

2022-04-21

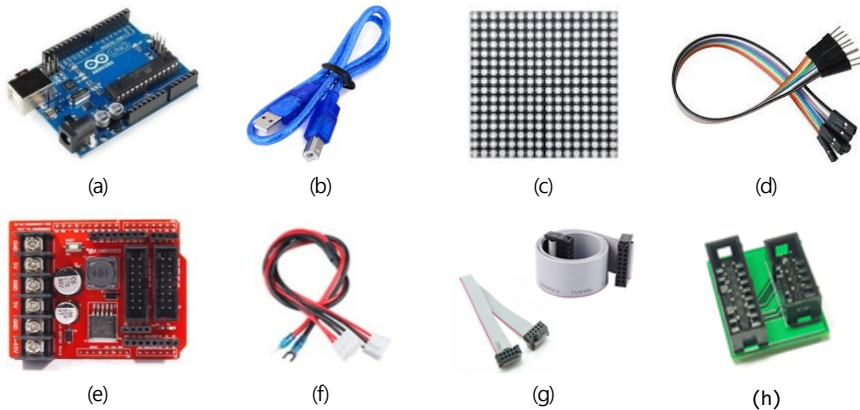
32X16 도트매트릭스 RGY LED모듈 키트 매뉴얼

주식회사 엘케이 임베디드
WWW.LKEMBEDDED.CO.KR

제1장 16X16 도트매트릭스 RGY LED 모듈 실습

이번 장에서는 아두이노 우노 R3를 이용하여 16x16 도트매트릭스 RGY LED 모듈 제어에 대해서 학습한다.

실습 준비물



<그림 1- 1 > 준비물

제품명	수량	설명	그림번호
아두이노 우노 R3	1	초보자도 사용 가능한 초소형 미니 컴퓨터	(a)
USB 케이블	1	프로그램 다운로드용 USB 데이터 케이블	(b)
16x16 도트매트릭스 LED 모듈	2	가로 16열, 세로 16행으로 구성된 RGY LED 모듈	(c)
점퍼 케이블(숫াম)	1	납땜없이 전기소자들을 연결해주는 선	(d)
도트매트릭스 LED셴드	1	도트매트릭스 LED 모듈 연결시 사용되는 셴드보드	(e)
전원 케이블	1	전원 2P 케이블	(f)
16P, 10P 플랫 케이블	1	데이터 16P 케이블	(g)
16P to 10P 커넥터	1	아두이노 도트매트릭스 LED셴드와 16x16 도트매트릭스 RGY LED 모듈을 연결시 사용되는 커넥터 변환보드	(h)

<표 1-1> 준비물 리스트

16x16 도트매트릭스 RGY LED 모듈

LED를 이용한 디스플레이는 이전에는 주로 7세그먼트였다. 그러나 대부분의 7세그먼트는 영문자와 숫자를 제한적으로 표시할 수 있어서 크기가 작고 정보 표현이 자유롭지 못했다.

현재는 다양한 문자 및 이미지 출력이 가능한 16x16 도트매트릭스 LED 모듈 혹은 32x16 도트매트릭스 LED 모듈이 대체되고 있다. 16x16 도트매트릭스 LED 모듈 혹은 32x16 도트매트릭스 LED 모듈은 서로 연결하여 확장이 가능하여 실내, 실외에서 대형 간판으로도 사용 가능하다.

본 실습에서 사용된 RGY LED 모듈 하드웨어는 가로 16행, 세로 16행으로 배열된 LED로 구성되었다. 각각의 도트에는 2개의 LED가 내장되어 있으며 한 도트당 적색 및 녹색으로 구동할 수 있다.

또한 각각의 도트는 적색 혹은 녹색을 LED를 선택하여 점등 가능하며 2개의 적색 및 녹색 LED를 동시에 구동시켰을때는 적색과 녹색이 혼합된 황색으로도 점등할 수 있다.

16x16 도트매트릭스 LED 모듈을 제어하는 방법은 스테틱 구동과 다이내믹 구동이 있다.

스태틱 구동은 시프트 레지스터의 비트를 LED 개수만큼 준비하고 데이터를 전송하여 점등하는 방식으로 한번에 모든 LED를 ON 혹은 OFF 할 수 있다.

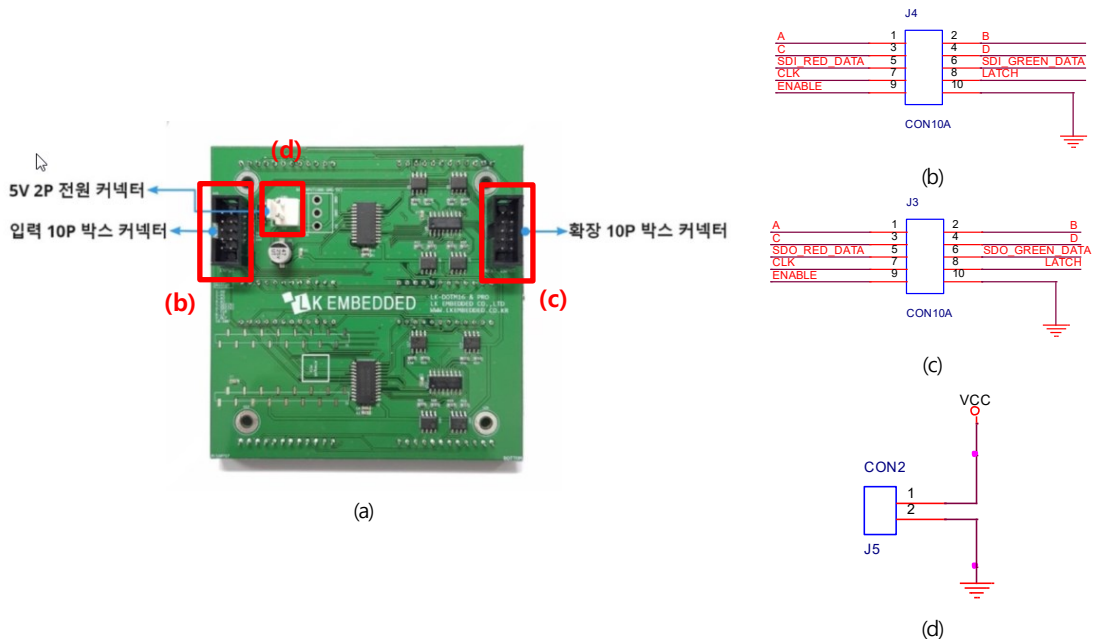
다이내믹 구동은 1행에서 16행을 같은 주기로 스캔하면서 각각의 열에 해당하는 LED를 ON 혹은 OFF하는 방식이다. 행과 열을 구분하기 위해 스캔 라인과 데이터 라인으로 구성되며 1행씩 스캔하면서 16개의 동기화된 데이터를 데이터 라인을 통해 RGY LED 모듈에 입력하는 방식이다.

다이내믹 구동은 스태틱 구동에 비해 휘도와 스캔 주파수가 낮은 경우에 깜빡거림이 발생하는 치명적인 단점이 있지만 구동 회로가 간단하기 때문에 저비용으로 구현이 가능하다.

이번 실습에서 구현할 RGY LED 모듈의 구동방법은 다이내믹 구동방식으로 해당하는 핀에 데이터를 입력하여 적색 또는 녹색 LED를 구동하도록 한다.

커넥터

<그림 1-2>처럼 RGY LED 모듈 후면에는 전원을 공급할수 있는 2P 커넥터 (d), 신호를 입력할수 있는 10P 박스 커넥터 (b) 그리고 RGY LED 모듈을 확장할 때 사용되는 10P 박스 커넥터 (c) 로 구성된다.



<그림 1-2> RGY LED 모듈 후면(a), 10P 박스 커넥터(b), 확장 10P 박스 커넥터(c), 2P 전원 커넥터(d)

• 10P 박스 커넥터

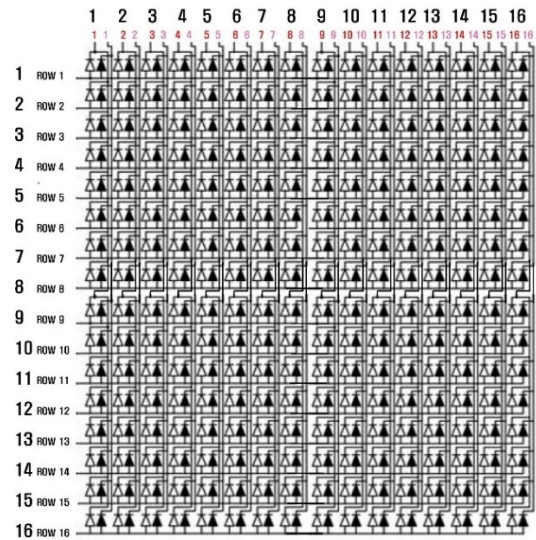
A, B, C, D

<그림 1-3>처럼 RGY LED 모듈은 세로 16행, 가로 16열의 구조로 LED가 구성된 하드웨어로 구성되었다. 10P 박스 커넥터 A, B, C, D 핀은 행 어드레스 입력 핀이므로 1행에서 16행 라인을 선택하기 위한 용도로 사용된다. 각각의 행은 <표 1-3>처럼 행 어드레스 핀 신호 설정에 따라 RGY LED 모듈 ROW1에서 ROW16 행을 하나씩 선택할 수 있다.

다이내믹 구동방식에서 행 어드레스 핀을 선택하고 다른 행으로 변화되기 전까지의 시간을 t 라고 했을 때 각각의 t 시간은 동일해야 한다. 왜냐하면 t 시간에 의해서 각 라인에 공급할 전력량이 결정되고 이를 통해서 RGY LED 모듈의 휘도가 균일하게 유지되기 때문이다.

	어드레스 핀 신호 설정	행 LED
1	A=0, B=0, C=0, D=0	ROW1
2	A=1, B=0, C=0, D=0	ROW2
3	A=0, B=1, C=0, D=0	ROW3
4	A=1, B=1, C=0, D=0	ROW4
5	A=0, B=0, C=1, D=0	ROW5
6	A=1, B=0, C=1, D=0	ROW6
7	A=0, B=1, C=1, D=0	ROW7
8	A=1, B=1, C=1, D=0	ROW8
9	A=0, B=0, C=0, D=1	ROW9
10	A=1, B=0, C=0, D=1	ROW10
11	A=0, B=1, C=0, D=1	ROW11
12	A=1, B=1, C=0, D=1	ROW12
13	A=0, B=0, C=1, D=1	ROW13
14	A=1, B=0, C=1, D=1	ROW14
15	A=0, B=1, C=1, D=1	ROW15
16	A=1, B=1, C=1, D=1	ROW16

<표 1-2> 어드레스 행 테이블



림 1-3 > 도트매트릭스 LED 모듈 내부 회로 다이어그램

<표 1-3> RGY LED 모듈 행 어드레스 선택

G1

시리얼 데이터 입력 16비트 시프트 레지스터 입력 핀으로 열(Column)에 해당하는 16개의 녹색 LED를 구동할 때 사용되는 입력 핀이다.

R1

시리얼 데이터 입력 16비트 시프트 레지스터 입력 핀으로 열(Column)에 해당하는 16개의 적색 LED를 구동할 때 사용되는 입력 핀이다.

LATCH

시리얼 데이터 저장 신호 입력 핀으로 G1, R1 핀에서 16개의 신호를 받은 후 데이터를 저장하는 신호를 입력받는 핀이다. 하이에서 로우로 변화하는 신호를 받을 때 데이터 신호를 내부적으로 저장한다.

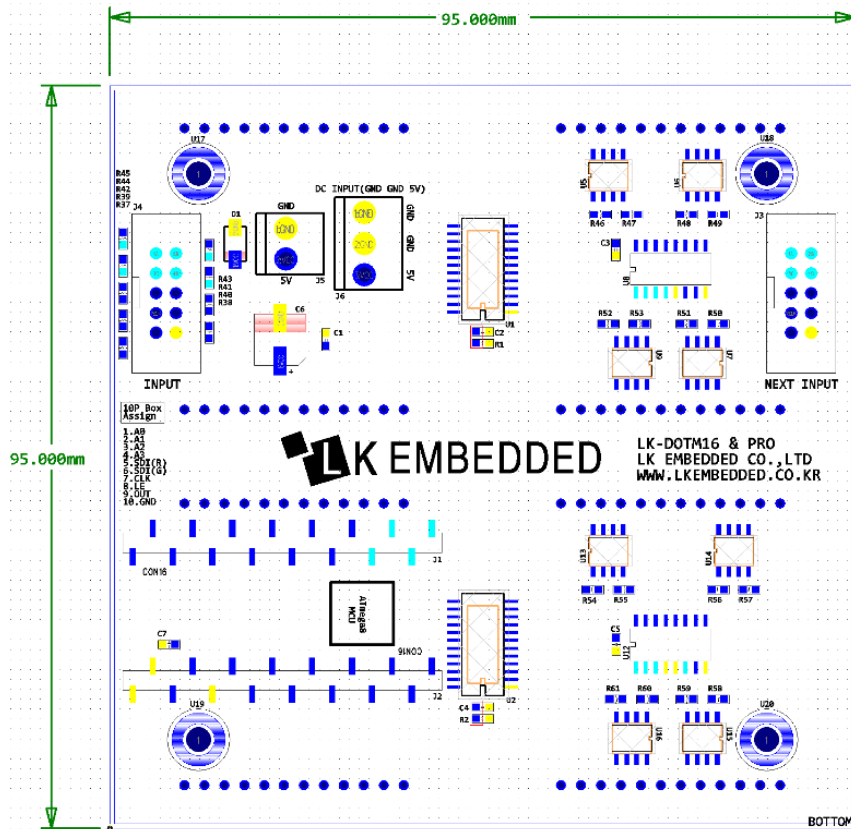
CLK

시리얼 클럭 입력 핀으로 G1, R1 핀에 데이터를 입력시 동기화 된 클럭 신호를 입력받는 핀이다. 클럭 신호가 라이징 엣지 때 G1, R1 핀에 신호를 입력받아 데이터를 레지스터에 전송한다.

OE

출력 활성화 입력 핀으로 로우일때 출력 드라이버가 활성화 되어 전체의 LED가 구동상태로 변한다.

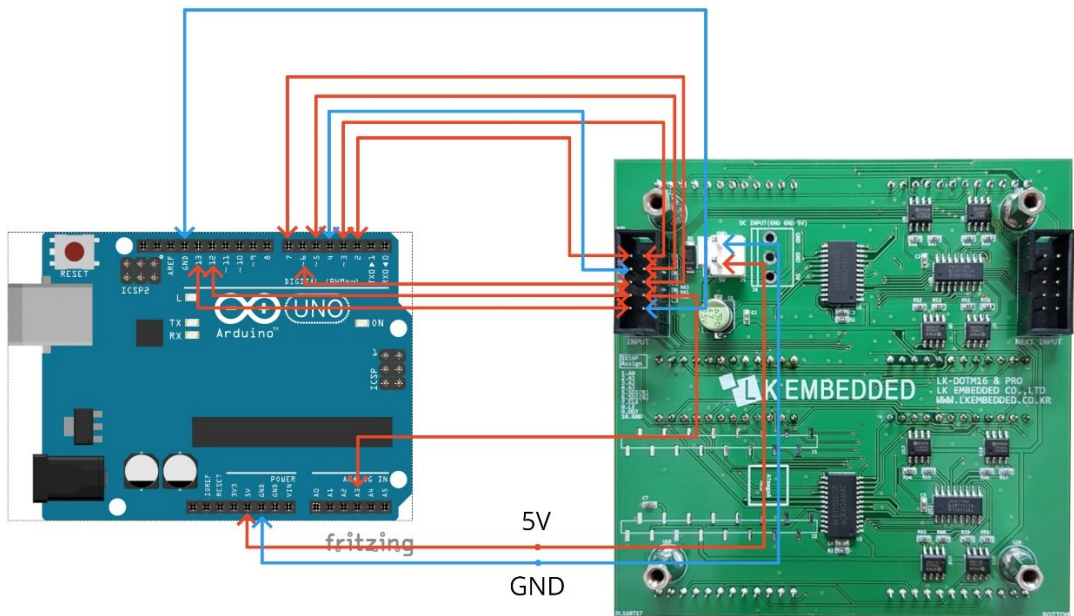
<그림 1-4 >에서는 RGY LED 모듈의 PCB 디멘전을 보여주고 있다.



<그림 1-4 > PCB Layout of Dimension

회로도

<그림 1-5>에서는 RGY LED 모듈과 아두이노 우노 R3를 점퍼 케이블을 이용하여 연결한 모습을 보여주고 있다.



<그림 1-5> 16x16 도트매트릭스 RGY LED 모듈과 아두이노 우노 R3가 연결된 사진

RGY LED 모듈 A, B, C, D 핀은 아두이노 우노 R3 디지털 2번, 3번, 4번, 5번 핀에 순서대로 연결한다. 그리고 나머지 R1,G1, CLK, LATCH, OE 핀은 아두이노 우노 R3 디지털 6, 7, 8, 9, 10번 핀에 각각 순서대로 연결한다.

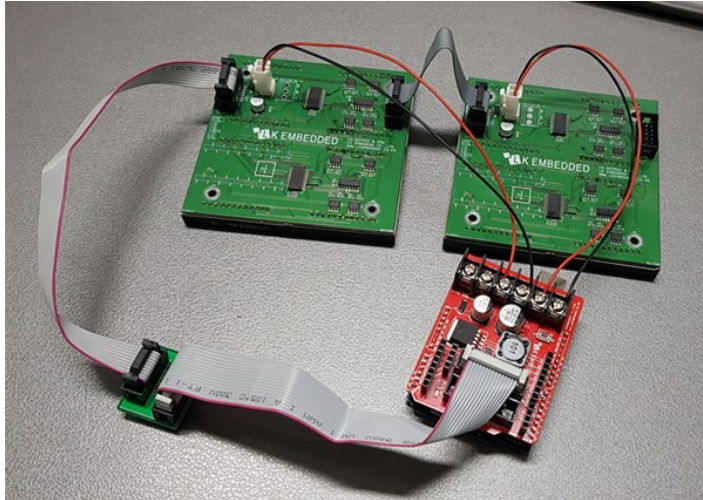
RGY LED 모듈 GND 핀은 아두이노 우노 R3의 GND 핀에 연결한다. 그리고 RGY LED 모듈 전원 VCC, GND 핀은 아두이노 우노 R3 VCC, GND 핀에 연결한다. <표 1-4>는 앞에서 설명한 내용을 요약한 핀 맵 테이블이다.

아두이노 우노 R3	10P 박스 커넥터	아두이노 우노 R3	확장 10P 박스 커넥터
디지털 2번	A	디지털 7번	DG1
디지털 3번	B	디지털 8번	CLK
디지털 4번	C	디지털 9번	LATCH
디지털 5번	D	디지털 10번	OE
디지털 6번	DR1	VCC, GND	VCC, GND (2P 전원 커넥터)

<표 1-4> 핀 맵

아두이노 우노 R3 도트매트릭스 LED 쉴드

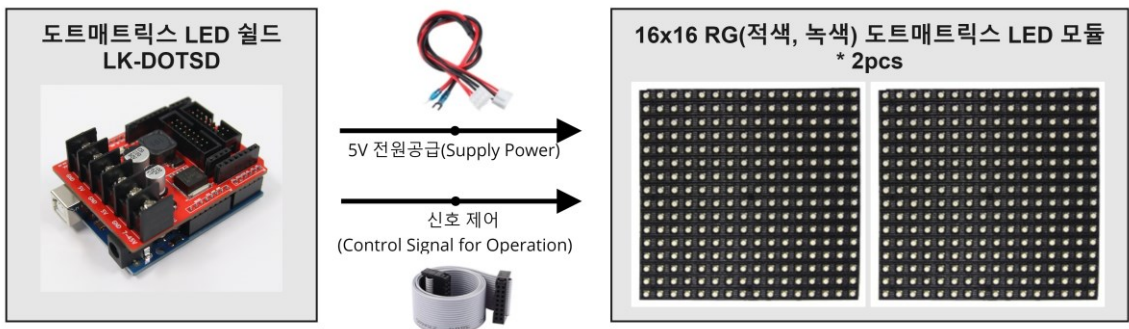
만일 아두이노 우노 R3 도트매트릭스 LED 쉴드가 준비된 경우에는 <그림 1-6>처럼 10P 플랫케이블과 2P 전원케이블을 이용하여 번거로움 없이 RGY LED 모듈과 아두이노 우노 R3를 연결할 수 있다.



<그림 1-6> 도트매트릭스 LED 쉴드를 사용하여 연결된 사진

<그림 1-7>은 도트매트릭스 LED 쉴드를 사용하여 RGY LED 모듈에 전원 및 신호공급에 대한 흐름을 보여주는 다이어그램이다.

도트매트릭스 LED 쉴드는 외부전원(7~45V, 3A)을 입력받아 아두이노 우노 R3 및 RGY LED 모듈에 안정적인 전원이 공급이 가능하다. 이로 인해서 400mA로 제한된 아두이노 우노 R3 USB 전원을 사용하지 않고 다수의 RGY LED 모듈을 확장 연결하여 구동할수 있어 실습 시 유용하게 사용될 수 있다.



<그림 1-7> 16x16 도트매트릭스 RGY LED 모듈과 도트매트릭스 LED 쉴드의 연결 다이어그램

참고지식

• 전류 소모

RGY LED 모듈은 적색, 녹색 LED가 256개가 내장되어 있으며 일반적인 전류 소모가 LED 1개당 15~20mA 일 때 512개의 LED를 스테틱으로 구동하면 전류 소모는 약 10A 이상이다. 그러나 다이내믹 방식으로 RGY LED 모듈을 구동하면 전류 소모는 이보다 훨씬 낮다.

RGY LED 모듈 전류 소모는 LED 점등 개수와 연관되며 실제로 “가” 를 다이내믹 구동방식을 이용하여 제어했을 때 전류 소모량은 300mA 이하이다.

예제코드

• 예제 1

아래의 예제코드는 아두이노 스케치 컴파일러에서 코드를 작성했다. 주요 프로그래밍 시나리오는 아두이노 우노 R3에서 16x16 도트매트릭스 RGY LED 모듈로 제어신호를 보내어 “LK” 문자를 출력하는 예제이다.

```
#include <avr/pgmspace.h>
#include <avr/interrupt.h>
#include <avr/io.h>

#define A_High PORTD|=0x04 // A
#define A_Low PORTD&=~0x04 // Arduino Uno 2 번핀
#define B_High PORTD|=0x08 // B
#define B_Low PORTD&=~0x08 // Arduino Uno 3 번핀
#define C_High PORTD|=0x10 // C
#define C_Low PORTD&=~0x10 // Arduino Uno 4 번핀
#define D_High PORTD|=0x20 // C
#define D_Low PORTD&=~0x20 // Arduino Uno 5 번핀
#define red_enable PORTD|=0x40 // Red
#define red_disable PORTD&=~0x40 // Arduino Uno 6 번핀
#define green_enable PORTD|=0x80 // Green
#define green_disable PORTD&=~0x80 // Arduino Uno 7 번핀
#define LE_enable PORTC|=0x08 // Latch_Enable(LE)
#define LE_disable PORTC&=~0x08 // Arduino Uno A3 번핀
#define Clk_enable PORTB|=0x10 // Clk
#define Clk_disable PORTB&=~0x10 // Arduino Uno 12 번핀
#define OE_enable PORTB&=~0x20 // Output_Enable(OE)
#define OE_disable PORTB|=0x20 // Arduino Uno 13 번핀
const int __attribute__((progmem)) string[][16] = {
0x00,0x00,0x00,0xc183,0x6183,0x3183,0x1983,0xd83,0x783,0xd83,0x1983,0x3183,0x6183,0xc1ff,0x180,0x00,
};
unsigned char Dot_char_cnt = 0; // 도트매트릭스에 표시될 문자 카운팅
unsigned char flag_cnt = 0; // 도트매트릭스에 표시될 색상 카운팅
unsigned int str_speed_cnt = 0;
bool flag_Oe = 0;
ISR(TIMER2_OVF_vect) {
    // 0.1mS 오버플로우 인터럽트 발생
    static unsigned int string_moving_speed_value = 500;
    // 도트매트릭스 시프트 속도 변수
    static unsigned int cnt = 0;
    if (flag_Oe) {
        // 스트로브 신호 활성화되면
        OE_enable; // OUTPUT 신호 활성화
```



```

        cnt++; // cnt 변수 증가
        if (cnt >= 5) {
            // OUTPUT 신호 주기 생성
            cnt = 0;
            flag_Oe = 0;
            OE_disable;
        }
    }
    str_speed_cnt++;
    if (str_speed_cnt > string_moving_speed_value)
    // 문자가 50mS 머문 후 시프트 시작
    {
        str_speed_cnt = 0;
        flag_cnt++; // 도트매트릭스 디스플레이 컬러 변경
        if (flag_cnt == 3) {
            flag_cnt = 0;
        }
    }
    TCNT2 = 0xe7;
}

void setup() {
    DDRD=0XFC;
    DDRC=0X08;
    DDRB=0X30;
    TCCR2A = 0x00;
    TCCR2B = 0x04; // 타이머 카운터 2 노멀 모드, 64 분주
    TIMSK2 = 0x01; // 타이머 카운터 2 인터럽트 허용
    TCNT2 = 0xe7; // 타이머 카운터 2 초기값: 231
    SREG = 0x80; // 인터럽트 허용
}

void row_dynamic(unsigned int i) {
    switch (i) // 행 이동
    {
        case 0:
            A_Low;
            B_Low;
            C_Low;
            D_Low;
            break; // 1 행 LED

        case 1:
            A_High;
            B_Low;
            C_Low;
            D_Low;
            break; // 2 행 LED

        case 2:
            A_Low;

```

```

        B_High;
        C_Low;
        D_Low;
        break; // 3 행 LED
case 3:
    A_High;
    B_High;
    C_Low;
    D_Low;
    break; // 4 행 LED
case 4:
    A_Low;
    B_Low;
    C_High;
    D_Low;
    break; // 5 행 LED
case 5:
    A_High;
    B_Low;
    C_High;
    D_Low;
    break; // 6 행 LED
case 6:
    A_Low;
    B_High;
    C_High;
    D_Low;
    break; // 7 행 LED
case 7:
    A_High;
    B_High;
    C_High;
    D_Low;
    break; // 8 행 LED
case 8:
    A_Low;
    B_Low;
    C_Low;
    D_High;
    break; // 9 행 LED
case 9:
    A_High;
    B_Low;
    C_Low;
    D_High;
    break; // 10 행 LED
case 10:
    A_Low;
    B_High;

```

```

        C_Low;
        D_High;
        break; // 11 행 LED
    case 11:
        A_High;
        B_High;
        C_Low;
        D_High;
        break; // 12 행 LED
    case 12:
        A_Low;
        B_Low;
        C_High;
        D_High;
        break; // 13 행 LED
    case 13:
        A_High;
        B_Low;
        C_High;
        D_High;
        break; // 14 행 LED
    case 14:
        A_Low;
        B_High;
        C_High;
        D_High;
        break; // 15 행 LED
    case 15:
        A_High;
        B_High;
        C_High;
        D_High;
        break; // 16 행 LED
    }
}

void shift_Register(unsigned char out) {
    // 점등 시키고자 하는 LED Bit 입력
    unsigned char clk = 0;
    for (clk = 0; clk < 8; clk++) {
        // 8 비트 데이터를 1 비트씩 시프트레지스터에 입력
        if (out & (0x80 >> clk)) {
            switch (flag_cnt) {
                case 0:
                    green_disable;
                    red_enable;
                    break; // 적색으로 LED 점등
                case 1:
                    green_enable;

```

```

        red_disable;
        break; // 녹색으로 LED 점등
    case 2:
        green_enable;
        red_enable;
        break; // 노란색으로 LED 점등
    }
} else {
    green_disable;
    red_disable;
}
Clk_enable; // CLK 신호 활성화
Clk_disable;
}
}
void ActivePulse() {
    // 스트로브 신호
    LE_enable;
    LE_disable; // LE 신호 활성화
    flag_Oe = 1; // OE 신호 활성화
}
void dot1_display(unsigned char first) // 도트매트릭스 문자 출력 함수
{
    static unsigned int i_cnt = 0;
    unsigned int buff1[16] = {
        0
    };
    unsigned char high1 = 0; // Dot1 상위 bit
    unsigned char low1 = 0; // Dot1 하위 bit
    register unsigned int i = 0;
    for (i_cnt = 0; i_cnt < 16; i_cnt++) {
        buff1[i_cnt] = pgm_read_word( & string[first][i_cnt]);
        // 출력 버퍼에 문자열 저장
    }
    for (i = 0; i < 16; i++) {
        high1 = (buff1[i] >> 8); // Dot1 상위 8bit 저장
        low1 = (buff1[i] & 0xff); // Dot1 하위 8bit 저장
        shift_Register(high1); // 상위 8bit LED 점등
        shift_Register(low1); // 하위 8bit LED 점등
        row_dynamic(i); // 0~15 행 접근
        ActivePulse(); // 스트로브 신호
    }
}
void loop() {
    dot1_display(Dot_char_cnt);
}

```

```
}
```

• 예제 2

아래의 예제코드는 아두이노 스케치 컴파일러에서 코드를 작성했다. 주요 프로그래밍 시나리오는 아두이노 우노 R3에서 16x16 도트매트릭스 RGY LED 모듈로 제어신호를 보내어 “LK” 문자를 출력하고 50ms 간격으로 문자를 왼쪽에서 오른쪽으로 시프트하는 예제이다.

```
#include <avr/pgmspace.h>
#include <avr/interrupt.h>
#include <avr/io.h>

#define A_High PORTD|=0x04 // A
#define A_Low PORTD&=~0x04 // Arduino Uno 2 번핀
#define B_High PORTD|=0x08 // B
#define B_Low PORTD&=~0x08 // Arduino Uno 3 번핀
#define C_High PORTD|=0x10 // C
#define C_Low PORTD&=~0x10 // Arduino Uno 4 번핀
#define D_High PORTD|=0x20 // D
#define D_Low PORTD&=~0x20 // Arduino Uno 5 번핀
#define red_enable PORTD|=0x40 // Red
#define red_disable PORTD&=~0x40 // Arduino Uno 6 번핀
#define green_enable PORTD|=0x80 // Green
#define green_disable PORTD&=~0x80 // Arduino Uno 7 번핀
#define LE_enable PORTC|=0x08 // Latch_Enable(LE)
#define LE_disable PORTC&=~0x08 // Arduino Uno A3 번핀
#define Clk_enable PORTB|=0x10 // Clk
#define Clk_disable PORTB&=~0x10 // Arduino Uno 12 번핀
#define OE_enable PORTB&=~0x20 // Output_Enable(OE)
#define OE_disable PORTB|=0x20 // Arduino Uno 13 번핀

const int __attribute__((progmem)) string[][16] = {
0x00,0x00,0x00,0xc183,0x6183,0x3183,0x1983,0xd83,0x783,0xd83,0x1983,0x3183,0x6183,0xc1ff,0x180,0x00,
}; // LK

unsigned char Dot_char_cnt = 0; // 도트매트릭스에 표시될 문자 카운팅
unsigned char flag_cnt = 0; // 도트매트릭스에 표시될 색상 카운팅
unsigned char move_motion = 0; // 도트매트릭스의 디스플레이 모션 방식 설정
unsigned int Move_cnt = 16;
unsigned int Move_cnt2 = 0; // 도트매트릭스 시프트 변수
unsigned int str_speed_cnt = 0;
bool flag_Oe = 0;
ISR(TIMER2_OVF_vect) {
    // 0.1ms 오버플로우 인터럽트 발생
    static unsigned int string_moving_speed_value = 500;
    // 도트매트릭스 시프트 속도 변수
```

```

static unsigned int cnt = 0;
if (flag_Oe) {
    // 스트로브 신호 활성화되면
    OE_enable; // OUTPUT 신호 활성화
    cnt++; // cnt 변수 증가
    if (cnt >= 5) {
        // OUTPUT 신호 주기 생성
        cnt = 0;
        flag_Oe = 0;
        OE_disable;
    }
}
str_speed_cnt++;
if (str_speed_cnt > string_moving_speed_value) {
    // 문자가 50ms 초 머문 후 시프트 시작
    str_speed_cnt = 0;
    Move_cnt--;
    Move_cnt2++;
    if (Move_cnt == 0 || Move_cnt2 == 16) {
        Move_cnt = 16;
        Move_cnt2 = 0;
        move_motion++;
        if (move_motion > 1) {
            move_motion = 0;
            flag_cnt++; // 도트매트릭스 디스플레이 컬러 변경
            if (flag_cnt == 3) {
                flag_cnt = 0;
            }
        }
    }
}
}
TCNT2 = 0xe7;
}

void setup() {
    DDRD=0XFC;
    DDRC=0X08;
    DDRB=0X30;

    TCCR2A = 0x00;
    TCCR2B = 0x04; // 타이머 카운터 2 노멀 모드, 64 분주
    TIMSK2 = 0x01; // 타이머 카운터 2 인터럽트 허용
    TCNT2 = 0xe7; // 타이머 카운터 2 초기값: 231
    SREG = 0x80; // 인터럽트 허용
}

void row_dynamic(unsigned int i) {
    static unsigned int str_cnt = 0;
    switch (i) // 행 이동

```

```

{
case 0:
    A_Low;
    B_Low;
    C_Low;
    D_Low;
    break; // 1 행 LED
case 1:
    A_High;
    B_Low;
    C_Low;
    D_Low;
    break; // 2 행 LED
case 2:
    A_Low;
    B_High;
    C_Low;
    D_Low;
    break; // 3 행 LED
case 3:
    A_High;
    B_High;
    C_Low;
    D_Low;
    break; // 4 행 LED
case 4:
    A_Low;
    B_Low;
    C_High;
    D_Low;
    break; // 5 행 LED
case 5:
    A_High;
    B_Low;
    C_High;
    D_Low;
    break; // 6 행 LED
case 6:
    A_Low;
    B_High;
    C_High;
    D_Low;
    break; // 7 행 LED
case 7:
    A_High;
    B_High;
    C_High;
    D_Low;
    break; // 8 행 LED

```



```

case 8:
    A_Low;
    B_Low;
    C_Low;
    D_High;
    break; // 9 행 LED
case 9:
    A_High;
    B_Low;
    C_Low;
    D_High;
    break; // 10 행 LED
case 10:
    A_Low;
    B_High;
    C_Low;
    D_High;
    break; // 11 행 LED
case 11:
    A_High;
    B_High;
    C_Low;
    D_High;
    break; // 12 행 LED
case 12:
    A_Low;
    B_Low;
    C_High;
    D_High;
    break; // 13 행 LED
case 13:
    A_High;
    B_Low;
    C_High;
    D_High;
    break; // 14 행 LED
case 14:
    A_Low;
    B_High;
    C_High;
    D_High;
    break; // 15 행 LED
case 15:
    A_High;
    B_High;
    C_High;
    D_High;
    break; // 16 행 LED
}

```

```

}
void shift_Register(unsigned char out) {
    // 점등 시키고자 하는 LED Bit 입력
    unsigned char clk = 0;
    for (clk = 0; clk < 8; clk++) {
        // 8비트 데이터를 1비트씩 시프트레지스터에 입력
        if (out & (0x80 >> clk)) {
            switch (flag_cnt) {
                case 0:
                    green_disable;
                    red_enable;
                    break; // 적색으로 LED 점등
                case 1:
                    green_enable;
                    red_disable;
                    break; // 녹색으로 LED 점등
                case 2:
                    green_enable;
                    red_enable;
                    break; // 노란색으로 LED 점등
            }
        }
        else {
            green_disable;
            red_disable;
        }
        Clk_enable; // CLK 신호 활성화
        Clk_disable;
    }
}

void ActivePulse() {
    // 스트로브 신호
    LE_enable;
    LE_disable; // LE 신호 활성화
    flag_Oe = 1; // OE 신호 활성화
}

void dot1_display_shift(unsigned char first) {
    // 도트매트릭스 오른쪽 시프트 함수
    static unsigned int i_cnt = 0;
    unsigned int buff1[16] = {
        0
    };
    unsigned char high1 = 0; // Dot1 상위 bit
    unsigned char low1 = 0; // Dot1 하위 bit
    register unsigned int i = 0;
    for (i_cnt = 0; i_cnt < 16; i_cnt++) {
        if (move_motion == 0) {

```

```

        buff1[i_cnt] = pgm_read_word( & string[first][i_cnt]) <<
Move_cnt;

        // 오른쪽, 도트매트릭스 안으로 이동
    }
    if (move_motion == 1) {
        buff1[i_cnt] = pgm_read_word( & string[first][i_cnt]) >>
Move_cnt2;

        // 오른쪽, 도트매트릭스 밖으로 이동
    }
}
for (i = 0; i < 16; i++) {
    high1 = (buff1[i] >> 8); // Dot1 상위 8bit 저장
    low1 = (buff1[i] & 0xff); // Dot1 하위 8bit 저장
    shift_Register(high1); // 상위 8bit LED 점등
    shift_Register(low1); // 하위 8bit LED 점등
    row_dynamic(i); // 0~15 행 접근
    ActivePulse(); // 스트로브 신호
}
}
void loop() {
    dot1_display_shift(Dot_char_cnt);
}

```

📖 참고지식

• 함수설명

🌈 void shift_Register(unsigned int out);

RGY LED 모듈내에 1열에서 16열까지의 LED를 ON 혹은 OFF 할 때 사용하는 함수이다. 이 함수를 통해 LED 색상 제어가 가능하며 클럭 신호에 동기화 시켜 데이터를 입력하면 각각의 열의 LED를 구동할 수 있다.

각각의 열에 해당하는 LED의 데이터는 클럭과 동기화하여 입력해야 하고 클럭 신호의 상승 엣지에서 R1(RED) 및 G1(GREEN) 데이터가 레지스터에 전송된다. 이때의 A, B, C, D 핀에 신호를 입력하여 각 행을 지정한 후 R1(RED)핀에 1을 입력하면 해당 도트의 LED는 적색으로 점등되고, G1(GREEN)핀에 1을 입력하면 해당 도트의 LED는 녹색으로 점등된다. 또한 R1 및 G1 핀에 동일하게 1을 입력하면 황색 LED가 점등된다.

RGY LED 모듈은 총 256개의 도트를 제어해야 한다. 이를 구현하기 위해서는 16개의 행을 선택하고 또 각각의 행마다 16개 데이터를 입력되어야 한다. 또한 16개 데이터에는 적색 혹은 녹색을 구동할 것인지에 대한 R1, G1 핀 신호 선택도 미리 준비되어야 한다.

RGY LED 모듈 LED 스캔 주기도 고려해야 한다. 각 행마다 머무는 시간이 일정치 않다면 전체 LED 점등이 불안하거나 깜박거림 현상이 나타날 수 있기 때문이다. 이를 해결하기 위해서는 타이머 카운터 인터럽트를 이용하여 스캔 주기의 정확성을 확보해야 한다.

각각의 행마다 16개의 데이터 및 클럭 신호가 입력되면 LATCH 신호에 의하여 데이터가 저장되며 이때 OE(Out Enable)신호를 입력하여 드라이버를 구동시켜 LED가 점등된다.

위에서 설명한 내용을 바탕으로 아래 코드에서는 해당 기능을 구현하고 있다. 만일 RGY LED 모듈을 여러개로 병렬로 접속하는 경우에는 시프트 레지스터가 16비트, 32비트, 48비트, 64비트로 확장되는 것을 연상하며 코드를 작성하도록 한다.


예컨대 16x16 도트매트릭스 RGY LED 모듈 2개를 연결하여 사용하는 경우라면 32개의 데이터를 1행마다 시프트하여 입력해 사용하면 된다. 만일 RGY LED 모듈 4개를 연결하여 사용하는 경우라면 64개의 데이터를 1행마다 시프트하여 입력하면 다수의 LED를 쉽게 확장하여 사용할 수 있게 된다.

```
void LED_Shift(unsigned char out) // 데이터 입력 함수
{
    unsigned char clk = 0;
    for (clk = 0; clk < 8; clk++) { //8 비트 데이터를 한비트씩 입력
        if (out & (0x80 >> clk)) {
            Data_Green = 0;
            Data_Red = 1;
            Clk = 1;
            Clk = 0;
        } else {
            Data_Green = 0;
            Data_Red = 0;
            Clk = 1;
            Clk = 0;
        }
    }
}

void ActivePulse() // 구동 펄스 함수
{
    Latch = 1;
    Latch = 0;
    En = 0;
    delay_us(900);
    En = 1;
}

En=0; //ENABLE LOW
LED_Shift(0x01); //첫번째 도트매트릭스 모듈 0~7
LED_Shift(0x00); //첫번째 도트매트릭스 모듈 8~15

LED_Shift(0x00); //두번째 도트매트릭스 모듈 0~7
LED_Shift(0x00); //두번째 도트매트릭스 모듈 8~15
ActivePulse(); //구동 펄스 출력
En=1; //ENABLE HIGH
```


 void row_dynamic(unsigned int i);

RGY LED 모듈의 행을 제어하는 함수이다. 행 어드레스 핀 A, B, C, D를 통해 각각의 행을 제어하여 원하는 행에 LED를 ON 혹은 OFF할 때 사용된다. 각각의 행의 머무는 시간은 1mS이며

이 시간이 주기적으로 같아야지만 보다 선명한 LED 출력이 가능하다.

 void ActivePulse(void);

RGY LED 모듈 LATCH 핀에 펄스를 입력하여 16개의 LED 데이터를 저장할 때 사용되는 함수이다.

 void dot1_display_shift(unsigned char first)

RGY LED 모듈의 출력 문자를 왼쪽, 오른쪽, 위, 아래 등 원하는 방향으로 시프트하여 디스플레이 하는 함수이다.

END Epilog

기술지원

이 책의 모든 예제들은 시뮬레이션 및 검증이 진행되었습니다. 그러나 소스 코드를 책으로 옮기는 편집 과정에서 오류가 발생되었을 수도 있습니다. 만일 소스 코드에 오류가 발견되거나 이 책의 부족한 부분을 알려 주시면 개정판에서 반영하도록 하겠습니다.

이 책의 예제 소스는 다음의 사이트에서 다운로드할 수 있으며, 책에서 궁금한 부분은 상담 문의 게시판으로 문의 바랍니다.

LK임베디드 홈페이지 문의: WWW.LKEMBEDDED.CO.KR 상담 문의 게시판

저자에게 이메일 기술 문의: LKN9270@LKEMBEDDED.CO.KR

제품 구입 관련 전화문의: Tel. 02-968-8617

A/S 관련 안내

제품과 관련된 모든 기술 문의는 LK임베디드 홈페이지 상담 문의 게시판을 이용해 주시면 최대한 신속하게 답변을 받아 보실 수 있습니다.

제품 초기 불량 시 제품 구입일로부터 7일 이내에 제품 교환 및 환불을 요청할 수 있으며, 학습자 과실로 인하여 하자가 발생하였을 경우에는 수리비가 청구될 수 있고, A/S 기간은 6개월입니다. 또한 AS 관련 제품은 택배 비용 선불로 보내야 합니다.

단순 변심으로 인한 제품 교환 및 환불 요청은 불가능하니 이점 양해바랍니다.

- 문의전화: Tel. 02-968-8617
- LK 홈페이지 주소: WWW.LKEMBEDDED.CO.KR

감사의 글

LK임베디드 제품을 구입해 주셔서 감사합니다. 당사는 아두이노, AVR, PIC, ARM, FPGA를 사용하시는 고객님의 편의를 증진시키기 위해서, 마이컴 교육 및 신제품 연구 개발을 위해서 항상 노력하고 있습니다. 앞으로도 끊임없는 도전 정신을 바탕으로 신제품 개발, 완벽한 품질 보증 체계 확립, 고객 서비스를 통해 고객의 마음을 편하게 하는데 정진할 것입니다. 본 제품을 활용하여 마이컴 학습 및 제품 개발에 큰 도움되시기 바랍니다