Lights Musical

Autor:
Orzață Miruna-Narcisa

Introducere

Planuiesc sa imi construiesc un waw Player cu ajutorul caruia sa redau melodii de pe un card SD. Voi putea schimba melodiile prin intermediul butoanelor, iar LCD-ul si led-ul RGB vor oferi o interfata prietenoasa utilizatorului, afisandu-se astfel mai multe informatii despre melodia curenta impreuna cu o animatie de culori.

M-am gandit la acest proiect din dorinta de a realiza un dispozitiv interesant, ce cuprinde atat culoare, cat si note muzicale. Cu alte cuvinte, mi-am dorit sa construiesc ceva distractiv si palpitant cu ajutorul conceptelor studiate la laborator.

De asemenea, intentionez sa fac o comparatie intre ceea ce reusesc sa realizez de una singura si ceea ce exista deja in industrie, la performante mult mai mari.

Descriere generală

Diagrama bloc:

Hardware Design

Pentru realizarea proiectului voi avea nevoie de:

Componente	Numar
Arduino Uno	1
Difuzor	1
Modul card SD	1
Card SD	1
Led RGB	1

Rezistente 1k	4
Breadboard	1
Butoane	2
Ecran LCD	1
Potentiometru	1
Fire tata-tata	30
Condensator	1

Schema cablaj:

Software Design

Voi folosi drept mediu de dezvoltare **Arduino IDE**.

Bibliotecile folosite:
→ #include <SD.h>
→ #include <LiquidCrystal_I2C.h>
→ #include <SPI.h>

COD

```
#include "SD.h"
#include "SPI.h"
```

```
#define Rpin1 10
#define Gpin1 9
#define Bpin1 8
#define Rpin2 14
#define Gpin2 15
#define Bpin2 16
#define Rpin3 17
#define Gpin3 18
#define Bpin3 19
```

```
uint8_t signal_values[] = {250, 240, 230, 220, 210, 200, 190, 180, 170, 160,
150, 140, 130,
                           120, 110, 100, 90, 80, 70, 60, 50, 40, 30,
20, 10};
File myFile;
```

```
// change this to make the song slower or faster
uint16_t tempo = 110;
```

```
uint8_t buzzer = 7;
```

```
short int melody[120];
uint8_t len = 0;
```

```
uint8_t notes;
volatile uint8_t song = 0;
volatile uint8_t lights_dance = 1;
```

```
void RGBColor(uint8_t red, uint8_t green, uint8_t blue) {
```

```
  analogWrite(Rpin1, red);
  analogWrite(Gpin1, green);
  analogWrite(Bpin1, blue);
```

```
  analogWrite(Rpin2, red);
  analogWrite(Gpin2, green);
  analogWrite(Bpin2, blue);
```

```
  analogWrite(Rpin3, red);
  analogWrite(Gpin3, green);
  analogWrite(Bpin3, blue);
}
```

```
void lights(uint8_t signal) {
```

```
  if (signal >= signal_values[0]) {
    // blue
    RGBColor(0, 0, 255);
  } else if (signal >= signal_values[1]) {
    // Azure
    RGBColor(0, 255, 255);
  } else if (signal >= signal_values[2]) {
    // Cyan
    RGBColor(0, 127, 255);
  } else if (signal >= signal_values[3]) {
    // Aqua marine
    RGBColor(0, 255, 127);
  } else if (signal >= signal_values[4]) {
    // Green
    RGBColor(0, 255, 0);
  } else if (signal >= signal_values[5]) {
    // Yellow
    RGBColor(255, 255, 0);
  } else if (signal >= signal_values[6]) {
    // Magenta
    RGBColor(255, 0, 255);
  } else if (signal >= signal_values[7]) {
    // Rose
    RGBColor(255, 0, 127);
```

```
  } else if (signal >= signal_values[8]) {
    // Orange
    RGBColor(255, 127, 0);
  } else if (signal >= signal_values[9]) {
    // Red
    RGBColor(255, 0, 0);
  } else if (signal >= signal_values[9]) {
    // Purple
    RGBColor(128, 0, 128);
  } else if (signal >= signal_values[10]) {
    // Gold
    RGBColor(255, 215, 0);
  } else if (signal >= signal_values[11]) {
    // Spring green
    RGBColor(0, 250, 154);
  } else if (signal >= signal_values[12]) {
    // Turquoise
    RGBColor(64, 224, 208);
  } else if (signal >= signal_values[13]) {
    // Indigo
    RGBColor(75, 0, 130);
  } else if (signal >= signal_values[14]) {
    // Pink
    RGBColor(255, 192, 203);
  } else if (signal >= signal_values[15]) {
    // Lavender
    RGBColor(230, 230, 250);
  } else if (signal >= signal_values[16]) {
    // Chocolate
    RGBColor(210, 105, 30);
  } else if (signal >= signal_values[17]) {
    // Sea green
    RGBColor(46, 139, 87);
  } else if (signal >= signal_values[18]) {
    // Olive
    RGBColor(128, 128, 0);
  } else if (signal >= signal_values[19]) {
    // Maroon
    RGBColor(128, 0, 0);
  } else if (signal >= signal_values[20]) {
    // Midnight blue
    RGBColor(25, 25, 112);
  } else if (signal >= signal_values[21]) {
    // Medium violet red
    RGBColor(199, 21, 133);
  } else if (signal >= signal_values[22]) {
    // Orchid
    RGBColor(218, 112, 214);
  } else if (signal >= signal_values[23]) {
    // Steel blue
    RGBColor(70, 130, 180);
```

```
  } else if (signal >= signal_values[24]) {
    // Coral
    RGBColor(255, 127, 80);
  } else {
    // White
    RGBColor(255,255,255);
  }
}
```

```
void button1Pressed()
{
  song++;
  song = song % 5;
}
```

```
void button2Pressed()
{
  lights_dance = !lights_dance;
}
```

```
void readFile(char *file_name) {
  Serial.print("Init SD card...");
```

```
  if (!SD.begin(4)) {
    Serial.println("init failed!");
    while (1);
  }
  Serial.println("init done.");

  // open the file for reading:
  myFile = SD.open(file_name);
  if (myFile) {
    Serial.println(file_name);
```

```
    uint8_t i = 0;
    // read from the file until there's nothing else in it:
    while (myFile.available()) {
      short int note = myFile.parseInt();
      melody[i] = note;
      if (i > 120)
        break;
      i++;
    }
    len = i;
    notes = len / 2;
    // close the file:
    myFile.close();
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening file");
```

```
  }
}
```

```
void sing(uint8_t song_id) {
  // this calculates the duration of a whole note in ms
  int wholenote = (60000 * 4) / tempo;
```

```
  int divider = 0, noteDuration = 0;
  // iterate over the notes of the melody.
  // the array is twice the number of notes (notes + durations)
  for (uint8_t thisNote = 0; thisNote < notes * 2; thisNote = thisNote + 2)
{
    if (song_id != song) {
      return;
    }
```

```
    if (lights_dance == 1) {
      lights(abs(melody[thisNote]) % 255);
    } else {
      RGBColor(46, 139, 87);
    }
```

```
    // calculates the duration of each note
    divider = melody[thisNote + 1];
    if (divider > 0) {
      // regular note, just proceed
      noteDuration = (wholenote) / divider;
    } else if (divider < 0) {
      // dotted notes are represented with negative durations!!
      noteDuration = (wholenote) / abs(divider);
      noteDuration *= 1.5; // increases the duration in half for dotted
notes
    }
```

```
    // we only play the note for 90% of the duration, leaving 10% as a pause
    tone(buzzer, melody[thisNote], noteDuration * 0.9);
```

```
    // Wait for the specief duration before playing the next note
    delay(noteDuration);
```

```
    // stop the waveform generation before the next note
    noTone(buzzer);
  }
}
```

```
void play(char *song_name, uint8_t song_id) {
  readFile(song_name);
  sing(song_id);
}
```

```
void setup()
{
  // Start with the LEDs off.
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(14, OUTPUT);
  pinMode(15, OUTPUT);
  pinMode(16, OUTPUT);
  pinMode(17, OUTPUT);
  pinMode(18, OUTPUT);
  pinMode(19, OUTPUT);
```

```
  pinMode(2, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(2), button1Pressed, FALLING);
```

```
  pinMode(3, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(3), button2Pressed, FALLING);

  // Open serial communications and wait for port to open
  Serial.begin(9600);
  while (!Serial) {
      // wait for serial port to connect
  }
}
```

```
void loop()
{
  switch(song) {
    case 0:
      play("got.txt", 0);
      break;
    case 1:
      play("furelise.txt", 1);
      break;
    case 2:
      play("starwars.txt", 2);
      break;
    case 3:
      play("takeonme.txt", 3);
      break;
    case 4:
      play("tears.txt", 4);
      break;
    default:
      break;
  }
}
```

Comentarii

Ideea initiala a proiectului a fost de a reda melodii .wav de pe un card SD si de adauga un fundal luminos pentru a armoniza culorile si notele muzicale, iar display-ul LCD trebuia sa afiseze informatii despre melodia curenta.

Din motive necunoscute, prin intermediul bibliotecii TMRpcm, nu am reusit sa redau fisierele .wav de pe cardul SD.

Din aceasta cauza, am ales sa stochez pe card mai multe fisiere .txt cu note pentru fiecare melodie in parte, ca apoi sa citesc notele dintr-un anumit fisier intr-un vector si sa le redau cu ajutorul functiei “note” din Arduino IDE.

Cu ajutorul primului buton se poate schimba melodia curenta, iar cel de-al doilea buton are rolul de a porni si de a opri animatia led-urilor. In timpul animatiei luminoase, toate cele 3 led-uri isi schimba culorile in functie de nota redata curent de difuzor.

De asemenea, am incercat sa ameliorez debouncing-ul provocat de contactele imperfecte din structura interna a butonului prin adaugarea unui condensator in serie cu fiecare buton. Nu a functionat dupa cum ma asteptam, dar, in general, apare o singura intrerupere externa atunci cand apas pe buton.

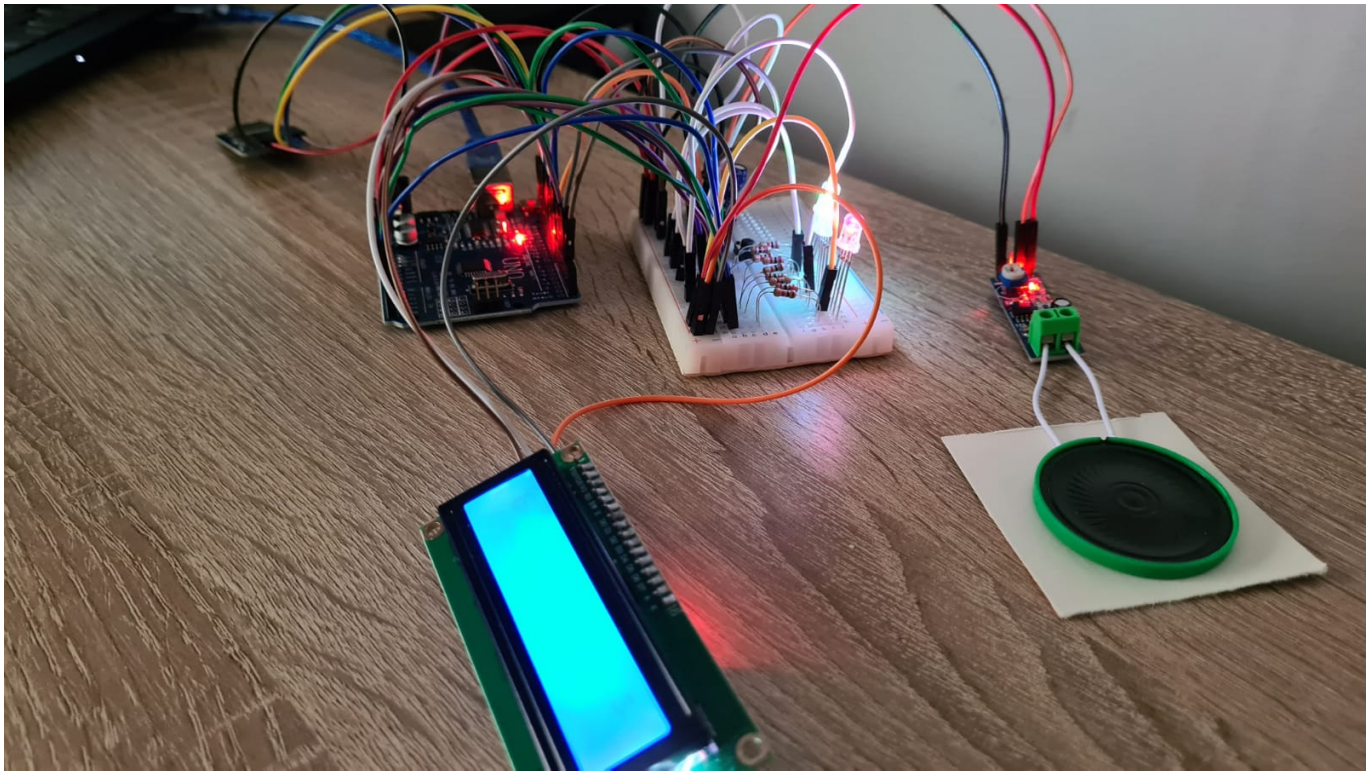
In plus, din cauza spatiului de stocare limitat prezent pe placuta Arduino, am fost nevoita sa inregistrez varianta proiectului in care nu folosesc display-ul LCD, deoarece componentele nu mai functionau cum trebuie la adaugarea bibliotecii “LiquidCrystal_I2C”. Bineinteles, am incercat sa reduc foarte mult memoria consumata in program prin micsorarea vectorului de note, dar tot am intampinat erori de performanta.

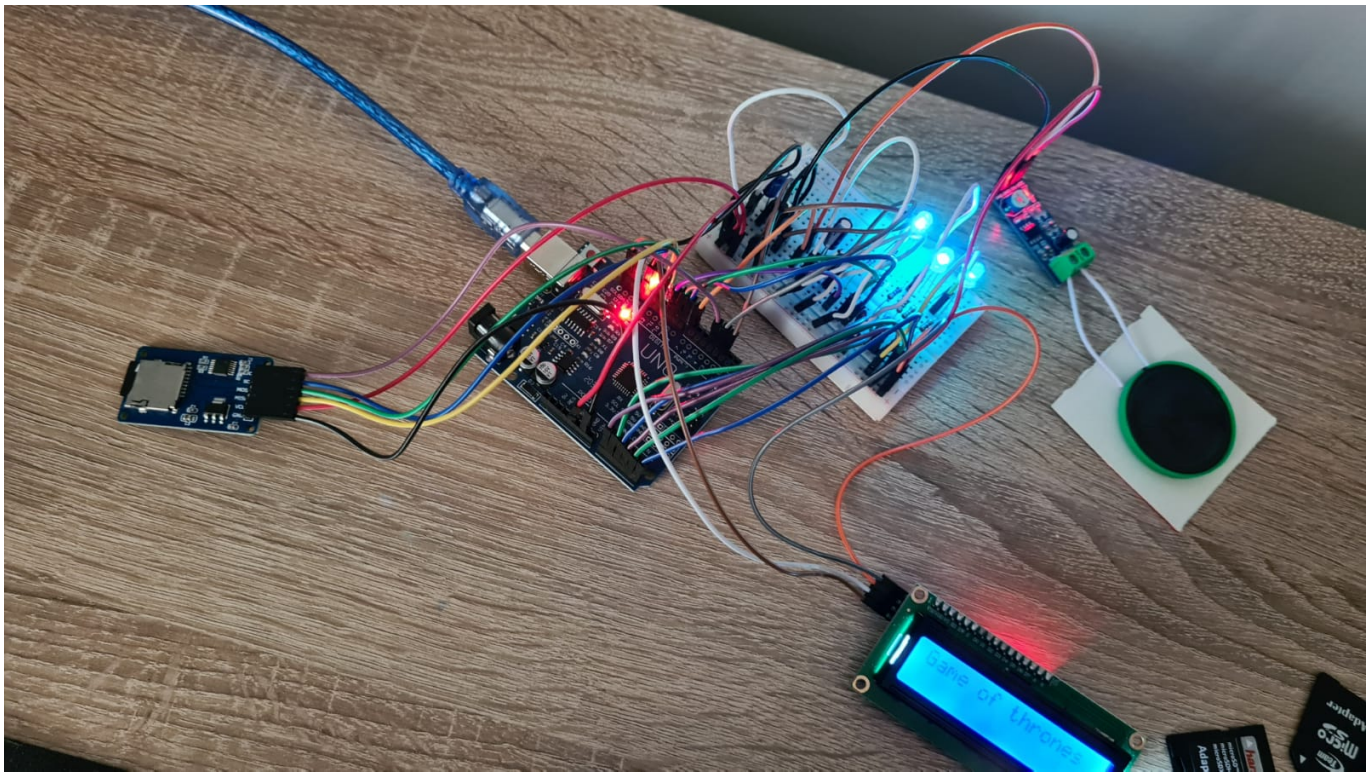
In arhiva se poate gasi si varianta codului cu biblioteca pentru LCD in caz ca se doreste implementarea proiectului pe o placuta de dezvoltare cu o capacitate de memorie mai mare. Tin sa mentionez ca aceasta versiune functioneaza, dar sunetul pare deteriorat, in sensul ca se aude ca si cum se sar anumite note sau se redau uneori alte note decat cele originale.

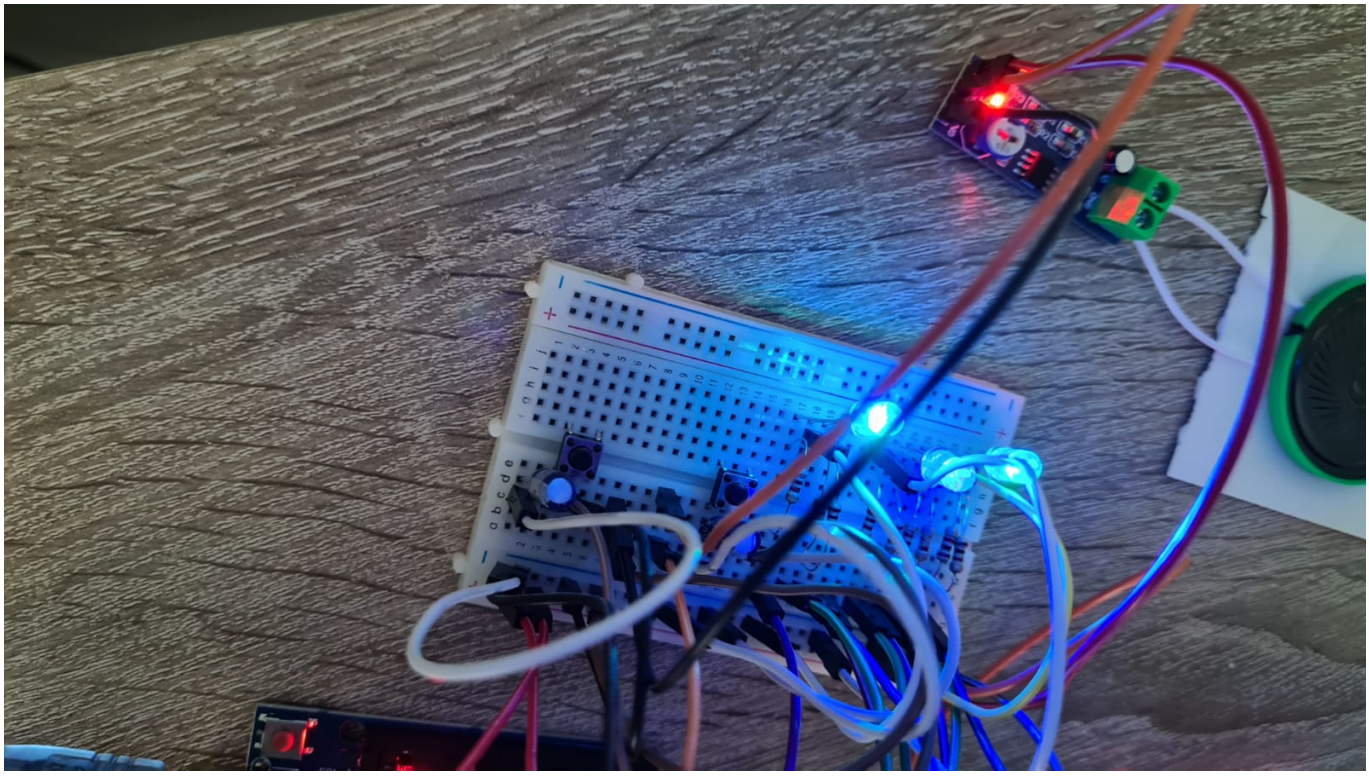
Din punct de vedere hardware, versiunea finala a proiectului a suferit cateva modificari, deoarece am avut nevoie de a economisi cativa pini pentru inca 2 led-uri RGB si a trebuit sa aleg un LCD cu interfata I2C, in loc de unul obisnuit.

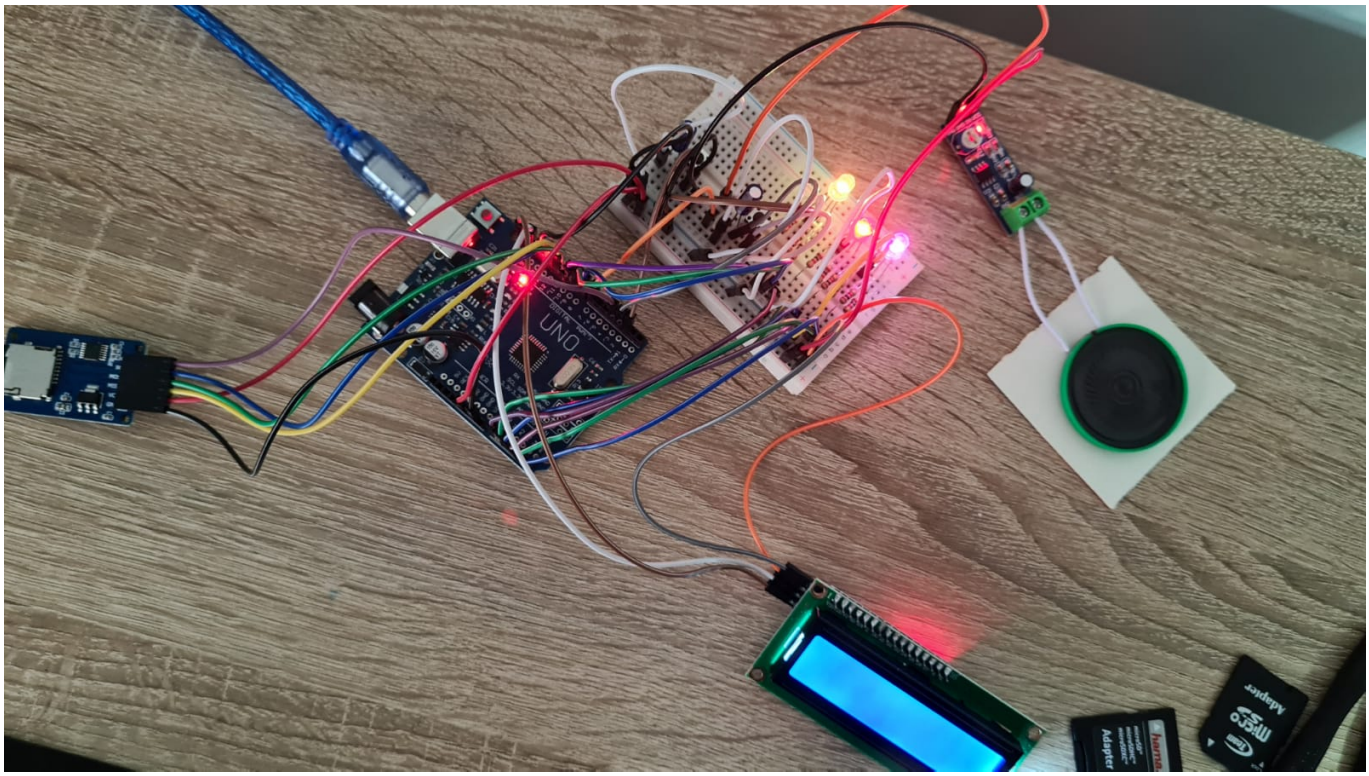
Rezultate Obținute

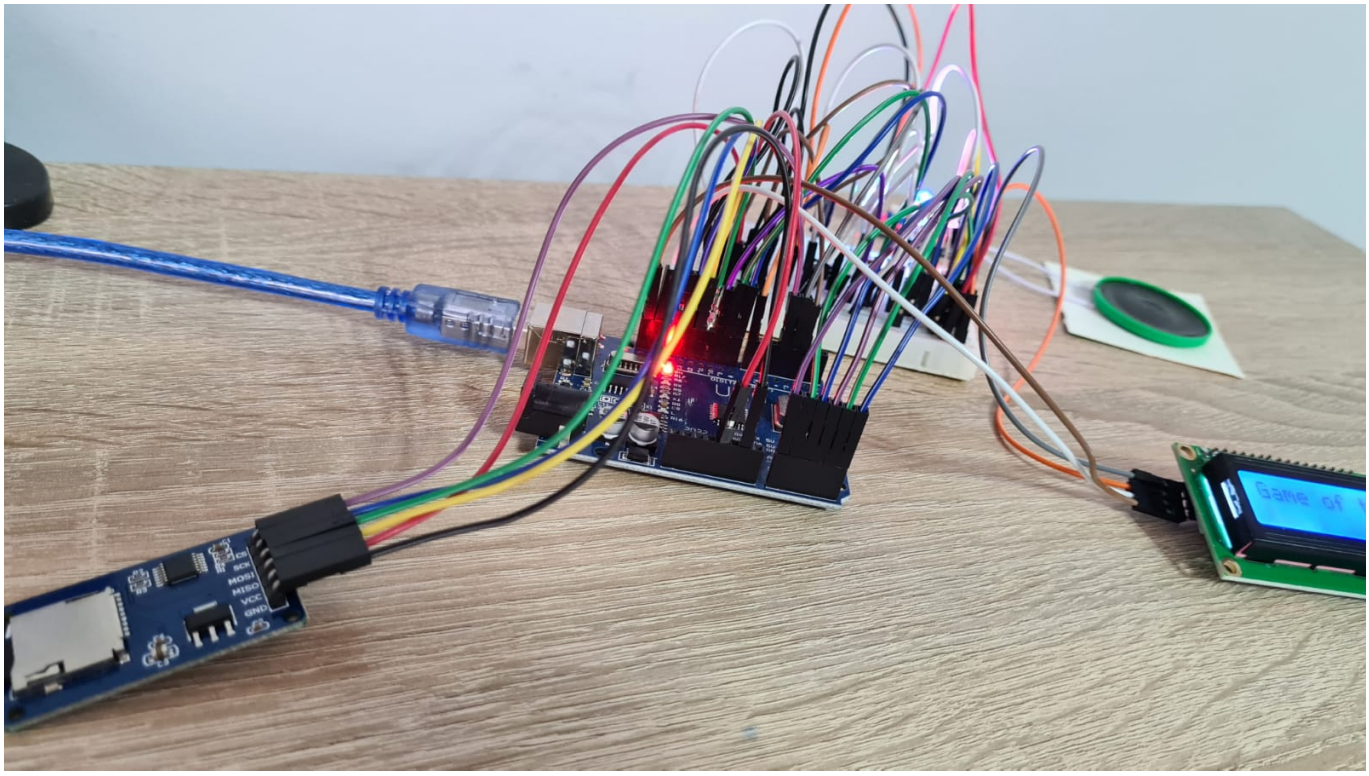
La acest link se afla un scurt videoclip demonstrativ cu intreaga functionalitate a proiectului: https://www.youtube.com/watch?v=sQiFtnPFtMM

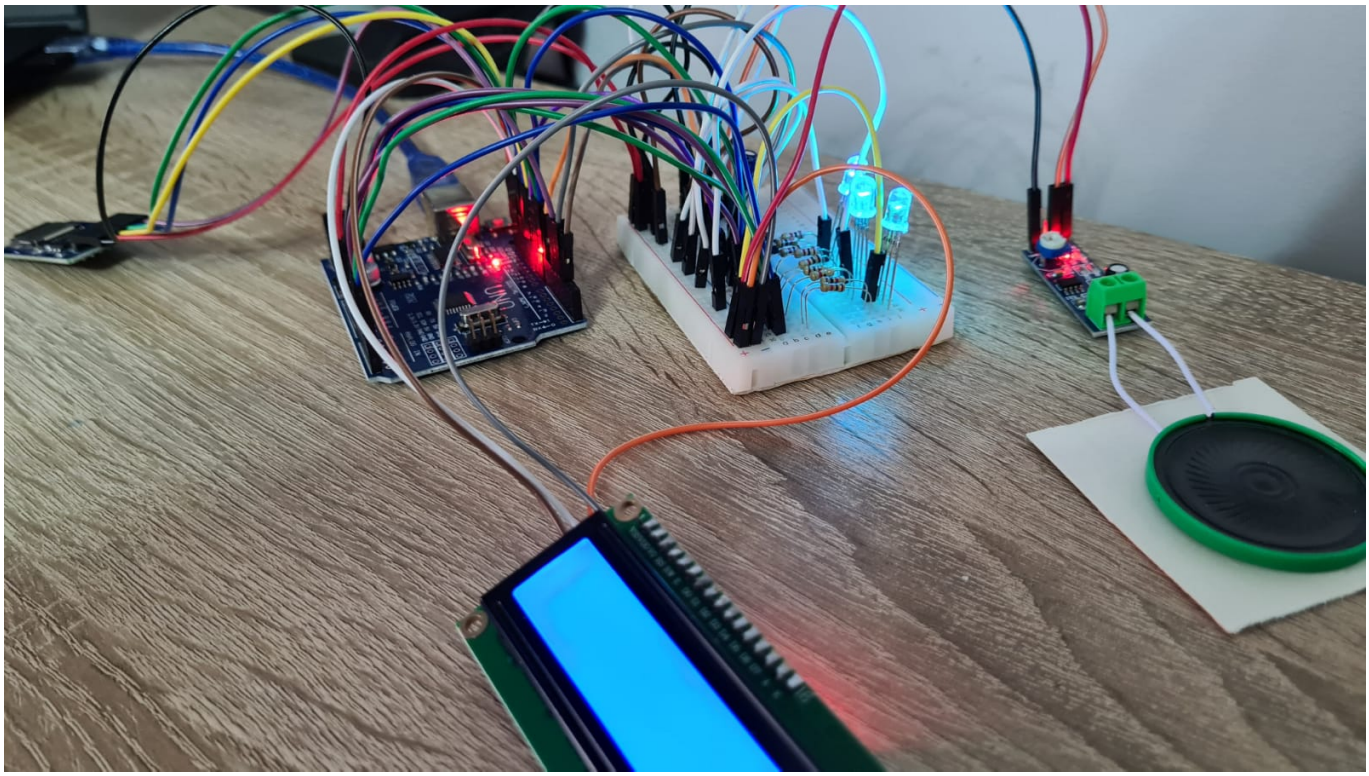







UNO





Concluzii

Cu toate ca am intampinat mai multe probleme tehnice pe parcursul implementarii proiectului, intr-un final am ajuns la un rezultat interesant. Mi-a facut placere sa experimentez anumite functionalitati, sa ma documentez din tutoriale si sa pun in aplicare notiunile studiate la laborator.

Download

Link pagina: https://ocw.cs.pub.ro/courses/pm/prj2021/abasoc/lightsmusical
Arhiva proiect: proiect_pm.zip
Pdf pagina: pagina_pm.pdf

Jurnal

27 Aprilie → alegerea temei pentru proiect
1 Mai → realizarea diagramei bloc si a schemei cablaj
3 Mai → comandarea pieselor necesare
7 Mai → cablarea fizica a componentelor
23 Mai → implementarea software
1 Iunie → finalizare proiect

Bibliografie/Resurse

Diagrama bloc → https://app.diagrams.net/
Schema cablaj → https://fritzing.org
LCD with I2C interface library → https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library
SD library → https://github.com/arduino-libraries/SD