Timer / Interrupt

- Timer
- Interrupt

Timer

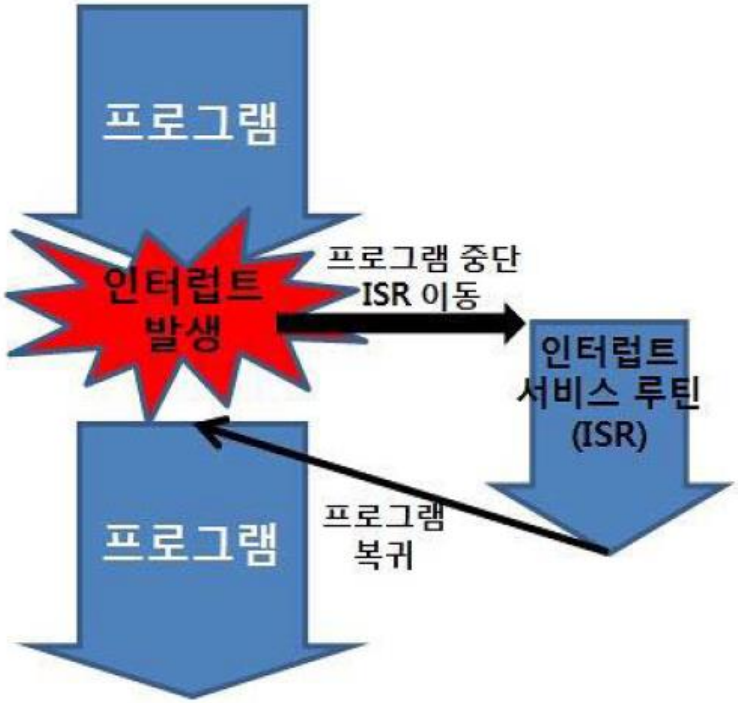
1. 클럭과 카운터

- 동기 디지털 회로에서 클럭(Clock)은 두 개 이상의 회로 동작을 통합(동기화)하는 데에 쓰이는 신호이다.
- 클럭 신호는 높고(High) 낮은(Low) 상태 사이에서 진동하며 시간의 흐름에 따른 변화를 나타낸다.
- 동기화를 위해 클럭 신호를 사용하는 회로는 오르는 부분(상승 에지, Rising Edge), 떨어지는 부분(하강 에지, falling Edge)에서 활성화될 수 있다.
- 클럭은 주어진 일을 정확한 시간에 처리하기 위해 사용된다.
- 일반적으로 높은 신호를 "1"값으로 표시하며 낮은 신호를 "0"값으로 표시하는데, 클럭에서 "0"과 "1"이 번갈아 나타나는 시간의 단위를 주기라고 한다.
- 1초당 몇 번의 주기가 포함되는가를 주파수 라고 한다. 예를 들어, 1MHz라고 하면, 0과 1이 반복되는 주기가 1,000,000번 임을 뜻한다.

2. 인터럽트

- 마이크로프로세서(CPU)가 프로그램을 실행하고 있을 때, 입출력 하드웨어 등의 장치나 내, 외부의 어떤 변화에 의한 예외 상황이 발생하여 처리가 필요할 경우에 마이크로프로세서에게 알려 프로그램의 실행을 정지하고 변화에 대응하는 다른 프로그램을 처리할 수 있도록 하는 것을 말한다.
- 임베디드 시스템의 마이크로 컨트롤러도 비슷한 상황이 많이 발생한다. 여러 가지 일을 처리해야만 하고, 동시에 처리할 수 없는 일들을 순서를 가지고 차근히 처리해 나가야 한다.
- 급한 일은 먼저 처리해 주어야 장치가 원활하게 동작할 수 있을 것이다. 이 급한 일로 인해 인터럽트가 발생한다.
- 인터럽트 발생에 따른 일의 처리는 지금 하고 있는 일과 앞으로 발생되는 일들의 경중에 따라 중요도(Priority)가 결정되고, 지금 하는 일보다 중요도가 낮은 일은 처리가 안될 수도 있다.

2. 인터럽트 (계속)



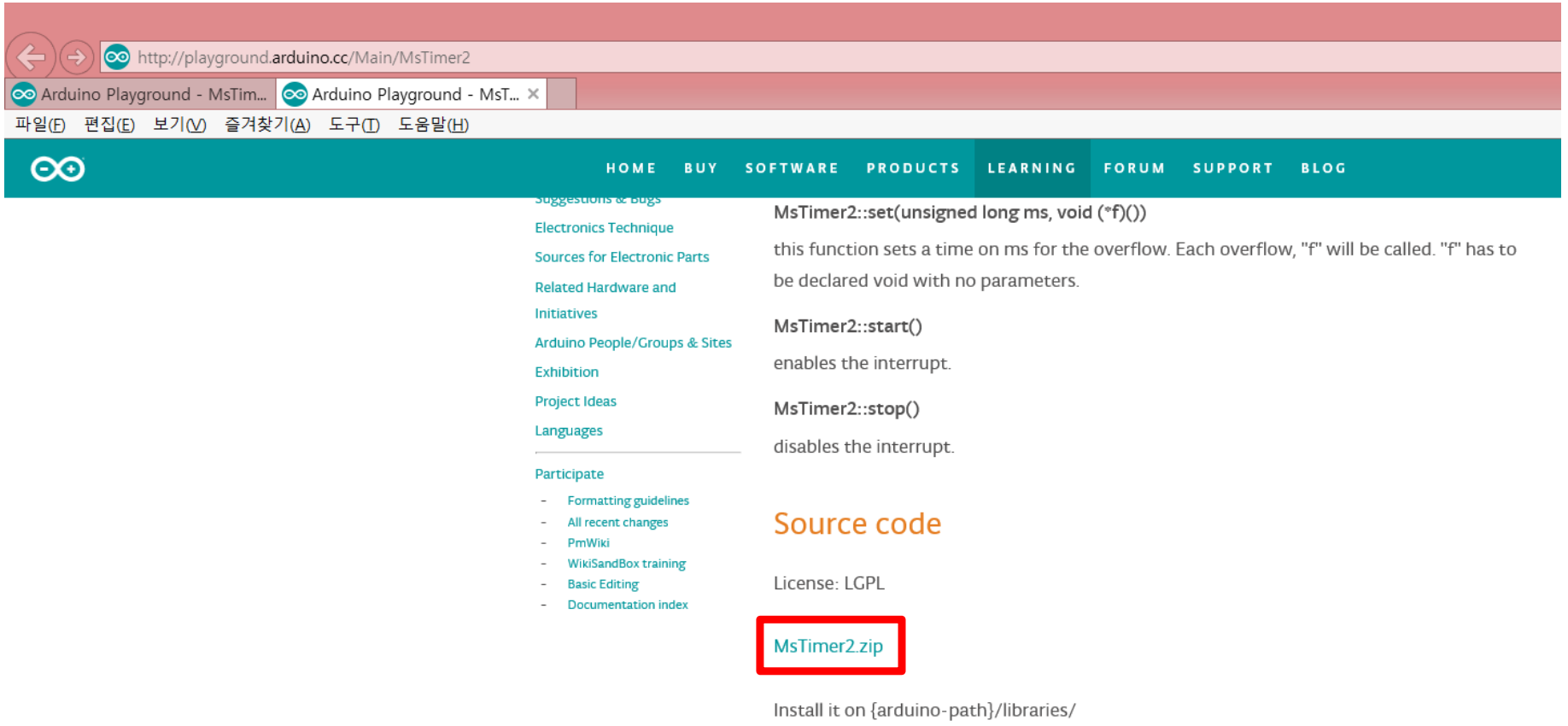
인터럽트가 걸리면 해당 서비스 루틴이 실행되어야 하는데, 현재 진행중인 프로그램이 영향을 받으면 안되므로 서브루틴의 경우처럼 나중에 되돌아 올 복귀 주소가 자동적으로 스택에 저장되었다가 인터럽트 서비스 루틴의 마지막에서 복귀 명령을 만나면 자동으로 저장한 복귀 주소를 찾아 인터럽트 발생 전의 위치로 되돌아 온다.

3. 타이머를 이용하여 LED 제어하기

- 내부 클럭을 사용하여 타이머를 동작 시켜서 LED를 제어하는 실험
- delay() 함수
  - ☺ 소프트웨어로 구현되어 있어서 정확한 시간 제어가 어렵고 지연 시간 동안에 원하지 않는 작업을 수행함으로 MCU의 동작을 정지시켜 원하는 다른 작업을 수행하는데 어려움이 있다.
- 타이머/카운터
  - ☺ 레지스터에 시간을 설정하고 원하는 시간이 되면 작업을 수행하므로 다른 작업과 병행하여 동작하여 일의 효율성을 높일 수 있다.
  - ☺ 하지만 이 타이머/카운터를 사용하여 제어하는 방법을 사용하려면 Atmega2560의 내부 레지스터 설정까지 알아야 한다.
  - ☺ 따라서 Timer를 사용하기 위해 MsTimer2라는 라이브러리를 사용한다.

참고: MsTimer2 라이브러리 다운로드

- 홈페이지 http://playground.arduino.cc/Main/MsTimer2 에 접속
- 아래쪽 Source code -> MsTimer2.zip 파일 클릭 후 다운로드



3.1 TimerLED 예제 핀 배열

Device	MegaADK	LED MODULE
PIN NO	14	RED 1
PIN NO	15	RED 2
PIN NO	16	RED 3
PIN NO	17	RED 4
PIN NO	18	RED 5
PIN NO	19	RED 6
PIN NO	20	RED 7
PIN NO	21	RED 8

PIN NO	0	GREED 9
PIN NO	1	GREED 10
PIN NO	2	GREED 11
PIN NO	3	GREED 12
PIN NO	4	GREED 13
PIN NO	5	GREED 14
PIN NO	6	GREED 15
PIN NO	7	GREED 16

3.2 TimerLED 예제

```
#include <MsTimer2.h> //타이머 관련 라이브러리 선언
int Ledpin[16] = {14,15,16,17,18,19,20,21,0,1,2,3,4,5,6,7};  // 핀 배열로 정의
int Counter = 0; //카운터 관련 변수 선언
void setup(){
        // PinMode()함수를 통해 LED와 연결된 핀을 출력으로 설정
        for(int i = 0; i<16 ; i++)
        {
                pinMode(Ledpin[i],OUTPUT);
        }
        MsTimer2::set(500, ledcontrol); // 타이머 인터럽트 설정, 500ms주기
        MsTimer2::start(); // 타이머 시작
}
```

3.2 TimerLED 예제 (계속)

```
//타이머 인터럽트 서비스 루틴
void ledcontrol(){
        static boolean output = HIGH;  // 변수 초기화
        // digitalWrite() 함수로 LED 와 연결된 핀 변수 값 출력
        digitalWrite(Ledpin[Counter],output);
        Counter++; //카운터 변수 값 증가

        //카운터 값이 15 보다 크면 카운터 값을 0 으로 초기화 후 output 변수를 반전
        if(Counter>15)
        {
                Counter = 0;
                output = !output;
        }
}

void loop(){
}
```

Interrupt

1. attachInterrupt()

- 인터럽트가 발생할 때 호출될 인터럽트 서비스 루틴(ISR)을 지정한다.
- 보드에서 사용할 수 있는 인터럽트 핀

BOARD	Interrupt 0	Interrupt 1	Interrupt 2	Interrupt 3	Interrupt 4	Interrupt 5
Uno,Ethenet	2	3	-	-	-	-
MEGA2560	2	3	21	20	19	18
레오나르도	3	2	0	1	7	-

- 인터럽트 서비스 루틴에서는 delay()가 작동하지 않기 때문에 주의하여 코딩 해야 한다.
- 일반적으로 인터럽트 서비스 루틴은 가능할 짧고 빠르게 처리해야 한다.

1. attachInterrupt() (계속)

- attachInterrupt()의 3개의 인자
  - ☺ 첫 번째 인자 : Interrupt number
  - ☺ 두 번째 인자 : 인터럽트 서비스 루틴
  - ☺ 세 번째 인자 : mode
    - LOW - 핀이 LOW일 때 인터럽트 발생
    - CHANGE - 핀의 값이 변경될 때 인터럽트 발생
    - RISING - 핀에 LOW에서 HIGH로 변경될 때 인터럽트 발생
    - FALLING - 핀이 HIGH에서 LOW로 변경될 때 인터럽트 발생
    - HIGH - 핀이 HIGH 일 때 인터럽트 발생(Arduino Due 만 해당)

1. attachInterrupt() (계속)

```
int pin = 13; // 핀 번호 변수에 초기화
Volatile int state = LOW; // state 변수에 LOW로 초기화
Void setup(){
        pinMode(pin, OUTPUT); // 13번 핀을 출력으로 설정
        attachInterrupt(0, blink, CHANGE); // 외부 인터럽트 설정
}
Void loop(){
        digitalWrite(pin, state); digitalWrite함수로 13번 핀에 state 출력
}
Void blink(){
        state = !state; // state를 반전하여 state에 저장
}
```

- **detachInterrupt()**
  - ☺ 지정된 인터럽트를 해제한다. 사용방법은 detachInterrpt(인터럽트 번호); 이다.
- **interrupts()**
  - ☺ 전체 인터럽트를 활성화 시키는 함수. 인터럽트는 어떤 중요한 작업이 백그라운드에서 수행할 수 있도록 기본적으로 활성화되어 있다.
- **noInterrupts()**
  - ☺ 전체 인터럽트를 비활성화 시키는 함수

1.1 InterruptControl 예제 핀 배열

- Keypad 모듈과 LED 모듈을 이용하여 버튼 입력을 인터럽트로 처리하여 LED를 제어하는 프로그램

LED 핀 결선

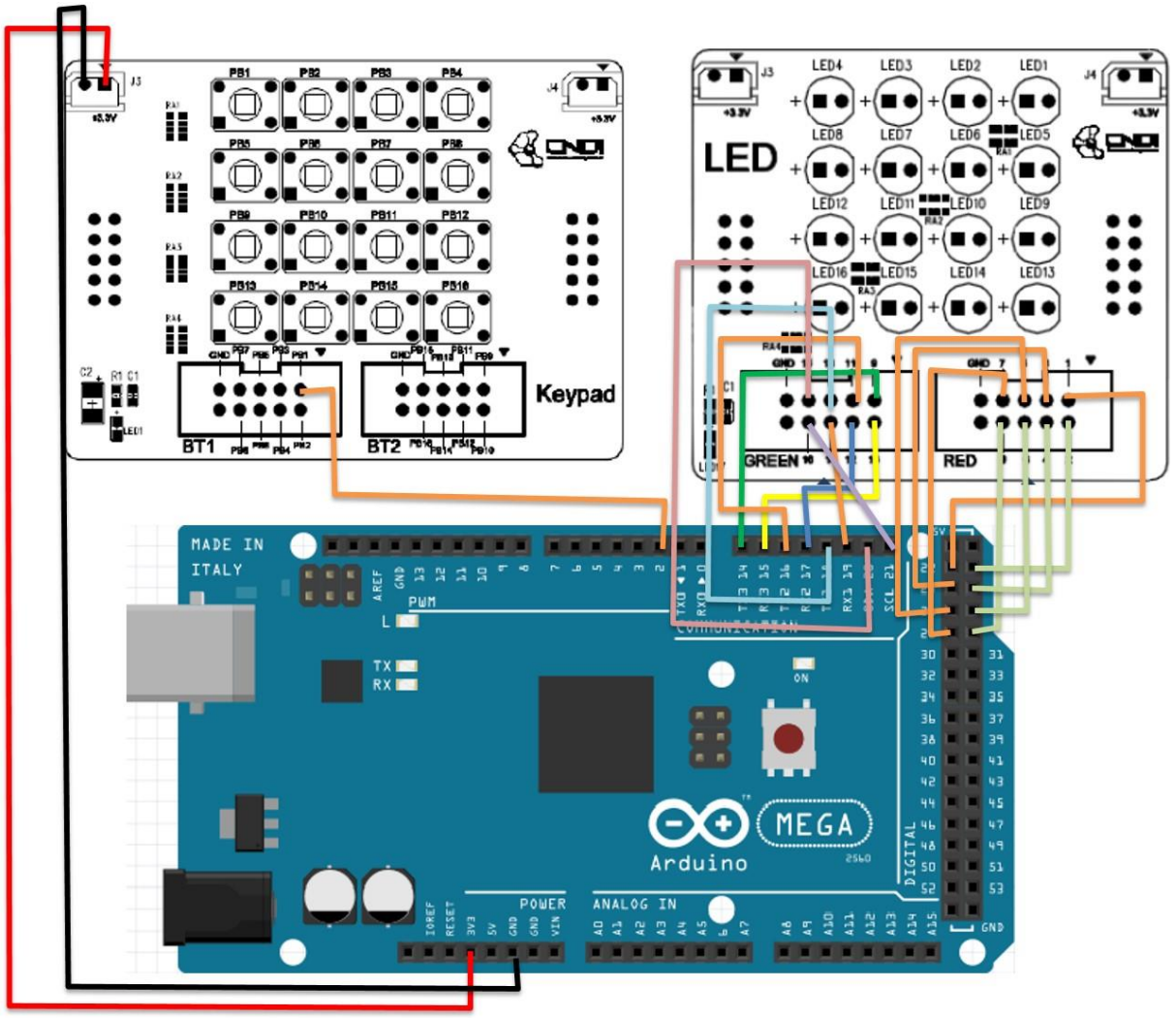
Device	MegaADK	LED MODULE
PIN NO	22	RED 1
PIN NO	23	RED 2
PIN NO	24	RED 3
PIN NO	25	RED 4
PIN NO	26	RED 5
PIN NO	27	RED 6
PIN NO	28	RED 7
PIN NO	29	RED 8

PIN NO	30	GREED 9
PIN NO	31	GREED 10
PIN NO	32	GREED 11
PIN NO	33	GREED 12
PIN NO	34	GREED 13
PIN NO	35	GREED 14
PIN NO	36	GREED 15
PIN NO	37	GREED 16

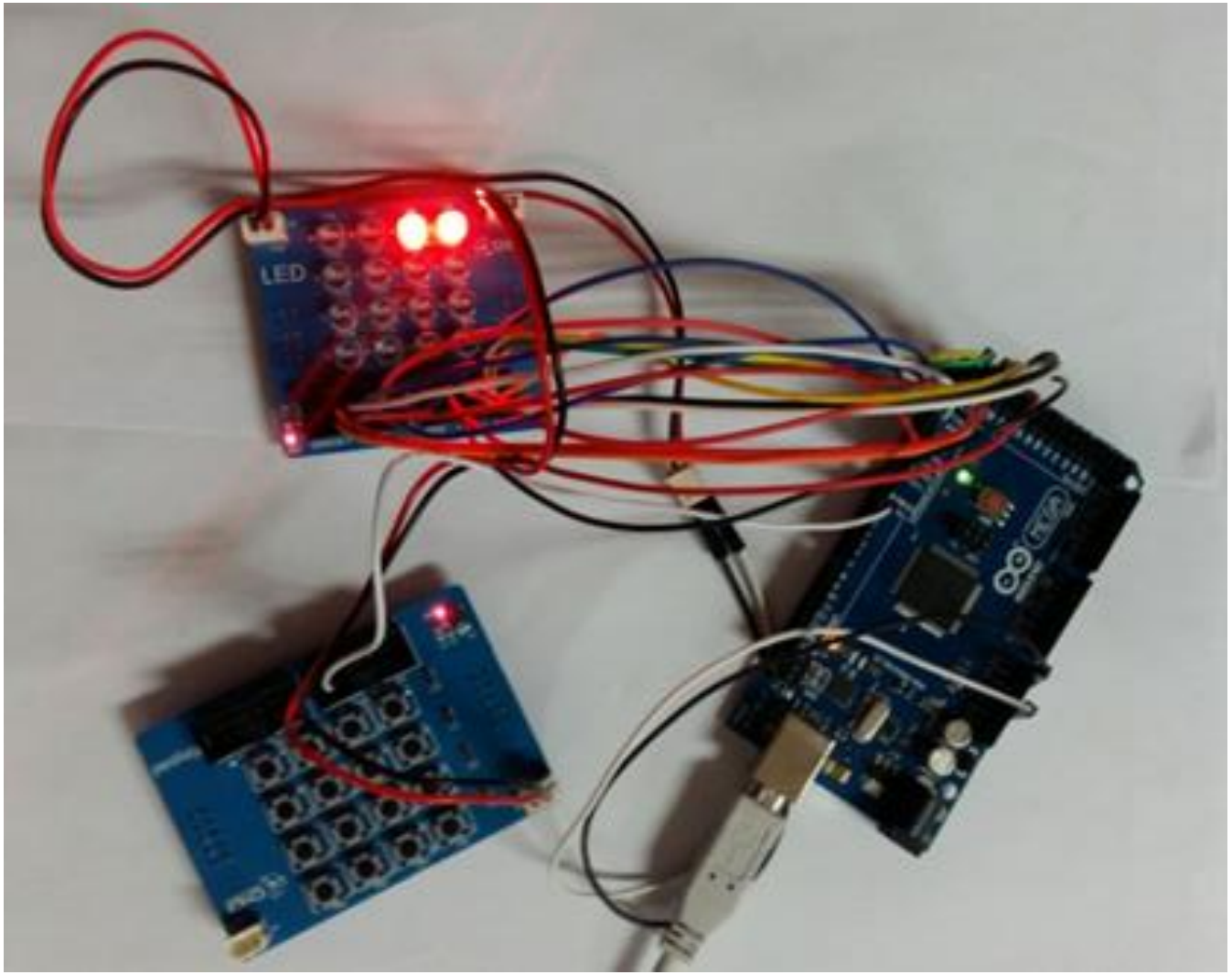
Keypad 핀 결선

Device	MegaADK	Keypad MODULE
PIN NO	2	BT1 PB1

1.2 InterruptControl 예제 핀 결선



1.3 InterruptControl 예제 실제 핀 결선



1.4 InterruptControl 예제

```
#include <MsTimer2.h> //타이머 관련 라이브러리 선언
//LED와 연결된 핀 배열로 정의
int ledpin[16] = {22,23,24,25,26,27,28,29,14,15,16,17,18,19,20,21};

volatile int counter = 0; //인터럽트 관련 변수 선언
volatile byte state = HIGH; //인터럽트 관련 변수 선언

void setup(){
        Serial.begin(9600);
        // LED와 연결된 핀을 출력으로 설정
        for(int i = 0; i<16; i++)
                pinMode(ledpin[i],OUTPUT);

        // 외부인터럽트 설정
        // 외부인터럽트 0번은 디지털 2번 핀에서 발생한다.
        attachInterrupt(0, LedControl, FALLING);
        MsTimer2::set(800, Interruptcontrol); // 타이머 설정
}
```

1.4 InterruptControl 예제 (계속)

```
void Interruptcontrol(){
        attachInterrupt(0,LedControl,FALLING); //외부인터럽트 설정
        MsTimer2::stop(); //타이머 정지
}

void LedControl(){
        counter++; //카운터 변수 증가
        MsTimer2::start(); //타이머 카운트 시작
        detachInterrupt(0); //외부인터럽트 0번 중지
        Serial.println(counter);

        // 카운터 값이 15보다 크면 카운터 변수를 0으로 초기화 하고
        // state 변수를 반전
            시킨다.
        if(counter>15){
                counter = 0;
                state = !state;
        }
}
```

1.4 InterruptControl 예제 (계속)

```
void loop(){
        digitalWrite(ledpin[counter], state); //digitalWrite()함수로 LED 제어
        delay(200); //지연 200ms
}
```

1.5 InterruptControl 예제 설명

- 외부 인터럽트 0번 즉 digital 2번 핀에 입력되는 버튼의 Falling edge(HIGH 에서 LOW 변화되는 신호) 신호를 인터럽트로 인식하여 LED를 제어
- 버튼이 입력되는 횟수를 카운트하여 그 카운트 값만큼 LED를 제어
- 카운트 값이 16개가 넘어가게 되면 카운트를 0으로 초기화 하고 LED에 출력되는 신호를 반전시킨다.
- 버튼을 한번 누를 때마다 LED가 한 개씩 점등되고 LED 모듈의 LED가 다 점등된 상태에서 다시 버튼을 누르면 하나씩 소등되는 동작