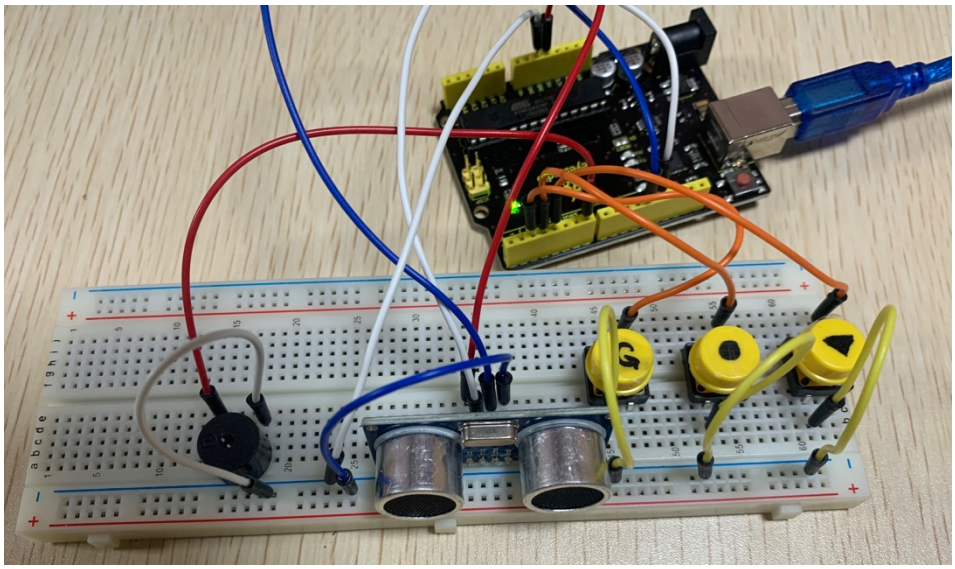
제목 : 거리 위의 작곡

#멜로디 #작곡 #음감훈련 #재미

주제 :

- 거리(street)위에서 거리(distance)를 이용하여 노래를 작곡해본다.
- 거리와 버튼푸시 지속시간에 따라 달라지는 음의 높이와 길이를 통해 관련 개념을 학습할 수 있다.
- 음을 기록할 때는 소리가 나지 않기 때문에 어떻게 멜로디가 기록될지 상상하는 재미와 동시에 음감을 훈련할 수 있다.

재료 :



좌측부터

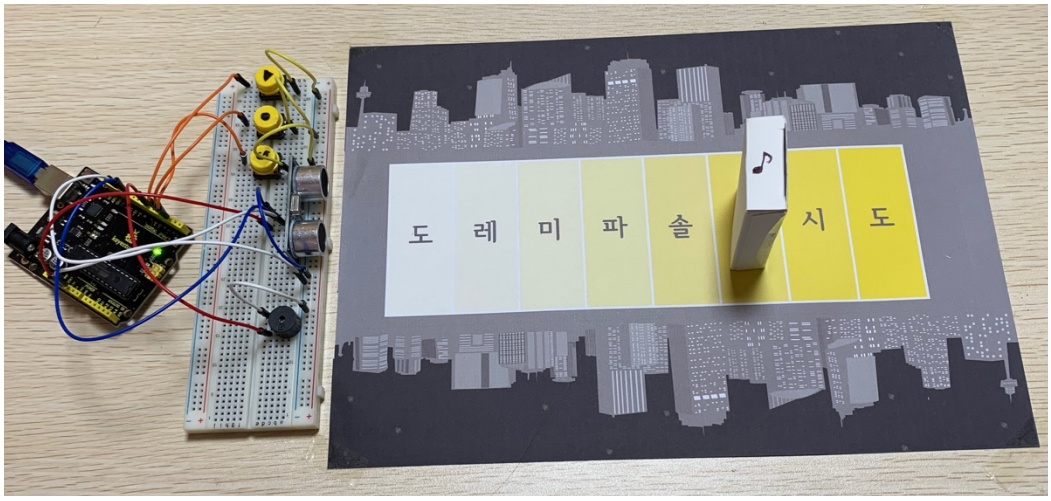
1. 수동 부저
2. 초음파 센서
3. 버튼 3개 (Reset, Record, Play)

동작설명 :

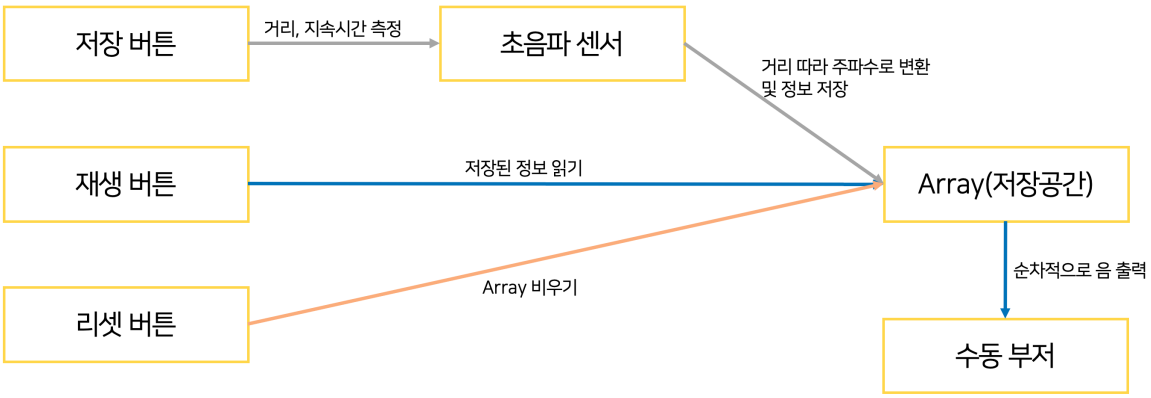
1. 시작
2. 원하는 음에 해당하는 칸에 물체를 놓고 저장버튼(a) 누르기

- 초음파 센서로 물체와의 거리와 버튼 누른 지속시간 측정하여 해당하는 2개의 Array에 각각 저장함

3. 2번 여러 번 반복
4. 재생버튼(b) 누르기
5. Array에 저장된 음의 높이가 지속시간 반영하여 차례대로 재생
6. 리셋버튼(c) 누르면 1번으로 돌아감.



시스템 구성도 :



보완점 :

- 칸 위에 올려놓는 물체는 원래 사람 형태의 인형이나 피규어로 하고 싶었는데, 얇아서 그런지 센서가 인식하지 못하는 문제가 있었고, 거리 측정도 가끔 제대로 안될 때가 있었는데 더 정밀하고 좋은 센서를 사용하면 좋을 것이다.
- 재생버튼을 누르면 노래가 나오는 동안 아무 버튼도 작동시킬 수 없는데, 버튼을 하나 더 만들어서 노래를 중간에 멈추고 다른 버튼을 작동시킬 수 있게하면 더욱 발전된 작품이 될 것이다.
- 배열 크기를 30개로 제한해서 최대 30개의 음까지 저장할 수 있는데, 저장공간을 더 늘려도 좋을 것 같다.
- 지금은 '도레미파솔라시도' 8가지 음만 가능하지만 음역대를 넓히면 더 다양한 멜로디를 작곡할 수 있을 것이다.

자기평가 :

다른 과목에서 프로그래밍한 경험이 몇 번 있었지만, 아두이노는 메이커스 워크샵' 강의에서 난생처음 접했다. 그래서 처음 몇 주는 강의가 어렵게 느껴졌었는데, 그래도 실습을 몇 번 따라해보니 프로그래밍 언어나 부품 연결 등의 개념을 이해할 수 있어서 다행이었다.

기획서 제출 당시 기획했던 부분을 정확하게 그대로 재현한 것 같고, 완성도 있게 작품이 나온 것 같아서 뿌듯하다. 코드도 이 정도면 꽤 간결하고 효율적으로 짠 편이라고 생각한다. 코드에 문제가 있어 작동이 잘 안될 때 코드 사이사이마다 시리얼모니터에 글자를 출력하게 했더니, 어디가 잘못된 건지 파악할 수 있었다.

밑에 깔아놓은 종이도 일러스트로 직접 제작하여 프린트했다. 센서가 측정하는 거리를 정확하게 계산해서 종이에 표시하는 작업이 조금 오래 걸렸지만, 그만큼 정확도가 높은 작품이 되었다.

완성도, 정확성, 예술성을 종합적으로 고려해봤을 때 이 작품은 A+를 받아도 괜찮을 것 같다.

코드 :

```
#define trigPin 13
#define echoPin 12
#define btn_reset 2
#define btn_record 3
#define btn_play 4
#define piezo 7
int scale [30];
int duration [30];
int i=0;

void setup()
{
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(btn_reset, INPUT_PULLUP);
  pinMode(btn_record, INPUT_PULLUP);
  pinMode(btn_play, INPUT_PULLUP);
  pinMode(piezo,OUTPUT);
  Serial.begin(9600);
}

void loop(){
  bool reset = digitalRead(btn_reset);
  bool record = digitalRead(btn_record);
  bool play = digitalRead(btn_play);

  if (reset==LOW){
    for(int t=0;t<=i;t++){
     scale[t]=0;
     duration[t]=0;
    }
    i=0;
    Serial.println("reset");
    delay(500);
```

```
  }
  if (record==LOW){
    int s_time = millis();
    while(1){
        Serial.println("while loop");
        bool new_record = digitalRead(btn_record);
        if(new_record==HIGH){
          break;}
      }
    int e_time = millis();
      scale[i]=measure_distance();
      duration[i]=e_time-s_time;
      if(scale[i]==1000){
         duration[i]=0;}

    Serial.println("Record");
    Serial.println(i);
    Serial.println(scale[i]);
    Serial.println(duration[i]);
    delay(500);
    i++;
  }
  if (play==LOW){
    for(int t=0;t<=i;t++){
      tone(7, scale[t]);
      delay(duration[t]);
      noTone(7);
      delay(300);
    }
    Serial.println("PLAY");
    delay(1000);
  }
  delay(100);
}

int measure_distance(){
  long duration, distance;                    // 각 변수를 선언합니다.
  digitalWrite(trigPin, LOW);                 // trigPin 에 LOW 를 출력하고
  delayMicroseconds(2);                       // 2 마이크로초가 지나면
  digitalWrite(trigPin, HIGH);                // trigPin 에 HIGH 를 출력합니다.
  delayMicroseconds(10);                      // trigPin 을 10 마이크로초 동안 기다렸다가
  digitalWrite(trigPin, LOW);                 // trigPin 에 LOW 를 출력합니다.

  duration = pulseIn(echoPin, HIGH);    // echoPin 핀에서 펄스값을 받아옵니다.
  distance = duration * 17 / 1000;
  //거리에 따른 주파수 반환
  if (distance>=2 && distance<5){
    return 262;
  }
  else if (distance>=5 && distance<8){
    return 294;
  }
  else if (distance>=8 && distance<11){
    return 330;
  }
  else if (distance>=11&& distance<14){
    return 349;
  }
```

```
    else if (distance>=14 && distance<17){
      return 392;
    }
    else if (distance>=17 && distance<20){
      return 440;
    }
    else if (distance>=20 && distance<23){
      return 494;
    }
    else if (distance>=23 && distance<26){
      return 523;
    }
    else {return 1000;}
}
```