NS-RX231를 위한 소프트웨어가이드

<스피커>

목차

1 프로젝트 가져오기

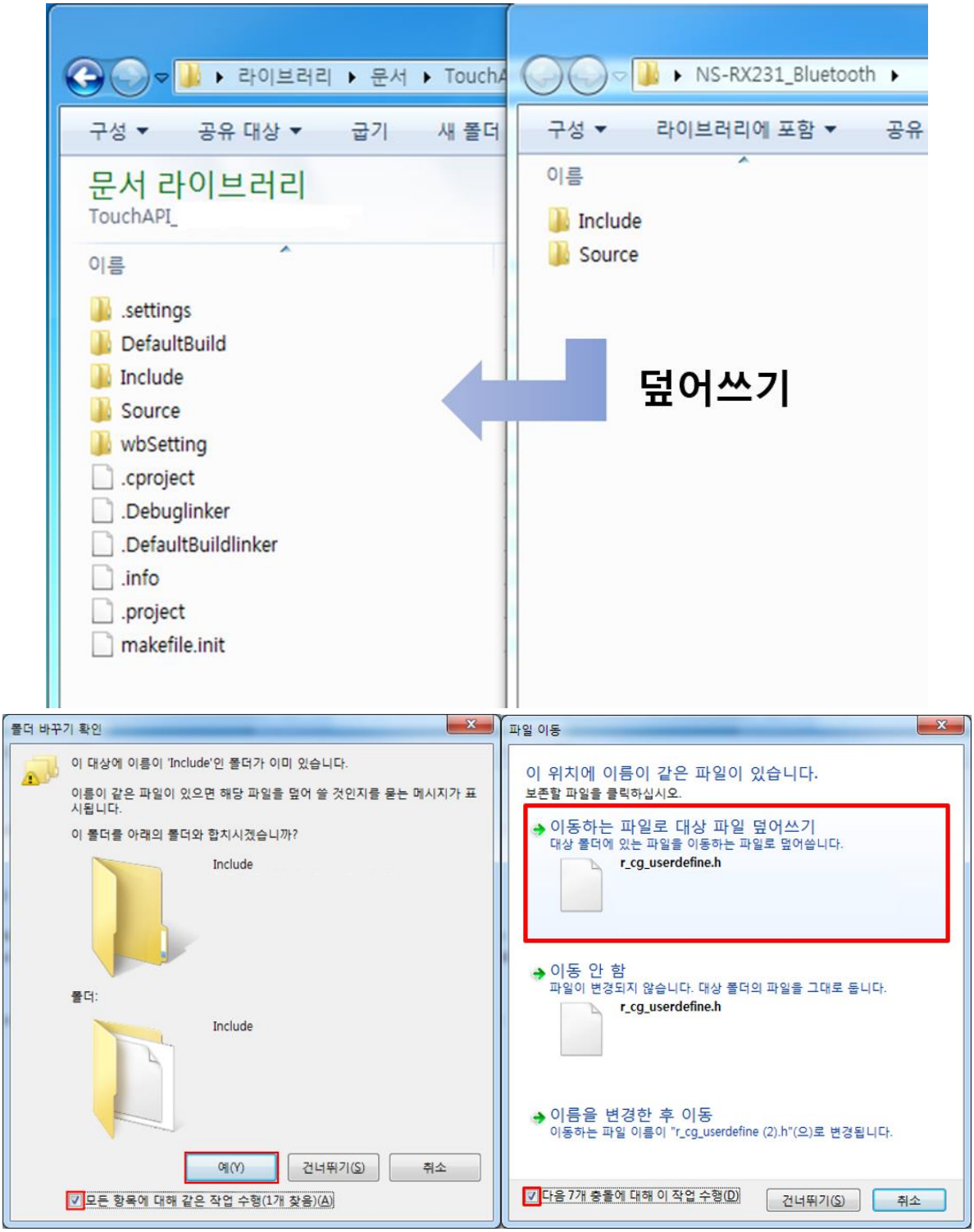


그림 1-1 소스파일 덮어쓰기

첨부된 소스파일을 Workbench6 First step guide 마법사로 만든 프로젝트에 덮어쓰기 한 후, e2studio에서 실행합니다.

2 개요

8 비트 타이머가 4 개 있으면 두 개씩 묶어 16 비트 타이머로 사용할 수 있습니다. 16 비트 타이머의 최대 장점은 8 비트 타이머의 TCNT 값의 범위는 0~255(2^8-1)인데 반해, 16 비트 타이머는 0~65535(2^{16}-1)가 되는 것입니다. 지정할 수 있는 범위가 넓어지기 때문에 타이머 설정시간을 세밀하게 조정할 수 있습니다.

타이머는 내부의 카운터 값을 증가시켜 일정 값이 되는 시점에 인터럽트를 발생시킬 수 있습니다. 예를 들어, 오버플로우 인터럽트 같은 경우에는 카운터 값이 최대값을 넘어가는(오버플로우) 시점에서 인터럽트가 발생합니다.

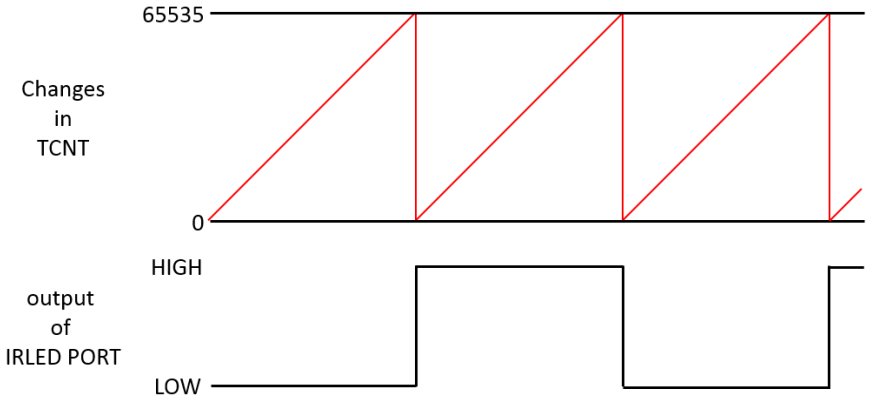


그림 2-1 오버플로우 인터럽트의 예시

일치 인터럽트의 경우, 비교 값을 지정하여 카운터가 지정한 비교값과 일치하는 시점에 인터럽트가 발생하게 됩니다.

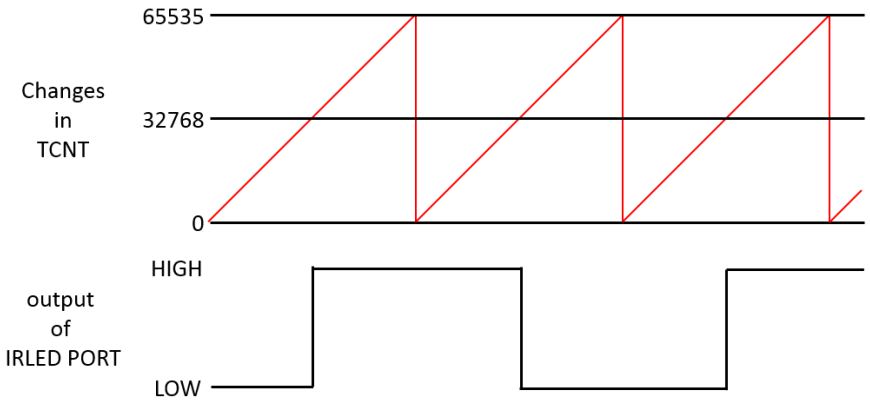


그림 2-2 비교 인터럽트 수행 카운터 리셋 안함

일치 인터럽트를 사용하지만 동시에 카운터를 재설정 해주지 않으면 그림 2-2 와 같이 되게 됩니다. 그림 2-1 의 오버플로우 인터럽트에 비해 발생시기가 빠른 것 만으로는 기대하는 결과를 얻을 수 없습니다.

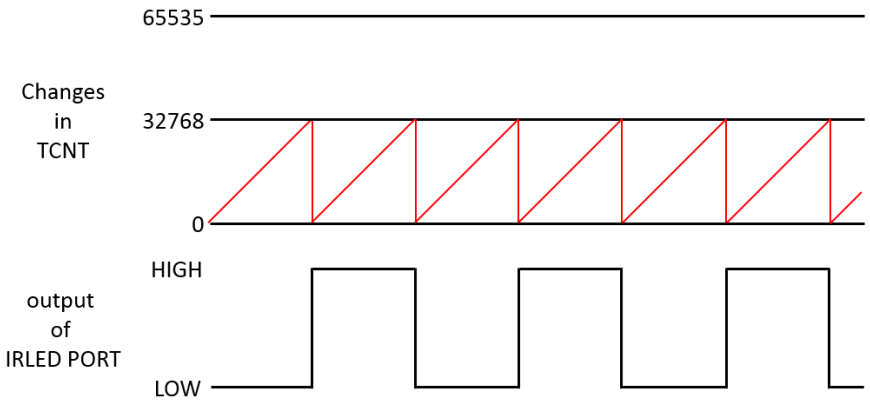


그림 2-3 비교 인터럽트 수행 카운터 리셋

위의 그림과 같이 일치 인터럽트 실행과 동시에 카운터를 재설정하면 원하는 간격으로 인터럽트를 발생시킬 수 있게 됩니다.

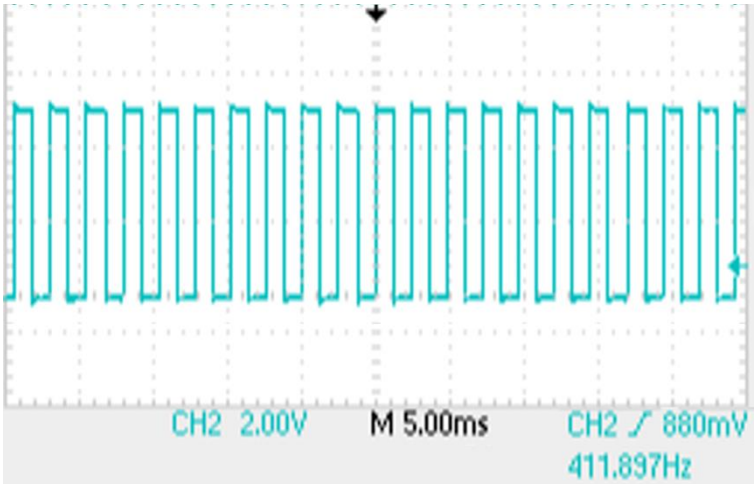


그림 2-4 값 32768을 넣었을 때 오실로스코프의 파형

$$\text{TCORA value} * \text{Desired Frequency[Hz]} = \text{Timer Operating Frequency[Hz]}$$

$$7\text{FFF}(32768 - 1) * 411.897[\text{Hz}] = 13497040[\text{Hz}]$$

피아노 음계 중 중간 도(C)를 예로 들면, 523hz 의 주파수를 가지고 있습니다. 1hz 는 1 초에 1 회 진동한다는 뜻을 가지고 있습니다. 즉, 중간 도(C)를 표현하려면, 1 초에 523 회 진동시켜야 합니다. 소리를 표현하기 위해서는 타이머의 클리어 시간을 조절하는 것으로 구현합니다.

그리고 다음의 계산식에서 원하는 주파수를 생성 할 수 있습니다.

$$\frac{\text{Timer Operating Frequency[Hz]}}{\text{Desired Frequency[Hz]}} / \text{Prescaler} = \text{TCORA value}$$

$$\frac{13497040[\text{Hz}]}{523[\text{Hz}]} / 1 = 25806$$

3 소스코드

```
void request(touch_result_t ts_result){
    uint8_t i, count = 0;
    uint8_t ledValue[16] = {0,};

    ledValue[0] = 0x01;
    ledValue[1] = 0x02;
    ledValue[7] = 0x08;
    ledValue[8] = 0x20;
    ledValue[12] = 0x40;
    ledValue[15] = 0x80;

    PORTB.PODR.BYTE = 0x00;

    if(ts_result.button[0] >> 15){
        PORTB.PODR.BYTE = ledValue[0];
        count++;
    }
    for(i = 0 ; i < 16 ; i++){
        if((ts_result.button[1] >> i) & 0x01){
            PORTB.PODR.BYTE |= ledValue[i+1];
            count++;
        }
    }
    tone(count);
}

void tone(uint8_t temp){
    uint16_t scaleArr[6] = {523,587,659,699,784,880};
    if(temp != 0){
            result = 13497061/scaleArr[temp-1];
            TMR01.TCORA = result;
            tmrLength = 19+temp*3;
        }
}
```

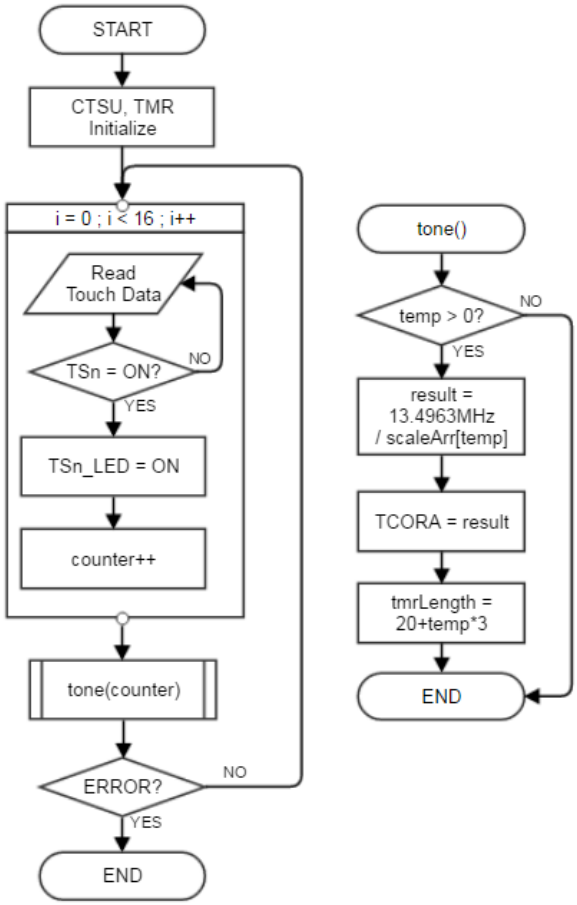


그림 3-1 소스코드 및 순서도

원하는 주파수에 맞춘 TCORA 값의 변화에 따라, tmrLength 를 조정합니다.
타이머는 계속 실행중인 상태이며, tmrLength 가 1 이상일 경우 tmrLength 를 1 씩 빼며 소리를 내고 있는 시간을 유지합니다.

버튼을 터치하면 변수 count 가 1 증가합니다. 멀티터치는 터치 한 만큼 count 값이 커지고, 그에 대응하여 도(C), 레(D), 미(E), 파(F), 솔(G), 라(A)로 주파수가 올라갑니다.

4 디버깅

그림 4-1 NS-RX231을 전원 어댑터와 E1디버거를 연결한 모습.

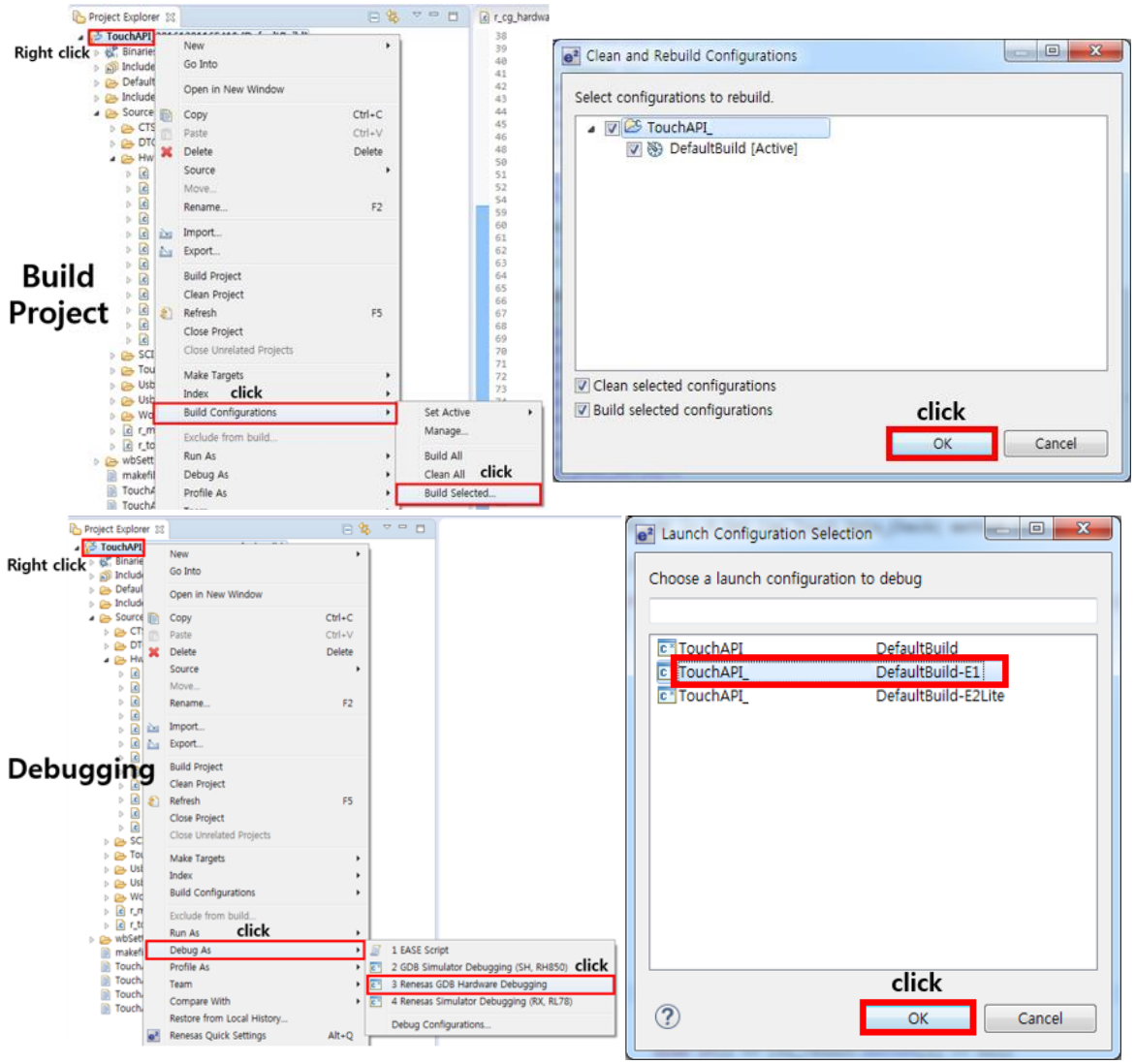
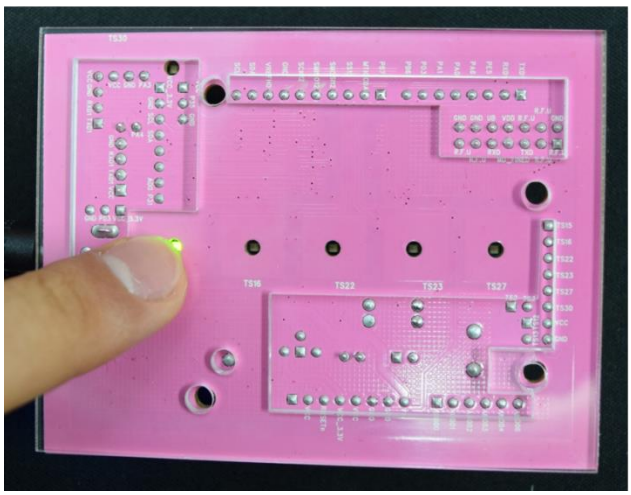


그림 4-2 프로젝트 빌드 및 디버깅

5 실행



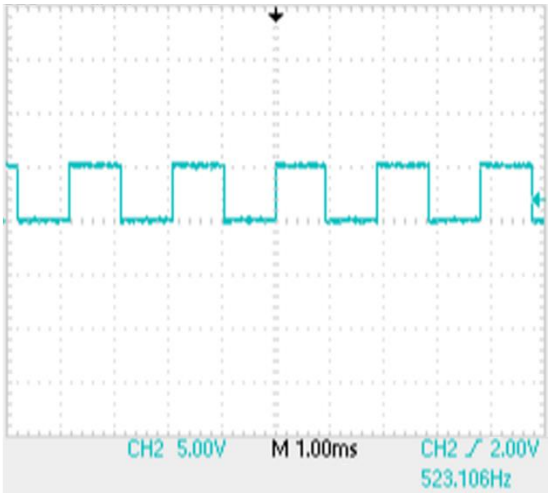
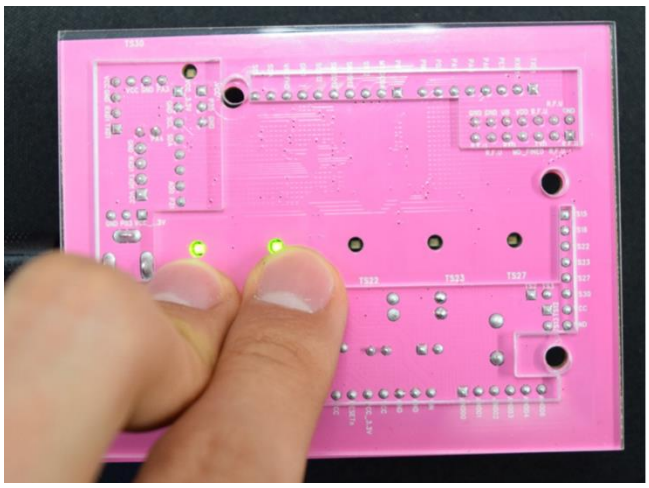


그림 5-1 터치 카운트가 1일 때 → 도 523[Hz]



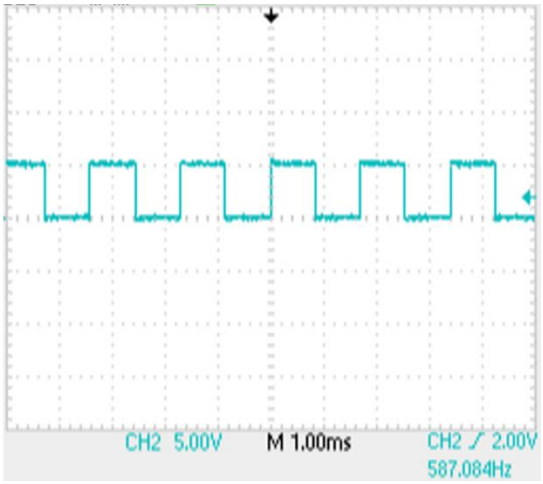
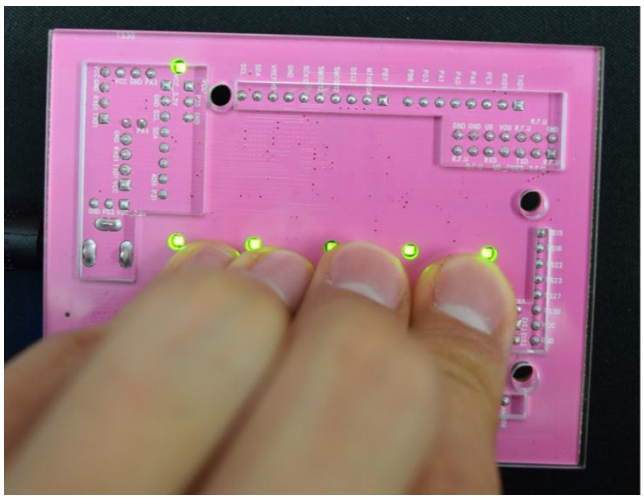


그림 5-2 터치 카운트가 2일 때 → 레 587[Hz]



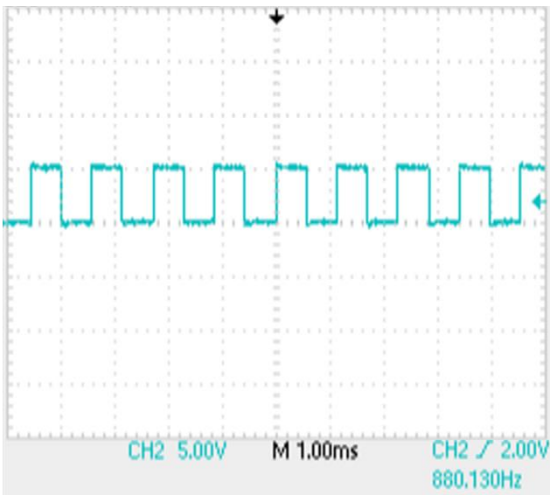


그림 5-3 터치 카운트가 6일 때 → 라 880[Hz]

6 회로도

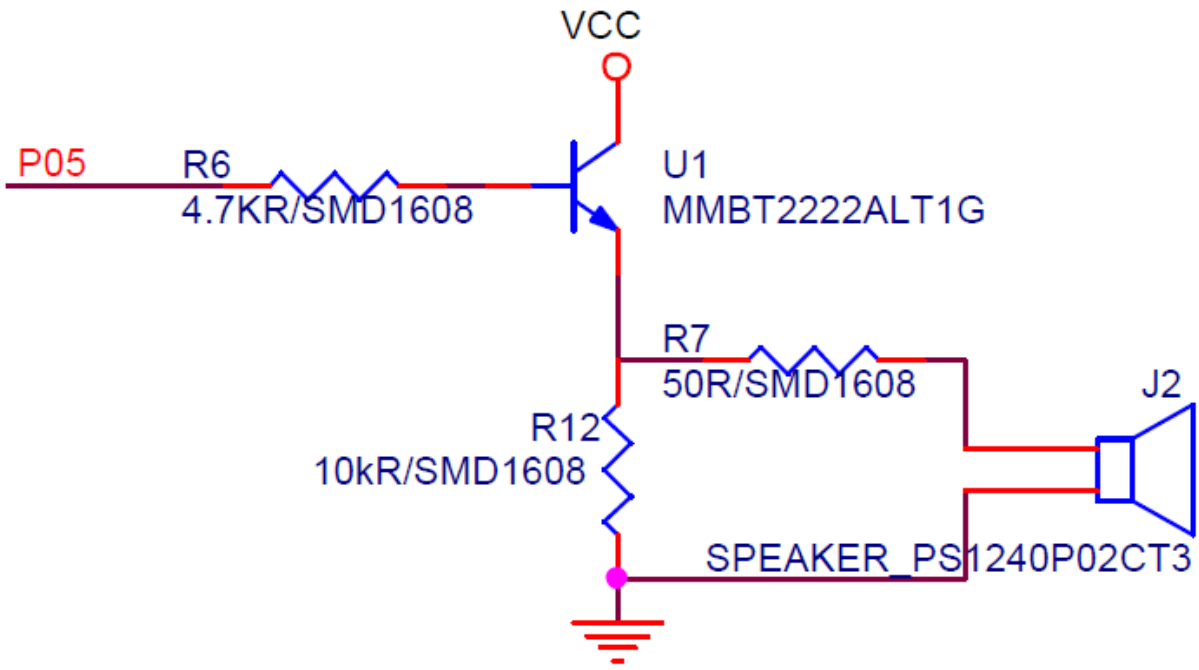


그림 6-1 NS-RX231의 스피커부분 회로도