

Timer Interrupts

Lesson 07

Arduino Uno는 3개의 내장 타이머를 가짐

• **Timer0:**

- 8 bit 타이머: 0~255까지 카운트 가능
- `delay()`, `millis()`, `micros()` 함수가 이 타이머를 사용
- 따라서 `timer0` 레지스터를 직접 컨트롤하면 이러한 함수들이 제대로 작동하지 않음

• **Timer1:**

- 16 bit 타이머: 0~65535까지 카운트 가능
- Servo library가 사용

• **Timer2:**

- 8 bit 타이머
- `tone()` 함수가 사용

TimerOne 라이브러리

- 타이머를 직접 제어하는 것은 다소 복잡함
- TimerOne 서드파티 라이브러리를 사용하면 **ATMega 328 기반의 아두이노에서 16비트 타이머(Timer1)를 제어할 수 있음**
 - Timer1은 Servo 라이브러리 및 아두이노의 9번, 10번의 PWM을 위해서 사용됨
 - 따라서 아두이노 프로그램에서 TimerOne라이브러리를 사용하면 9번과 10번 핀에서는 analogWrite() 함수를 실행할 수 없고, Servo 라이브러리도 사용할 수 없다.


TimerOne 라이브러리

다운로드

- <http://code.google.com/p/arduino-timerone/downloads/list>

아두이노의 라이브러리 폴더로 복사



- Sketch->IncludeLibrary->Add .ZIP Library 메뉴를 이용

 **arduino-timerone**
Timer One Library for Arduino

Search projects

[Project Home](#) [Downloads](#) [Wiki](#) [Issues](#) [Source](#)

Search Current downloads ▼ for

1 - 2 of 2						
Filename ▼	Summary + Labels ▼	Uploaded ▼	ReleaseDate ▼	Size ▼	DownloadCount ▼	...
 TimerOne-r11.zip	Timer One r11 Featured	Oct 2013	Oct 2013	11.2 KB	40448	
 TimerOne-v9.zip	TimerOne-v9	Feb 2012	Feb 2012	5.7 KB	48961	

1 - 2 of 2

실습 01: TimerOne라이브러리를 이용하여 LED 깜빡이기

```
#include <TimerOne.h>
```

```
const int LED=13; ← 13번 핀에 LED 연결, 저항(300 Ω정도) 까먹지 말것
```

```
void setup()
{
    pinMode(LED, OUTPUT);
    Timer1.initialize(1000000); // 타이머를 1000000us(1초)로 설정
    Timer1.attachInterrupt(blinky); // 타이머 인터럽트가 발생할 때마다 blinky()실행
}

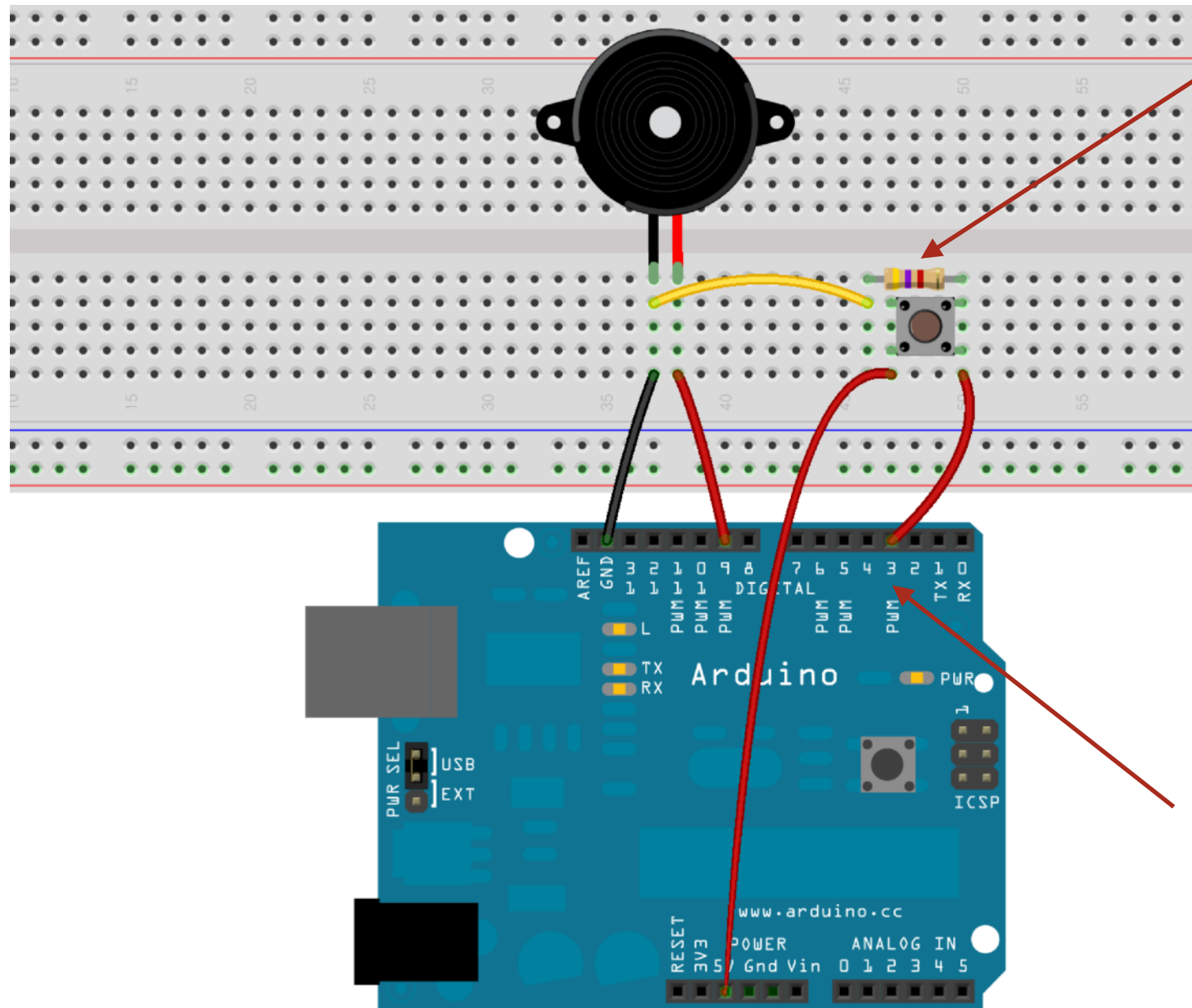
void loop()
{
    // Put any other code here
}

void blinky()
{
    digitalWrite(LED, !digitalRead(LED)); //LED 켜짐 상태 바꿈
}
```

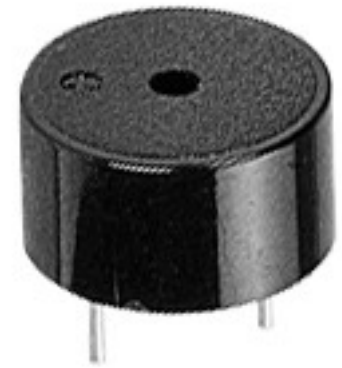
실습 02: 인터럽트로 작동하는 사운드 시스템

- **스피커의 음을 도부터 시까지 순서대로 출력하는 사운드 시스템을 만들어보자.**
 - 타이머 인터럽트는 현재 선택된 음의 모든 옥타브를 순차적으로 출력한다.
 - 버튼을 누를 때마다 다음 음으로 이동한다.
- **changeKey() 함수:**
 - 버튼 인터럽트를 호출할 때마다 실행하며 다음 음으로 이동할 수 있도록 주파수 값을 변경한다.
- **changePitch() 함수:**
 - 타이머 인터럽트로 0.5초마다 호출하며 tone()함수를 사용하여 스피커에서 특정 주파수의 음을 출력한다.
- **loop문:**
 - 0.1초마다 현재 음계, 옥타브, 주파수를 시리얼 모니터로 출력한다.

인터럽트로 작동하는 사운드 시스템 제작하기



풀다운 레지스터($10\text{ K}\Omega$)



Piezo Buzzer
(square wave를 소리로 변환)

Int1을 사용하기 위해서
3번 핀에 연결
(참고: Int0는 2번 핀)

사운드 시스템 프로그램 작성하기

```
#include <TimerOne.h>           //TimerOne 라이브러리 삽입

const int BUTTON_INT=1;          // 3번 핀을 사용하는 인터럽트 버튼 상수 정의
const int SPEAKER=9;             // 9번 핀을 사용하는 스피커 상수 정의

//음계 설정
#define NOTE_C 65
#define NOTE_D 73
#define NOTE_E 82
#define NOTE_F 87
#define NOTE_G 98
#define NOTE_A 110
#define NOTE_B 123

//인터럽트 내에서 값이 변경될 휘발성 변수 선언
volatile int key= NOTE_C;
volatile int octave_multiplier=1;
```


사운드 시스템 프로그램 작성하기

```
void setup()  
{  
    Serial.begin(9600);  
    pinMode(SPEAKER, OUTPUT);  
  
    //RISING EDGE를 감지할 수 있도록 BUTTON_INT 핀 신호를 인터럽트로 설정  
    attachInterrupt(BUTTON_INT, changeKey, RISING);  
  
    //타이머 인터럽트 설정  
    Timer1.initialize(500000); //0.5초로 타이머 설정  
    Timer1.attachInterrupt(changePitch); //타이머 인터럽트가 발생하면  
                                         //changePitch() 실행  
}
```

사운드 시스템 프로그램 작성하기

```
void changeKey()  
{  
    octave_multiplier=1;  
    if(key==NOTE_C)  
        key=NOTE_D;  
    else if(key==NOTE_D)  
        key=NOTE_E;  
    else if(key==NOTE_E)  
        key=NOTE_F;  
    else if(key==NOTE_F)  
        key=NOTE_G;  
    else if(key==NOTE_G)  
        key=NOTE_A;  
    else if(key==NOTE_A)  
        key=NOTE_B;  
    else if(key==NOTE_B)  
        key=NOTE_C;  
}
```

사운드 시스템 프로그램 작성하기

//타이머 인터럽트 실행

```
void changePitch()
```

```
{  
    octave_multiplier = octave_multiplier*2;  
    if(octave_multiplier>16) octave_multiplier= 1;  
    tone(SPEAKER, key*octave_multiplier);  
}
```

Generates a square wave of the specified frequency (and 50% duty cycle) on a pin.

```
void loop()
```

```
{  
    Serial.print("Key: ");  
    Serial.print(key);  
    Serial.print(" Multiplier: ");  
    Serial.print(octave_multiplier);  
    Serial.print(" Frequency: ");  
    Serial.println(key*octave_multiplier);  
    delay(100);  
}
```

Arduino Timer 직접 제어하기

Clear Timer on Compare Match (CTC) 모드로 타이머 직접 제어하기

- 타이머(TCNT 레지스터의 값)는 아두이노 클록 사이클마다 1씩 증가
 - 8비트 타이머는 0에서 최대 $2^8 - 1$ 까지 증가
 - 16비트 타이머는 0부터 최대 $2^{16} - 1$ 까지 증가
- **Output Compare Register (OCR) 레지스터에 저장된 값에 도달하면 0으로 초기화되면서 인터럽트를 발생**

- **ATmega328은 16MHz로 동작**
- **타이머는 255 (8비트) 혹은 65535 (16비트) 까지만 표현할 수 있으므로 매 클록마다 카운터가 증가되면 너무 빨라서 사용하기 어려움**
- **8, 64, 256, 혹은 1024 클록마다 카운터가 1 증가하도록 설정할 수 있음. 이 값을 prescaler라고 부름**
- **prescaler는 TCCRx 레지스터에서 설정가능**

Clock select and timer frequency

예를 들어 **timer1**에서 **2Hz**로 인터럽트를 발생하려면

1. CPU frequency 16Mhz for Arduino
2. maximum timer counter value (256 for 8bit, 65536 for 16bit timer)
3. Divide CPU frequency through the choosen prescaler ($16000000 / 256 = 62500$)
4. Divide result through the desired frequency ($62500 / 2\text{Hz} = 31250$)
5. Verify the result against the maximum timer counter value ($31250 < 65536$ success) if fail, choose bigger prescaler.

Timer 설정 레지스터: TCCR_x

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00

Timer/Counter Control Register 0 A

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
TCCR0B	FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00

Timer/Counter Control Register 0 B

Timer 설정 레지스터: TCCRx

MODE	WGM02	WGM01	WGM00	DESCRIPTION	TOP
0	0	0	0	Normal	0xFF
1	0	0	1	PWM, Phase Corrected	0xFF
2	0	1	0	CTC	OCR0A
3	0	1	1	Fast PWM	0xFF
4	1	0	0	Reserved	-
5	1	0	1	Fast PWM, Phase Corrected	OCR0A
6	1	1	0	Reserved	-
7	1	1	1	Fast PWM	OCR0A

CTC
mode

Waveform Generator Mode bits

CS02	CS01	CS00	DESCRIPTION
0	0	0	Timer/Counter0 Disabled
0	0	1	No Prescaling
0	1	0	Clock / 8
0	1	1	Clock / 64
1	0	0	Clock / 256
1	0	1	Clock / 1024
1	1	0	External clock source on T0 pin, Clock on Falling edge
1	1	1	External clock source on T0 pin, Clock on rising edge

prescaler

CS bits

TIMSKn 레지스터

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
TIMSK0	-	-	-	-	-	OCIE0B	OCIE0A	TOIE0

Timer/Counter Interrupt Mask Register

enable interrupt

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
TIFR0	-	-	-	-	-	OCF0B	OCF0A	TOV0

Timer/Counter Interrupt Flag Register

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
TCNT0								

Timer/Counter Register (stores the counter value)

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
OCR0A								

Output Compare Register

Toggle Three LEDs using Three Timers

```
//timer0 will interrupt at 2kHz  
//timer1 will interrupt at 1Hz  
//timer2 will interrupt at 8kHz
```

```
//storage variables  
boolean toggle0 = 0;  
boolean toggle1 = 0;  
boolean toggle2 = 0;
```

```
void setup(){
```

```
    //set pins as outputs  
    pinMode(8, OUTPUT);  
    pinMode(9, OUTPUT);  
    pinMode(13, OUTPUT);
```

← 3개의 핀에 LED 연결

```
    cli();//stop interrupts ← clear global interrupt flag
```

Toggle LED

```
//set timer0 interrupt at 2kHz
TCCR0A = 0; // set entire TCCR2A register to 0
TCCR0B = 0; // same for TCCR2B
TCNT0 = 0; //initialize counter value to 0

// set compare match register for 2khz increments
OCR0A = 124; // = (16*10^6) / (2000*64) - 1 (must be <256)

// turn on CTC mode
TCCR0A |= (1 << WGM01); ← CTC 모드

// Set CS01 and CS00 bits for 64 prescaler
TCCR0B |= (1 << CS01) | (1 << CS00);

// enable timer compare interrupt
TIMSK0 |= (1 << OCIE0A);
```

Toggle LED

```
//set timer1 interrupt at 1Hz
TCCR1A = 0; // set entire TCCR1A register to 0
TCCR1B = 0; // same for TCCR1B
TCNT1  = 0; // initialize counter value to 0

// set compare match register for 1hz increments
OCR1A = 15624; // = (16*10^6) / (1*1024) - 1 (must be <65536)

// turn on CTC mode
TCCR1B |= (1 << WGM12);

// Set CS12 and CS10 bits for 1024 prescaler
TCCR1B |= (1 << CS12) | (1 << CS10);

// enable timer compare interrupt
TIMSK1 |= (1 << OCIE1A);
```

Toggle LED

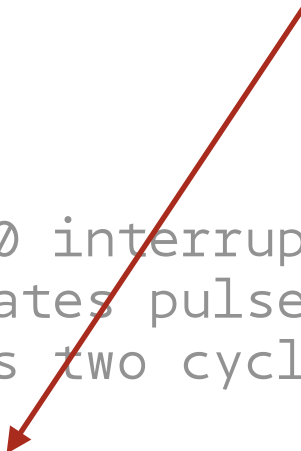
```
//set timer2 interrupt at 8kHz
TCCR2A = 0; // set entire TCCR2A register to 0
TCCR2B = 0; // same for TCCR2B
TCNT2  = 0; // initialize counter value to 0
// set compare match register for 8kHz increments
OCR2A = 249; // = (16*10^6) / (8000*8) - 1 (must be <256)
// turn on CTC mode
TCCR2A |= (1 << WGM21);
// Set CS21 bit for 8 prescaler
TCCR2B |= (1 << CS21);
// enable timer compare interrupt
TIMSK2 |= (1 << OCIE2A);

sei(); //allow interrupts

} //end setup
```

Toggle LED

AVR Interrupt Vector Number
for Timer/Counter0 Compare Match A



```
// timer0 interrupt 2kHz toggles pin 8
// generates pulse wave of frequency 2kHz/2 = 1kHz
// (takes two cycles for full wave- toggle high then toggle low)

ISR(TIMER0_COMPA_vect){
    if (toggle0){
        digitalWrite(8,HIGH);
        toggle0 = 0;
    }
    else{
        digitalWrite(8,LOW);
        toggle0 = 1;
    }
}
```

Toggle LED

```
//timer1 interrupt 1Hz toggles pin 13 (LED)
//generates pulse wave of frequency 1Hz/2 = 0.5kHz
// (takes two cycles for full wave- toggle high then toggle low)

ISR(TIMER1_COMPA_vect){
    if (toggle1){
        digitalWrite(13,HIGH);
        toggle1 = 0;
    }
    else{
        digitalWrite(13,LOW);
        toggle1 = 1;
    }
}
```


Toggle LED

```
//timer1 interrupt 8kHz toggles pin 9
//generates pulse wave of frequency 8kHz/2 = 4kHz
//(takes two cycles for full wave- toggle high then toggle low)

ISR(TIMER2_COMPA_vect){
    if (toggle2){
        digitalWrite(9,HIGH);
        toggle2 = 0;
    }
    else{
        digitalWrite(9,LOW);
        toggle2 = 1;
    }
}

void loop(){
    //do other things here
}
```