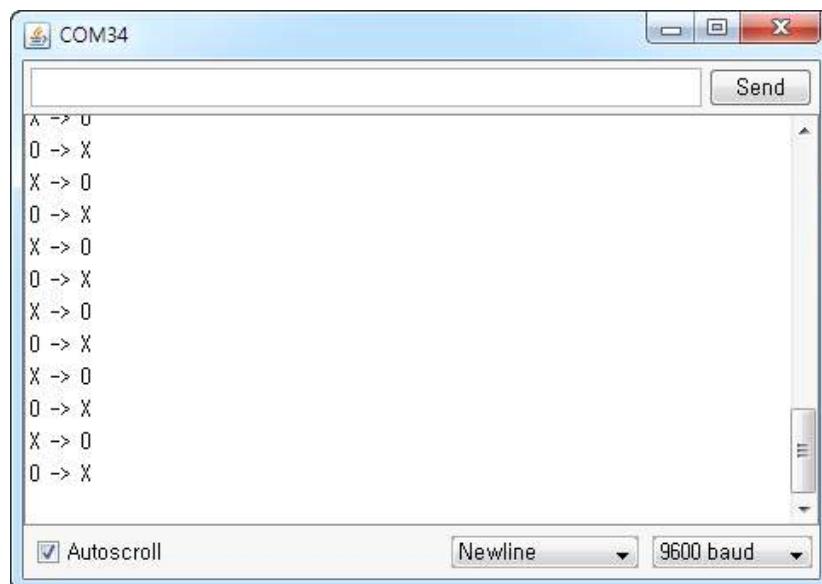


그림 11. 스케치 5 실행 결과

스케치 5를 실행시키면 버튼을 누르거나 떼는 순간에만 시리얼 모니터로 메시지가 출력되는 것을 확인할 수 있다. 하지만 계속해서 버튼을 눌렀다 떼기를 반복하다 보면 버튼을 누르거나

때는 순간에 메시지가 두 번 이상 출력되는 경우가 있음을 알 수 있다. 이는 버튼을 누르거나 떼는 때 두 개의 접점이 완전히 연결되거나 떨어지기 이전에 기계적인 진동에 의해 미세하게 연결되고 떨어지기를 반복하기 때문에 발생한다. 이러한 현상을 바운스 현상(bounce effect) 또는 채터링(chattering)이라고 하며 이를 없애는 것을 디바운스(de-bounce)라고 한다. 디바운스는 하드웨어적인 회로를 통해 구현할 수도 있으며 소프트웨어적으로도 처리가 가능하다. 디바운스를 위해 사용할 수 있는 전용 라이브러리에는 Bounce 라이브러리¹⁾가 있다. 하지만 디바운스를 위해서는 버튼의 상태 변화 사이의 최소 시간 간격을 설정함으로써도 구현할 수 있으며 이를 위해서는 dealy 함수를 사용할 수 있다. 스케치 6은 디바운스를 위해 버튼의 상태 변화를 출력한 이후 delay 함수를 추가한 것 이외에는 동일하다. 스케치 6에서 시간 지연은 100밀리초로 설정되어 있으므로 0.1초 이내에 발생하는 버튼의 상태 변화는 감지할 수 없다.

스케치 6. 디바운스 기능이 추가된 버튼 상태 변화 감지

```
int buttonPin = 9;
// 이전 버튼 상태와 현재 버튼 상태 저장을 위한 변수
boolean button_previous, button_current;

void setup() {
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);           // 9번 핀을 입력으로 사용

  // 버튼이 눌러지지 않은 상태로 초기화
  button_previous = false;
  button_current = false;
}

void loop() {
  button_current = digitalRead(buttonPin);    // 현재 버튼 상태 읽기

  if(button_current != button_previous){      // 이전 버튼 상태와 다른 경우
    button_previous = button_current;         // 이전 버튼 상태 갱신

    if(button_current == HIGH){               // 버튼이 눌러진 경우
```

1) <https://github.com/thomasfredericks/Bounce-Arduino-Wiring>

```
        Serial.println("X -> O");
    }
    else{                                     // 버튼이 눌러지지 않은 경우
        Serial.println("O -> X");
    }

    // 디바운스를 위한 시간 지연
    delay(100);
}
}
```

스케치 6의 실행 결과는 그림 11과 동일하며 버튼을 천천히 누르거나 떼는 경우에도 스케치 5에 비해 채터링 현상이 거의 발생하지 않음을 확인할 수 있다.