به نام خدا

گزارش کار آزمایش نهم آزمایشگاه ریزپردازنده و زبان اسمبلی

امیرپارسا سلمان خواه

۹۸۳۱۰۳۴

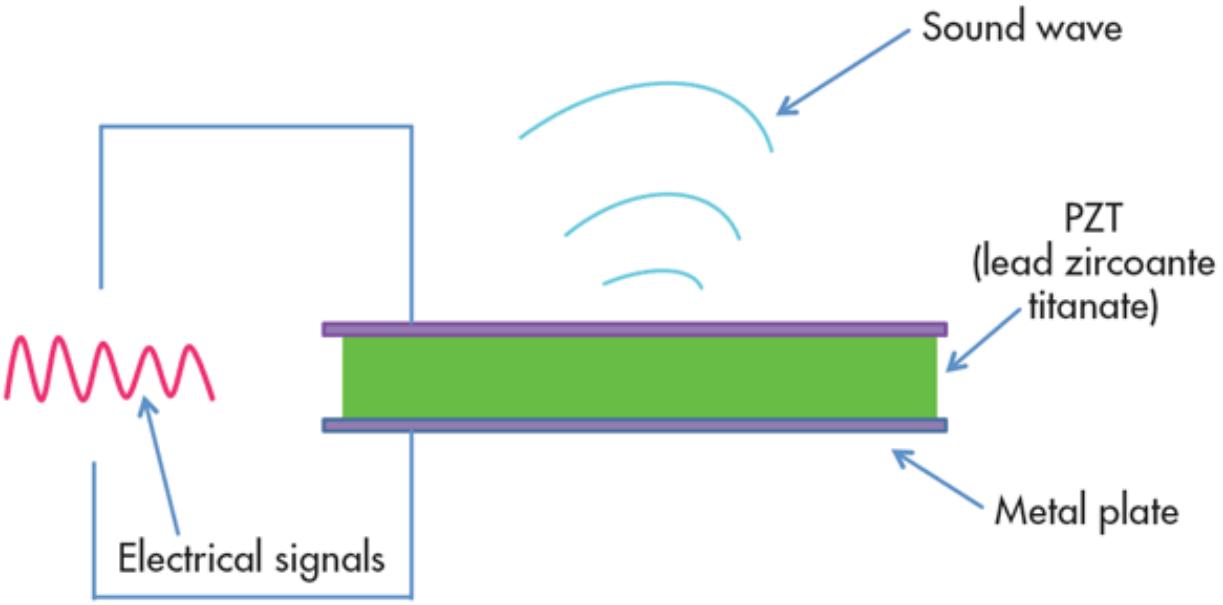
پرسش: چند مورد از کاربرد های پیزوالکتریک در دنیای واقعی را نام ببرید.

از پیزوالکتریک در کاربرد هایی نظیر اسپیکر و بلندگو، موتور، فندک های الکتریکی و سیستم های شناسایی موانع در زیردریایی ها استفاده می شود.

پرسش: اسپیکر پیزوالکتریک ما چطور کار می‌کند؟ فکر می‌کنید چرا این روش کار انتخاب شده است؟

پدیده ی پیزوالکتریک در بعضی از کریستال ها که ساختار غیر متقارن دارند دیده می شود. این پدیده به این شکل است که با اعمال یک نیروی مکانیکی به کریستال، ساختار آن فشرده می شود و در نتیجه ی آن یک جریان الکتریکی تولید می شود. در واقع با پدیده ی پیزوالکتریک می توان انرژی مکانیکی را به انرژی الکتریکی تبدیل کرد.

اما چیزی که در اسپیکر های پیزوالکتریک استفاده می شود، عکس این پدیده می باشد. به این معنی که با اعمال یک جریان الکتریکی به یک کریستال که در بین دو صفحه ی فلزی قرار گرفته است، کریستال باز تر می شود و در نتیجه ی آن انرژی مکانیکی را به شکل موج های صوتی از خود آزاد می کند.



روش پیزوالکتریک برای ساخت اسپیکر ها مزایای زیادی نسبت به روش الکترودینامیک دارد. روش الکترودینامیک نیازمند سیم پیچ و آهن ربا است و این قطعات جا گیر هستند و نمی توان از آن ها در فضای کوچک استفاده کرد. از طرفی کیفیت صدای این اسپیکر ها نیز از اسپیکر های پیزوالکتریک پایین تر است. همچنین در برخی محیط ها مانند زیر آب نیز نمی توان از اسپیکر های الکترودینامیک استفاده

کرد. این در حالی است که برای ساخت سیستم شناسایی موانع در زیردریایی ها از پدیده ی پیزوالکتریک استفاده می شود.

کد های مربوط به آزمایش اول(پخش آهنگ با فشردن دکمه ی کی پد):

فایل pitches.h

```
#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
```

```
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
```

```
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
```

فایل themes.h

```
// notes in the jingleBellsMelody:
int jingleBellsMelody[] = {
  NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_G5, NOTE_C5, NOTE_D5,
  NOTE_E5,
  NOTE_F5, NOTE_F5, NOTE_F5, NOTE_F5,
  NOTE_F5, NOTE_E5, NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_D5, NOTE_D5, NOTE_E5,
  NOTE_D5, NOTE_G5
};

// NOTE_durations: 4 = quarter note, 8 = eighth note, etc.:
int jingleBellsNoteDurations[] = {

  4, 4, 2,
  4, 4, 2,
  4, 4, 3, 8,
  1,
  4, 4, 4, 4,
  4, 4, 4, 8, 8,
  4, 4, 4, 4,
  2, 2
};

int odeToJoyMelody[] = {
  NOTE_E5, NOTE_E5, NOTE_F5, NOTE_G5,
  NOTE_G5, NOTE_F5, NOTE_E5, NOTE_D5,
  NOTE_C5, NOTE_C5, NOTE_D5, NOTE_E5,
  NOTE_E5, NOTE_D5, NOTE_D5,
  NOTE_E5, NOTE_E5, NOTE_F5, NOTE_G5,
  NOTE_G5, NOTE_F5, NOTE_E5, NOTE_D5,
  NOTE_C5, NOTE_C5, NOTE_D5, NOTE_E5,
  NOTE_D5, NOTE_C5, NOTE_C5,
  NOTE_D5, NOTE_D5, NOTE_E5, NOTE_C5,
  NOTE_D5, NOTE_E5, NOTE_F5, NOTE_E5, NOTE_C5,
  NOTE_D5, NOTE_E5, NOTE_F5, NOTE_E5, NOTE_D5,
  NOTE_C5, NOTE_D5,
};

int odeToJoyNoteDurations[] = {
```

```
  4, 4, 4, 4,
  4, 4, 4, 4,
  4, 4, 4, 4,
  3, 8, 2,
  4, 4, 4, 4,
  4, 4, 4, 4,
  4, 4, 4, 4,
  3, 8, 2,
  4, 4, 4, 4,
  4, 4, 4, 4, 4,
  4, 4, 4, 4, 4,
  4, 4
};

int underworldMelody[] = {
  NOTE_C4, NOTE_C5, NOTE_A3, NOTE_A4,
  NOTE_AS3, NOTE_AS4, 0,
  0,
  NOTE_C4, NOTE_C5, NOTE_A3, NOTE_A4,
  NOTE_AS3, NOTE_AS4, 0,
  0,
  NOTE_F3, NOTE_F4, NOTE_D3, NOTE_D4,
  NOTE_DS3, NOTE_DS4, 0,
  0,
  NOTE_F3, NOTE_F4, NOTE_D3, NOTE_D4,
  NOTE_DS3, NOTE_DS4, 0,
  0, NOTE_DS4, NOTE_CS4, NOTE_D4,
  NOTE_CS4, NOTE_DS4,
  NOTE_DS4, NOTE_GS3,
  NOTE_G3, NOTE_CS4,
  NOTE_C4, NOTE_FS4, NOTE_F4, NOTE_E3, NOTE_AS4, NOTE_A4,
  NOTE_GS4, NOTE_DS4, NOTE_B3,
  NOTE_AS3, NOTE_A3, NOTE_GS3,
  0, 0, 0
};
//Underwolrd tempo
int underworldNoteDurations[] = {
  12, 12, 12, 12,
  12, 12, 6,
  3,
  12, 12, 12, 12,
  12, 12, 6,
  3,
  12, 12, 12, 12,
  12, 12, 6,
  3,
```

```
  12, 12, 12, 12,
  12, 12, 6,
  6, 18, 18, 18,
  6, 6,
  6, 6,
  6, 6,
  18, 18, 18, 18, 18, 18,
  10, 10, 10,
  10, 10, 10,
  3, 3, 3
};
```

فایل piezo.ino

```
#include "pitches.h"
#include "themes.h"
#include <Keypad.h>

#define PIEZO_PIN 10

#define NOTE_BASE_TIME 1000

const byte ROWS = 4;
const byte COLS = 4;

char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

#define KEYPAD1 9
#define KEYPAD2 8
#define KEYPAD3 7
#define KEYPAD4 6
#define KEYPAD5 5
#define KEYPAD6 4
#define KEYPAD7 3
#define KEYPAD8 2

byte rowPins[ROWS] = {KEYPAD1, KEYPAD2, KEYPAD3, KEYPAD4}; //connect to the
row pinouts of the keypad
byte colPins[COLS] = {KEYPAD5, KEYPAD6, KEYPAD7, KEYPAD8}; //connect to the
column pinouts of the keypad

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
```

```
void play_music(int melody[], int note_duration[], int melody_length);

void setup() {

  // iterate over the notes of the jingleBellsMelody:
}

void loop() {
  char c = keypad.waitForKey();
  if (c == '1'){
    play_music(jingleBellsMelody, jingleBellsNoteDurations,
sizeof(jingleBellsMelody)/sizeof(int));
  }
  else if (c == '2'){
    play_music(odeToJoyMelody, odeToJoyNoteDurations, sizeof(odeToJoyMelody)/sizeof(int));
  }
  else if (c == '3'){
    play_music(underworldMelody, underworldNoteDurations,
sizeof(underworldMelody)/sizeof(int));
  }
  // no need to repeat the jingleBellsMelody.
}

void play_music(int melody[], int note_duration[], int melody_length){

  for (int thisNote= 0; thisNote< melody_length; thisNote++) {

    // to calculate the NOTE_duration, take NOTE_BASE_TIME divided by the NOTE_type.

    //e.g. quarter NOTE_= NOTE_BASE_TIME / 4, eighth NOTE_= NOTE_BASE_TIME/8, etc.

    int noteDuration = NOTE_BASE_TIME / note_duration[thisNote];

    tone(PIEZO_PIN, melody[thisNote], noteDuration);

    // to distinguish the notes, set a minimum time between them.

    // the note's duration + 30% seems to work well:

    int pauseBetweenNotes = noteDuration * 1.30;

    delay(pauseBetweenNotes);

    // stop the tone playing:
```

```
    noTone(PIEZO_PIN);
  }

}
```

کد های مربوط به آزمایش دوم(زیر و بم شدن صدا با پتانسیومتر):

```
#include "pitches.h"
#include "themes.h"
#include <Keypad.h>

#define PIEZO_PIN 10
#define POT_PIN A0

#define NOTE_BASE_TIME 1000

const byte ROWS = 4;
const byte COLS = 4;

char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

#define KEYPAD1 9
#define KEYPAD2 8
#define KEYPAD3 7
#define KEYPAD4 6
#define KEYPAD5 5
#define KEYPAD6 4
#define KEYPAD7 3
#define KEYPAD8 2

byte rowPins[ROWS] = {KEYPAD1, KEYPAD2, KEYPAD3, KEYPAD4}; //connect to the
row pinouts of the keypad
byte colPins[COLS] = {KEYPAD5, KEYPAD6, KEYPAD7, KEYPAD8}; //connect to the
column pinouts of the keypad

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

void play_music(int melody[], int note_duration[], int melody_length);

void setup() {
  pinMode(POT_PIN, INPUT);
  // iterate over the notes of the jingleBellsMelody:
```

```
}

void loop() {
  char c = keypad.waitForKey();
  if (c == '1'){
    play_music(jingleBellsMelody, jingleBellsNoteDurations,
sizeof(jingleBellsMelody)/sizeof(int));
  }
  else if (c == '2'){
    play_music(odeToJoyMelody, odeToJoyNoteDurations, sizeof(odeToJoyMelody)/sizeof(int));
  }
  else if (c == '3'){
    play_music(underworldMelody, underworldNoteDurations,
sizeof(underworldMelody)/sizeof(int));
  }
   no need to repeat the jingleBellsMelody.
}

void play_music(int melody[], int note_duration[], int melody_length){

  for (int thisNote= 0; thisNote< melody_length; thisNote++) {

    // to calculate the NOTE_duration, take NOTE_BASE_TIME divided by the NOTE_type.

    //e.g. quarter NOTE_= NOTE_BASE_TIME / 4, eighth NOTE_= NOTE_BASE_TIME/8, etc.

    int noteDuration = NOTE_BASE_TIME / note_duration[thisNote];

    // Calculating pot value
    int pot = analogRead(POT_PIN);
    int scale = map(pot, 0, 1023, 0, 2);

    int note = melody[thisNote];
    if (scale == 0){
      note = note / 2;
    }
    else if (scale == 2){
      note = note * 2;
    }

    tone(PIEZO_PIN, note, noteDuration);

    // to distinguish the notes, set a minimum time between them.

    // the note's duration + 30% seems to work well:
```

```
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // stop the tone playing:
    noTone(PIEZO_PIN);
  }
}
```