Dmytro Matviichuk 240545
Konrad Kieda 250997

Termin: pt TN 17:15-20:15

# Urządzenia peryferyjne

## Temat: Obsługa karty dźwiękowej

Kod źródłowy(GitHub ): https://github.com/dmatviichuk/Urzadzenia_Peryferyjne

## 1. Cel ćwiczenia

Celem ćwiczenia było napisanie programu, który wykryje urządzenia typu karta dźwiękowa.
Zadania do wykonania:

1.Korzystając z przykładowej aplikacji odtworzyć dźwięk za pomocą trzech sposobów: ActiveX,Waveform and Auxiliary Audio i DirectSound.
2.Napisać program, odtwarzający dźwięk z wykorzystaniem komendy PlaySound() oraz z wykorzystaniem ActiveX`a.
3.Napisać program zczytujący nagłówek WAVa i wyświetlić poszczególne wartości.
4.Napisać program odtwarzający WAVa za pomocą DirectSounda i Waveform and Auxiliary Audio z wykorzystaniem komendy waveOutWrite().
5.Napisać program odtwarzający wybrany (z listy plików) plik w formacie MP3 (wykorzystać WindowsMediaPlayer)
6.Napisać aplikację która sczytuje z mikrofonu dane i zapisuje je do pliku.
7. Wyświetlić na ekranie moc sygnału (np. słupki) dla kilku podprzedziałów częstotliwości

## 2. Wstęp

**Karta dźwiękowa, karta muzyczna** – komputerowa karta rozszerzeń umożliwiająca rejestrację, przetwarzanie i odtwarzanie dźwięku, słuchanie muzyki. Obecnie układy dźwiękowe wystarczające do zastosowań amatorskich są zazwyczaj wbudowywane w płytę główną komputera, nie stanowiąc już karty rozszerzenia. Z powodów historycznych są jednak określane mianem „zintegrowana karta dźwiękowa”. Pojawiły się również zewnętrzne karty dźwiękowe podłączane do komputera przez port USB.

**DirectSound** – część pakietu DirectX umożliwiającą szybki dostęp do karty dźwiękowej m.in. odtwarzanie i nagrywanie dźwięku. Obsługuje efekty dźwiękowe (jak np. echo). Ta część DirectX działa tylko pod systemem Windows.

**ActiveX** – rodzaj komponentów i kontrolek możliwy do użycia w programach pisanych za pomocą takich narzędzi jak Delphi, Visual Basic, C++, Java, Power Builder i wielu innych. Technologia ActiveX pozwala na przekazywanie danych pomiędzy różnymi aplikacjami działającymi pod kontrolą systemów operacyjnych Windows.

**WAV** – format plików dźwiękowych stworzony przez Microsoft oraz IBM. WAVE bazuje na formacie RIFF, poszerzając go o informacje o strumieniu audio, takie jak użyty kodek, częstotliwość próbkowania czy liczba kanałów. WAV podobnie jak RIFF został przewidziany dla komputerów IBM PC, toteż wszystkie zmienne zapisywane są w formacie little endian.

## 3. Infromancja o programie oraz fragmenty kodu

Program został napisany w języku C#.

*3.1 Biblioteki które zostały użyte w naszej aplikacji:*

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;
using System.Media;
using System.Runtime.InteropServices;
using System.Windows.Forms;
using AxWMPLib;
```

*3.2 Odtworzenie pliku .wav w aplikacji*

Z prawej strony znajduje się komponent ActiveX - Windows Media Player. Na lewym panelu aplikacji do obsługi pliku .wav w celu odtwarzania dźwięku uzywa się trzech przycisków: ChooseFile, Play, Stop. Została wykorzystana klasa System.Media.SoundPlayer

```
private void fileDialogBtn_Click(object sender, EventArgs e)
{
    OpenFileDialog fileDialog = new OpenFileDialog();
    fileDialog.Filter = "Audio Files(*.wav)|*.wav";
    if (fileDialog.ShowDialog() == DialogResult.OK)
     AudioCardController.filepath = fileDialog.FileName;
    filepath = fileDialog.FileName;
    Console.WriteLine(filepath);
}

private void SoundClick_Click(object sender, EventArgs e)
{
    audio.PlaySound(filepath);
}

private void StopSound_btn_Click(object sender, EventArgs e)
{
    audio.StopSound();
}
```

*3.2 Funkcja PlaySound*

```
public void PlaySound(string fileName) //gets .wav file and plays it.
{
    if (fileName != null)
    {
        player = new SoundPlayer(fileName);
        player.Play();
    }
    else
    {
        MessageBox.Show("Choose .wav file first");
    }
}
public void StopSound()
{
    player?.Stop();
}

public void PlayMediaPlayer(string fileName)
```

```
        {
            if (fileName != null)
            {
                mediaPlayer.URL = fileName;
                mediaPlayer.Ctlcontrols.play();
            }
        }
```

*3.3 Odtworzyc  .wav w aplikacji i wyświetlić dane nagłówka pliku:*

```
namespace AudioCard
{
    public struct Naglowki
    {
        public byte[] riffID;
        public uint size;
        public byte[] wavID;
        public byte[] fmtID;
        public uint fmtSize;
        public ushort format;
        public ushort channels;
        public uint sampleRate;
        public uint bytePerSec;
        public ushort blockSize;
        public ushort bit;
        public byte[] dataID;
        public uint dataSize;

        public override string ToString()
        {
            //słowny zapis typu kanału
            string tempChannel;
            if (channels == 1)
                tempChannel = "mono";
            else if (channels == 2)
                tempChannel = "stereo";
            else
            {
                tempChannel = "unrecognized";
            }

            return "riffID: " + riffID + "\n" +
                   "size: " + size + "\n" +
                   "fmtSize: " + fmtSize + "\n" +
                   "dataSize: " + dataSize + "\n" +
                   "sampleRate: " + sampleRate + "\n" +
                   "channel: " + channels + " " + tempChannel + "\n";
        }
    }
}

private void button4_Click(object sender, EventArgs e)
        {
```

```
        try
        {
            label1.Text = AudioCardController.GetHeadersAndDetails();
        }
        catch (Exception)
        {
            MessageBox.Show(@"Nie można wczytać załadowanego pliku!",
                @"Uwaga",
                MessageBoxButtons.OK,
                MessageBoxIcon.Exclamation,
                MessageBoxDefaultButton.Button1);
        }
    }
```

3.5 Microfony

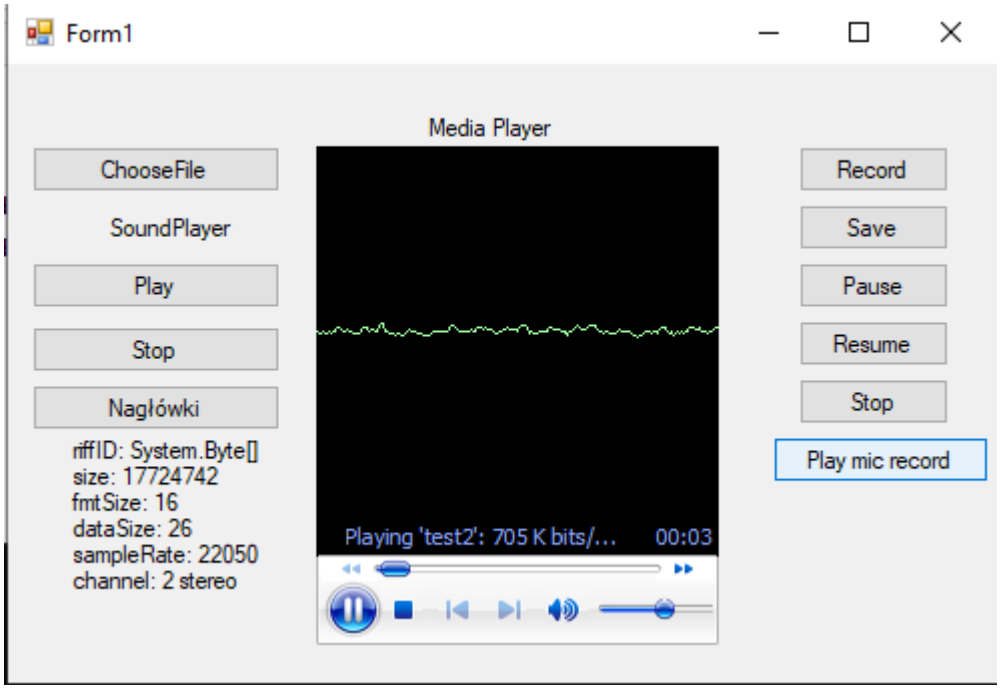
Z samej prawej strony znajduję się obluga microfonu:

```
    public void MicroPhoneOn()
    {
        micSendString("open new type waveaudio alias wavfile", null, 0, IntPtr.Zero);
        micSendString("record wavfile", null, 0, IntPtr.Zero);
    }

    public void SaveMicro()
    {
        micSendString("save wavfile test.wav", null, 0, IntPtr.Zero);
        micSendString("close wavfile", null, 0, IntPtr.Zero);
    }
    public void micPlay(string fileName)
    {
        string command = "Open wavefile " + fileName + " alias MediaFile";
        Console.WriteLine(command);
        micSendString(command,null,0,IntPtr.Zero);
        command = "Play "+fileName +" from 5";
        micSendString(command,null,0,IntPtr.Zero);
    }

    public void micPause(string filename)
    {
        string command= "pause "+filename;
        micSendString(command, null, 0, IntPtr.Zero);
    }

    public void micResume(string filename)
    {
        string command = "resume " + filename;
        micSendString(command, null, 0, IntPtr.Zero);
```

```
}

public void micStop(string filename)
{
    string command = "stop " + filename;
    micSendString(command, null, 0, IntPtr.Zero);
    command = "close " + filename;
    micSendString(command, null, 0, IntPtr.Zero);
}
```

### 4. Działanie aplikacji

Został stworzony prosty graficzny interfejs uzytkownika z dostępem do wszystkich funkcji programu jak np. odtwarzanie i zatrzymywanie plików d´zwi¸ekowych. Wygląd został przedstawiony na rysunku:




### 5. Wnioski

Podczas laboratoriów nauczyliśmy się podstawowej obsługi karty muzycznej. Opanowaliśmy w podstawowym stopniu biblioteki DirectSound, API i ActiveX która w znaczącym stopniu ułatwia wykorzystywanie podstawowych funkcji karty muzycznej.