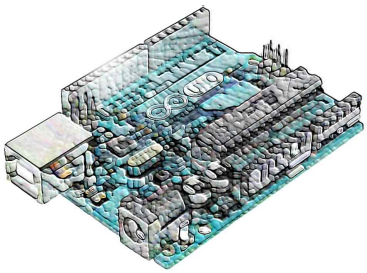


* 본 자료는 서경대학교 아두이노 프로그래밍 수업 수강자를 위해 작성된 강의 교재입니다.

강의교재– 아두이노 프로그래밍

5장 스피커

서경대학교 김진헌

5

스피커

내용

"(보류) 💣" 표기 부분은 강좌 후반부에 다시 검토할 예정입니다. 처음 공부할 때는 생략해 주세요.

5.1 스피커

스피커는 음원 파형을 전류로 제공하면 자체에 내장된 진동판으로 공기 중에 음파를 생성하여 소리를 발생하는 장치이다.

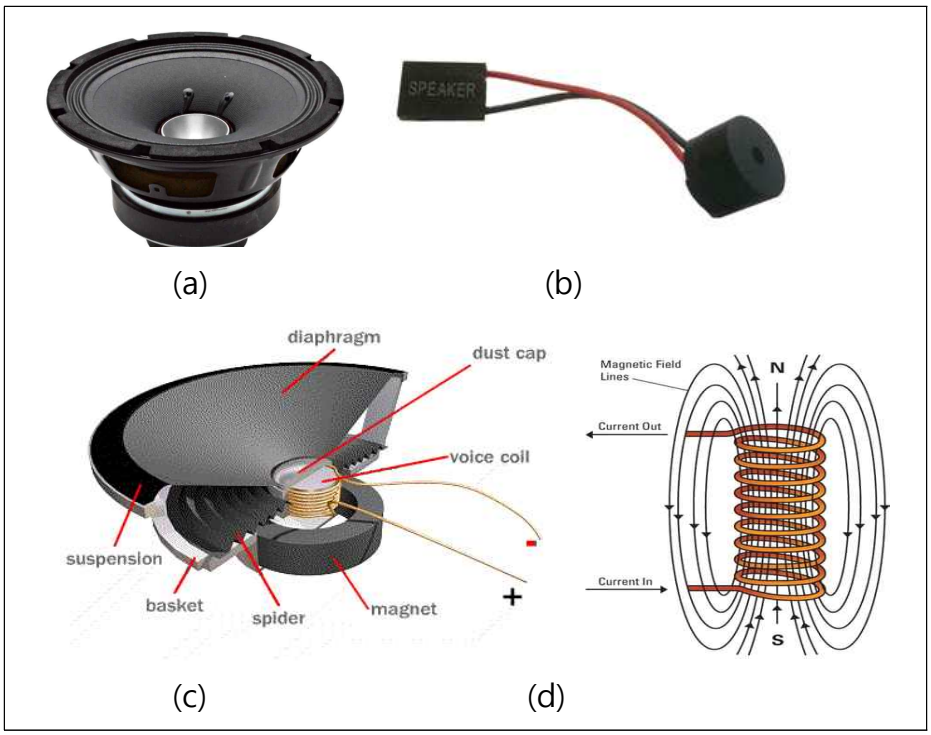
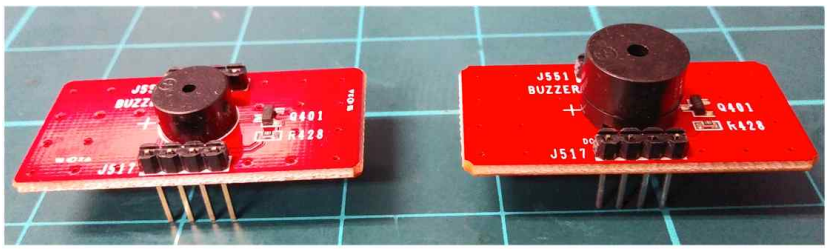


그림 5.1.1 스피커 및 내부 구조

그림 5.1.1은 이러한 스피커의 외형과 내부 동작원리를 보여주는 그림을 보인 것이다. (a)는 보편적인 자기형 스피커의 외형을 보인 것이다. (b)는 PC의 부팅과정에서 발생하는 오류를 소리로 알리는 용도로 사용 PC 내장용 스피커를 보인 것이다. (a) 형태의 스피커는 구동을 위해 큰 전력을 소비하지만 (b) 형태는 십여 mA의 전류만드로도 구동이 가능하여 디지털 출력단자의 전류만으로도 아두이노 소리 발생 실습에 적절하게 활용될 수 있다.

(c)의 그림은 스피커의 소리 발생 원리를 보여주는 것이다. 코일에 인가된 전류에 의해 자계가 발생하여 연결되어 있는 진동판(diaphragm)이 진동하여 소리를 발생한다. 그림 (d)는 코일을 따라 흐르는 전류에 의해 자계가 생성되는 모습을 보인 것이다. 전류의 방향이 바뀌면 자계는 그림에 보인 방향의 반대 방향으로 형성된다. 그림 (c)에 보인 바와 같이 코일은 영구자석에 전류의 방향에 따라 상하 진동하면서 소리를 발생하며 전류의 크기에 따라 그 음량이 결정된다[1].

그림 5.1.2의 우측에는 본 실험에서 사용되는 스피커 모듈을 보였다. 좌측 그림은 스피커로 착각하기 쉬운 부저(buzzer)의 모습을 보인 것이다[2]. 우측의 스피커 모듈은 높이만 조금 높고 실크 인쇄에는 buzzer라고 적혀 있어서 장착하는데 주의가 필요하다[3]. 스피커는 자체에 파형 생성회로가 없기 때문에 외부에서 파형신호를 제공해야 한다.



Buzzer Speaker

그림 5.1.2 부저(좌)와 스피커(우)

1) 이번 실험에서는 전류의 방향은 바뀌지 않고 단지 전류가 ON/OFF 만으로 소리를 생성한다. 이 경우에는 진동판은 한 방향으로만 운동할 것인데 그래도 소리를 발생하는 원리는 변함없이 적용된다.
2) 부저는 전원만 제공되면 내부의 발진회로에 의해 고정된 파형의 음향을 출력한다.
3) 실크(silk) 인쇄라함은 PCB에 희색으로 표기된 문자나 그림을 말한다. 스피커 모듈의 PCB에 J551 BUZZER 이라고 잘못 인쇄되어 있다.

5.2 주파수와 주기

반복적인 신호를 모델링할 때는 그림 5.2.1과 같이 정현파(Sinusoids) 신호를 이용할 수 있다. 이는 시간이 증가함에 따라 신호의 크기가 변화하는 모습을 보인 것인데 그림 5.2.1(a)에 보인 바와 같이 신호의 최대값을 진폭(amplitude)라 하고 하며 같은 여러 번의 파형이 반복될 경우 신호가 1회의 진동을 완성하는 시간[초]을 주기(period, T)라고 하고 이것이 1초 동안 반복되는 회수[Hz]를 주파수(frequency, f)라고 한다[4]. 보통 사람의 가청 주파수대는 16Hz ~ 20,000Hz 범위로 알려져 있다.

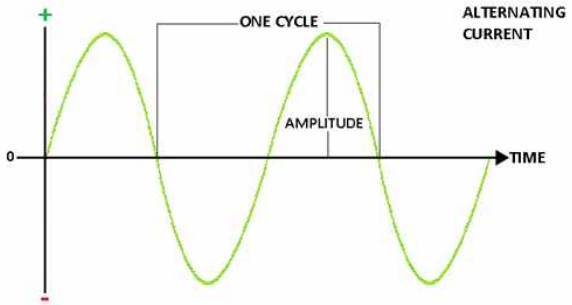


그림 5.2.1(a) 정현파 신호

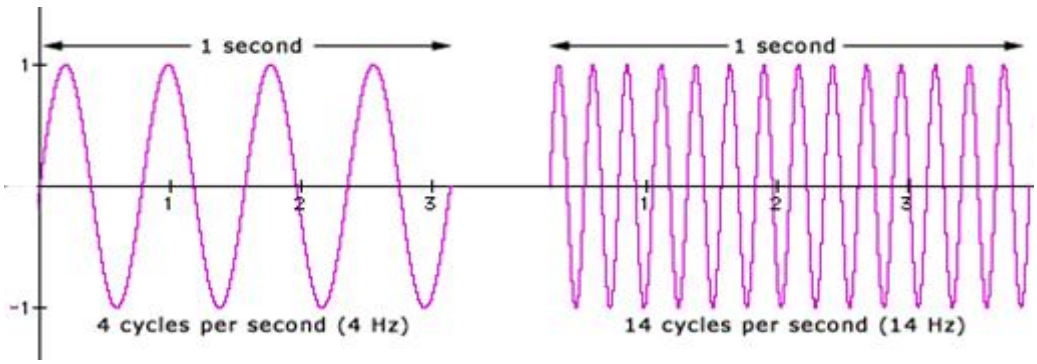


그림 5.2.1(b) 정현파 신호 (좌) f=4[Hz], (우) f=14[Hz]

---

4) 주기와 주파수는 역수의 관계가 있다. 즉, $T=\frac{1}{f}$. 혹은 $f=\frac{1}{T}$.

그림 (b)에는 1초 동안 관찰된 2개의 정현파 신호를 보인 것이다. 두 파형의 값의 범위가 [-1 ~ +1]의 범위에 이르므로 두 파형은 진폭은 1이라 할 수 있다.

좌측의 파형은 1초 동안 파형이 4회 반복되므로 주파수는 4[Hz]이고 우측의 파형의 주파수는 14[Hz]이다. 주기 "T=1/f"이므로 각 파형의 주기는 각각 1/4[초], 1/14[초]이다.

5.3 GPIO 기반의 스피커 음정 생성

원래 스피커의 진동판을 전후 방향으로 구동하기 위해서는 -전압과 + 전압을 교변하여 제공해야 하지만 그림 5.3.1(a)와 같은 디지털 데이터(로직 1과 0)로 만든 펄스(pulse) 신호로도 스피커를 구동시킬 수 있다. 스피커의 코일에 H(High, 논리 1)가 제공되면 이와 연결된 진동판이 어떤 방향이던지 움직일 것이다. L(Low, 논리 0)가 제공되면 복원력에 의해 그 반대 방향으로 움직여 원위치로 복구할 것이다.

이렇게 진동하는 시간 간격을 제어하여 필요한 주파수의 음정(tone)을 생성해 낼 수 있다. 그림 5.3.1(a)는 아두이노의 함수 digitalWrite()를 이용하여 H, L 펄스 신호를 생성하는 모습을 보인 것이다.

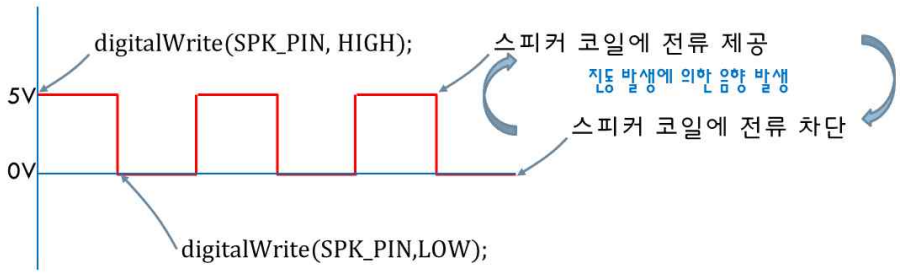


그림 5.3.1(a) 디지털 데이터 출력을 통한 펄스 신호 생성

주파수는 이렇게 진동하는 펄스의 시간 간격으로 정해지는데 가령 다음과 같이 H 혹은 L를 출력한 후 각각 delay(1)을 수행하였다고 가정해 보자.

```
void loop () {
        digitalWrite(SPK_PIN, HIGH);
        delay(1);                  // 1ms
        digitalWrite(SPK_PIN, LOW);
        delay(1);                  // 1ms
}
```

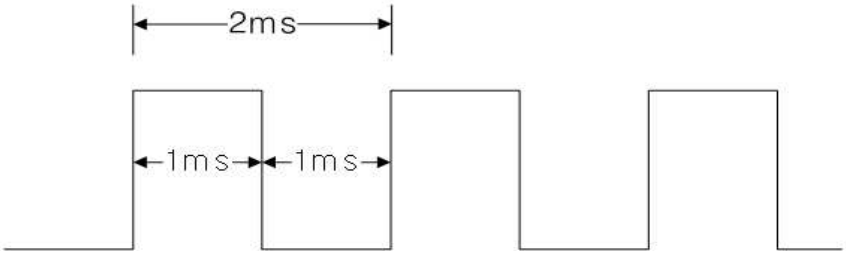


그림 5.3.1(b) 주기가 2ms(주파수 50Hz)인 펄스 신호

위 루틴의 수행으로 그림 5.3.1(b)와 같이 GPIO 출력 단자는 H 상태가 1ms, L 상태가 1ms 유지되는 파형이 반복하여 출력한다. H와 L 한 개의 사이클이 완성되는 시간은 1ms + 1ms = 2ms이다. 따라서 본 파형의 주기(T)는 2ms($=2\times10^{-3}$[초])이다. 주파수 $f=\frac{1}{T}$이므로 주파수는 500[Hz]이다[5]. 이로써 GPIO 출력함수만으로도 delay( ) 시간을 조절하면 출력되는 주파수를 조절할 수 있다는 사실을 알 수 있다[6].

---

5) $f=\frac{1}{2\cdot10^{-3}}=\frac{10^3}{2}=500$

6) delay() 함수는 1ms 이하의 지연시간은 지정할 수 없다. 따라서 이 함수로는 최대 500Hz까지 생성할 수 있다.

예제(1) - digitalWrite()/delay()로 소리 만들기

❐ 예제 1 : GPIO 함수를 이용해 스피커에 100[Hz]의 소리를 발생시키시오.	
SPK_01.ino : 아두이노 표준 함수, digitalWrite()	
01	#define SPK_PIN 11
02	int X;
03	int Y=100;
04	void setup( ) {
05	pinMode(SPK_PIN, OUTPUT);
06	X = 1000/(Y*2);
07	}
08	void loop( ) {
09	digitalWrite(SPK_PIN, HIGH);
10	delay(X); // 지연 시간을 적게 설정하면 고주파 신호 생성
11	digitalWrite(SPK_PIN, LOW);
12	delay(X);
13	}

주기=$\frac{1}{\text{주파수}}$. $T=\frac{1}{100}=10\times\frac{1}{1000}=10[ms]$. 따라서 H와 L가 각각 5ms의 지연 시간을 갖고 반복되어야 한다. 희망하는 주파수가 Y[Hz]라면 ms 지연함수의 인자 X는 다음 식으로 정리할 수 있다[7]. 예를 들어 Y=100이라면 X=5, Y=200이라면 X=2.5이다.

```
X[ms] = 1000/(Y[Hz] * 2);          X ≥ 1.
```

이 방법은 X=1이 최소 하한선인데 이때의 주파수는 500[Hz]이고, 주파수로서는 최대 상한선이다.

7) 지연 시간 X가 부동소수로 나오는 경우는 근사화시켜 정수로 만들어 사용해야 한다. delay() 함수가 정수만을 입력받기 때문이다.

미션: 아래 루틴이 몇 Hz의 파형을 생성해내는지 예측하시오.

```
void loop( ) {    // 사례 1
      digitalWrite(SPK_PIN, HIGH);
      delay(2);                  // 2ms
      digitalWrite(SPK_PIN, LOW);
      delay(2);                  // 2ms
}
// 풀이) T = 2+2 = 4ms. f=1/T = ?[Hz]
```

예제(2) - digitalWrite()/delayMicroseconds()로 소리 만들기

❒ 예제 2 : 지정한 특정 주파수(≥ 500)의 파형 신호를 출력하시오.

SPK_02.ino : 아두이노 표준 함수, digitalWrite()

```
01  #define SPK_PIN   11
02
03  void setup( ) {
04    pinMode(SPK_PIN, OUTPUT);
05  }
06  int desiredFreq = 10000;
07  int X = 1000000 / ( desiredFreq * 2) ;
08  void loop( ) {
09    digitalWrite(SPK_PIN, HIGH);
10    delayMicroseconds(X);
11    digitalWrite(SPK_PIN, LOW);
12    delayMicroseconds(X);
13  }
```

좀 더 고주파의 음향을 발생시키기 위해서는 delay() 시간을 더 줄여주어야 한다. 아두이노에서는 us(micro second, 10^{-6}) 단위의 지연 시간을 만들어 내기 위해 delayMicroseconds() 함수를 제공하고 있다. 이 함수는 지연되는 시간을 us 단위로 지원한다. 이 함수를 사용할 경우 희망하는 주파수가 Y[Hz]라면 microsec 지연함수의 인자 X는 다음 식으로 정리할 수 있다.

X[us] = 1,000,000/(Y[Hz] * 2); X ≥ 1.

위의 공식에 의하면 생성 가능한 최대 주파수 Y=500,000[Hz]이다. 하지만 사람의 가청 주파수 최대 범위는 20~20,000[Hz]이기 때문에 들을 수도 없겠지만, 일반 스피커로는 이런 가청 주파수 범위를 넘어서는 주파수는 잘 생성

하지 못한다. 고음용 트위터(tweeter)가 문서상 성능이 30,000[Hz] 남짓하며 저주파 전용 우퍼(woofer)도 저주파 하한대 영역은 70~30[Hz] 수준이다[8].

미션: 아래 루틴이 각각 몇 Hz의 파형을 생성해내는지 예측하시오.

```
void loop( ) {    // 사례 2. 지연시간이 us 단위로 주어진다.
      digitalWrit2(SPK_PIN, HIGH);
      delayMicroseconds(200);              // 200us
      digitalWrite(SPK_PIN, LOW);
      delayMicroseconds(200);              // 200us
}
// 풀이) T = 200+200 = 400us. f=1/T = ?[Hz]
```

```
void loop( ){    // 사례 3. duty rate = 90%
      digitalWrite(SPK_PIN, HIGH);
      delayMicroseconds(90);               // 90us
      digitalWrite(SPK_PIN, LOW);
      delayMicroseconds(10);               // 10us
}
// 풀이) T = 90+10 = 100us. f=1/T = ?[Hz]
```

사례 3의 경우는 H와 L의 비율이 50:50이 아닌 경우이다. H와 L의 비율을 듀티비(duty rate, ratio)라고 한다. 같은 주파수에 대해 듀티비가 달라졌을 때 이를 느낄 수 있는 지의 여부는 각자의 판단에 맡기기로 한다.

8) 참고로 Tinkercad 시뮬레이션의 경우는 음정의 왜곡이 포함되는 한계가 있다.

5.4 스피커 제어 표준함수

아두이노는 특정 주파수의 신호 생성을 위해 tone()/noTone() 함수를 지원한다. tone() 함수는 GPIO로 지정 가능한 단자에 대해 지정된 주파수를 갖는 듀티비(duty rate, ratio) 50%의 구형파(square wave) 신호를 만들어 내는 함수[9]이다.

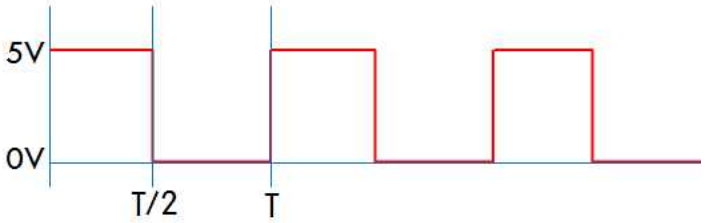


그림 5.4.1 듀티비 50%의 구형파 사례

듀티비는 전체 주기에서 High Pulse 폭이 차지하는 비율을 말한다. 그림 5.4.1에 보인 사례처럼 듀티비 50%의 구형파는 High pulse 폭과 Low Pulse 폭이 같다. 만약 같은 주기를 가진 파형이면서 듀티비가 낮다면 High pulse의 폭이 이보다 줄어들 것이다.

다음은 tone() 함수와 noTone() 함수의 사용법을 보인 것이다.

9) 아두이노 공식 문서에 의하면 이 함수로 만들어 낼 수 있는 최저 주파수는 31Hz이다.

void tone(pinNun, frequency) **void tone(pinNum, frequency, duration)**		
기 능	지정된 번호 단자를 출력모드로 설정하여 지정된 주파수의 펄스 신호를 생성한다. 따라서 따로 pinMode()를 통해 출력모드로 설정할 필요는 없다. **한 프로그램에서 1개의 tone() 함수만 수행될 수 있다.** 같은 프로그램에서 동시에 여러 핀에 대해 tone() 함수를 적용할 수 없다.	
매개변수	uint8_t pinNo	표준 아두이노 함수는 특별히 핀의 번호에 대해 제한을 두고 있지 않다. 다만 아두이노의 경우에는 3, 11번을 PWM 용도로 쓸 때는 간섭을 일으킨다고 주의를 주고 있다. **31Hz 미만의 주파수는 지원하지 않는다.**
	unsigned int freq	구형파를 만들기 위한 진동 수
	unsigned int duration	신호를 출력하는 시간[ms]. 지정하지 않으면 무한 출력된다. 유의! - 어떤 경우든 출력을 마칠 동안 기다리지 않고 다음 줄의 명령어로 진행한다. 따라서 다음 줄에 tone() 함수를 수행하면 이 시간은 지켜지지 않는다.
리턴 값	void	없음

void noTone(pinNo)		
기 능	지정된 번호에 해당하는 핀에 구형파 출력을 정지	
매개 변수	uint8_t pinNo	구형파가 출력되는 핀 번호
리턴 값	void	없음

5.5 tone() 함수 활용 예제

예제(3) - 고정 주파수의 음향 출력

특정 주파수를 출력하는 tone( ) 함수를 통해 스피커에 고정된 주파수의 음향을 출력하는 예제를 통해 이 함수의 활용법을 익혀 보자. 이 함수는 지정된 단자에 대하여 지정한 주파수를 출력하게 한다. 주파수 발생을 멈추려면 noTone( ) 함수를 사용한다.

❐ 예제 3 : tone()/noTone()를 사용해서 스피커에 고정된 음정을 출력한다.

SPK_03.ino : 아두이노 표준 함수 -tone()/noTone()

```
01  #define SPK_PIN  11
02  void setup( ) {
03    //pinMode(SPK_PIN, OUTPUT);  // tone() 함수 사용시 필요없음.
04  }
05  void loop( ) {
06    tone(SPK_PIN, 524);  // 주파수 524Hz로 설정하여 출력
07    delay(1500);         // 1.5초 동안 계속 출력
08    noTone(SPK_PIN);     // 출력 중지
09    delay(500);          // 0.5초 동안 묵음 상태 지속
10  }
```

예제(4) - 고정 주파수의 음향 출력, duration이 있는 경우

tone( ) 함수의 파라미터 중 마지막 인자는 음향이 출력되는 시간이지만 그 음향을 출력하는 도중에도 다음에 있는 함수의 수행이 가능함을 유의해야 한다. 이는 정현파 신호의 발생이 delay( ) 함수를 바탕으로 이루어진 것이 아니라 주기적인 인터럽트에 기반으로 설계되어 있기 때문이다[10].

예제 4의 프로그램은 06번 줄의 경우에는 지정한 대로 500Hz 주파수의 음향이 1초 동안 출력되지 않는다. 06번 줄을 수행해서 소리가 나는 동안 07번의 0.5초 지연 시간을 거친 후 08번에서 새로운 2000Hz 파형의 출력이 이루어지기 때문이다. 반면 08번의 0.5초의 출력은 의도한 대로 이루어진다. 09번에서 2초 동안 기다리기 때문이다. 결과적은 2KHz 음향이 0.5초 동안 출력된 후 1.5초의 공백이 있을 것이다.

❐ 예제 4 : tone()/noTone()를 사용해서 스피커에 고정된 음정을 출력한다. 음향 출력 시간 허용 시간을 하는 경우

SPK_04.ino : 아두이노 표준 함수 -tone()

```
01  #define SPK_PIN  11
02  void setup( ) {
03    //pinMode(SPK_PIN, OUTPUT);  // tone() 함수 사용시 필요없음.
04  }
05  void loop( ) {
06    tone(SPK_PIN, 500, 1000);  // 500Hz. 1 초 동안 출력. 실제로는 0.5초
07    delay(500);                // 0.5 동안 지연 후 다음 명령 수행
08    tone(SPK_PIN, 2000, 500);  // 2KHz, 0.5초 출력. 실제로도 0.5초
09    delay(2000);               // 2초 동안 지연. 이 중 0.5초는 2KHz 출력.
10  }
```

10) 보통은 타이머 장치로 구현하는 것이 일반적이지만 그러려면 특정 단자만 사용해야 한다. tone() 함수는 단자를 가리지 않는 것으로 보아 타이머를 사용하지 않는 것으로 판단된다.

예제(5) - 시리얼 모니터로부터 입력받은 주파수를 출력하기

시리얼 모니터는 아두이노의 문자 출력을 내보내는 곳이기도 하지만 아두이노에게 전달하는 사용자의 입력을 전달하는 창구이기로 하다. 시리얼 모니터에서 입력받는 전송창을 그림 5.5.1에 보였다.

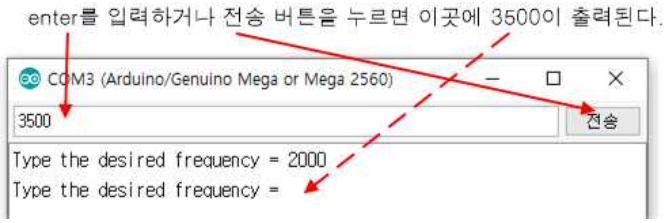


그림 5.5.1 시리얼 모니터의 사용자 입력창

❒ 예제 5 : 시리얼 모니터의 사용자 입력창에서 전송한 주파수(스트링) 값을 정수 값으로 전달받아 스피커로 음향 출력하기.

SPK_05.ino : 아두이노 표준 클래스 -Serial()

```
01  #define SPK_PIN 11
02  int freq;         // 정수형 주파수.
03  String freqS;     // 스트링형 주파수. 시리얼 모니터에서는 문자열로 입력됨.
04  void setup() {
05      Serial.begin(9600);      delay(50);
06      Serial.print("Type the desired frequency = ");
07  }
08  void loop() {
09    while(Serial.available() == true) { // 입력이 들어왔으면 True.
10      freqS = Serial.readString();  // 시리얼 모니터의 입력을 스트링으로 읽는다.
11      freq = freqS.toInt(); // 스트링 문자열을 정수형으로 바꾼다.
12      Serial.println(freq);
13      tone(SPK_PIN, freq, 1500);
14            Serial.print("Type the desired frequency = ");
15    }
16  }
```

예제(6) - 한 옥타브 음정 출력

음악을 연주하기 위한 특정 옥타브(octave)의 주파수를 보이면 다음과 같다. 본 프로그램은 해당 주파수를 순차적으로 출력하였다.

음정	도	레	미	파	솔	라	시	도
주파수(Hz)	262	294	330	347	392	440	494	524

❒ 예제 6 : tone()/noTone() 함수를 사용해서 한 옥타브를 연주한다.

SPK_06.ino : 아두이노 표준 함수 -tone()/noTone()

```
01  #define SPK_PIN  11
02  int scales[]={262, 294, 330, 347, 392, 440, 494, 524};
03  void setup( ) {
04    //pinMode(SPK_PIN, OUTPUT);  // tone() 함수 사용시 필요없음.
05  }
06  void loop() {
07      for(int nIdx=0; nIdx<8; nIdx++)
08       {
09            tone(SPK_PIN, scales[nIdx]);
10            delay(500);
11        }
12  }
```

예제(7) - 연속적인 주파수의 변화를 갖는 음향의 생성

❒ 예제 7 : 1ms 단위로 연속적으로 주파수가 올라가면서 출력하는 프로그램을 설계하시오. 이때 저음은 100Hz, 고음은 3000Hz로 설정한다.

SPK_07.ino : 아두이노 표준 함수 - tone()/noTone()

```
01  #define  SPK_PIN  11
02  void setup( ) {
03    //pinMode(SPK_PIN, OUTPUT);  // tone() 함수 사용시 필요없음.
04  }
05  void loop( ) {
06  #define MaxF 3000
07  #define MinF 100
08    for (int freq =MinF; freq < MaxF; freq=freq+1 ) {
09        tone(SPK_PIN, freq);
10        delay(1);
11    }
12  }
```

5.6 응용 예제

예제 (8) - 응용: PC 키보드를 입력장치로 활용하여 연주한다.

(보류)

PC의 키보드를 활용하여 한 옥타브의 음정을 연주하는 프로그램을 작성을 검토해 보자. Serial Monitor를 사용하면 연주할 음정에 해당하는 키를 입력한 후 매번 enter 키를 입력해야 하기 때문에 불편하다. 엔터 키를 입력하지 않고 피아노처럼 숫자키만 입력해도 사용자의 입력이 전달이 되게 하려면는 엔터키 없이 전송가능한 터미널 에뮬레이터를 사용해야 한다[11].

응용 9	PC 키보드 1~8번 숫자 키로 음정을 연주하는 프로그램을 설계하시오. Serial.read(), Serial.available() 함수를 이용하여 시리얼 모니터의 키 입력을 이용하여 연주한다.		
사용 센서	스피커	**사용 함수**	digitalWrite() 혹은 tone()

☞ **프로그램 작성을 위한 도움말**

1. PC의 키 입력 결과를 아두이노에서 획득하는 방법

아두이노가 직렬통신으로 데이터를 입력받는 함수를 호출한다.

a= Serial.read(); // PC 키보드로 입력한 데이터(ASCII)를 직렬통신을 통해 입력받는다.

2. PC에서 키보드 데이터를 아두이노로 전달하는 방법

1) Serial Monitor 이용 – 상부의 입력란에서 입력한 결과를 직렬통신 포트로 전송. Enter 키를 입력해야 1바이트 전송 가능. 불편함.

2) Tera Term 같은 Terminal Emulator 활용. -> 프로그램을 다운 받아 사용

*** 테라텀과 Serial Monitor는 동시 수행이 불가하다. COM port를 공유할 수 없으므로 사용에 주의할 것.**

11) 본 사례는 6장의 직렬통신을 모두 학습한 후 다시 돌아와서 구현하기로 한다.

5.7 tone() 함수의 동작 원리

이 절은 14장의 인터럽트를 학습해야 검토할 수 있습니다. 인터럽트가 여러 부분에 걸쳐 관계되어 있어 해당 챕터에 관심가져야 할 내용을 기록하였습니다.

❐ tone() 함수의 펄스 발생 원리에 대하여(보류)

주파수 정해진 펄스의 생성은 보통 타이머(Timer 혹은 Timer/Counter) 장치를 활용한다. PC의 경우 그림 5.7.1에 보인 바와 같이 타이머를 활용해 다양한 주파수의 소리를 생성하고 있다.

타이머는 그림에 보인 바와 같이 고정된 주파수의 클록 신호를 입력받아 이를 원하는 주파수의 클록으로 만들어 내는 장치이다. 내부적으로는 다수의 레지스터가 관계되어 입력되는 클록의 개수를 세는 동작을 하는 카운터가 내장되어 있어 타이머/카운터라고 부르기도 한다.

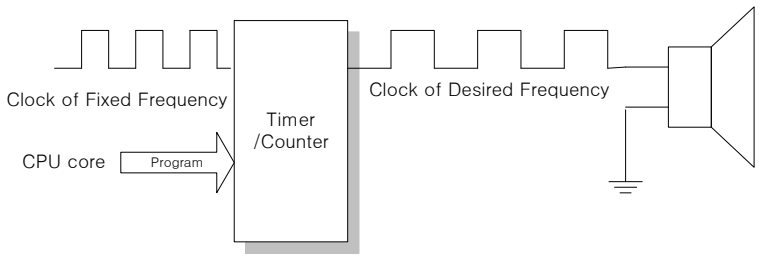


그림 5.7.1 타이머의 동작 개념도

타이머를 클록 발생의 음원으로 사용하려면 출력 단자는 타이머와 연결되어 있어야 한다. Arduino Mega의 경우에는 타이머는 2~13, 44~46 번 단자에 연결되어 있다.

그러나 아두이노의 공식 사이트에는 tone() 함수의 사용 가능한 핀 번호에 대해 이들 단자만 사용할 수 있다는 언급을 발견할 수 없다. 이는 어떤

GPIO 단자이던지 사용할 수 있다고 해석해도 무방하다[12]. 실제로 Ardu-Ez보드에서 피에조 소자가 연결되어 있는 58번 단자는 타이머에 연결되어 있지 않는데도 tone()함수를 사용해도 동작이 잘 되는 것을 관찰하였다[13].

명제: 이를 종합하면 아두이노의 tone() 함수는 타이머로 구현하지 않고 주기적 인터럽트를 발생시켜 그 서비스 루틴((ISR, Interrupt Service Routine)) 에서 digitalWrite() 함수를 통해 High와 Low를 교번하여 출력하는 것으로 판단된다[14]. 예를 들어 1초에 2000번 주기적 인터럽트를 발생하게 하면 2000회 ISR을 수행하게 되는데 이때 GPIO에 H와 L를 교대로 출력하면 결과적으로 1000Hz의 듀티비 50%의 파형을 생성해 낼 수 있다.

논거 1: 주기적 인터럽트는 보통 타이머가 발생시키는데 아마도 타이머 2가 이를 담당하는 것으로 보인다. 타이머 2는 표 5.7.1에 보인 바와 같이 단자 3, 11번에 연결되어 있으므로 이 때문에 같은 단자에 PWM 동작을 시키면 간섭이 일어난다고 판단된다. 그래서 아두이노 공식 사이트에서는 3번 11번을 쓰면 tone() 함수에 문제가 있을 수 있다고 기술하였다.

---

12) Arduino Mega가 아니라면 단자 3, 11번은 tone() 함수를 사용할 때 PWM 출력 동작과 간섭을 일으킬 것이라고 주의를 주고 있다. PWM 동작은 타이머가 사용되는 동작이다.

13) 확인 결과 tone() 함수는 GPIO로 활용되는 단자를 모두 사용할 수 있는 것으로 판단된다. tone() 함수는 타이머를 1채널 쓰긴 하지만 사용하지 않는 타이머 채널을 검색해서 활용하기 때문에 단자를 사용하지 않는다.

14) 이 추론의 타당성에 대해서 여러분들과 함께 토론하고 싶습니다. 사실의 진위여부가 중요한 것이 아니라 이러한 추론을 근거를 들어 설명하고 논리적으로 분석하고 이해하는 능력의 계발이 더 필요하겠지요. 지식에 대한 암기가 중요한 것이 아니라 분석하고 이해하는 것이 더 중요하다는 취지에서 생각해 볼만한 명제입니다.

표 5.7.1 아두이노의 타이머와 연결된 단자 번호

단자번호 (논리적 단자 번호)	연결된 타이머 채널	비고
Pins 5 and 6	timer0	
Pins 9 and 10	timer1	
Pins 11 and 3	timer2	tone() 함수의 펄스 발생을 위한 주기적 인터럽트 발생에 관여

만약 tone() 함수가 독립적으로 동작하는 타이머를 사용한다면 다수의 핀에 대해서 동시 설정이 가능해야 되는데 아두이노 공식 사이트의 설명에 따르면 tone() 함수에 의해 어떤 단자가 펄스를 출력한다면 다른 단자에 대해서 tone() 함수 설정을 할 수 없다고 기술되어 있다. 타이머는 H/W 장치이기 때문에 본래는 독립동작이 가능하기 때문에 설정된 여러 주파수의 펄스를 동시에 출력할 수 있어야 정상인데 이것이 금지되어 있는 것을 보면 이 함수에 의한 펄스 출력이 타이머의 출력단자를 사용하지 않는다고 판단하게 하는 근거가 된다. => 논거 2: tone()함수는 원래 여러 개가 동시에 함께 작동할 수 있어야 정상이다.

논거 3: tone() 함수가 타이머를 사용한다면 특정 단자만 지원해야 한다. 타이머 출력단자는 미리 정해진 단자에만 사용 가능하기 때문이다. 하지만 아두이노에서는 모든 GPIO 단자를 tone(0 함수의 출력단자로 사용할 수 있다고 하였다.

표 5.7.2 아두이노 메가의 타이머와 연결된 핀 및 Ardu-Ez의 활용

PWM OUT Pin No. (물리적 단자 번호)	Register	Timer/ Counter	Counter Length	Usages in Ardu-Ez
13	TIMER0A	0	8	
4	TIMER0B			
11	TIMER1A	1	16	Buzzer, Speaker
12	TIMER1B			
10	TIMER2A	2	8	
9	TIMER2B			
5	TIMER3A	3	16	
2	TIMER3B			
3	TIMER3C			Step Motor Pulse
6	TIMER4A	4	16	
7	TIMER4B			DC 모터제어(정방향)
8	TIMER4C			DC 모터제어(역방향)
46	TIMER5A	5	16	
45	TIMER5B			
44	TIMER5C			

아두이노 메가의 경우에는 표 5.7.2와 같이 6개의 타이머를 이용해 총 15개의 PWM 출력단을 지원한다. 이 중 11개는 16비트 카운터를 사용하며, 4개는 8비트 카운터를 사용한다. 타이머는 Arduino Mega의 경우에는 2~13, 44~46 번 단자를 사용한다.

❒ 부저에 tone( ) 함수를 사용하면 어떤 일이 일어날까?

- 논의의 가치가 별로 없지만 실제로 실험 교재에 수록되고 운영되었던 일이라서 혹시나 하는 마음에 거론하였습니다. 현재의 상황에서라도 이 정도는 답할 수 있어야 하겠습니다.

tone() 함수를 부저에 대해 적용하면 올바른 톤의 소리를 들을 수는 없다. 이는 부저 자체가 스스로 특정 주파수의 신호를 만들어 내고 있기 때문에 tone 함수를 사용하면 지정한 주파수와 부저의 고유 주파수가 부적절하게 혼합되어 실제 귀에 들리는 주파수가 변형되기 때문이다. 부저의 출력 주파수를 프로그램으로 바꾸는 작업은 주파수의 왜곡을 감수해야 한다.

5.8 고찰

다음 문제 혹은 미션에 대해 적절한 답안을 제시할 수 있는지 검토해 보면서 그동안 습득한 지식을 정리해 보자.

1. 18,000Hz의 주파수를 갖는 신호를 스피커로 보냈는데 그 음향을 들을 수 없었다. 그 이유를 2가지로 제시하여 추론하시오.

2. tone() 함수로 10Hz를 생성하여 스피커로 출력하였는데 소리를 들을 수 없었다. 그 이유를 진단하시오.

3. digitalWrite() 함수를 반복적으로 사용하여 스피커를 진동시킬 수 있다. 이를 이용해 tone() 함수를 쓰지 않고 스피커의 출력 주파수를 제어할 수 있는가?

4. tone() 함수에 사용되는 단자는 타이머에 연결되지 않아도 쓸 수 있는가?

5. 서로 다른 단자와 주파수를 갖는 tone( ) 함수를 수행시키고 2개의 스피커를 연결하여 두 개의 음정을 갖는 화음으로 연주를 하고자 한다. 예를 들어 13번 단자와 11번 단자를 각각 524Hz와 2018Hz 주파수의 신호를 동시에 3초간 출력시키고자 한다. 이것이 불가능한 이유를 설명하시오.

6. 예외적으로 특정 조건 하에서는 위의 미션을 tone() 함수를 사용하지 않고 digitalWrite()와 적절한 delay()를 추가하여 수행할 수 있는 것으로 추정된다[15]. 이를 구현하는 방안에 대해 아이디어를 제시하거나 프로그램을 작성하시오.

---

15) 향후에 기술할 주기적 인터럽트를 적용하여 구현하는 방안도 가능할 것으로 기대된다.

7. "도" 음정을 스피커로 출력할 수 있는 2가지 방안을 제시하시오.

8. digitalWrite 함수를 이용하여 tone(), noTone()과 같은 동작을 하는 함수 my_tone(), my_noTone()을 제작하시오. tone 함수는 다음과 같이 2개의 파라미터만 지원하는 것으로 가정한다. tone(단자, 주파수);

9. delay() 함수를 이용하여 tone() 함수를 설계하면 delay() 함수가 수행을 마칠 때까지는 tone() 함수를 나갈 수 없다. 이를 극복하는 방안이 있을까? (보류) 💣