아두이노프로그래밍

6차과제

14장8절고찰 일부연습 문제풀이

2020.06.07.일

컴퓨터공학과

2019305059

이현수

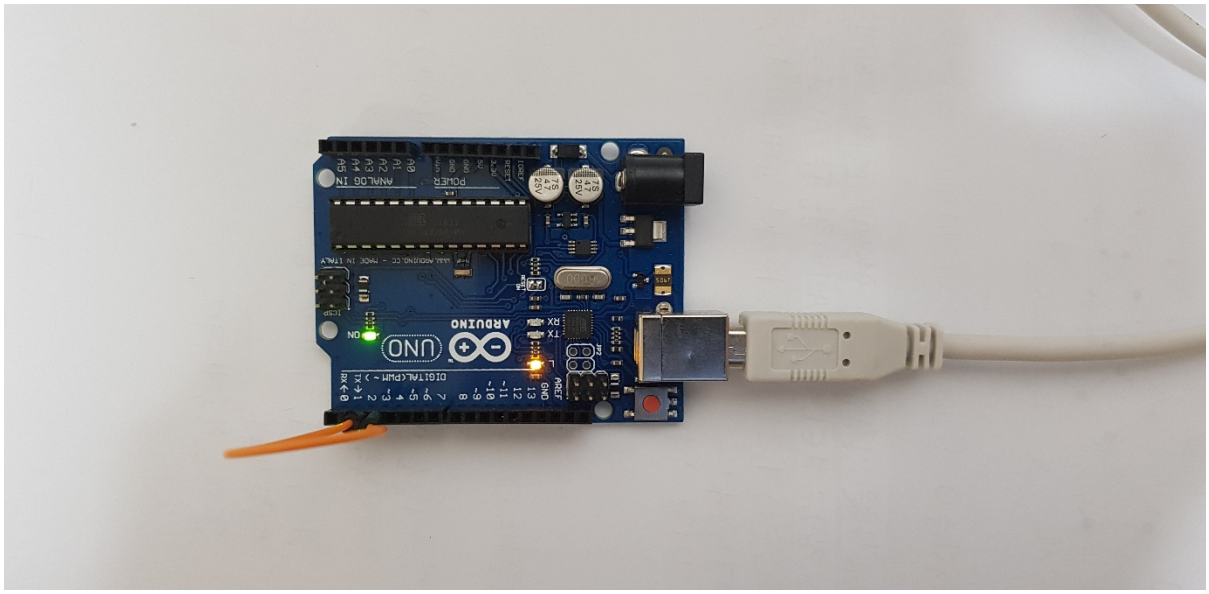
▪11번

- analogWrite() 함수는 490Hz의 PWM 펄스를 발생.

1) 이를 이용해 390Hz의 periodic interrupt를 발생.

2) 이 주기적 인터럽트를 이용해 시계를 만들 수 있다. PWM 펄스를 이용하여 스톱워치 프로그램을 작성.

▪회로도



아두이노우노 보드에 주황색 점퍼케이블을 GPIO단자 2, 3번 단자를 연결시킨다.

▪소스코드

```
#define intpin 2
#define analogpin 3

volatile int t=0;
volatile int count=0;
int ctrint=-1;
```

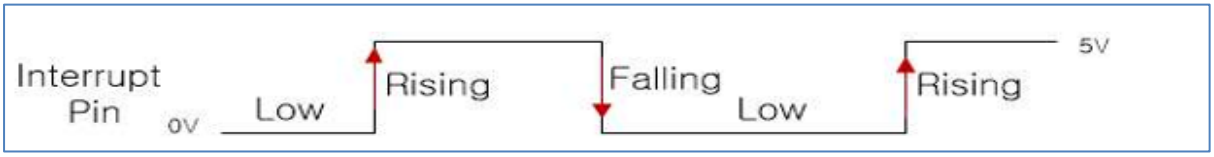
인터럽트를 위한 핀단자 2, PWM 제어 표준함수 analogWrite()함수를 위한 핀 3을 정의한다.

그리고 인터럽트 시 호출할 함수에서 사용할 변수 t와 count를 선언한다.
t변수의 경우 시간, 초가 저장될 변수이다. 처음에는 0초로 초기화 한다.
count변수는 인터럽트 발생 때마다 1씩 증감시켜서 1초를 계산하는데 이용되는 변수이다.

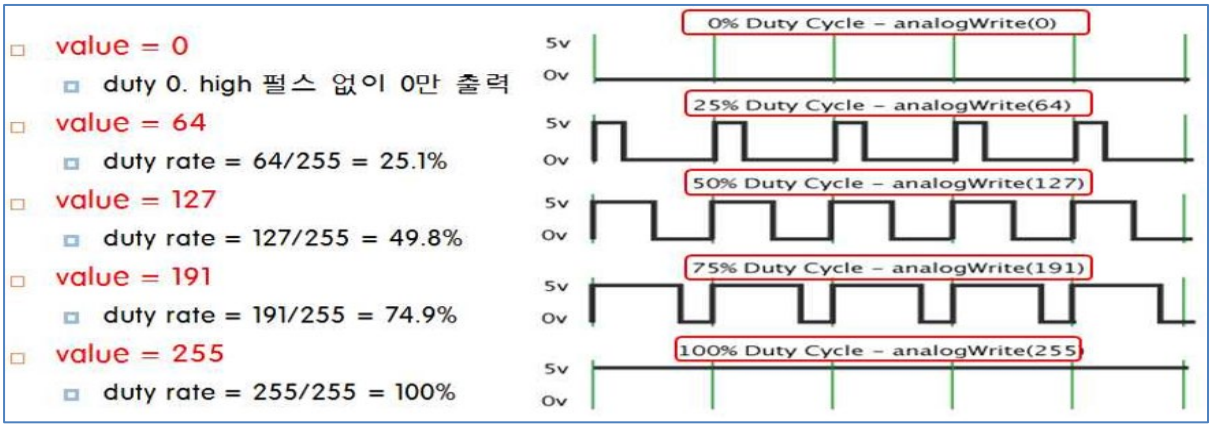
그리고 t변수 출력을 제어할 ctrint변수를 전역변수로 선언한다.

```
void setup() {
  pinMode(intpin, INPUT);
  attachInterrupt(digitalPinToInterrupt(intpin), counting, RISING);
  Serial.begin(9600);
  analogWrite(analogpin,0);
  Serial.println("(a)스톱워치 시작 (b)스톱워치 중지 (c)스톱워치 초기화");
}
```

setup함수에서는 인터럽트를 위한 핀 2번을 입력모드로 설정한다.



인터럽트는 RISING일 때, 즉 LOW에서 HIGH가 될 때 인터럽트가 발생하도록 설정한다.
인터럽트 발생 시 counting함수가 호출된다. 그리고 통신속도를 9600으로 설정한다.



그리고 analogWrite(anlogpin, 0);을 통해서 처음에는 주파수 0Hz를 내보낸다. 그래서 처음에는 인터럽트가 발생하지 않게 설정한다.

```
void counting(){
  count++;
  if(count==490){
    t++;
    count=0;
  }
}
```

인터럽트가 발생될 때 counting함수가 호출된다.

counting함수는 인터럽트가 발생할 때마다 함수내에서 전역변수 count를 1씩 증가시키고, 490이 될 때 시간을 1초 증가시킨 후 다시 count변수를 0으로 초기화해서 1초를 다시 계산할 수 있게 한다.

BOARD	PWM PINS	PWM FREQUENCY
Uno, Nano, Mini	3, 5, 6, 9, 10, 11	490 Hz (pins 5 and 6: 980 Hz)
Mega	2 - 13, 44 - 46	490 Hz (pins 4 and 13: 980 Hz)
Leonardo, Micro, Yún	3, 5, 6, 9, 10, 11, 13	490 Hz (pins 3 and 11: 980 Hz)
Uno WiFi Rev.2	3, 5, 6, 9, 10	976 Hz
MKR boards *	0 - 8, 10, A3 (18), A4 (19)	732 Hz
MKR1000 WiFi *	0 - 8, 10, 11, A3 (18), A4 (19)	732 Hz
Zero *	3 - 13, A0 (14), A1 (15)	732 Hz
Due **	2-13	1000 Hz
101	3, 5, 6, 9	pins 3 and 9: 490 Hz, pins 5 and 6: 980 Hz

analogWrite()함수의 경우 490Hz가 나온다. 즉 1초에 파형이 490번 움직이는 것이다.

그래서 인터럽트는 1/490초마다 발생하므로, 1초를 계산하기 위해서는 counting()함수에서 count변수를 490번 증가시켜 490이 되는순간 1초가 된 것이므로 t 시간변수를 1증가시켜 1초를 계산하는 원리이다.

```
void loop() {
  if(Serial.available()>0){
    char inchar=Serial.read();
    if(inchar=='a'){
      analogWrite(analogpin,100);
    }
    else if(inchar=='b'){
      analogWrite(analogpin,0);
    }
    else if(inchar=='c'){
      t=0;
      analogWrite(analogpin,0);
    }
  }
  if(ctrint!=t){
    Serial.print("\r"); Serial.print(t);
    ctrint=t;
  }
}
```

loop함수에서는 조건문 두개가 있다. 한 개는 사용자 입력을 기다리고, 다른 한 개는 시간변수 t를 출력시키는 조건문이다.

첫번재 조건문

만약 사용자가 입력을 했을경우
그것이 a, b, c일 경우에 각각 기능을 실행한다.

a가 입력됐을 때 스톱워치 시작 기능을 실행한다.
이때 analogWrite(analogpin, 100);을 해준다. 그렇게 되면 듀티비와 상관없이 항상 490Hz를 내보낸다.
프로그램 시작 시 스톱워치가 멈춰있기 때문에 스톱워치를 시작하려면 반드시 a를 입력해야 한다.
a입력 시 스톱워치가 작동되고, 1초마다 1씩 증가한다.

b가 입력됐을 때 스톱워치 중지 기능을 실행한다.
analogWrite(analogpin, 0);을 통해 인터럽트가 발생하지 않게 한다. 그렇게 되면 스톱워치가 멈춘다.

c가 입력됐을 때 스톱워치 초기화 기능을 실행한다.
t=0; analogWrite(analogpin, 0);가 실행되, 시간변수가 0으로 초기화되고 인터럽트가 발생하지 않게 설정한다. 만약 다시 스톱워치를 실행하고 싶다면 a를 입력하면 된다.

두번째 조건문

시간출력을 제어하기 위한 조건문이다. 만약 두번째 조건문이 없다면 숫자가 계속 출력되게 된다.

▪실행

프로그램을 처음 시작하면 숫자 0에서 계속 멈춰있다. 여기서 스톱워치를 실행하고 싶으면 a를 입력하면 스톱워치가 작동하고 b를 입력하면 스톱워치가 중지된다. 그러다 다시 a를 입력하면 스톱워치가 이어서 카운트한다. 그러다 c를 입력하면 숫자가 0으로 출력되고 스톱워치는 멈춘다.
만약 스톱워치 초기화 후 다시 스톱워치를 실행하려면 a를 입력하면 된다.

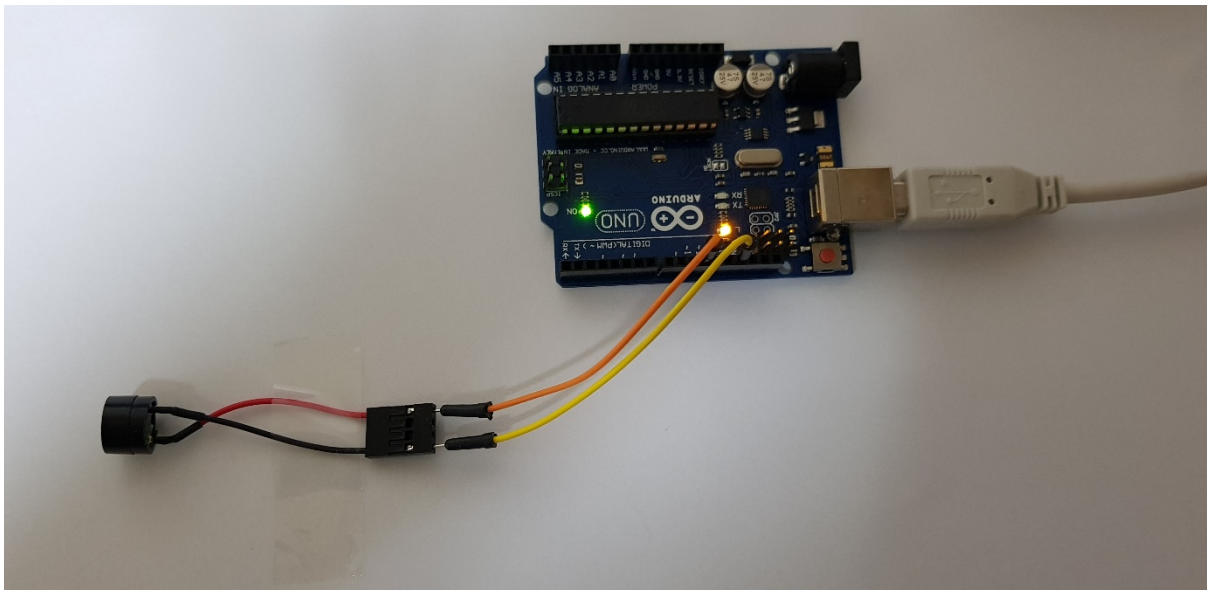
▪실험과정에서의 경험과 습득 사실

- 실험을 통해서 analogWrite()함수가 실제로 490Hz를 발생시킨다는 사실을 확인함.

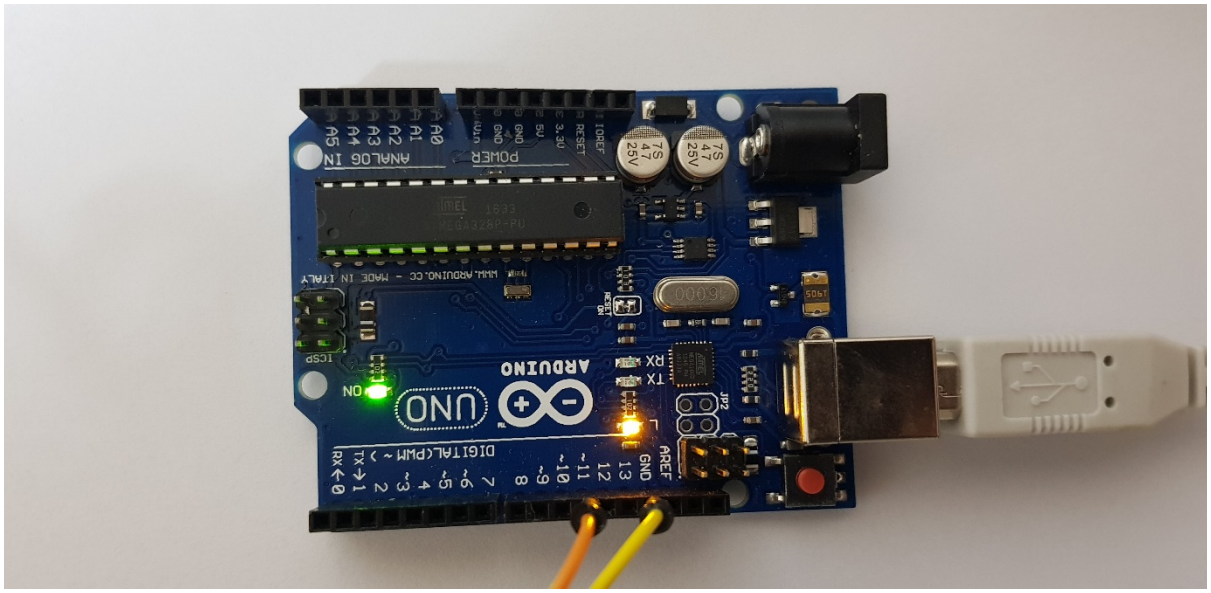
▪12번

- 클럭 소스(MsTimer2)를 이용하여 1,000Hz 인터럽트를 발생. 이를 이용하여 지정한 주파수의 음정을 생성하는 myTone(pin, frequency)과 myNoTone() 함수를 제작.
- 이 함수를 기반으로 직렬 통신(시리얼 모니터)으로 입력받은 주파수를 2초간 출력하는 프로그램을 설계. 생성 할 수 없는 주파수를 입력하면 범위 밖이라고 거절하고, 입력한 주파수를 시리얼 모니터에 알려줌.

▪회로도



아두이노우노 보드에 스피커를 연결한다.



스피커의 (+)는 GPIO 11번단자에, (-)는 GND에 연결한다.

▪소스코드

```
#include<MsTimer2.h>

volatile int frequen;
volatile double detime;
volatile int state=0;
volatile int SPK;
```

MsTimer2를 사용하기 위해서 해더파일을 선언한다.

그리고 인터럽트 내에서 사용할 수 있게끔 전역변수 선언 시 앞에 volatile을 적는다.

frequen변수는 주파수를 저장.
Detime변수는 HIGH, LOW에서 지속될 딜레이 시간.
state변수는 myTone, myNoTone함수 제어.
SPK변수는 스피커 단자번호를 저장.

```
void setup() {
  pinMode(11,OUTPUT);
  MsTimer2::set(1000, fre);
  MsTimer2::start();
  Serial.begin(9600);
}
```

setup함수에서 11번(스피커)을 출력모드로 설정한다.

그리고 MsTimer2를 1초마다 fre함수를 호출하게 인터럽트를 설정한다.

그리고 인터럽트를 시작하고, 통신속도를 9600으로 설정한다.

```
void loop() {
  AAA:
  Serial.print("주파수를 입력하세요 : ");
  int x=inputFromTerm();
  if(x>500){
    Serial.print("Hz");
    Serial.println(" 잘못된 범위 주파수!");
    goto AAA;
  }
  else{
    Serial.println("Hz");
    myTone(11,x);
    delay(2000);
    myNoTone();
  }
}
```

loop함수에서는 사용자로부터 숫자를 입력받는다. 입력받을 때는 inputFromTerm함수를 사용한다.

```
int inputFromTerm(){
  String inString;
  while(1){
    if(Serial.available()>0){
      char inchar=Serial.read();
      if(inchar=='\n') break;
     Serial.print(inchar);
     inString+=(char)inchar;
    }
  }
  return(inString.toInt());
}
```

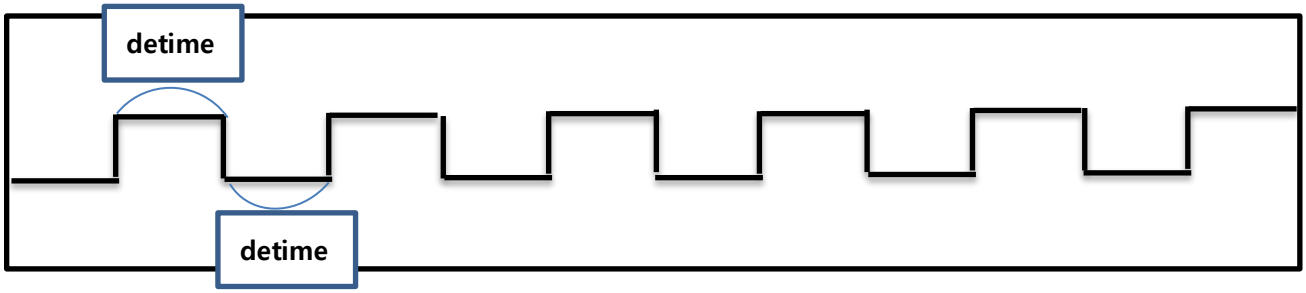
inputFromTerm함수는 String형 변수 inString을 선언 후 무한반복문을 통해서 사용자로부터 문자를 한 개씩 입력받아 그 문자를 출력하고 inString에 더한다. 그 후 사용자가 엔터키('\n')를 누르면 무한반복문에서 빠져나와 inString을 정수로 변환 후 반환한다.

만약 사용자가 입력한 숫자가 500초과라면 " 잘못된 범위 주파수"를 출력하고 사용자로부터 입력을 다시 받는다.

만약 사용자가 500이하 범위의 숫자를 입력하면 myTone함수를 통해 스피커에 소리가 나오고, delay(2000);을 통해 2초동안 소리 출력 후 myNoTone함수를 통해 소리를 멈춘다.

```
void myTone(int pin, int frequency){
  detime=((1/(double)frequency)*0.5)*1000000;
  frequen=frequency;
  SPK=pin;
  state=1;
}
```

myTone()함수에서는 detime=((1/(double)frequency)*0.5)*1000000; 식을 사용해서 전역변수 detime에 값을 저장한다.



detime이란 파형에서 HIGH 혹은 LOW가 지속되는 시간이다. 단위는 마이크로이다.

그리고 myTone()함수의 매개변수 pin(스피커 핀단자)를 전역변수 SPK에 저장하고, 매개변수 frequency(사용자입력주파수)를 전역변수 frequen에 저장한다.
그리고 state전역변수를 1로 초기화한다. 이 순간 MsTimer2의 인터럽트 fre함수에 중요한 변화가 생긴다.

```
void fre(){
  int n=0;
  if(state==0){
    while(n<1000){
      digitalWrite(2,1); delayMicroseconds(500);
      digitalWrite(2,1); delayMicroseconds(500);
      n++;
    }
  }
  else if(state==1){
    while(n<frequen){
      digitalWrite(SPK,1); delayMicroseconds(detime);
      digitalWrite(SPK,0); delayMicroseconds(detime);
      n++;
    }
  }
}
```

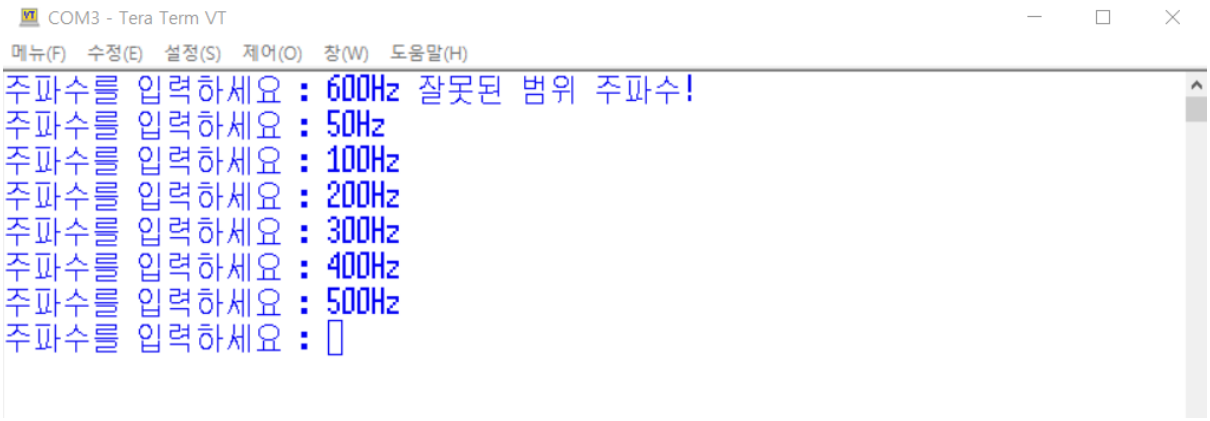
fre()함수는 MsTimer2의 인터럽트 시 1초마다 호출되는 함수로 평소에는 state가 0이기 때문에 아무 의미없는 핀단자에 1000Hz주파수를 내보낸다.

하지만 myTone()함수에서 state를 1로 바꿔주고 SPK, detime변수를 myTone()함수에서 해당하는 핀번호와 주파수에 맞는 detime을 초기화해줬기 때문에 state가 1이 만족이 되서 1초마다 frequen개의 LOW, HIGH 파형이 출력이되 즉 frequen(Hz)가 스피커단자(SPK)를 통해서 나오게 된다.

```
void myNoTone(){
  state=0;
}
```

myNoTone()함수에서는 state전역변수를 0으로 초기화한다. 그렇게 되면 fre()함수 내에 조건문에 의해 스피커에 출력이 멈춰진다.

▪실행



테라텀을 통해 실행한다.

터미널 설정에서 줄바꿈 설정에서 수신, 송신 모두 LF로 설정해야 정상적으로 동작한다.

500초과 숫자를 입력하면 “잘못된 범위 주파수!”가 출력이 되고, 다시 사용자입력을 기다린다.

사용자가 입력한 숫자의 주파수로 스피커를 통해서 소리가 2초동안 나온 후 다시 사용자입력을 기다린다.

▪실험과정에서의 경험과 습득 사실

- digitalWrite()함수를 통해서 스피커에 소리를 나오게 할 수 있었다.

▪13번

- 20KHz의 클럭 소스를 이용하여 인터럽트를 발생시키고자 한다. 이를 기반으로 tone() 함수를 설계하고 한 옥타브의 음정을 연주하고자 한다. 위과정에 대한 타당성 및 실행 방안을 다음 관점에서 기술.

1. 20KHz의 클럭 소스를 타이머로부터 생성하는 방법의 가능성 타진

타이머를 사용하기 위해서 타이머 라이브러리를 다운해야 한다. MsTimer2 라이브러리를 다운받는다고 한다면 MsTimer2::set(1000, 함수); MsTimer2::start();를 한다. 20KHz는 20000Hz이다. 그러므로 호출될 함수안에서 digitalWrite(핀번호, HIGH); digitalWrite(핀번호, LOW);가 20000만 반복되게 하면된다.

```
void 함수이름(){
    int n=0;
    while(n<20000){
      digitalWrite(핀번호, HIGH);   delayMicroseconds(25);
      digitalWrite(핀번호, LOW);   delayMicroseconds(25);
      n++;
    }
}
```

이런식으로 함수를 짜주면 20KHz의 클럭소스가 확보된다.

2. 20KHz의 주파수를 가진 외부 OSC 소자 혹은 H/W회로를 이용하여 클럭소스를 확보하는 방법

20KHz의 주파수를 가진 외부 OSC 소자 혹은 H/W회로를 확보했다면 OSC 소자 혹은 H/W회로를 아두이노 보드에 연결한다. OSC 소자 혹은 H/W회로를 아두이노 보드의 GPIO단자에 연결한다. 그 연결한 핀을 A번핀이라고 했을 때, A번 핀을 digitalRead()함수를 통해 값을 읽어들여 1 혹은 0값을 그대로 digitalWrite()함수를 통해서 출력시켜 20KHz의 주파수를 생성한다.

3. 1혹은 2의 방법으로 클럭소스가 확보되었다고 하고 자체 tone() 함수를 제작하시오.

클럭소스가 확보된 후 클럭소스를 이용해 1초마다 인터럽트를 발생하게 한다. 그래서 인터럽트가 발생하면 digitalWrite()함수를 통해 스피커에 1, 0을 반복 출력시켜서 소리가 나오게 한다.

digitalWrite()함수 사이사이에 delayMicroseconds(N);을 해주어 해당하는 주파수가 스피커를 통해 출력되게 해준다.