

Chapter 6. 디지털 데이터 출력 - LED

마이크로컨트롤러는 디지털 데이터만을 처리할 수 있다. 아날로그 데이터 역시 입력이 가능하지만 실제 처리는 아날로그-디지털 변환기(ADC)를 거쳐 디지털 형태로 변환하여야 처리가 가능하다. 아두이노 메가2560에서 사용 가능한 디지털 입출력 핀은 최대 70개이다. 디지털 데이터 출력을 위해 이 장에서는 발광 다이오드(Light Emitting Diode, LED)를 사용한다.

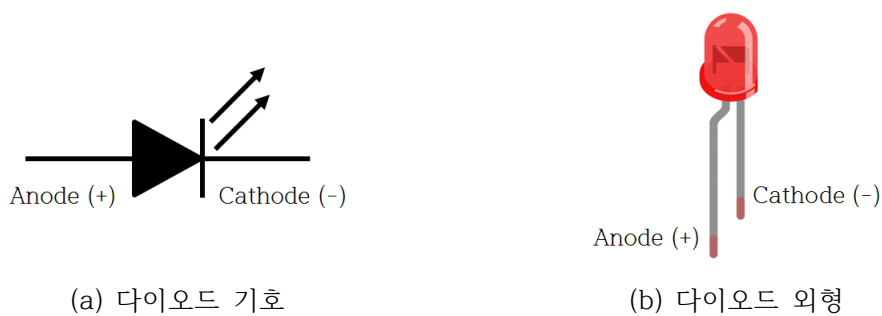


그림 1. 발광 다이오드 (LED)

1. 블링크

아두이노 메가2560과 학습보드에는 디지털 13번 핀에 연결된 LED가 포함되어 있으므로 별도의 LED 연결 없이도 디지털 데이터 출력을 실습해 볼 수 있다.

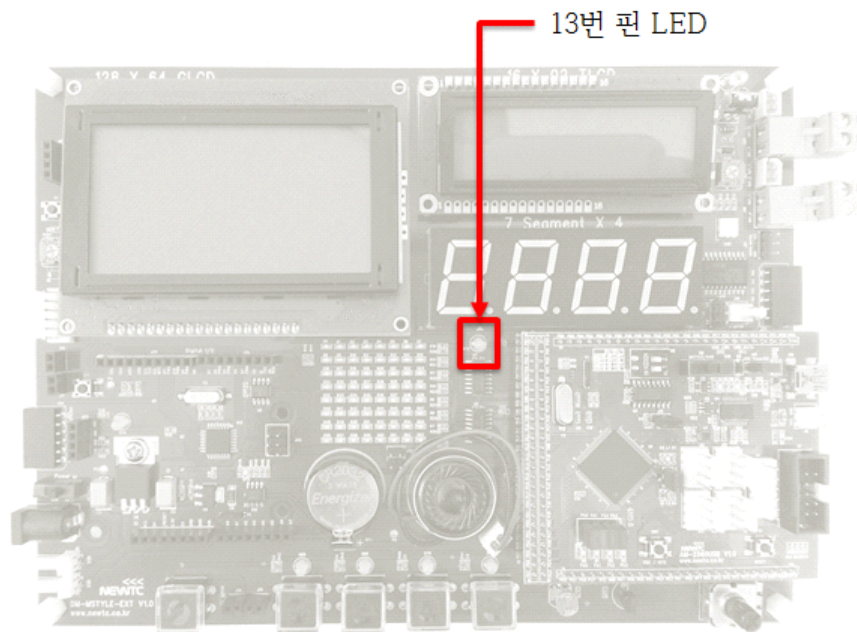


그림 2. 학습보드의 디지털 13번 핀 LED

아두이노 프로그램을 실행시키고 ‘파일 → 예제 → 01.Basics → Blink’를 선택하여 블링크 스케치를 열어보자. 스케치 1은 블링크 스케치에서 핀 번호를 쉽게 변경할 수 있도록 수정한 예이다.

스케치 1. 블링크

```
int led = 13;                                // 13번 핀에 LED 연결

void setup() {
  pinMode(led, OUTPUT);                      // 13번 핀을 출력 모드로 설정
}

void loop() {
  digitalWrite(led, HIGH);                   // 13번 핀으로 HIGH (1 또는 ON) 출력
  delay(1000);                               // 1초 (1000ms) 대기
  digitalWrite(led, LOW);                    // 13번 핀으로 LOW (0 또는 OFF) 출력
  delay(1000);                               // 1초 (1000ms) 대기
}
```

블링크 스케치에서는 디지털 출력 핀으로 데이터를 출력하기 위해 pinMode와 digitalWrite 함수를 사용한다. 디지털 데이터를 출력할 수 있는 핀은 입력을 위해서도 사용이 가능하므로 입출력을 지정하기 위해 pinMode 함수가 사용된다.

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

핀 모드가 설정되면 비트 단위의 데이터 출력을 위해 digitalWrite 함수가 사용된다. digitalWrite 함수는 지정한 핀으로 이진 데이터를 출력하기 위해 사용한다.

```
void digitalWrite(uint8_t pin, uint8_t value)
- 매개변수
  pin : 핀 번호
  value : HIGH(1) 또는 LOW(0)
- 반환값 : 없음
```

학습보드의 디지털 2번에서 디지털 5번까지 연결된 LED를 점멸하는 스케치를 작성해 보자.

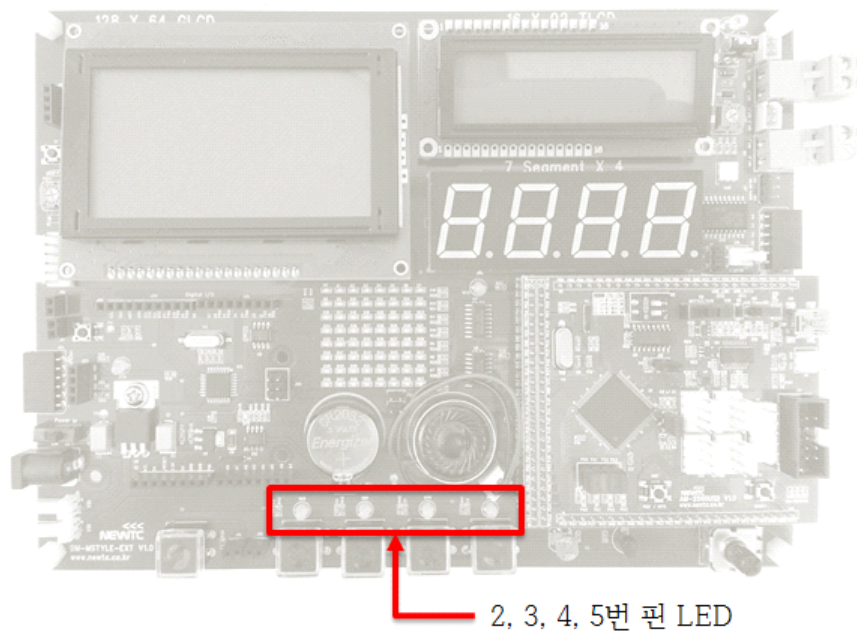


그림 3. 학습보드의 LED

스케치 2는 4개의 LED를 순차적으로 1초에 하나씩만 순서대로 켜는 스케치의 예이다.

스케치 2. 4개 LED 순차적으로 점등

```
int pins[] = {2, 3, 4, 5};           // LED 연결 핀
int state = 0;                       // 현재 켜 LED 인덱스

void setup() {
    Serial.begin(9600);               // 시리얼 통신 초기화

    for(int i = 0; i < 4; i++){
        pinMode(pins[i], OUTPUT);    // 출력 모드로 설정
        digitalWrite(pins[i], LOW);  // 꺼진 상태로 시작
    }
}

void loop() {
    for(int i = 0; i < 4; i++){
        if(i == state){               // 하나의 LED만 켜
```

```
Serial.print("O ");
digitalWrite(pins[i], HIGH);
}
else{                                // 나머지는 끄
    Serial.print("X ");
    digitalWrite(pins[i], LOW);
}
}
Serial.println();

state = (state + 1) % 4;              // 켜 LED 바꿈

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

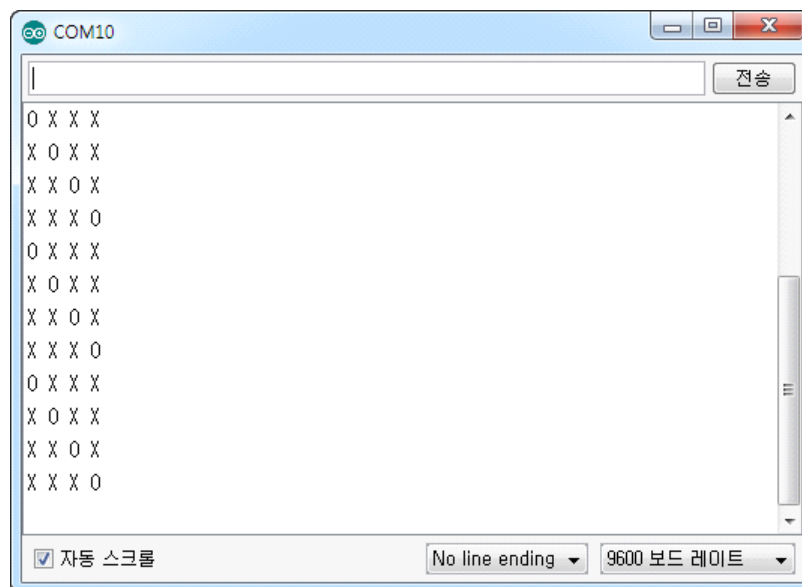


그림 4. 스케치 2 실행 결과

2. 시리얼 모니터로 LED 제어

시리얼 모니터에서 현재 켜 LED의 번호를 입력받고 해당 LED만을 켜는 스케치를 작성해 보자. LED 번호는 1에서 4 사이의 값을 입력 받고, 잘못된 값을 입력한 경우에는 잘못된 입력

임을 알려주도록 한다. 스케치 3은 시리얼 모니터를 통해 LED를 제어하는 스케치의 예이다.

스케치 3. 시리얼 모니터를 통한 LED 점멸 제어

```
int pins[] = {2, 3, 4, 5};           // LED 연결 핀
int state = -1;                       // 현재 켜 LED 인덱스

void setup() {
    Serial.begin(9600);               // 시리얼 통신 초기화

    for(int i = 0; i < 4; i++){
        pinMode(pins[i], OUTPUT);    // 출력 모드로 설정
        digitalWrite(pins[i], LOW);  // 꺼진 상태로 시작
    }
}

void loop() {
    if(Serial.available()){           // 입력 확인
        char data = Serial.read();    // 데이터 읽기
        Serial.println(String("You entered \"") + data + "\""); // 입력 문자 표시

        if(data >= '1' && data <= '4'){ // 유효한 입력인 경우
            state = data - '0' - 1;    // LED 인덱스로 변환
            Serial.print("LED ");
            Serial.print(state + 1);
            Serial.println(" is ON...");
        }
        else{                          // 잘못된 입력인 경우
            Serial.println("* Invalid LED number...");
            state = -1;
        }

        for(int i = 0; i < 4; i++){
            if(i == state){             // 하나의 LED만 켜
                Serial.print("O ");
                digitalWrite(pins[i], HIGH);
            }
        }
    }
}
```

```
    }  
    else{                                     // 나머지는 끄  
        Serial.print("X ");  
        digitalWrite(pins[i], LOW);  
    }  
}  
Serial.println();  
}  
}
```

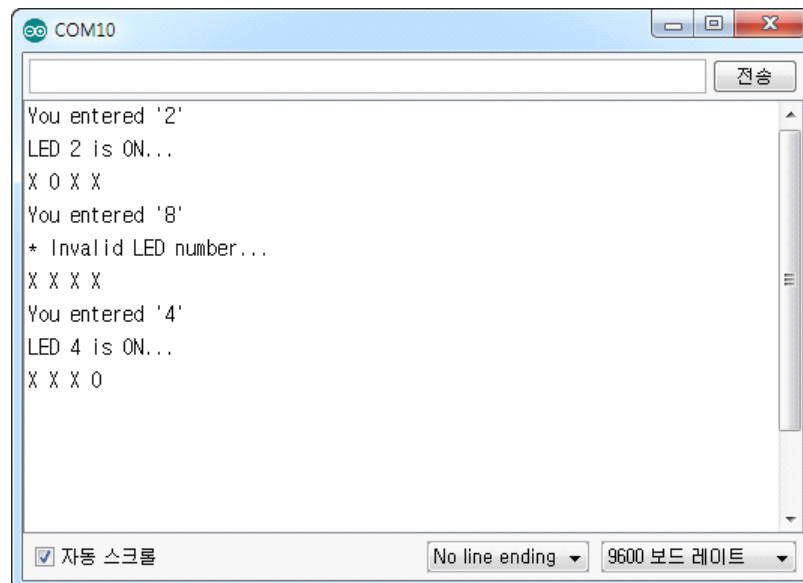


그림 5. 스케치 3 실행 결과