7. BUZZER와 7조 센서

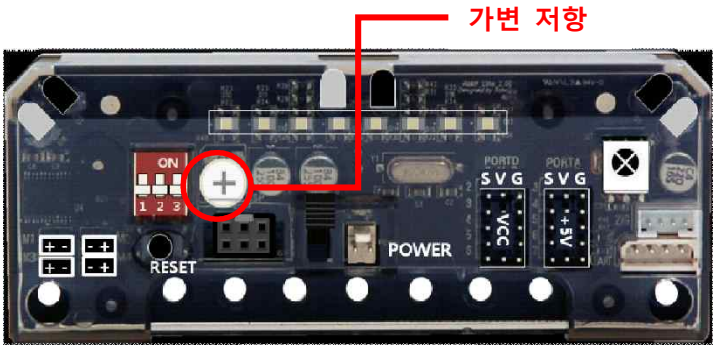
1) 개요

-Buzzer

피에조 장치(piezo device)는 전기 에너지를 진동으로 바꾸어주는 장치의 일종으로서 공기 진동을 발생시켜 소리를 만들어 낸다. 소리에는 높낮이가 있고 이 높낮이는 1초 동안 스피커가 몇 번 진동하며 공기를 진동하는 지의 여부에 따라 결정된다. 이처럼 1초 동안 진동하는 진동수를 Hz라 하며 이 진동수에 의해 형성되는 소리의 높낮이를 피치(pitch)라 한다. 아두이노에서는 tone이라는 함수를 이용하여 제어기에 연결된 스피커를 일정한 주파수로 진동시켜 소리를 만들어 낼 수 있다.

-7조 센서

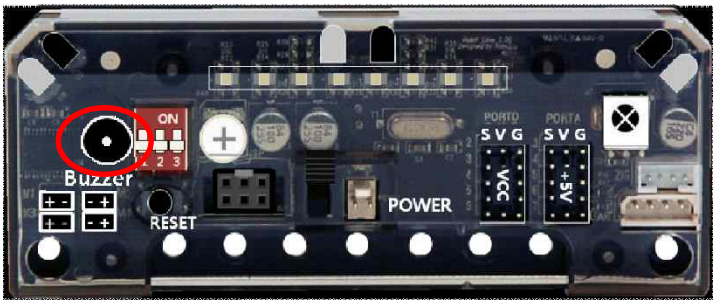
스마트 인벤터 보드의 특징 중 하나는 라인 트레이서 로봇을 쉽게 만들 수 있도록 바닥의 7개의 IR센서를 배치하였다는 점이다. 바닥에 있는 7개의 IR센서는 센서로부터 1cm 이내의 물체를 감지하여 단순히 HIGH(감지됨) 나 LOW(감지 안됨)으로 알려주거나 흰색 바탕의 검은 색을 감지하여 HIGH(감지 안됨)나 LOW(감지됨)로 검은색 감지 여부를 알려준다. 바닥의 7조 센서는 제어기 윗부분의 가변 저항을 돌려서 감도를 조절할 수 있다. 7조 센서를 처음 사용하거나 환경이 바뀐 경우에는 반드시 감도를 조절하여야 한다.



2) 연결 방법

-Buzzer

스마트 인벤터 제어기는 피에조 스피커를 내장하고 있으며 7번 핀과 연결되어 있다. 이 7번 핀에 일정한 Hz의 신호를 인가 함으로서 비프(beep)음을 만들어 낼 수 있다. 따라서 프로그램 코딩 시 스피커 출력 핀을 7번으로 설정해 주기만 하면 되며 따로 연결 방법이나 핀 배열을 주의할 필요가 없다.



- 7조 센서

7조 센서는 스마트 인벤터 보드에 내장된 센서이므로 별도의 연결은 필요하지 않다. 다만 7개의 센서 각각은 핀 번호를 갖고 있으므로 스위치 값을 읽는 것처럼 프로그래밍하면(3장 스위치 참조) 각각의 센서 감지 여부를 알 수 있다. 자세한 것은 아래 예제에서 살펴보기로 한다.

3) 아두이노 함수 설명

- Buzzer

비프음을 만들어 내는 tone함수는 따로 라이브러리를 추가하지 않고서도 사용할 수 있다. 다만 피치 데이터를 한데 묶은 pitches.h 파일을 필요로 하는 예제가 있으며 그렇지 않은 예제도 있으므로 예제를 실행하기 전에 먼저 살펴볼 필요가 있다. pitches.h 파일은 아두이노 IDE의 tone예제 폴더에 함께 포함되어 있는 것을 사용하거나 아두이노 홈페이지에서 얻을 수 있다.

tone(pin, frequency)
tone(pin, frequency, duration)

tone() 함수의 첫번째 파라메터인 pin은 스피커와 연결된 핀을 의미한다. Frequency에는 피치를 입력한다. duration은 음이 재생되어야 하는 시간으로서 밀리세컨드(1/1000초) 단위로 입력한다.

noTone(pin)

해당 핀의 음 생성 신호를 멈춘다. 다른 음계를 연주하고 싶다면 이 함수를 사용하여 일단 기존의 음 생성을 멈추어야 한다.

-7조 센서

pinMode(핀번호, INPUT);

7개 센서들의 상태를 읽는 것은 스위치의 상태를 읽는 것과 동일하다. 따라서 7개의 센서들은 입력 장치로 볼 수 있으며 센서에 연결된 각각의 핀들을 입력(INPUT)으로 설정하여야 한다.

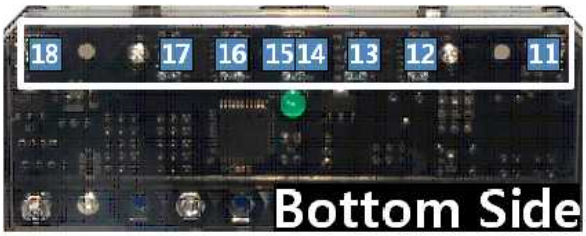
digitalRead(pin);

pin: 디지털 핀 번호

반환: HIGH(1), LOW(0)

핀의 상태를 읽어오는 함수로서 디지털 형식의 값인 HIGH(1) 아니면 LOW(1)의 값으로 알려준다. HIGH일 때에는 물체가 감지되었거나 검은색이 감지되지 않은 경우이며 LOW일 때에는 물체가 감지되지 않았거나 검은색이 감지된 경우이다.

여기서 유의해야 할 것은 14번과 15번 핀은 가운데 센서에 연결되어 있어 가운데 센서를 감지하면 핀의 상태가 동일하게 변경된다는 것이다.



4)프로그래밍

예제1 : 짧은 멜로디를 연주하는 예제이다. 이 예제를 실행하기 전에 예제가 들어있는 폴더에 음계 데이터가 들어있는 pitches.h 파일이 함께 들어 있어야 한다.

```
#include "pitches.h"                //피치 정의 헤더 파일 포함

int melody[] = { NOTE_C4, NOTE_G3,NOTE_G3, NOTE_A3, NOTE_G3,0, NOTE_B3, NOTE_C4};
//연주할 피치들을 정의함
int noteDurations[] = {  4, 8, 8, 4,4,4,4,4 };  //각 피치의 연주해야 할 길이를 음표로 나타냄(4: 4분음표 8:8분음표)

void setup() {
```

```
  for (int thisNote = 0; thisNote < 8; thisNote++)
  {
      int noteDuration = 1000/noteDurations[thisNote];     //음표를 1/1000초로 환산
      tone(7, melody[thisNote],noteDuration);              //7번 핀에 스피커 연결
      int pauseBetweenNotes = noteDuration * 1.30;         //음 사이의 간격 계산
      delay(pauseBetweenNotes);
       noTone(8);
  }
}

void loop() {
  // no need to repeat the melody.
}
```

예제검토: 연주해야 할 데이터들은 두 개의 배열 변수 안에 저장되었다. Melody[]배열은 연주해야 할 음계들을 갖고 있다. pitches.h파일을 열어보면 옥타브 및 음계 별 피치 정의가 되어 있음을 볼 수 있는데 예를 들어 NOTE_C4의 C는 음계를, 4는 옥타브를 의미한다.

noteDuration[]은 음들의 길이에 대한 데이터가 있는 배열로서 1/1000초 단위가 아니라 음표 단위로 표시하였다. 4는 4분 음표를, 8은 8분 음표를 의미한다.

이 음표들은 for문 바로 아래 줄 계산식이 들어 있는 코드에서 순서대로 실제 1/1000초 단위로 환산하여 tone함수의 파라메터로 보내어 지는 것을 볼 수 있다. 또한 음과 음 사이에 쉼을 주기 위해서 pauseBetweenNotes를 계산해서 딜레이를 주어 쉼표를 만들어 내기도 하였다.

예제2: 7조 센서 를 감지시킬 때 음계가 연주되도록 하는 프로그램이다.

```
#include "pitches.h"

int pitch = 0;

void setup() {
  // put your setup code here, to run once:

  pinMode(11, INPUT);
  pinMode(12, INPUT);
```

```
  pinMode(13, INPUT);
  pinMode(14, INPUT);
  pinMode(15, INPUT);
  pinMode(16, INPUT);
  pinMode(17, INPUT);
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:

  int s1 = digitalRead(11);
  int s2 = digitalRead(12);
  int s3 = digitalRead(13);
  int s4 = digitalRead(14);
  int s5 = digitalRead(15);
  int s6 = digitalRead(16);
  int s7 = digitalRead(17);
  int s8 = digitalRead(18);


  if(s1) pitch = NOTE_C3;
  else if(s2) pitch = NOTE_D3;
  else if(s3) pitch = NOTE_E3;
  else if(s4) pitch = NOTE_F3;
  else if(s6) pitch = NOTE_G3;
  else if(s7) pitch = NOTE_A4;
  else if(s8) pitch = NOTE_B4;
  else pitch = 0;

  tone(7, pitch, 100);
}
```