

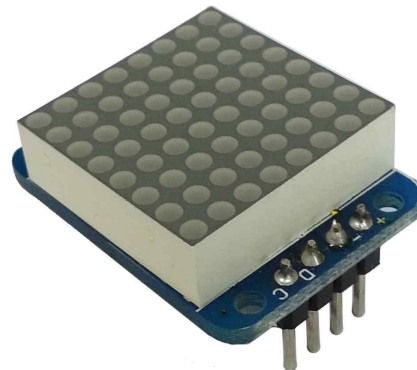
Chapter 118. LED 매트릭스

1. LED 매트릭스

LED 매트릭스는 LED를 행과 열의 매트릭스 형태로 배치하여 문자나 기호를 표시할 수 있도록 만들어진 장치로 다양한 분야에서 출력장치로 사용되고 있다. 특히 8×8 크기의 매트릭스는 알파벳이나 숫자 등 ASCII 코드에 정의된 문자를 표시할 수 있어 흔히 사용된다. 시중에 판매되고 있는 8×8 LED 매트릭스에는 LED만을 포함하고 있는 제품에서부터 LED 매트릭스를 제어하기 위한 회로가 포함되어 있는 제품에 이르기까지 다양한 종류의 제품이 판매되고 있다. 또한 매트릭스의 도트를 구성하는 LED도 단색, 이색, RGB 등 다양한 종류가 사용되고 있다. 이 장에서는 제어를 위한 별도의 회로를 포함하고 있지 않으면서 단색 LED를 사용하여 만들어진 8×8 크기의 LED 매트릭스를 사용한다.



(a) LED 매트릭스



(b) 제어 회로가 포함된 LED 매트릭스

그림 1. 8×8 LED 매트릭스

8×8 LED 매트릭스는 16개의 핀을 가지고 있으며 각 핀은 8개 행과 8개 열에 대응한다. LED 매트릭스의 뒷면을 살펴보면 1번 핀에 해당하는 위치에 숫자 '1'이 표시되어 있는 것을 볼 수 있다. 모듈에 따라서는 뒷면에 표시된 숫자를 확인하기 어려운 경우도 있으니 주의해서 살펴보아야 한다. 뒷면의 숫자를 확인하기 어려운 경우에는 일반적으로 형번이 기록된 면을 아래로 놓았을 때 가장 왼쪽 핀이 1번 핀이며 반시계 방향으로 핀 번호는 증가한다.



그림 2. LED 매트릭스의 핀 배치

실험에 사용한 LED 매트릭스 모듈의 내부는 행에 해당하는 핀을 (+)에, 열에 해당하는 핀을 (-)에 연결하였을 때 해당 위치의 LED가 켜지도록 그림 3과 같이 구성되어 있으며, 이를 공통 행 양극 (common-row anode) 방식 또는 양극 방식이라고 한다.¹⁾

1) 흔히 공통 행 양극 (common-row anode) 방식을 양극 (anode) 방식이라고 부르지만 공통 열 양극 (common-column anode) 방식을 양극 방식이라고 지칭하는 경우도 있으므로 반드시 모듈의 회로도를 확인하여야 한다.

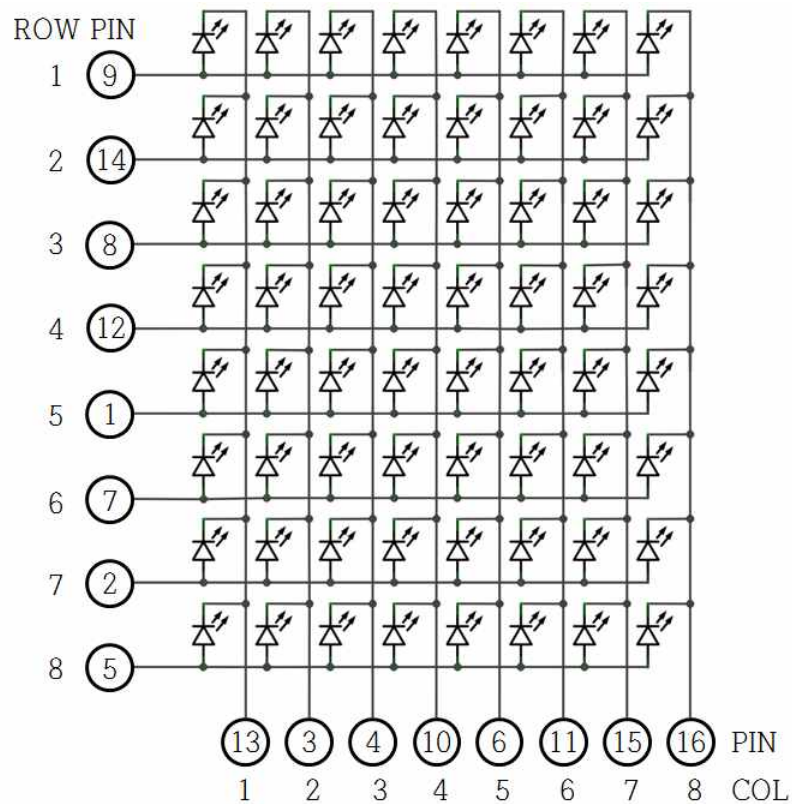


그림 3. LED 매트릭스 내부 회로도

양극 방식과 음극 방식 LED 매트릭스의 차이는 특정 위치의 LED를 켜는 방법에 있다. 그림 4는 4×4 크기의 양극 방식과 음극 방식의 LED 매트릭스를 비교한 것이다. D행 4열 LED를 켜는 경우를 생각해 보자. 양극 방식의 경우 행에는 (+)를, 열에는 (-)를 가해주어야 해당 LED만 켜지므로 행에는 (LOW LOW LOW HIGH)를, 열에는 (HIGH HIGH HIGH LOW)를 가해주어야 한다. 반면 음극 방식의 경우에는 양극 방식과 반대로 행에는 (HIGH HIGH HIGH LOW)를, 열에는 (LOW LOW LOW HIGH)를 가해주어야 한다.

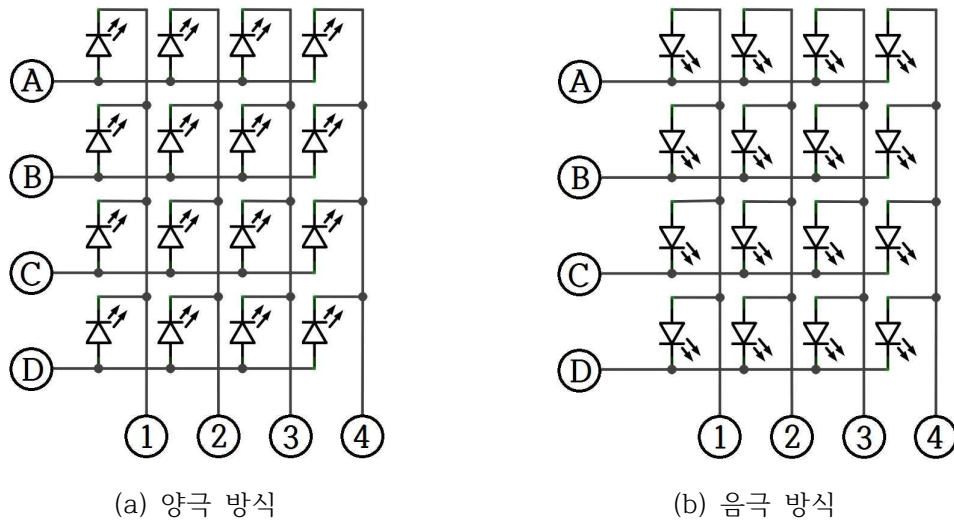


그림 4. 양극 방식과 음극 방식의 매트릭스

8×8 크기의 LED 매트릭스에 있는 모든 LED를 개별적으로 제어하기 위해서는 64개의 제어핀이 필요하다. 하지만 LED 매트릭스의 경우 16개의 제어핀만을 사용하므로 동시에 모든 LED를 제어하는 것은 불가능하다. 4자리 7세그먼트 표시장치를 제어하기 위해 한 번에 한 자리의 숫자만을 표시하고, 잔상효과를 이용하여 4자리 숫자가 동시에 표시되는 것과 같은 효과를 얻었다. LED 매트릭스의 경우도 마찬가지이다. 한 번에 하나의 행 또는 열만을 표시하고 잔상효과를 이용하여 LED 매트릭스 전체가 동시에 표시되는 것과 같은 효과를 얻을 수 있다. LED 매트릭스에서 한 번에 하나의 행을 제어하는 방식을 행 단위 스캔 방식이라 하며, 한 번에 하나의 열을 제어하는 방식을 열 단위 스캔 방식이라 한다. 여기서는 양극 방식의 LED 매트릭스를 기준으로 설명하지만 음극 방식의 경우에도 마찬가지이다. 먼저 행 단위 스캔의 예를 살펴보자. 행 단위 스캔에서는 한 번에 하나의 행에 포함되어 있는 8개 LED만을 제어할 수 있다. 4번째 행의 2번, 8번 열에 포함된 LED를 켜고 나머지 열에 포함된 LED는 끄기 위해서는 행에는 00010000_2 을 열에는 10111110_2 을 가해주면 된다. 행의 경우 하나의 행 만을 선택하여야 하므로 '1'의 값이 주어지는 행은 하나인 반면, 열의 경우에는 '0'의 값이 주어지는 열은 8개까지 가능하다.

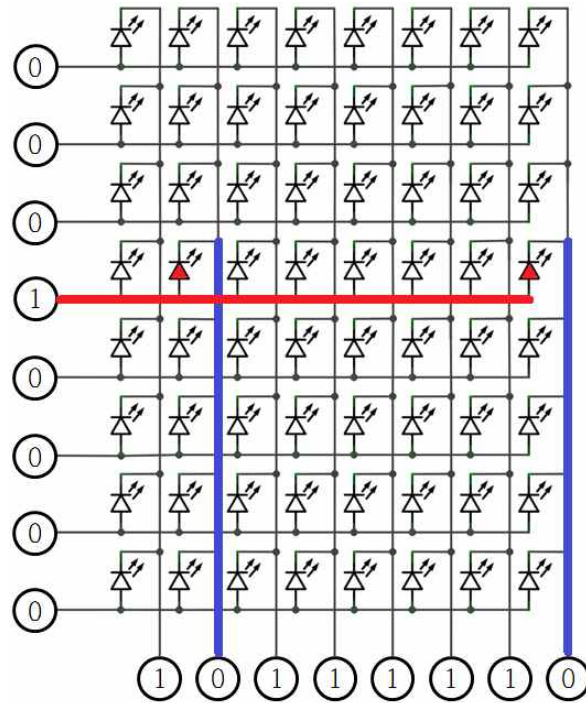


그림 5. 행 단위 스캔의 예

열 단위 스캔의 경우도 행 단위 스캔과 유사하다. 다만 하나의 열만을 선택하여야 하므로 '0'의 값이 주어지는 열은 하나인 반면, 행의 경우에는 '1'의 값이 주어지는 행은 8개까지 가능하다는 점에서 행 단위 스캔과 반대이다. 그림 6은 열 단위 스캔을 통해 4번째 열에서 1, 3, 6, 8번 행의 LED만을 켜고 나머지는 끄는 예로 행에는 10100101_2 을 열에는 11101111_2 을 가해주면 된다.

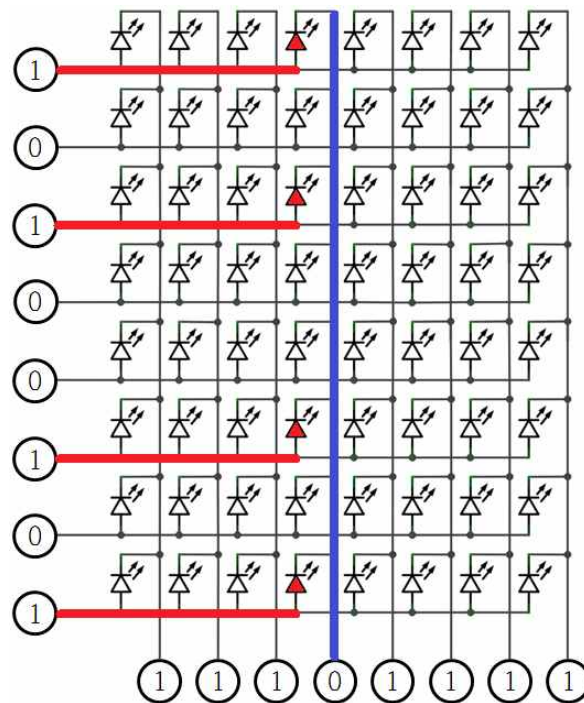


그림 6. 열 단위 스캔의 예

LED 매트릭스를 사용하기 위해서는 몇 가지 더 고려해야 할 점들이 있으며 이 중 하나가 LED 보호를 위한 저항의 연결이다. 저항의 연결에서 주의해야 할 점은 하나의 LED만 켜지는 쪽에 저항을 연결하여야 한다는 점이다. 행 단위 스캔의 경우 하나의 행에는 2개 이상의 LED가 켜질 수 있지만 하나의 열에는 하나의 LED만 켜질 수 있다. 따라서 저항은 열에 연결하여야 한다. 2개 이상의 LED가 켜질 수 있는 행에 저항을 연결한다면 켜지는 LED의 수에 따라 공급되는 전력이 달라질 수 있어 LED가 균일한 밝기를 보여주지 못할 수 있다. 유사하게 열 단위 스캔의 경우 하나의 LED만 켜질 수 있는 행에 저항을 연결하여야 한다.

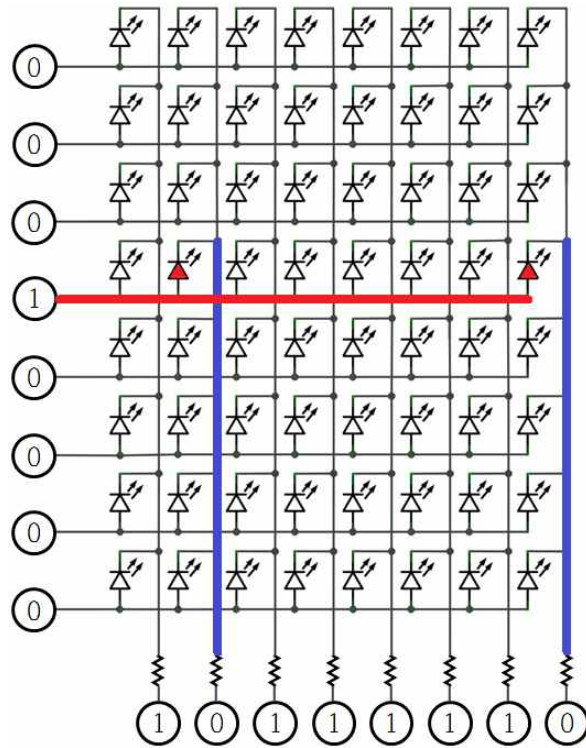


그림 7. 행 단위 스캔에서의 저항 연결

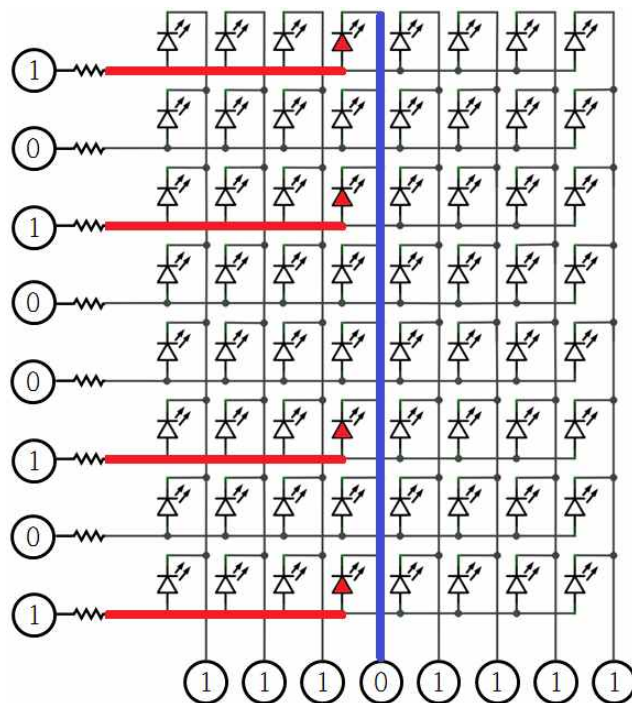


그림 8. 열 단위 스캔에서의 저항 연결

2. LED 매트릭스 제어

이 장에서 사용하는 LED 매트릭스는 양극 (common-row anode) 방식이며 열 단위 스캔 방식을 사용한다. LED 매트릭스를 그림 9와 같이 연결한다.

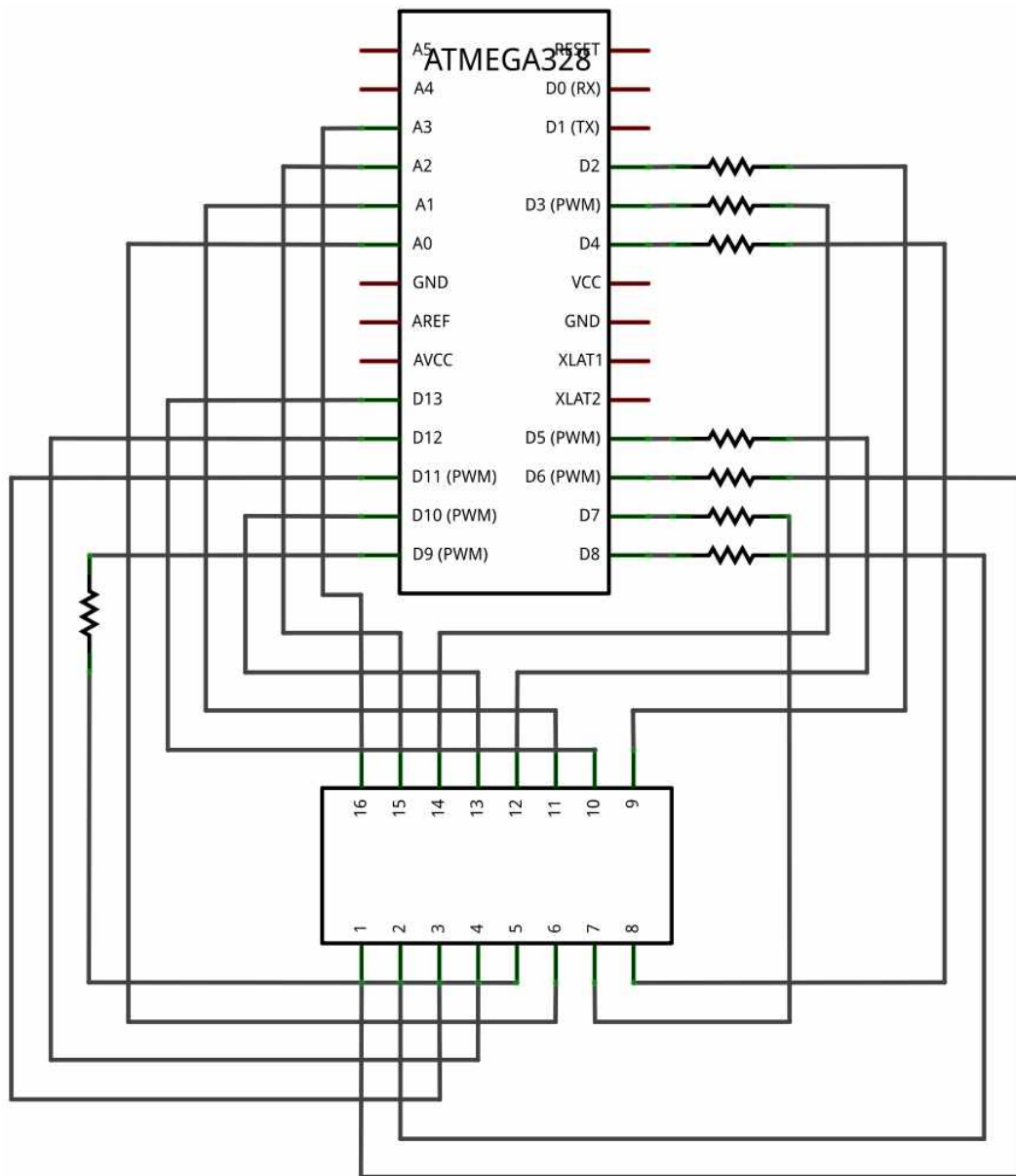


그림 9. LED 매트릭스 연결 회로도

회로를 연결하였으면 먼저 'LEDMatrixTest'라는 이름으로 프로젝트를 생성하고, 프로젝트에

‘LEDMatrixTest.h’ 파일을 추가해 보자. 코드 1은 추가한 헤더 파일의 내용으로 LED 매트릭스 제어를 위한 핀의 DDR 레지스터, PORT 레지스터, 핀 번호를 정의한 내용이다.

코드 1. LEDMatrixTest.h

```
#ifndef LEDMATRIXTEST_H_
#define LEDMATRIXTEST_H_

// DDR register for row
#define MATRIX_ROW1_DDR      DDRD
#define MATRIX_ROW2_DDR      DDRD
#define MATRIX_ROW3_DDR      DDRD
#define MATRIX_ROW4_DDR      DDRD
#define MATRIX_ROW5_DDR      DDRD
#define MATRIX_ROW6_DDR      DDRD
#define MATRIX_ROW7_DDR      DDRB
#define MATRIX_ROW8_DDR      DDRB

// PORT register for row
#define MATRIX_ROW1_PORT     PORTD
#define MATRIX_ROW2_PORT     PORTD
#define MATRIX_ROW3_PORT     PORTD
#define MATRIX_ROW4_PORT     PORTD
#define MATRIX_ROW5_PORT     PORTD
#define MATRIX_ROW6_PORT     PORTD
#define MATRIX_ROW7_PORT     PORTB
#define MATRIX_ROW8_PORT     PORTB

// pin number for row
#define MATRIX_ROW1_PIN      PD2
#define MATRIX_ROW2_PIN      PD3
#define MATRIX_ROW3_PIN      PD4
#define MATRIX_ROW4_PIN      PD5
#define MATRIX_ROW5_PIN      PD6
#define MATRIX_ROW6_PIN      PD7
#define MATRIX_ROW7_PIN      PB0
```

```
#define MATRIX_ROW8_PIN          PB1

// DDR register for column
#define MATRIX_COL1_DDR          DDRB
#define MATRIX_COL2_DDR          DDRB
#define MATRIX_COL3_DDR          DDRB
#define MATRIX_COL4_DDR          DDRB
#define MATRIX_COL5_DDR          DDRC
#define MATRIX_COL6_DDR          DDRC
#define MATRIX_COL7_DDR          DDRC
#define MATRIX_COL8_DDR          DDRC

// PORT register for column
#define MATRIX_COL1_PORT          PORTB
#define MATRIX_COL2_PORT          PORTB
#define MATRIX_COL3_PORT          PORTB
#define MATRIX_COL4_PORT          PORTB
#define MATRIX_COL5_PORT          PORTC
#define MATRIX_COL6_PORT          PORTC
#define MATRIX_COL7_PORT          PORTC
#define MATRIX_COL8_PORT          PORTC

// bit number for column
#define MATRIX_COL1_PIN          PB2
#define MATRIX_COL2_PIN          PB3
#define MATRIX_COL3_PIN          PB4
#define MATRIX_COL4_PIN          PB5
#define MATRIX_COL5_PIN          PC0
#define MATRIX_COL6_PIN          PC1
#define MATRIX_COL7_PIN          PC2
#define MATRIX_COL8_PIN          PC3

#endif /* LEDMATRIXTEST_H_ */
```

코드 2는 열 단위 스캔을 사용하여 각 열에서 행을 변화시키면서 0.1초 간격으로 하나의 LED만을 켜는 코드의 예이다.

코드 2. LEDMatrixTest.c

```
#define F_CPU 16000000L
#include <avr/io.h>
#include <util/delay.h>
#include "LEDMatrixTest.h"

void LEDMatrix_init(void)
{
    // 해당 column 핀을 출력으로 설정
    MATRIX_COL1_DDR |= (1 << MATRIX_COL1_PIN);
    MATRIX_COL2_DDR |= (1 << MATRIX_COL2_PIN);
    MATRIX_COL3_DDR |= (1 << MATRIX_COL3_PIN);
    MATRIX_COL4_DDR |= (1 << MATRIX_COL4_PIN);
    MATRIX_COL5_DDR |= (1 << MATRIX_COL5_PIN);
    MATRIX_COL6_DDR |= (1 << MATRIX_COL6_PIN);
    MATRIX_COL7_DDR |= (1 << MATRIX_COL7_PIN);
    MATRIX_COL8_DDR |= (1 << MATRIX_COL8_PIN);

    // 해당 row 핀을 출력으로 설정
    MATRIX_ROW1_DDR |= (1 << MATRIX_ROW1_PIN);
    MATRIX_ROW2_DDR |= (1 << MATRIX_ROW2_PIN);
    MATRIX_ROW3_DDR |= (1 << MATRIX_ROW3_PIN);
    MATRIX_ROW4_DDR |= (1 << MATRIX_ROW4_PIN);
    MATRIX_ROW5_DDR |= (1 << MATRIX_ROW5_PIN);
    MATRIX_ROW6_DDR |= (1 << MATRIX_ROW6_PIN);
    MATRIX_ROW7_DDR |= (1 << MATRIX_ROW7_PIN);
    MATRIX_ROW8_DDR |= (1 << MATRIX_ROW8_PIN);
}

void LEDMatrix_writeColumn(uint8_t col_data)    // column 데이터 출력
{
    if(col_data & (1 << 0)) MATRIX_COL1_PORT |= (1 << MATRIX_COL1_PIN); // ON
    else MATRIX_COL1_PORT &= ~(1 << MATRIX_COL1_PIN); // OFF
    if(col_data & (1 << 1)) MATRIX_COL2_PORT |= (1 << MATRIX_COL2_PIN);
```

```
else MATRIX_COL2_PORT &= ~(1 << MATRIX_COL2_PIN);
if(col_data & (1 << 2)) MATRIX_COL3_PORT |= (1 << MATRIX_COL3_PIN);
else MATRIX_COL3_PORT &= ~(1 << MATRIX_COL3_PIN);
if(col_data & (1 << 3)) MATRIX_COL4_PORT |= (1 << MATRIX_COL4_PIN);
else MATRIX_COL4_PORT &= ~(1 << MATRIX_COL4_PIN);
if(col_data & (1 << 4)) MATRIX_COL5_PORT |= (1 << MATRIX_COL5_PIN);
else MATRIX_COL5_PORT &= ~(1 << MATRIX_COL5_PIN);
if(col_data & (1 << 5)) MATRIX_COL6_PORT |= (1 << MATRIX_COL6_PIN);
else MATRIX_COL6_PORT &= ~(1 << MATRIX_COL6_PIN);
if(col_data & (1 << 6)) MATRIX_COL7_PORT |= (1 << MATRIX_COL7_PIN);
else MATRIX_COL7_PORT &= ~(1 << MATRIX_COL7_PIN);
if(col_data & (1 << 7)) MATRIX_COL8_PORT |= (1 << MATRIX_COL8_PIN);
else MATRIX_COL8_PORT &= ~(1 << MATRIX_COL8_PIN);
}

void LEDMatrix_writeRow(uint8_t row_data)      // row 데이터 출력
{
    if(row_data & (1 << 7)) MATRIX_ROW1_PORT |= (1 << MATRIX_ROW1_PIN);
    else MATRIX_ROW1_PORT &= ~(1 << MATRIX_ROW1_PIN);
    if(row_data & (1 << 6)) MATRIX_ROW2_PORT |= (1 << MATRIX_ROW2_PIN);
    else MATRIX_ROW2_PORT &= ~(1 << MATRIX_ROW2_PIN);
    if(row_data & (1 << 5)) MATRIX_ROW3_PORT |= (1 << MATRIX_ROW3_PIN);
    else MATRIX_ROW3_PORT &= ~(1 << MATRIX_ROW3_PIN);
    if(row_data & (1 << 4)) MATRIX_ROW4_PORT |= (1 << MATRIX_ROW4_PIN);
    else MATRIX_ROW4_PORT &= ~(1 << MATRIX_ROW4_PIN);
    if(row_data & (1 << 3)) MATRIX_ROW5_PORT |= (1 << MATRIX_ROW5_PIN);
    else MATRIX_ROW5_PORT &= ~(1 << MATRIX_ROW5_PIN);
    if(row_data & (1 << 2)) MATRIX_ROW6_PORT |= (1 << MATRIX_ROW6_PIN);
    else MATRIX_ROW6_PORT &= ~(1 << MATRIX_ROW6_PIN);
    if(row_data & (1 << 1)) MATRIX_ROW7_PORT |= (1 << MATRIX_ROW7_PIN);
    else MATRIX_ROW7_PORT &= ~(1 << MATRIX_ROW7_PIN);
    if(row_data & (1 << 0)) MATRIX_ROW8_PORT |= (1 << MATRIX_ROW8_PIN);
    else MATRIX_ROW8_PORT &= ~(1 << MATRIX_ROW8_PIN);
}

int main(void)
```

```

{
    LEDMatrix_init();

    while(1)
    {
        for(int i = 0; i < 8; i++){
            // 열 단위 스캔이므로 하나의 열에만 LOW를 가하고 나머지는 HIGH를 가함
            uint8_t col_data = ~(1 << i);
            LEDMatrix_writeColumn(col_data);

            for(int j = 0; j < 8; j++){
                // 하나의 열에서 켜고 싶은 위치의 LED에 HIGH를
                // 끄고 싶은 위치의 LED에 LOW를 가함
                uint8_t row_data = 1 << (7 - j);
                LEDMatrix_writeRow(row_data);

                _delay_ms(100);
            }
        }
    }
}

```

코드 1과 2를 사용하면 하나의 LED만을 켤 수 있지만, row_data 값을 조절함으로써 두 개 이상의 LED를 켤 수 있다. 하지만 서로 다른 열에 있는 LED를 동시에 켜기 위해서는 약간의 수정이 필요하다. 예를 들어 그림 10의 스마일 문자를 나타내고 싶다고 가정해 보자.

								1열	0b00111100
								2열	0b01000010
								3열	0b10101001
								4열	0b10000101
								5열	0b10000101
								6열	0b10101001
								7열	0b01000010
								8열	0b00111100

그림 10. 스마일 문자

열 단위 스캔을 사용하므로 열 단위로 데이터를 관리하는 것이 편리하다. 열 단위 데이터를 이용하여 스마일 문자를 나타내는 코드의 예가 코드 3이다. 코드 3에는 main 함수만을 나타내었고으므로 DMatrix_init, LEDMatrix_writeColumn, LEDMatrix_writeRow 함수는 포함시키지 않았으므로 코드 2를 참고하도록 한다.

코드 3. 스마일 문자 표시

```
int main(void)
{
    LEDMatrix_init();

    uint8_t smile[] = {
        0b00111100,
        0b01000010,
        0b10101001,
        0b10000101,
        0b10000101,
        0b10101001,
        0b01000010,
        0b00111100 };

    while(1)
    {
        for(int i = 0; i < 8; i++){
            // 열 단위 스캔이므로 하나의 열에만 LOW를 가하고 나머지는 HIGH를 가함
            uint8_t col_data = ~(1 << i);
            LEDMatrix_writeColumn(col_data);
            // 각 열에 대응하는 스마일 문자 데이터를 출력
            LEDMatrix_writeRow(smile[i]);

            _delay_ms(1);
        }
    }
}
```

코드 3에서 `_delay_ms` 함수는 LED의 반응 시간이 느려서 LED가 완전히 켜질 때까지 대기하기 위해 사용한 것이다. 코드 2에서 `delay(2)` 명령을 삭제하면 LED의 밝기가 어두워지거나 인접한 LED에 영향을 미치는 것을 확인할 수 있다. 이는 너무 빠른 열 단위의 전환으로 인해 LED를 켜기에 충분한 전류를 공급받지 못하기 때문이다. 반면 `delay` 함수 내에 숫자를 크게 하면 열과 열 사이의 표시 간격이 길어져 전체적으로 표시되는 문자가 깜빡거리게 된다. 실험에 사용한 모듈의 경우 2ms 정도의 지연 시간에서 안정된 동작을 보여 주었다.

3. LED 매트릭스 제어 - 아두이노

아두이노 환경에서 LED 매트릭스를 제어하는 것은 기본적으로 위에서와 동일하다. 다만, 아두이노 환경에서는 각 입출력 핀을 개별적으로 다루는 것이 가능하므로 작성해야 하는 코드는 훨씬 간단해진다. 코드 4는 코드 1과 동일한 동작을 하는 코드를 아두이노 환경에서 작성한 예이다.

코드 4. LED 매트릭스 테스트 - 한 번에 하나만 켜기

```
int rows[] = {2, 3, 4, 5, 6, 7, 8, 9};           // 행 연결 핀
int cols[] = {10, 11, 12, 13, A0, A1, A2, A3};    // 열 연결 핀

void setup(){
    for(int i = 0; i < 8; i++){                    // 해당 핀을 출력으로 설정
        pinMode(rows[i], OUTPUT);
        pinMode(cols[i], OUTPUT);
    }
}

void loop(){
    for(int i = 0; i < 8; i++){
        // 열 단위 스캔이므로 하나의 열에만 LOW를 가하고 나머지는 HIGH를 가함
        byte col_data = ~(1 << i);
        for(int k = 0; k < 8; k++){
            digitalWrite(cols[k], (col_data >> k) & 0x01);

            for(int j = 0; j < 8; j++){
```

```

        // 하나의 열에서 켜고 싶은 위치의 LED에 HIGH를
        // 끄고 싶은 위치의 LED에 LOW를 가함
        byte row_data = 1 << (7 - j);
        for(int k = 0; k < 8; k++)
            digitalWrite(rows[k], (row_data >> (7 - k)) & 0x01);

        delay(100);
    }
}
}

```

코드 5는 아두이노 환경에서 스마일 문자를 표시하는 코드의 예이다.

코드 5. 스마일 문자 표시

```

int rows[] = {2, 3, 4, 5, 6, 7, 8, 9};           // 행 연결 핀
int cols[] = {10, 11, 12, 13, A0, A1, A2, A3};    // 열 연결 핀

byte smile[] = {
    0b00111100,
    0b01000010,
    0b10101001,
    0b10000101,
    0b10000101,
    0b10101001,
    0b01000010,
    0b00111100 };

void setup(){
    for(int i = 0; i < 8; i++){                    // 해당 핀을 출력으로 설정
        pinMode(rows[i], OUTPUT);
        pinMode(cols[i], OUTPUT);
    }
}

```

```
void loop(){
  for(int i = 0; i < 8; i++){
    // 열 단위 스캔이므로 하나의 열에만 LOW를 가하고 나머지는 HIGH를 가함
    byte col_data = ~(1 << i);
    for(int j = 0; j < 8; j++)
      digitalWrite(cols[j], (col_data >> j) & 0x01);

    // 각 열에 대응하는 스마일 문자 데이터를 출력
    for(int j = 0; j < 8; j++)
      digitalWrite(rows[j], (smile[i] >> (7 - j)) & 0x01);

    delay(2);
  }
}
```