**Nume, Prenume (Grupa) :**     Bohotineanu Florin(1406A)

**Tema proiectului:** *Muzica & Bluetooth*

Proiectul presupune redarea unor melodii prin generarea frecvențelor sunetelor muzicale pe pinii de ieșire și controlarea melodiei redate alături de tempoul acesteia prin comunicație bluetooth între un dispozitv: computer/laptop/tabletă/telefon și modulul bluetooth.

**Am folosit:** Plăcuța Arduino Uno

Modul bluetooth HC-05

În urma faptului că nu am avut o plăcuță raspberry pi la îndemână, a trebuit să improvizez.

Inițial am dorit să folosesc o interfață WEB făcută prin PHP ca în laboratorul 6 <http://embedac.ro/rrr/SI/Lab/Lab6/Laborator6.htm> însă pentru comunicația web, aveam nevoie de un server rulat pe plăcuța mea ceea ce era imposibil de făcut deoarece plăcuță nu suportă Real Time Operating System. M-am gândit atunci să fac PHP-ul să trimită o comandă prin bluetooth (comunicație serială până la urmă) dar nu există un plugin PHP care să mă ajute să fac asta. Așa că în final am decis să folosesc o interfață programată în C# care folosește SerialPort pentru comunicația bluetooth. În acest fel se programa o aplicație și pentru mobil, și pentru alte dispozitive care să comunice cu plăcuța mea.

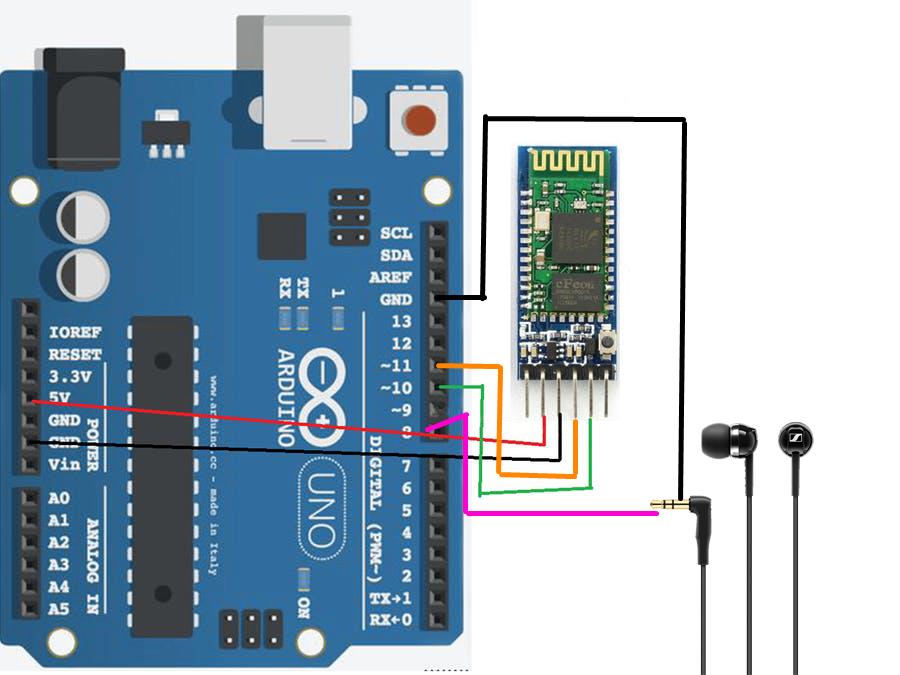
**Rezumat:** Plăcuța redă melodii și primește comenzi prin bluetooth de a schimba melodia/tempo-ul acesteia.

**Explorare documentara asupa temei,raport sintetic alternative solutie:**

Bluetooth : [https://components101.com/wireless/hc-05-bluetooth-modul](https://components101.com/wireless/hc-05-bluetooth-module)

C# bluetooth communication: <https://docs.microsoft.com/en-us/dotnet/api/system.io.ports.serialport?view=netframework-4.8>

Arduino songs player: <https://www.arduino.cc/en/Tutorial/PlayMelody>



**Cod interfațî C#:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO.Ports;

namespace BluetoothForm

{

public partial class Form1 : Form

{

SerialPort port = new SerialPort();

public Form1()

{

InitializeComponent();

selectButton.Enabled = false;

closeButton.Enabled = false;

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private static void SendSampleData()

{

// Instantiate the communications

// port with some basic settings

SerialPort port = new SerialPort( "COM1", 9600, Parity.None, 8, StopBits.One);

// Open the port for communications

port.Open();

// Write a string

port.Write("Hello World");

// Write a set of bytes

// Close the port

port.Close();

}

private void findPorts\_Click(object sender, EventArgs e)

{

Dictionary<string, string> comboSource = new Dictionary<string, string>();

foreach (string s in SerialPort.GetPortNames())

{

comboSource.Add(s,s);

}

portsBox.DataSource = new BindingSource(comboSource, null);

portsBox.DisplayMember = "Value";

portsBox.ValueMember = "Key";

selectButton.Enabled = true;

}

private void selectButton\_Click(object sender, EventArgs e)

{

string key = ((KeyValuePair<string, string>)portsBox.SelectedItem).Key;

string value = ((KeyValuePair<string, string>)portsBox.SelectedItem).Value;

//textBox1.Text = key;

// textBox2.Text = value;

port = new SerialPort(value, 9600, Parity.None, 8, StopBits.One);

//MessageBox.Show(value);

try

{

port.Open();

}

catch (Exception exp)

{

MessageBox.Show(exp.ToString());

}

//port.Write(new byte[] { 0x0A, 0xE2, 0xFF }, 0, 3);

//port.Write("2");

selectButton.Enabled = false;

closeButton.Enabled = true;

}

private void Form1\_FormClosed(object sender, FormClosedEventArgs e)

{

port.Close();

}

private void closeButton\_Click(object sender, EventArgs e)

{

port.Close();

selectButton.Enabled = true;

closeButton.Enabled = false;

}

private void beethovenButton\_Click(object sender, EventArgs e)

{

port.Write("1");

this.BackgroundImage = Image.FromFile("C:\\Users\\zeg68\\Documents\\Arduino\\SI\\Beethoven.jpg");

}

private void piratesButton\_Click(object sender, EventArgs e)

{

port.Write("2");

this.BackgroundImage = Image.FromFile("C:\\Users\\zeg68\\Documents\\Arduino\\SI\\Pirate.jpg");

}

private void crazyfrogButton\_Click(object sender, EventArgs e)

{

port.Write("3");

this.BackgroundImage = Image.FromFile("C:\\Users\\zeg68\\Documents\\Arduino\\SI\\CrazyFrog.jpg");

}

private void marioButton\_Click(object sender, EventArgs e)

{

port.Write("4");

this.BackgroundImage = Image.FromFile("C:\\Users\\zeg68\\Documents\\Arduino\\SI\\mario.jpg");

}

private void titanicButton\_Click(object sender, EventArgs e)

{

port.Write("5");

this.BackgroundImage = Image.FromFile("C:\\Users\\zeg68\\Documents\\Arduino\\SI\\titanic.jpg");

}

private void tempoplusButton\_Click(object sender, EventArgs e)

{

port.Write("+");

}

private void tempominusButton\_Click(object sender, EventArgs e)

{

port.Write("-");

}

private void tempoequalButton\_Click(object sender, EventArgs e)

{

port.Write("=");

}

}

}

**Cod Arduino UNO:**

#include"melodii.c"

#include <SoftwareSerial.h>

#define ARRAY\_SIZE(x) (sizeof((x)) / sizeof((x)[0]))

SoftwareSerial Bluetooth(10, 11);

int BluetoothData;

int melodie = 1;

void setup()

{

Serial.begin(9600);

Bluetooth.begin(9600);

Bluetooth.println("Introduceti un numar de la 1 la 5 pentru a selecta o melodie");

int\* melody = &melody\_Ode\_of\_Joy[0];

int\* durations = &noteDurations\_Ode\_of\_Joy[0];

Serial.print(ARRAY\_SIZE(melody));

Serial.print("\t");

//for (int thisNote = 0; thisNote < (sizeof(melody) / sizeof(int)) ; thisNote++)

for (int thisNote = 0; thisNote < ARRAY\_SIZE(melody); thisNote++)

{

Serial.print(\*(melody + thisNote));

Serial.print("\t");

}

Serial.print(\*(melody + 0));

Serial.print("\t");

Serial.print(\*(melody + 1));

Serial.print("\t");

Serial.print(\*(melody + 2));

Serial.print("\t");

}

int to\_int(int data)

{

if (data == '1') return 1;

if (data == '2') return 2;

if (data == '3') return 3;

if (data == '4') return 4;

if (data == '5') return 5;

}

void raiseTempo(float\* tempo)

{

if (\*tempo > 0.5)

\*tempo = \*tempo - (float)0.05;

}

void lowerTempo(float\* tempo)

{

if (\*tempo < 1.5)

\*tempo = \*tempo + (float)0.1;

}

void restoreTempo(float\* tempo)

{

\*tempo = 1;

}

void loop()

{

//int\* melody=&melody\_Ode\_of\_Joy[0];

//int\* durations=&noteDurations\_Ode\_of\_Joy[0];

if (melodie == 1)

{

// iterate over the notes of the melody:

for (int thisNote = 0; thisNote < (sizeof(melody\_Ode\_of\_Joy) / sizeof(int)); thisNote++)

{

//int thisNote;

//while(thisNote++){

// to calculate the note duration, take one second divided by the note type.

//e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.

if (Bluetooth.available())

{

BluetoothData = Bluetooth.read();

if (BluetoothData >= '1' && BluetoothData <= '5')

{

restoreTempo(&tempo\_Ode\_of\_Joy);

melodie = to\_int(BluetoothData);

break;

}

else if (BluetoothData == '+')

{

raiseTempo(&tempo\_Ode\_of\_Joy);

}

else if (BluetoothData == '-')

{

lowerTempo(&tempo\_Ode\_of\_Joy);

}

else if (BluetoothData == '=')

{

restoreTempo(&tempo\_Ode\_of\_Joy);

}

}

int noteDuration = 1000 \* tempo\_Ode\_of\_Joy / noteDurations\_Ode\_of\_Joy[thisNote];

tone(8, melody\_Ode\_of\_Joy[thisNote], noteDuration);

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

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

double pauseBetweenNotes = (double)noteDuration \* 1.30 \* tempo\_Ode\_of\_Joy;

delay(pauseBetweenNotes);

// stop the tone playing:

noTone(8);

}

}

else if (melodie == 2) //pirates

{

for (int thisNote = 0; thisNote < (sizeof(melody\_Pirates) / sizeof(int)); thisNote++)

{

if (Bluetooth.available())

{

BluetoothData = Bluetooth.read();

if (BluetoothData >= '1' && BluetoothData <= '5')

{

restoreTempo(&tempo\_Pirates);

melodie = to\_int(BluetoothData);

break;

}

else if (BluetoothData == '+')

{

raiseTempo(&tempo\_Pirates);

}

else if (BluetoothData == '-')

{

lowerTempo(&tempo\_Pirates);

}

else if (BluetoothData == '=')

{

restoreTempo(&tempo\_Pirates);

}

}

int noteDuration = 1000 \* tempo\_Pirates / noteDurations\_Pirates[thisNote];

tone(8, melody\_Pirates[thisNote], noteDuration);

double pauseBetweenNotes = (double)noteDuration \* 1.2 \* tempo\_Pirates;

delay(pauseBetweenNotes);

noTone(8);

}

}

else if (melodie == 3) //Crazy frog

{

for (int thisNote = 0; thisNote < (sizeof(melody\_Crazy\_Frog) / sizeof(int)); thisNote++)

{

if (Bluetooth.available())

{

BluetoothData = Bluetooth.read();

if (BluetoothData >= '1' && BluetoothData <= '5')

{

restoreTempo(&tempo\_Crazy\_Frog);

melodie = to\_int(BluetoothData);

break;

}

else if (BluetoothData == '+')

{

raiseTempo(&tempo\_Crazy\_Frog);

}

else if (BluetoothData == '-')

{

lowerTempo(&tempo\_Crazy\_Frog);

}

else if (BluetoothData == '=')

{

restoreTempo(&tempo\_Crazy\_Frog);

}

}

int noteDuration = 1000.0 \* tempo\_Crazy\_Frog / noteDurations\_Crazy\_Frog[thisNote];

tone(8, melody\_Crazy\_Frog[thisNote], noteDuration);

double pauseBetweenNotes = (double)noteDuration \* 1.5 \* tempo\_Crazy\_Frog;

delay(pauseBetweenNotes);

noTone(8);

}

}

else if (melodie == 4) //Mario

{

for (int thisNote = 0; thisNote < (sizeof(melody\_Mario) / sizeof(int)); thisNote++)

{

if (Bluetooth.available())

{

BluetoothData = Bluetooth.read();

if (BluetoothData >= '1' && BluetoothData <= '5')

{

restoreTempo(&tempo\_Mario);

melodie = to\_int(BluetoothData);

break;

}

else if (BluetoothData == '+')

{

raiseTempo(&tempo\_Mario);

}

else if (BluetoothData == '-')

{

lowerTempo(&tempo\_Mario);

}

else if (BluetoothData == '=')

{

restoreTempo(&tempo\_Mario);

}

}

int noteDuration = 1000 \* tempo\_Mario / noteDurations\_Mario[thisNote];

tone(8, melody\_Mario[thisNote], noteDuration);

double pauseBetweenNotes = (double)noteDuration \* 1.6 \* tempo\_Mario;

delay(pauseBetweenNotes);

noTone(8);

}

}

else if (melodie == 5) //Titanic

{

for (int thisNote = 0; thisNote < (sizeof(melody\_Titanic) / sizeof(int)); thisNote++)

{

if (Bluetooth.available())

{

BluetoothData = Bluetooth.read();

if (BluetoothData >= '1' && BluetoothData <= '5')

{

restoreTempo(&tempo\_Titanic);

melodie = to\_int(BluetoothData);

break;

}

else if (BluetoothData == '+')

{

raiseTempo(&tempo\_Titanic);

}

else if (BluetoothData == '-')

{

lowerTempo(&tempo\_Titanic);

}

else if (BluetoothData == '=')

{

restoreTempo(&tempo\_Titanic);

}

}

int noteDuration = 1000 \* tempo\_Titanic / noteDurations\_Titanic[thisNote];

tone(8, melody\_Titanic[thisNote], noteDuration);

double pauseBetweenNotes = (double)noteDuration \* 1.5 \* tempo\_Titanic;

delay(pauseBetweenNotes);

noTone(8);

}

}

}