**به نام خدا**

گزارش کار آزمایش نهم آزمایشگاه ریزپردازنده و زبان اسمبلی

امیرپارسا سلمان خواه

۹۸۳۱۰۳۴

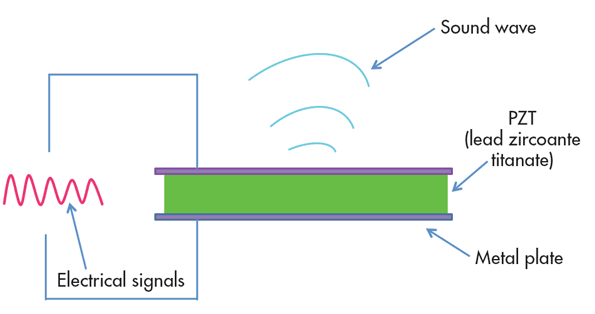


از پیزوالکتریک در کاربرد هایی نظیر اسپیکر و بلندگو، موتور، فندک های الکتریکی و سیستم های شناسایی موانع در زیردریایی ها استفاده می شود.



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 الکتروداینامیک دارد. روش الکتروداینامیک 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 الکتروداینامیک 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);

}

}