Karaoke Player

- **Student**: Florea Daria-Mihaela
- **Grupă**: 331CC
- **Asistent**: Ene Dragoș

Introducere

Karaoke Player este un proiect care își propune să simuleze un aparat pentru Karaoke, o activitate foarte îndrăgită atât de mine, cât și de multe alte persoane.

Dispozitivul va reda versurile unei melodii, în timp ce negativul acesteia se va auzi pe fundal, asemenea atmosferei de la Karaoke.

Descriere generală

Proiectul constă într-un Karaoke Player care va reda instrumentalul unor melodii printr-un difuzor (melodii vor fi stocate pe un card microSD) și versurile acesteia pe un ecran LCD. Vor exista și câteva leduri care vor lumina în funcție de notele melodiei și un led RGB care va putea fi controlat din telefon, prin intermediul unui modul Bluetooth.

Pornirea, respectiv oprirea programului pentru Karaoke se va face prin intermediul unui buton. De asemenea, dispozitivul va avea și funcția de schimbare a melodiei, tot printr-un buton.

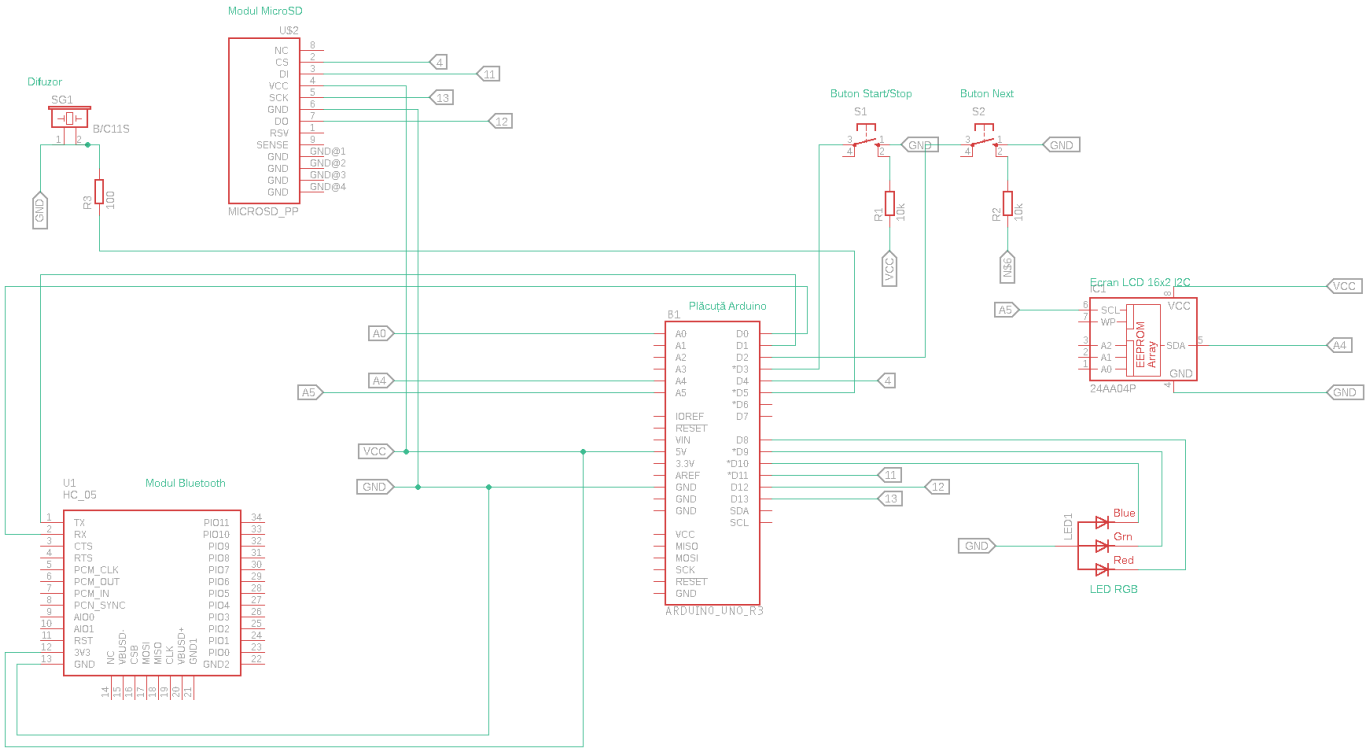
Hardware Design

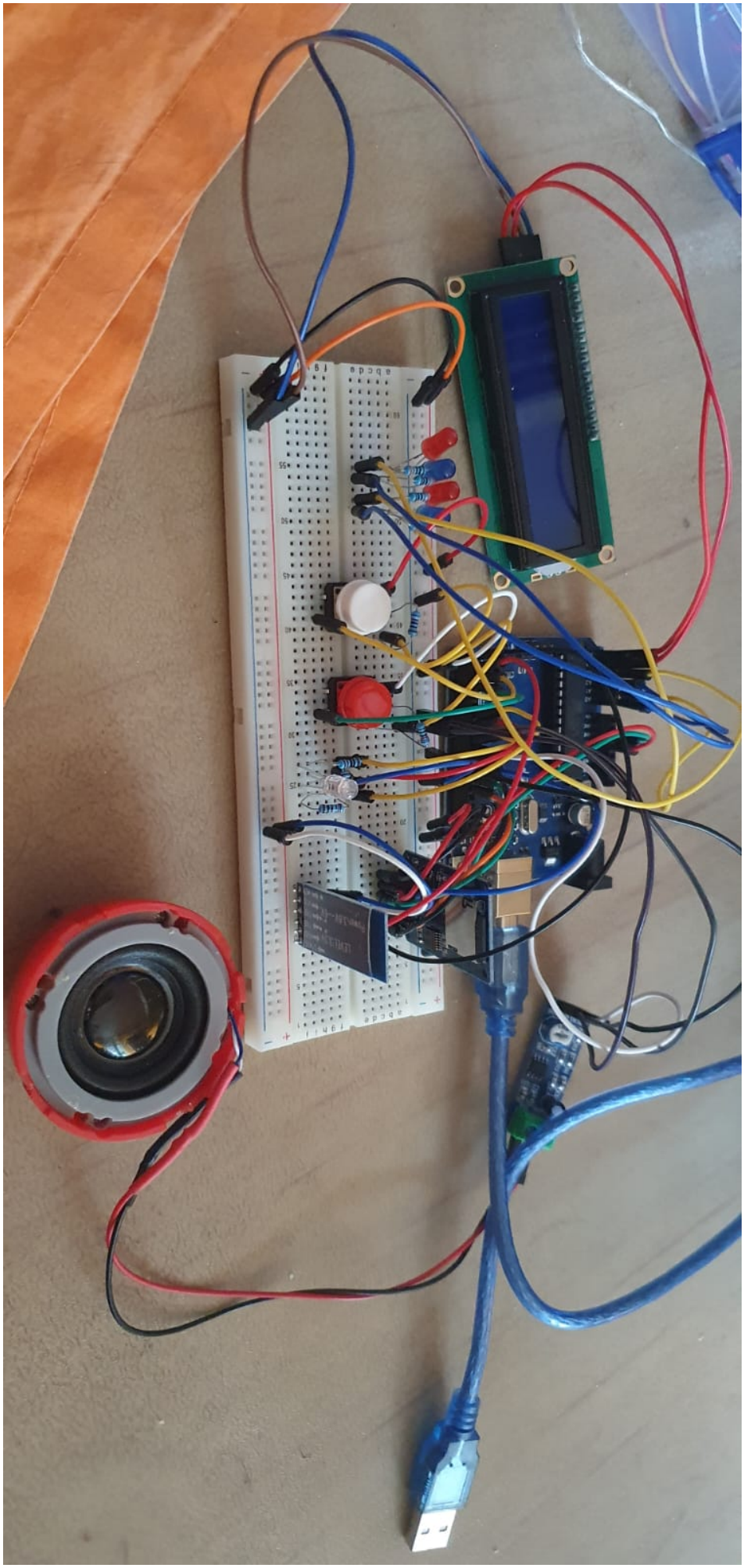
Lista pieselor folosite

- Arduino Uno
- Breadboard
- Modul Bluetooth
- Display LCD 16×2 I2C
- LED RGB
- LED-uri normale
- Difuzor
- Amplificator LM386

- Rezistențe
- 2 butoane
- Card microSD

Mai jos am atașat schema electrică a proiectului pe care am realizat-o în programul Eagle, respectiv o poză cu montajul/hard-ul proiectului.





Software Design

Biblioteci folosite

- TMRpcm.h - redarea melodiilor în format wav de pe cardul SD pe difuzor
- SD.h - citirea datelor de pe cardul microSD
- LiquidCrystal_I2C.h - afișarea mesajelor pe ecranul LCD 16×2 I2C

În funcția **loop** se verifică la fiecare moment de timp dacă s-a înregistrat o apăsare a unuia dintre butoane. Pentru a nu înregistra mai multe apăsări decât avem, am folosit funcția millis din laborator pentru a contoriza numărul de milisecunde dintre două apăsări de buton succesive. Dacă butonul de start/stop este apăsat, se pornește prima melodie, altfel melodia este oprită și se așteaptă o nouă apăsare a butonului. Butonul de next face tranziția între melodii. Pe cardul microSD avem stocate 3 melodii: Photograph, Tattoo și Someone like you.

Afișarea versurilor se face tot prin intermediul funcției millis, pe care o folosim pentru a contoriza perioada de timp dintre două versuri. Din cauza memoriei puține disponibile pe plăcuța Arduino Uno, se pot reda un număr mic de versuri pentru fiecare melodie.

Schimbarea melodiei se poate face și prin monitorul Serial care poate fi controlat pe laptop sau telefon datorită modulului Bluetooth. Se pot schimba atât melodiile, cât și alte caracteristici ale melodiei (calitate, volum etc.) sau chiar culoarea LED-ului RGB care redă lumina camerei.

Funcțiile **qtyLED0, qtyLED1** ș.a.m.d. sunt folosite pentru a reda culori în funcție de amplitudinea sunetului melodiei redate în acel moment.

Mai jos las atașat codul acestui proiect

```
#include <SD.h>
#define SD_ChipSelectPin 10
#include <TMRpcm.h>           //  Library to play wav file
#include <SPI.h>
#include <LiquidCrystal_I2C.h>

#define LED_ON    HIGH
#define LED_OFF   LOW

// RGB LEDs
const int LED_R = 8;
const int LED_G = 7;
const int LED_B = 6;

const int nextButtonPin = 2;  // the number of the pushbutton pin
int nextButtonState = 0;  // variable for reading the pushbutton status
int nextButtonTime, nextButtonLastTime = 0; // Interrupts

const int startButtonPin = 3;  // the number of the pushbutton pin
int startButtonState = 0;  // variable for reading the pushbutton status
int startButtonTime, startButtonLastTime = 0; // Interrupts
```

```
// Object for playing songs
TMRpcm tmrpcm;

// Couting periods for displaying lyrics
unsigned long current_time = 0, last_time = 0;

int current_song = 0;
int current_verse = 0;
int count = 0;

// Variables for LEDs
int waveform = 0;
int qtyLED = 0;
int timeLED = 25;

// Set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  // cli();
  tmrpcm.speakerPin = 9;

  Serial.begin(9600);

  // Buttons
  // initialize the pushbutton pin as an input:
  pinMode(nextButtonPin, INPUT);
  pinMode(startButtonPin, INPUT);

  // Song LEDs
  pinMode(17, OUTPUT);
  pinMode(16, OUTPUT);
  pinMode(15, OUTPUT);
  pinMode(14, OUTPUT);

  // RGB LEDs
  pinMode(LED_R, OUTPUT);
  pinMode(LED_G, OUTPUT);
  pinMode(LED_B, OUTPUT);

  digitalWrite(LED_R, LED_OFF);
  digitalWrite(LED_G, LED_OFF);
  digitalWrite(LED_B, LED_OFF);

  // Check SD card
  if (!SD.begin(SD_ChipSelectPin))
  {
    Serial.println("SD fail");
    return;
  }
  else
```

```
    Serial.println("SD ok");

  // Set volume speaker
  tmrpcm.volume(2);

  // initialize the LCD
  lcd.begin();

  // Turn on the blacklight and print a message.
  lcd.backlight();
  lcd.print("Karaoke Player");
}

void loop()
{
  // Photograph lyrics
  String song1[] = {
    "Loving can hurt",
    "loving can hurt",
    "sometimes"
  };

  int song1_length = 3;

  // Tattoo Lyrics
  String song2[] = {
    "I don't wanna go",
    "But baby, we both",
    "know"
  };
  int song2_length = 3;

  // Someone Like You Lyrics
  String song3[] = {
    "I heard that",
    "you're settled",
    "down"
  };
  int song3_length = 3;

  // read the state of the pushbuttons values:
  nextButtonState = digitalRead(nextButtonPin);
  nextButtonTime = millis();

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (nextButtonState == HIGH && nextButtonTime - nextButtonLastTime > 200)
{
    // Next song
    nextButtonLastTime = nextButtonTime;

    current_song++;
```

```
    if (current_song > 3) current_song = 1;

    switch (current_song) {
      case 1: tmrpcm.play("sng1.wav"); lcd.clear(); lcd.print("Photograph");
current_verse = 0; count = 0;  break;
      case 2: tmrpcm.play("sng2.wav"); lcd.clear(); lcd.print("Tattoo");
current_verse = 0; count = 0; break;
      case 3: tmrpcm.play("sng3.wav"); lcd.clear(); lcd.print("Someone like
you"); current_verse = 0; count = 0; break;
    }
  }

  startButtonState = digitalRead(startButtonPin);
  startButtonTime = millis();

  if (startButtonState == HIGH && startButtonTime - startButtonLastTime >
200) {
    // Next song
    startButtonLastTime = startButtonTime;

    if (current_song != 0)
    {
      current_song = 0;
      tmrpcm.stopPlayback();
      lcd.clear();
      lcd.print("Play a song!");
      lcd.setCursor(0, 1);
      lcd.print("Press start");
    }
    else {
      tmrpcm.play("sng1.wav");
      lcd.clear();
      lcd.print("Photograph");
      current_verse = 0;
      count = 0;
      current_song  = 1;
    }
  }

  current_time = millis();

  // 5 seconds after last period count
  if (current_time - last_time > 5000)
  {
    last_time = current_time;
    count ++;
    if (current_song == 1 && count > 5)
    {
      if (current_verse < song1_length)
      {
        lcd.clear();
```

```
      lcd.print(song1[current_verse]);
      current_verse++;
    }
  }

  if (current_song == 2 && count > 4)
  {
    if (current_verse < song2_length)
    {
      lcd.clear();
      lcd.print(song2[current_verse]);
      current_verse++;
    }
  }

  if (current_song == 3 && count > 3)
  {
    if (current_verse < song3_length)
    {
      lcd.clear();
      lcd.print(song3[current_verse]);
      current_verse++;
    }
  }
}

waveform = OCR1A; //read TIMER1 value, represent for wave form music
waveform = waveform - 256;  // middle point of volume(2) is 256
qtyLED = waveform / 16; //q'ty LED need to be

if (qtyLED <= 0)
{
  qtyLED0();
  delay(timeLED);
}

if (qtyLED == 1) {
  qtyLED0();
  delay(timeLED);
  qtyLED1();
  delay(timeLED);
}

if (qtyLED == 2) {
  qtyLED0();
  delay(timeLED);
  qtyLED1();
  delay(timeLED);
  qtyLED2();
  delay(timeLED);
}
```

```
  if (qtyLED == 3) {
    qtyLED0();
    delay(timeLED);
    qtyLED1();
    delay(timeLED);
    qtyLED2();
    delay(timeLED);
    qtyLED3();
    delay(timeLED);
  }

  if (qtyLED == 4) {
    qtyLED0();
    delay(timeLED);
    qtyLED1();
    delay(timeLED);
    qtyLED2();
    delay(timeLED);
    qtyLED3();
    delay(timeLED);
    qtyLED4();
    delay(timeLED);
  }

  if (Serial.available()) {
    switch (Serial.read()) {
      case 'd': tmrpcm.play("sng1.wav"); lcd.clear();
lcd.print("Photograph"); current_song = 1; current_verse = 0; count = 0;
break;
      case 'b': tmrpcm.play("sng2.wav"); lcd.clear(); lcd.print("Tattoo");
current_song = 2; current_verse = 0; count = 0; break;
      case 'r': tmrpcm.play("sng3.wav"); lcd.clear(); lcd.print("Someone
like you"); current_song = 3; current_verse = 0; count = 0; break;
      case 'p': tmrpcm.pause(); break;
      case '?': if (tmrpcm.isPlaying()) {
          Serial.println("A wav file is being played");
        } break;

      case 'S': tmrpcm.stopPlayback(); break;
      case '=': tmrpcm.volume(1); break;
      case '-': tmrpcm.volume(0); break;
      case '0': tmrpcm.quality(0); break;
      case '1': digitalWrite(LED_R, LED_ON); digitalWrite(LED_G, LED_OFF);
digitalWrite(LED_B, LED_OFF); break; // red
      case '2': digitalWrite(LED_R, LED_OFF); digitalWrite(LED_G, LED_ON);
digitalWrite(LED_B, LED_OFF); break; // green
      case '3': digitalWrite(LED_R, LED_ON); digitalWrite(LED_G, LED_OFF);
digitalWrite(LED_B, LED_ON); break; // purple
      case '4': digitalWrite(LED_R, LED_OFF); digitalWrite(LED_G, LED_OFF);
digitalWrite(LED_B, LED_ON); break; // blue
      default: break;
```

```
    }
  }
}

void qtyLED1() {
  digitalWrite(17, 0);
  digitalWrite(16, 0);
  digitalWrite(15, 0);
  digitalWrite(14, 1);
}

void qtyLED2() {
  digitalWrite(17, 0);
  digitalWrite(16, 0);
  digitalWrite(15, 1);
  digitalWrite(14, 1);
}

void qtyLED3() {
  digitalWrite(17, 0);
  digitalWrite(16, 1);
  digitalWrite(15, 1);
  digitalWrite(14, 1);
}

void qtyLED4() {
  digitalWrite(17, 1);
  digitalWrite(16, 1);
  digitalWrite(15, 1);
  digitalWrite(14, 1);
}

void qtyLED0() {
  digitalWrite(17, 0);
  digitalWrite(16, 0);
  digitalWrite(15, 0);
  digitalWrite(14, 0);
}
```

Rezultate Obţinute

Am obținut un proiect funcțional care are capacitatea de a reda 3 melodii prin intermediul celor 2 butoane (start/stop și next), care aprinde/stinge 4 LED-uri în funcție de amplitudinea sunetului la un moment dat, care afișează melodia/versurile melodiei pe un ecran LCD 16×2 și care aprinde un LED RGB în funcție de comenzile primite de pe telefon, datorită modulului Bluetooth.

Concluzii

Am obținut în urma acestui proiect un Karaoke player funcțional, care conține și alte funcționalități, pe lângă cele principale. Este important de menționat că încă se pot face modificări semnificative la proiect, în special pe partea de afișare a versurilor melodiilor.

Bibliografie/Resurse

Resurse Software

- Laboratoare PM
- LED-uri în funcție de amplitudinea sunetului - https://www.hackster.io/whitebank/arduino-music-player-with-leds-tutorial-cc6114
- Redare melodii wav - https://www.instructables.com/Playing-Wave-file-using-arduino/

Resurse Hardware

- Conectare difuzor la Arduino UNO - https://www.hackster.io/whitebank/arduino-music-player-with-leds-tutorial-cc6114
- Laboratoare PM

Export to PDF