Orga de lumini

Student: Epure Maria-Denisa
Grupa: 333CC

Introducere

Acest proiect consta intr-o orga de lumini ce va converti in mod automat semnalele audio (spre exemplu muzica) in efecte luminoase ritmice. Ideea de la care am pornit este banda LED inteligenta care isi poate sincroniza culorile si intensitatea cu muzica.

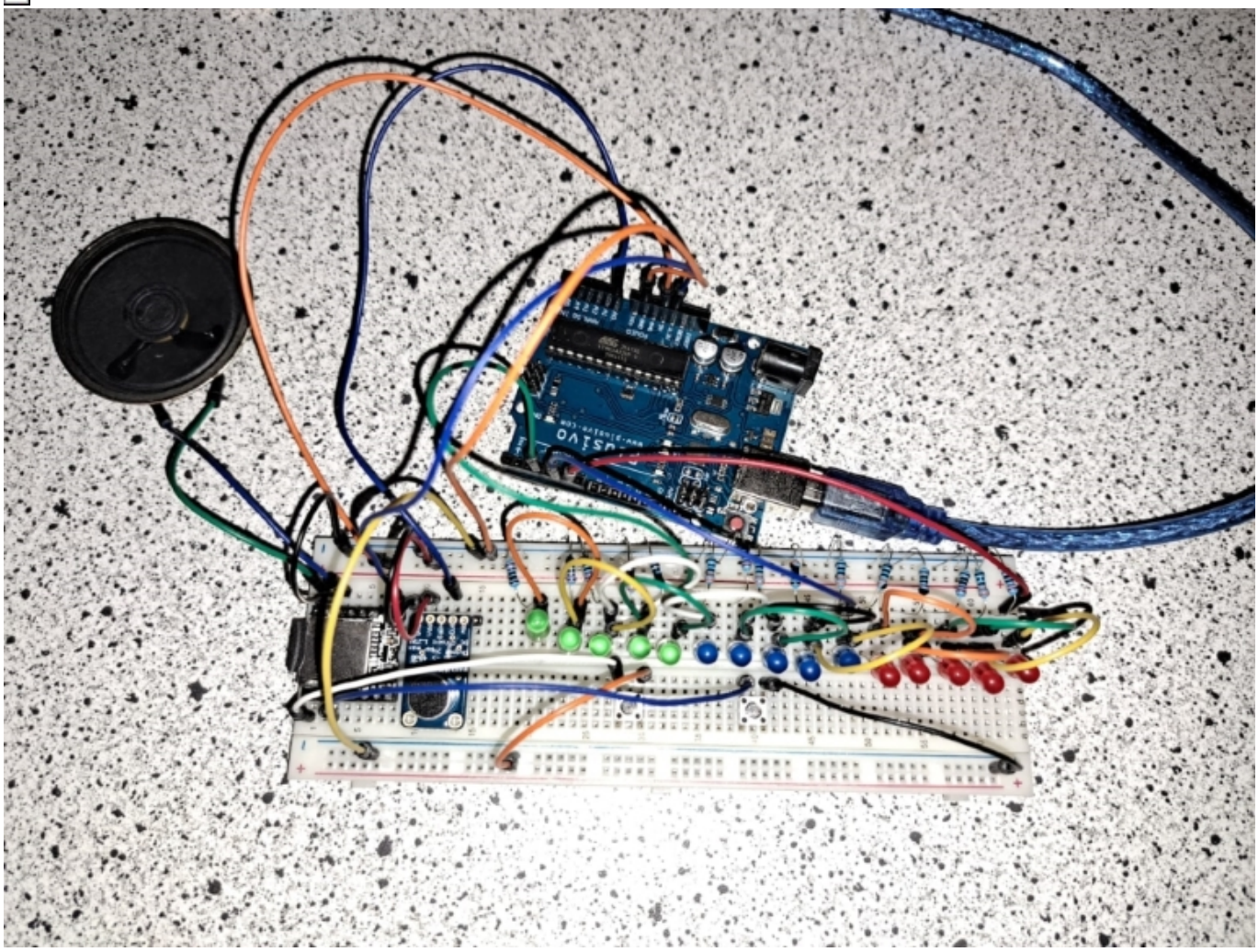
Descriere generală

Orga de lumini controleaza mai multe becuri LED-uri de trei culori, rosu, albastru si verde, pe care le aprinde prin separarea frecventelor sunetului din gamele joase, medii si inalte. Acesta se foloseste de un microphone module pentru a distinge intre aceste frecvente. Include si un speaker care reda muzica dintr-un card SD, ce ii da input microfonului. Cu ajutorul a doua butoane conectate la modulul DFPlayer se pot schimba melodiile si volumul speaker-ului.

Hardware Design

Lista de piese necesare:

- Arduino Uno;
- Breadboard;
- Speaker;
- LED-uri;
- DFPlayer Mini MP3 Player module;
- MicroSD card;
- MAX9841 Microphone module;
- Butoane
- Fire;
- Rezistente



Software Design

Mediu de dezvoltare : Arduino IDE

Librării şi surse 3rd-party :

- arduinoFFT.h

Implementare software :

Orga de lumini este compusa din 15 LED-uri, 5 rosii, 5 albastre, 5 verzi. LED-urile corespunzatoare fiecarei culori sunt conectate la cate un pin de tip PWM, si anume 3, 5, 6. Acesti pini sunt de folos pentru a controla semnalul venit de la microfon (prin pinul A0 de la care primeste input) si a-l translata corespunzator LED-urilor, rezultand in variatia intensitatii luminii.

```
#include <arduinoFFT.h>
#define sensorPin A0
#define REDPIN 6
#define GREENPIN 3
#define BLUEPIN 5
```

Microfonul va capta in continuu sunete pe care le reprezinta cu valori de la 0 la 1023. Aceste valori trebuie translatate si asignate cate unei culori, avand o anumita intensitate. Pentru aceasta este nevoie de folosirea bibliotecii “arduinoFFT”, care se va ocupa de transformarea acestor valori in frecvente. ArduinoFFT se foloseste de transformata Fourier pentru a descompune semnalul primit in sinusoide ce au frecvente si amplitudini diferite. Bazat pe un numar de valori primite ca input de la microfon (samples), se vor vor calcula frecventele cu ajutorul functiilor bibliotecii, sub forma vectoriala. Deci valorile intoarse sunt stocate in doi vectori, unul cu partea reala a frecventelor si unul cu partea imaginara.

```
#define SAMPLES 64      // 64 deoarece memoria este limitata, trebuie sa fie
putere a lui 2
double vReal[SAMPLES];
double vImag[SAMPLES];
```

Pentru a putea imparti valorile vectorilor cu o culoare, se va alege un interval de valori pentru fiecare culoare. In final, numarul de valori de frecvente calculate este jumatate din numarul de samples primit ca input, iar fara prima valoare (care este suma celorlalte), ne raman 31 de valori pe care le putem imparti intre culori.

```
#define Rrange 10                // frecvente mici
#define Grange 11                // frecvente medii
#define Brange 10                // frecvente mari
```

Avem nevoie de sampling frequency, care trebuie sa fie cel putin dublul frecventei pe care o primim ca input.

```
#define SAMPLING_FREQUENCY 1000
```

In functia de “setup” calculez cat dureaza primirea unui sampling si setez pinii de input, respectiv output.

```
void setup() {
                Serial.begin(115200);
                //pentru calculul a cat dureaza primirea ca input a unui
sampling
                sampling_period_us = round(1000000 * (1.0 /
SAMPLING_FREQUENCY));
                pinMode(sensorPin, INPUT);
                pinMode(REDPIN, OUTPUT);
                pinMode(GREENPIN, OUTPUT);
                pinMode(BLUEPIN, OUTPUT);
}
```

In functia “loop”:

- cat timp se pot citi samples, asignam valori primite de la microfon in vectorul de numere reale, iar pe cel de numere imaginare le initializam cu 0.

```
for (int i = 0; i < SAMPLES; i++) {
           microseconds = micros();  //overflow dupa 70 de minute
```

```
            vReal[i] = analogRead(sensorPin);
            vImag[i] = 0;
            // pentru citirea urmatorului sampling
            while (micros() < (microseconds + sampling_period_us)) {
            }
        }
```

Dupa ce am primit valorile de input urmeaza sa fie transformate in frecvente. Se folosesc functia “Windowing” pentru aplicarea functiei Hamming, pe valori, dupa care “Compute” si “ComplexToMagnitude” pentru a afla frecventele cu magnitudinea cea mai mare.

```
FFT.Windowing(vReal, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
FFT.Compute(vReal, vImag, SAMPLES, FFT_FORWARD);
FFT.ComplexToMagnitude(vReal, vImag, SAMPLES);
```

In functie de range-ul in care se afla in vector, valorile sunt impartite pe culori si calculeaza modulul. Se calculeaza valoarea frecventei si se aduna toate valorile din range-ul culorii.

```
 for(int i=1; i < SAMPLES/2; i++)
 {
  double a = pow(vReal[i], 2);
  double b = pow(vImag[i], 2);   // fiecare frecventa este asignata unei
culori
  if(i <= Rrange)
    Rmodule += sqrt(a + b);
  else if(i >= (Rrange+1) && i <= (Rrange+Grange))
    Gmodule += sqrt(a + b);
  else Bmodule += sqrt(a + b);
 }
```

Rezulta media in functie de suma calculata mai sus:

```
Rmodule /= Rrange;
Gmodule /= Grange;
Bmodule /= Brange;
```

In final, prin niste teste cu muzica am observat ca valorile calculate pentru verde si albastru erau foarte mici, asa ca prin inmultiri si impartiri am incercat sa le aduc la o medie normala.

```
Rvalue = Rmodule / 3;
Gvalue = 20 * Gmodule;
Bvalue = 20 * Bmodule;
```

Daca frecventa este prea mica, LED-urile nu se aprind:

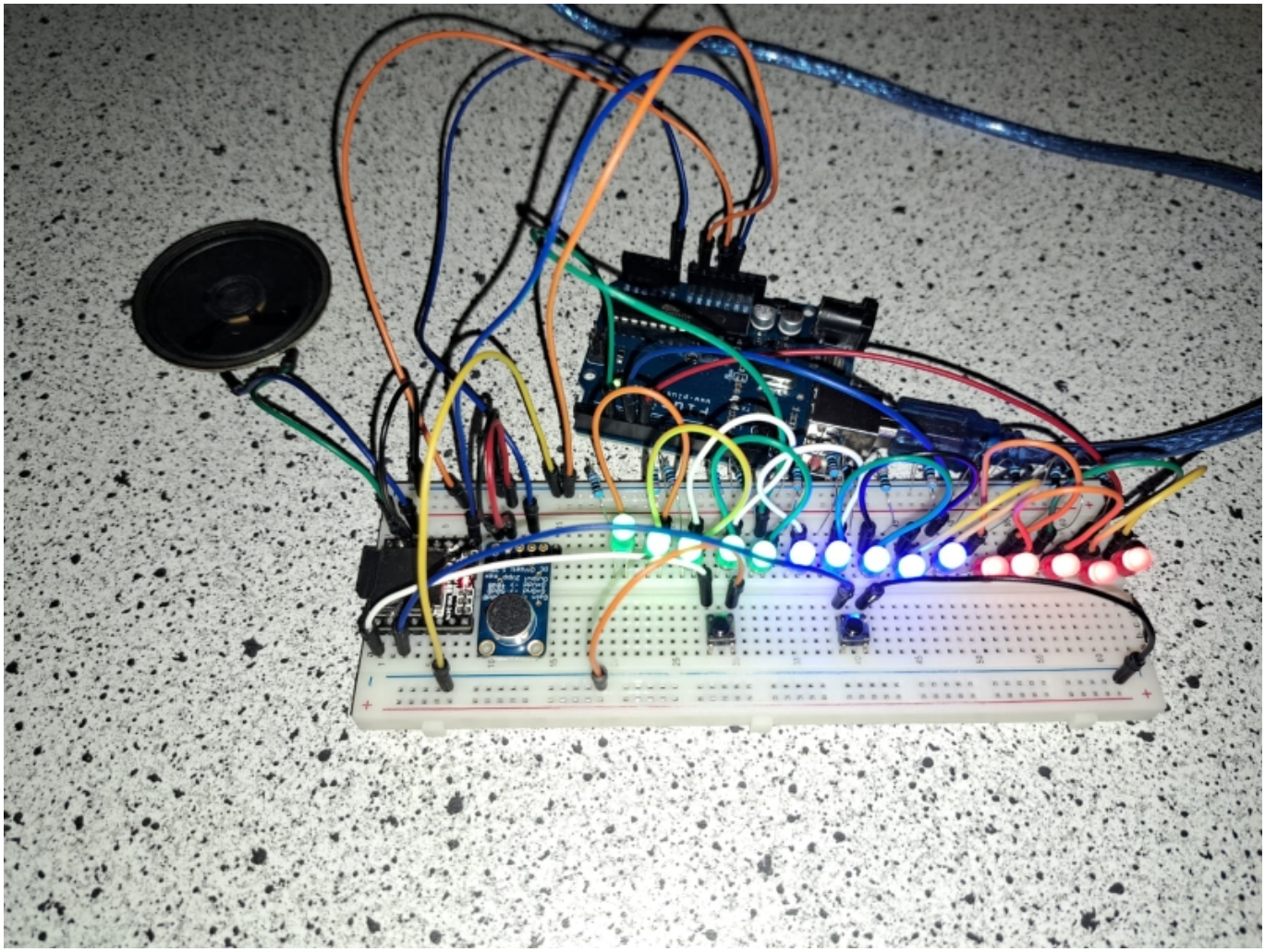
```
if (Rvalue < 115)
  Rvalue = 0;
if (Gvalue < 100)
  Gvalue = 0;
if (Bvalue < 120)
```

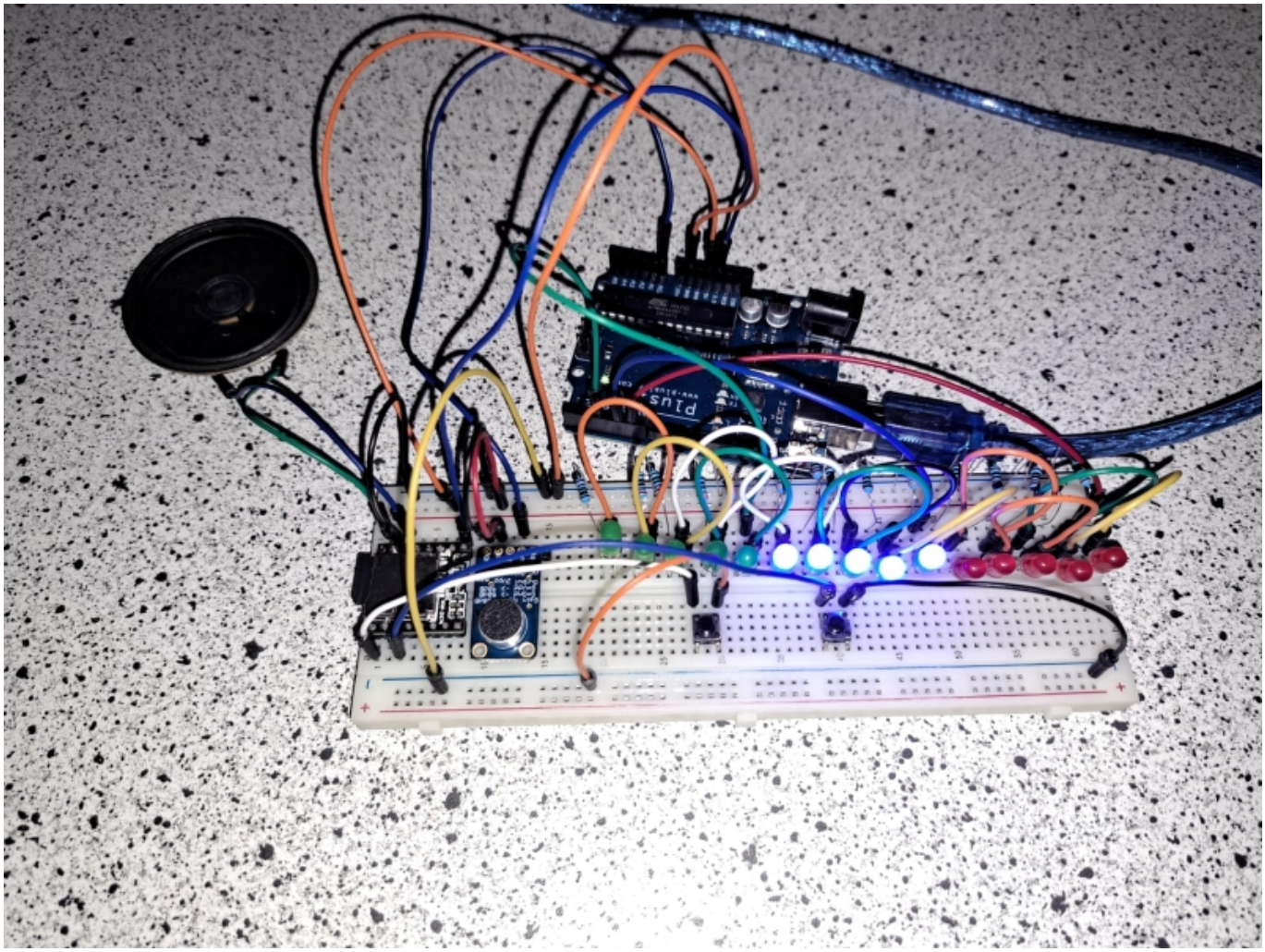
```
  Bvalue = 0;
```

```
analogWrite(REDPIN, Rvalue);
analogWrite(GREENPIN, Gvalue);
analogWrite(BLUEPIN, Bvalue);
```

Pentru redarea cantecelor si folosirea butoanelor nu a fost nevoie sa realizez o implementare din moment ce modulul DFPlayer vine cu aceste functionalitati integrate.

Rezultate Obţinute





Concluzii

In concluzie, a fost o experienta interesanta pentru mine, a fost prima oara cand am luat contact cu lumea hardware si cu toate ca nu este un proiect foarte complex, sunt mandra de rezultatele obtinute si pot spune ca am realizat singura ceva ce nu m-as fi gandit sa incerc in mod normal. Am avut dificultati pe parcurs, fie cu partea hardware, fie cu cea de cod. Initial am incercat sa folosesc un simplu modul de card SD, insa nu era mereu citit cardul SD si din acest motiv am renuntat la el si am adaugat un modul DFPlayer MP3 care mi-a usurat munca si mi-a permis sa adaug si butoane. In final, ma bucur ca am implementat ceva interesant ce imbina atat muzica, cat si jocuri de lumini.

Download

- Surse: orga_de_lumini.zip

Bibliografie/Resurse

- Link-uri:
- https://www.atomic14.com/2020/09/12/esp32-audio-input.html
- https://electropeak.com/learn/interfacing-max9814-electret-microphone-amplifier-module-with-arduino/
- https://www.tutorialspoint.com/fast-fourier-transform-fft-on-arduino
- https://wiki.dfrobot.com/DFPlayer_Mini_SKU_DFR0299