# Sound Keyboard

Autor: Iancu Alexandru-Gabriel

Grupa: 333CB

## Introducere

Aplicatia este un sound keyboard ce primeste la intrare, prin intermediul unui keypad, detalii relevante unei note muzicale: nota - A, B, ..., G; octava - 0, ..., 8; timbrul - jos, ridicat; durata - 1, 2, 4, 8, 16. Urmand ca pe baza intrarii sa produca sunetul corespondent, prin intermediul unui buzzer piezo. De asemenea, nota muzicala este afisata pe un ecran lcd (Ex: A1#).

Motivul pentru care am ales acest proiect este pentru ca ascult muzica destul de des si parea un proiect interesant si placut de realizat. Ideea de la care am pornit a fost sa fac un keyboard care poate reproduce orice nota. Am gasit o lista de define-uri cu note muzicale pentru a le folosi ca intrare pentru functia tone(). Pentru ca se diferentiau prin nota, octava si timbru m-am decis sa fac cumva ca acestea sa poata fi introduse de utilizator. Cum deja ii ceream 3 input-uri pentru o singura nota, am zis ca as putea la fel de bine sa-i cer si durata notei ca sa fie pachetul complet.

## Descriere generală

Aplicatia este formata din 2 placi Arduino care comunica prin I2C. Master-ul se ocupa de Input, acesta fiind conectat la un Keypad 4×4 prin care utilizatorul poate introduce nota. Slave-ul se ocupa de Output si acesta este conectat la un LCD care va afisa ce tasteaza, nota finala si erori si la un buzzer care va reda nota introdusa.

Arhitectura aplicatiei este caracterizata de primirea unei intrari urmand ca aceasta sa fie afisata in 2 forme diferite. Utilizatorul tasteaza folosind keypad-ul nota dorita si aceasta este afisata pe LCD si de asemenea este redata prin intermediul unui buzzer. O nota muzicala are 4 caracteristici si anume litera notei (A-G in conventia engleza), octava (sau nivelul ce merge de la 0 la 8 pentru ca pe acestea le are buzzer-ul), timbrul (jos sau ridicat) si durata sa (1 sec, 1/2 sec, .. , 1/16 sec). Pentru construirea notei este astfel nevoie de 4 apasari pe keypad:

- litera notei ⇒ se tasteaza butoanele 1-7 pentru notele A-G, corespondenta este liniara
- octava notei ⇒ corespondenta este 1:1 si anume la tastarea cifrei 0-8 se considera octava 0-8
- frecventa ⇒ '*' pe keypad este frecventa joasa (corespunde timbrului 'b'), '#' este frecventa mai ridicata (corespunde timbrului '#')
- durata ⇒ apasand butoanele 1, 2, 4, 8 se va genera sunetul pe durata 1, 1/2, 1/4, 1/8; apasand tasta

'A' se va genera durata 1/16

In cazul in care pe parcursul celor 4 tastari, o tastare este invalida se afiseaza un mesaj de eroare: “INVALID NOTE”, “INVALID NUMBER”, “INVALID PITCH”, “INVALID DURATION” timp de 1000ms, in functie de a cata tastare e. Daca toate tastarile sunt ok si anume individual se incadreaza in multimea valida aferenta, dar nota finala nu exista (Ex: A0b sau A0#), atunci se afiseaza nota respectiva alaturi de mesajul “INVALID NOTE” pe randul urmator. Daca utilizatorul a apasat deja niste taste, dar nu erau cele pe care le dorea, atunci poate sa faca RESET apasand pe tasta 'D'.

Pentru fiecare tastare corecta de keypad se afiseaza corespondentele discutate mai sus timp de 1000ms. Daca nota finala este corecta si exista, atunci se afiseaza la LCD cat timp este redata de buzzer. Durata de redare si afisare este n * 1000ms, ex. pentru n = 1/4 va fi 250 ms, asa ca pentru tasta x apelata, durata va fi 1000/x ms.

# Hardware Design

## Tinkercad

Varianta facuta in Tinkercad si vizibila mai sus e umpic diferita de varianta folosita in realitate. Intai o sa vorbesc de componentele initiale pe care le-am folosit in aplicatia Tinkercad si cum aveam de gand sa lucrez initial si ce schimbari am realizat.

Componentele Tinkercad: 2 x Arduino Uno, breadboard mic, keypad 4×4, buzzer piezo small, lcd 16×2, potentiometru 250kohm, 1 rezistor 220 ohmi, fire de legatura.

In cadrul aplicatiei de pe Tinkercad se folosesc 2 microcontrolere de tip Arduino Uno, unul folosit pentru intrare si celalalt pentru iesire. Cele 2 microcontrolere comunica prin porturile A4-A5 folosind protocolul I2C, cel de intrare fiind master-ul si cel de iesire slave-ul. Microcontroler-ul de intrare este conectat prin 8 port-uri la un keypad 4×4. Pin-urile pentru linii sunt 6-9, iar cele pentru coloane sunt 2-5. Arduino de iesire este conectat prin port-ul 8 la un buzzer piezo small si prin port-urile 2-5 si 11-12 la un lcd 16×2.

- port-urile 2-5 sunt conectate la pin-urile 11-14 de pe LCD si anume pin-urile pentru date (Data 4-7)
- port-ul 12 este conectat la pin-ul 4 de pe LCD, acesta este register select-ul care schimba de pe modul de scriere caractere pe modul de executare instructiuni (repozitionarea cursorului, golirea ecranului, etc.)
- port-ul 11 este conectat la pin-ul 6 de pe LCD care este enable-ul prin care se spune lcd-ului ca datele sunt pregatite pentru a fi citite

Se foloseste un potentiometru de 250k pentru a varia contrastul si luminozitatea ecranului, acesta fiind conectat la port-ul 3 al lcd-ului de contrast voltage. Potentiometrul si lcd-ul primesc ambele tensiune de la Arduino. In mare am folosit potentiometrul pentru pornirea si oprirea lcd-ului, cand acesta era pe maxim era pornit, cand acesta era pe minim lcd-ul era oprit. Rezistor-ul de 220 de ohmi este folosit pentru a seta luminozitatea backlight-ului.

## Fizic

Acum o sa discut despre ce modificari am adus pentru varianta concreta.

Componentele folosite: 2 x Arduino Uno, breadboard mic (atat e nevoie pentru proiect, dar cum nu aveau in stock am luat un breadboard mai mare), keypad 4×4, buzzer, LCD 16×2 I2C, fire de legatura.

In mare ce s-a schimbat a fost folosirea unui LCD 16×2 I2C in locul celui normal la care ar fi trebuit sa lipesc firele de legatura.

### LCD

N-am mai avut nevoie de potentiometru sau rezistenta pentru ca acestea sunt deja incluse in LCD-ul I2C. Potentiometrul fiind ca un surub pe care il poti intoarce cu o surubelnita pentru a controla luminozitatea backlight-ului. Acesta are pini in care pot baga fire mama-tata nefiind nevoie sa lipesc eu firele. Cum se vede pe desenul de mai jos unul este pentru Ground, unul pentru alimentare (5V) si ceilalti 2 pentru comunicarea I2C cu placuta.

Ma gandeam initial ca acuma ca nu mai am nevoie de potentiometru si rezistenta nu o sa mai am nevoie nici de breadboard. Problema era ca pin-urile A4-A5 ale Arduino-ului de output (slave-ul) le foloseam pentru comunicarea cu master-ul. Asa ca a trebuit in final sa folosesc breadbord-ul pentru a putea sa conectez Arduino-ul de output atat cu LCD-ul cat si cu Master-ul. Asa presupun ca toate cele 3 (cei 2 Arduino si LCD-ul) ar fi conectate prin I2C, astfel as putea sa comunic si prin Master la LCD desi n-am de gand sa fac asta pentru a pastra modularitatea. LCD-ul va fi considerat drept Slave in comunicarea I2C cu Arduino.

### Buzzer

Mai jos e diagrama buzzer-ului obisnuit care se conecteaza direct la breadboard. In sine diagrama e asemanatoare pentru buzzer-ul luat de mine (cel din diagrama de deasupra), diferenta este ca al meu are o intrare de tip mama pentru a putea folosi fire tata-tata cu acesta si Arduino. Initial il luasem pe acesta pentru ca credeam ca nu mai am nevoie de breadboard ca o sa folosesc LCD I2C. Comanda pentru buzzer am dat-o inainte de comanda pentru breadboard, asa ca am ramas cu acest buzzer desi am folosit in final breadboard-ul. Buzzer-ul ca oricare are 1 pin pentru Ground(firul negru) si unul pentru alimentare (firul rosu).

### Keypad

Keypad-ul e controlat matriceal, acesta fiind transpus intr-o matrice de 4×4. Cate 1 pin folosit pentru fiecare linie (4 linii ⇒ 4 pin-uri) si cate 1 pin folosit pentru fiecare coloana (4 coloane ⇒ 4 pin-uri), in total ajungandu-se la 8 pin-uri.

## Software Design

Keypad-ul este implementat folosind biblioteca Keypad.h. Creez un keymap cu toate tastele de pe keypad si 2 vectori in care completez pinii pentru coloane, respectiv pinii pentru linii. Toate aceste date sunt apoi folosite de constructorul keypad din biblioteca. Pentru a face rost de cheia tastata, apelez functia getKey() a clasei Keypad. Cum elementele din map sunt char-uri mi se va intoarce tot un char. [4] [5]

```
/*** MASTER ***/

#include <Keypad.h>

const byte numRows= 4;
const byte numCols= 4;

char keymap[numRows][numCols]=
{
{'1', '2', '3', 'A'},
{'4', '5', '6', 'B'},
{'7', '8', '9', 'C'},
{'*', '0', '#', 'D'}
};

byte rowPins[numRows] = {9,8,7,6};
byte colPins[numCols]= {5,4,3,2};
Keypad keypad= Keypad(makeKeymap(keymap), rowPins, colPins, numRows,
numCols);

void loop() {
  char buttonKey = keypad.getKey();
  ...
}
```

Pentru trimiterea tastei apasate catre microcontroler-ul de iesire folosesc protocolul I2C. Daca se apasa o tasta(valoarea intoarsa de Keymap este nenula), atunci se proceseaza acea valoare si se trimite caracterul prin intermediul clasei Wire. Apoi in slave am o variabila statica pe care o actualizez cand se activeaza evenimentul(sunt octeti primiti de la master). Cand se primeste char-ul, variabila devine diferita de 0 si apoi este procesata in loop(), dupa ce se termina de procesat se reseteaza pentru a fi folosita de octetii viitori. [7]

```
/*** MASTER ***/
```

```
#include <Wire.h>
#define I2C_SLAVE_ADDRESS 0x05

void setup() {
  Wire.begin();
}

void loop() {
  char buttonKey = keypad.getKey();
  if (buttonKey != 0) {
    char sendChar = processKey(buttonKey);
    Wire.beginTransmission(I2C_SLAVE_ADDRESS);
    Wire.write(sendChar);
    Wire.endTransmission();
  }
}

/*** SLAVE ***/

#include <Wire.h>
#define I2C_SLAVE_ADDRESS 0x05
static char i2cReadCharacter = 0;

void receiveEvent(int numBytes) {
  i2cReadCharacter = Wire.read();
}

void setup() {
  ...
  Wire.begin(I2C_SLAVE_ADDRESS);
  Wire.onReceive(receiveEvent);
}

void loop() {
  if (i2cReadCharacter != 0) {
    ...
    i2cReadCharacter = 0;
  } else if (...) {
    ...
  }
}
```

Procesarea din master a butonului apasat implica realizarea corespondentei dintre tasta apasata si semnificatia acesteia, cum au fost prezentate anterior. Insa cifrele 1-7 sunt folosite atat de nota propriu-zisa cat si de octava acesteia, asa ca a fost nevoie de o diferentiere. Pentru asta folosesc variabila step in care se salveaza numarul pas-ului curent din crearea notei. Este initializata cu valoarea 0 si incrementata pentru fiecare tastare valida, urmand ca atunci cand ultimul pas s-a realizat cu succes(tastarea duratei) aceasta sa se resteze pentru urmatoarea nota a utilizatorului. Asa ca pentru nota propriu-zisa pas-ul va fi 0 si se proceseaza corespunzator, iar dupa incrementare se ajunge la pas-ul 1 cand se asteapta octava, facandu-se astfel diferentierea dintre cele 2 stari.

```
/*** MASTER ***/

static int step = 0;

char getNote(char buttonKey) {
  step = 1;
  switch(buttonKey) {
    case '1': return 'A';
    case '2': return 'B';
    case '3': return 'C';
    case '4': return 'D';
    case '5': return 'E';
    case '6': return 'F';
    case '7': return 'G';
    default: step = 0; return -1;
  }
}

char getOctave(char buttonKey) {
  if (buttonKey >= '0' && buttonKey <= '8') {
    step = 2;
    return buttonKey;
  }

  return -2;
}

char getPitch(char buttonKey) {
  step = 3;
  if (buttonKey == '*') {
    return 'b';
  }

  if (buttonKey == '#') {
    return '#';
  }

  step = 2;
  return -3;
}

int getDuration(char buttonKey) {
  step = 0;
  switch(buttonKey) {
    case '1': return 1;
    case '2': return 2;
    case '4': return 4;
    case '8': return 8;
    case 'A': return 16;
    default: step = 3; return -4;
  }
```

```
}

char processKey(char buttonKey) {
  if (buttonKey == 'D') {
    step = 0;
    return 'R';
  }

  if (step == 0) {
    return getNote(buttonKey);
  }

  if (step == 1) {
    return getOctave(buttonKey);
  }

  if (step == 2) {
    return getPitch(buttonKey);
  }

  return getDuration(buttonKey);
}

void loop() {
  char buttonKey = keypad.getKey();
  if (buttonKey != 0) {
    char sendChar = processKey(buttonKey);
    ...
  }
}
```

In interiorul Arduino-ului de iesire fiecare valoare primita este salvata intr-o variabila corespunzatoare. Acestea urmand sa fie afisate impreuna pe LCD cand nota este redata la buzzer. Dupa ce se proceseaza durata, la urmatorul loop se va afisa si reda nota. Cum loop() se face la milisecunda, utilizatorul nu va putea sa intrerupa executarea(ar necesita timp de reactie inuman) si programul se va comporta in esenta secvential.

```
/*** SLAVE ***/

char note = ' ';
char number = ' ';
char pitch = ' ';
bool noteShown = false;
int noteDuration;
char duration = ' ';

void printCharacter(char character) {
  if (character >= 'A' && character <= 'G') {
    note = character;
  }
```

```
  if (character >= '0' && character <= '8') {
    number = character;
  }

  if (character == 'b' || character == '#') {
    pitch = character;
  }

  if ((character % 2 == 0 && character >= 2 && character <= 16)
      || character == 1) {
    noteDuration = character;
    noteShown = false;
    if (character == 16) {
      lcd.print("16");
    } else {
      // transform din cifra in ascii pentru afisare
      char durationChar = character + 48;
      lcd.print(durationChar);
    }

    return;
  }

  lcd.print(character);
}

void printToLCD(char character) {
  switch(character) {
    case 'R': lcd.print("RESET"); break;
    case -1: lcd.print("INVALID NOTE"); break;
    case -2: lcd.print("INVALID OCTAVE"); break;
    case -3: lcd.print("INVALID PITCH"); break;
    case -4: lcd.print("INVALID DURATION"); break;
    default: printCharacter(character);
  }
}void printCharacter(char character) {
  if (character >= 'A' && character <= 'G') {
    note = character;
  }

  if (character >= '0' && character <= '8') {
    number = character;
  }

  if (character == 'b' || character == '#') {
    pitch = character;
  }

  if ((character % 2 == 0 && character >= 2 && character <= 16)
      || character == 1) {
```

```
    noteDuration = character;
    noteShown = false;
    if (character == 16) {
      lcd.print("16");
    } else {
      // transform din cifra in ascii pentru afisare
      char durationChar = character + 48;
      lcd.print(durationChar);
    }

    return;
  }

  lcd.print(character);
}

void printToLCD(char character) {
  switch(character) {
    case 'R': lcd.print("RESET"); break;
    case -1: lcd.print("INVALID NOTE"); break;
    case -2: lcd.print("INVALID OCTAVE"); break;
    case -3: lcd.print("INVALID PITCH"); break;
    case -4: lcd.print("INVALID DURATION"); break;
    default: printCharacter(character);
  }
}

void loop() {
  if (i2cReadCharacter != 0) {
    lcd.clear();
    printToLCD(i2cReadCharacter);
    delay(1000);
    lcd.clear();
    i2cReadCharacter = 0;
  } else if (noteShown == false) {
    lcd.print(note);
    lcd.print(number);
    lcd.print(pitch);
    int sound = noteMap();
    playSound(sound);
    noteShown = true;
  }
}
```

Pentru redarea notelor ma folosesc de o insiruire de define-uri ce fac legatura dintre valoarea buzzer-ului si nota corespunzatoare. Insa pentru a folosi aceste define-uri trebuie sa fac legatura dintre acestea si nota formata din variabilele salvate(note + octave + pitch). Pentru asta am creat o pseudo-mapa care pur si simplu verifica cu if toate posibilitatile ce se regasesc in lista de define-uri si aceasta intoarce valoarea corespondenta sirului de caractere dat ca parametru. Nota urmeaza sa fie apoi redata folosind functia tone() cu define-ul corespunzator gasit prin pseudo-mapa si cu frecventa ca 1000/duration, unde duration este valoarea tastei discutate anterior('8' ⇒ 8, 'A' ⇒ 16, etc.). Apoi se

face un delay de durata respectiva inainte sa se faca noTone(), pentru a reda nota doar o data timp de durata precizata. [3]

```
/*** SLAVE ***/

/*** MUSICAL NOTES ***/

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
...

int noteMap() {
  char fullNote[4] = {note, number, pitch};
  if (!strcmp(fullNote, "B0b")) {
    return NOTE_B0;
  }

  if (!strcmp(fullNote, "C1b")) {
    return NOTE_C1;
  }

  if (!strcmp(fullNote, "D1b")) {
    return NOTE_D1;
  }

  if (!strcmp(fullNote, "E1b")) {
    return NOTE_E1;
  }

  if (!strcmp(fullNote, "F1b")) {
    return NOTE_F1;
  }

  ...

  return -1;
}

void playSound(int sound) {
  if (sound != -1) {
    tone(8, NOTE_C4, 1000/noteDuration);
    delay(1000/noteDuration);
    noTone(8);
    lcd.clear();
    return;
```

```
  }

  lcd.setCursor(0, 1);
  lcd.print("INVALID NOTE");
  delay(1000);
  lcd.clear();
}

void loop() {
  if (i2cReadCharacter != 0) {
    ...
  } else if (noteShown == false) {
    ...
    int sound = noteMap();
    playSound(sound);
    noteShown = true;
  }
}
```

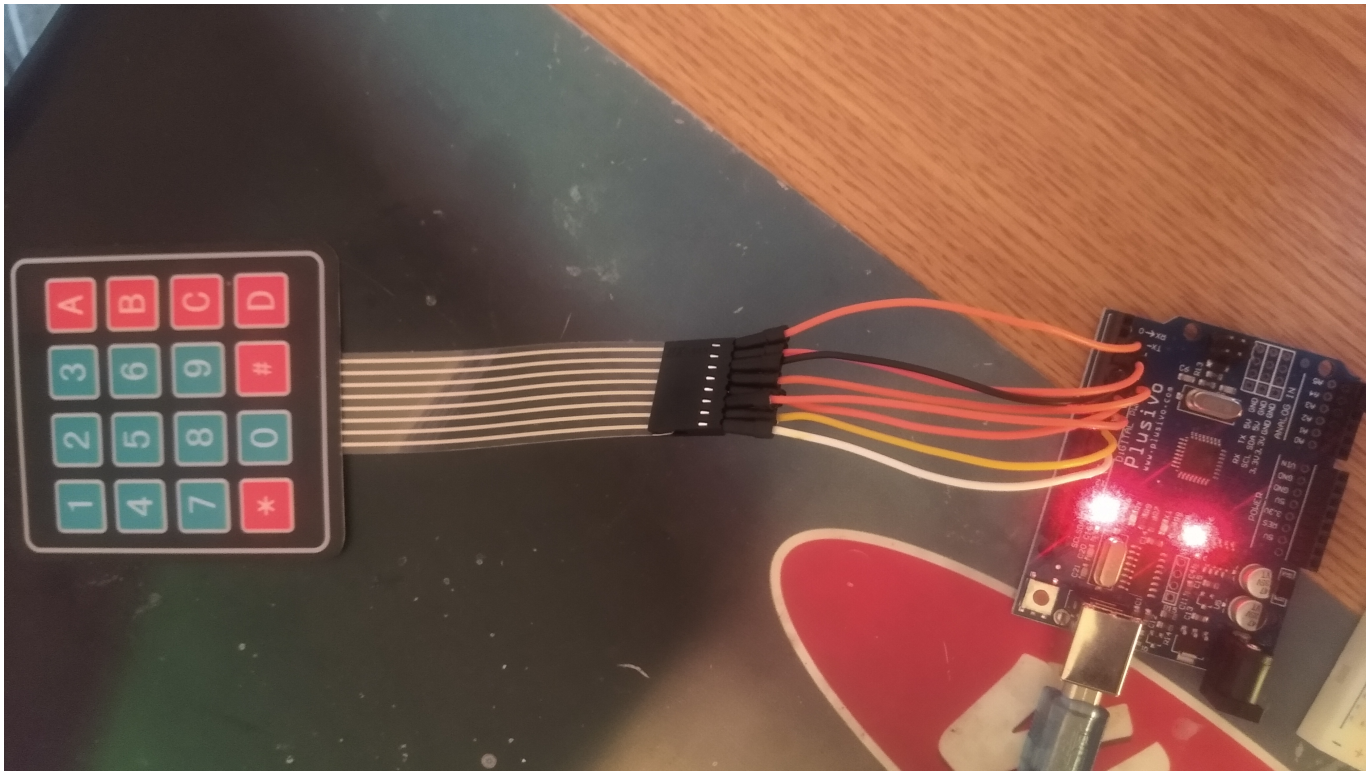
Pentru LCD am schimbat biblioteca din LiquidCrystal.h pe care am folosit-o in Tinkercad cu LiquidCrystal_I2C.h care e compatibila cu LCD-urile I2C. Functiile folosite in mare sunt aceleasi: print(), setCursor(), etc. Diferenta este modul in care se declara LCD-ul si se face setup-ul la LCD:

```
/*** SLAVE ***/

#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  lcd.init();
  lcd.backlight();
  ...
}
```

# Rezultate Obţinute

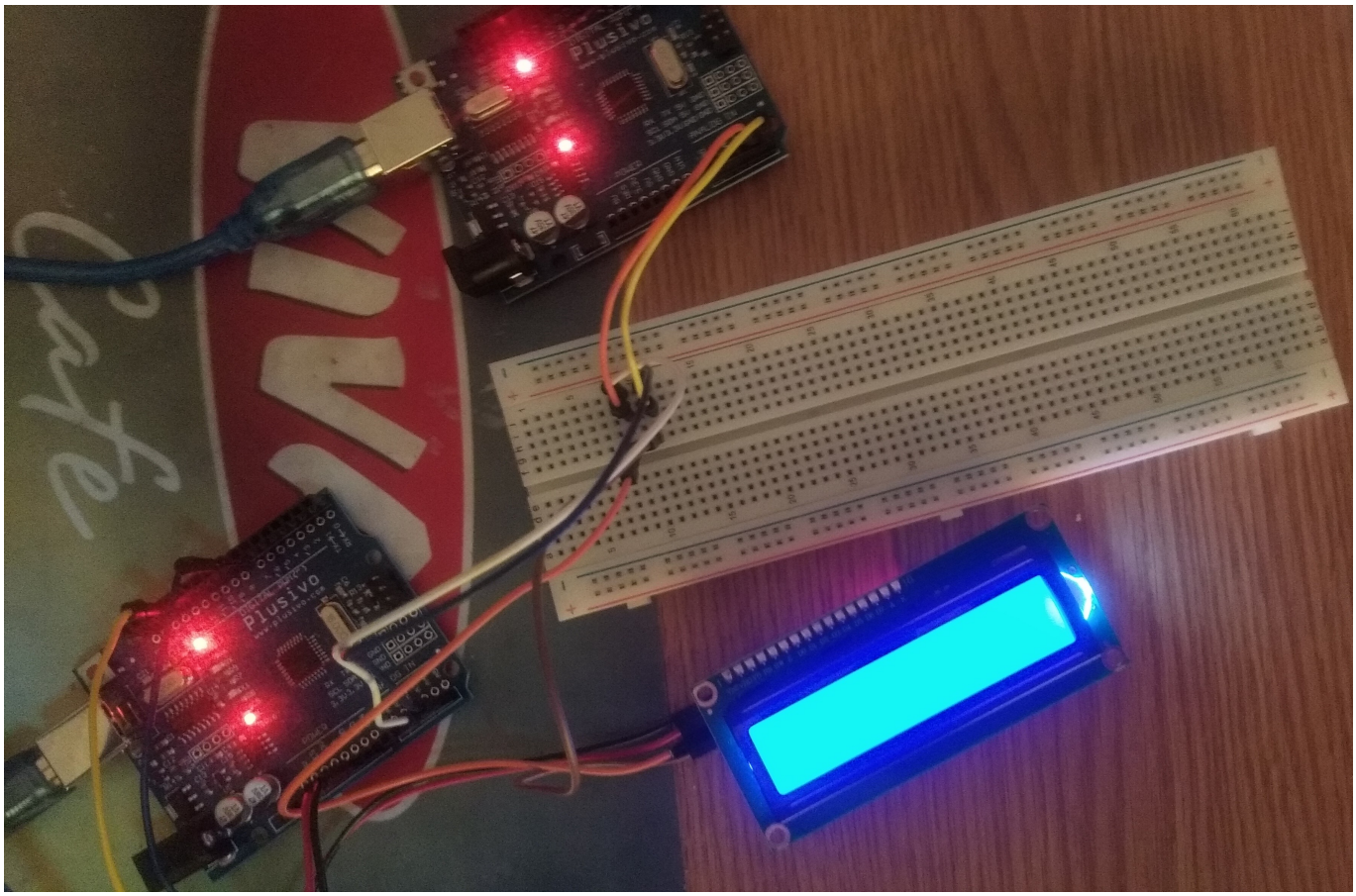
# Poza 1

In imagine apare Arduino-ul de input care are 8 pini conectati la un Keypad 4×4.



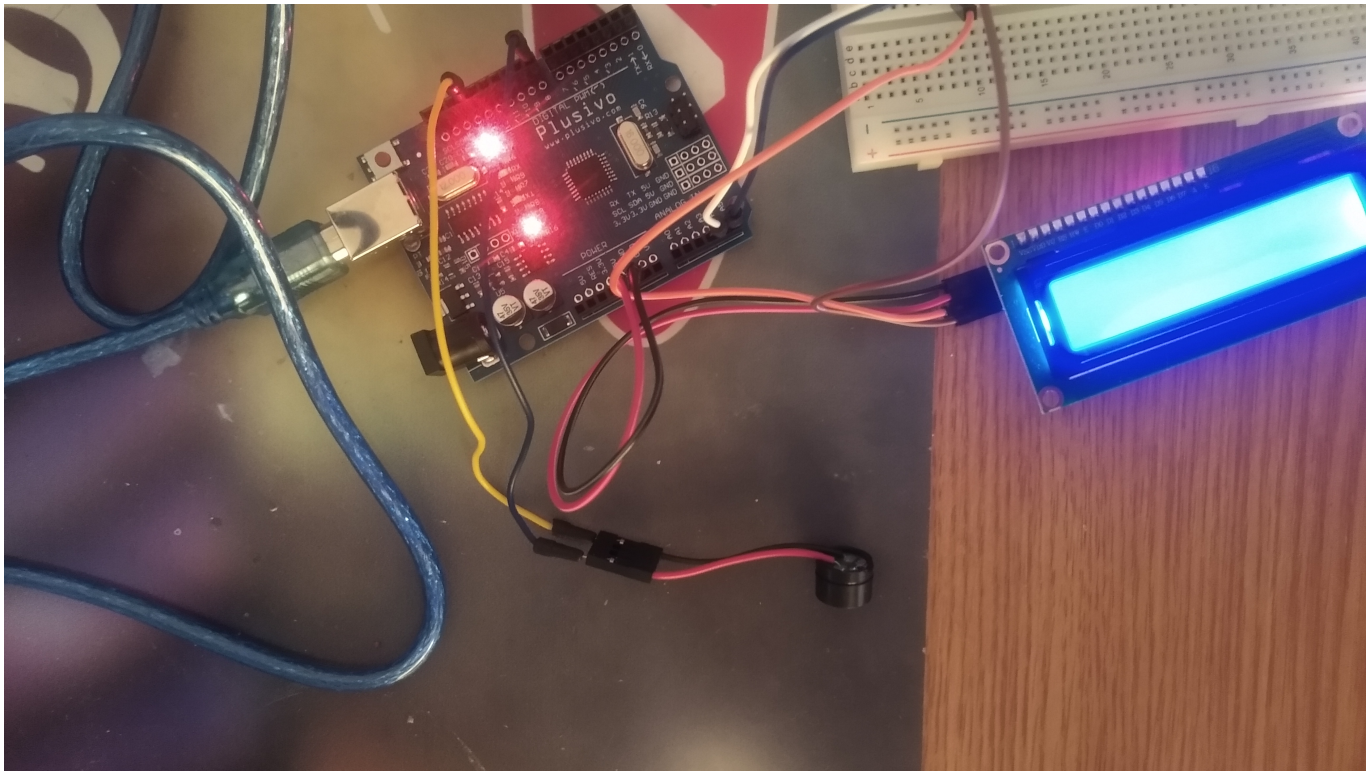

## Poza 2

Se poate vedea cum comunica prin I2C Master si Slave-ul si cum comunica Slave-ul sau Arduino-ul de Output cu LCD-ul tot prin I2C. Toate sunt conectate intre ele prin breadboard.



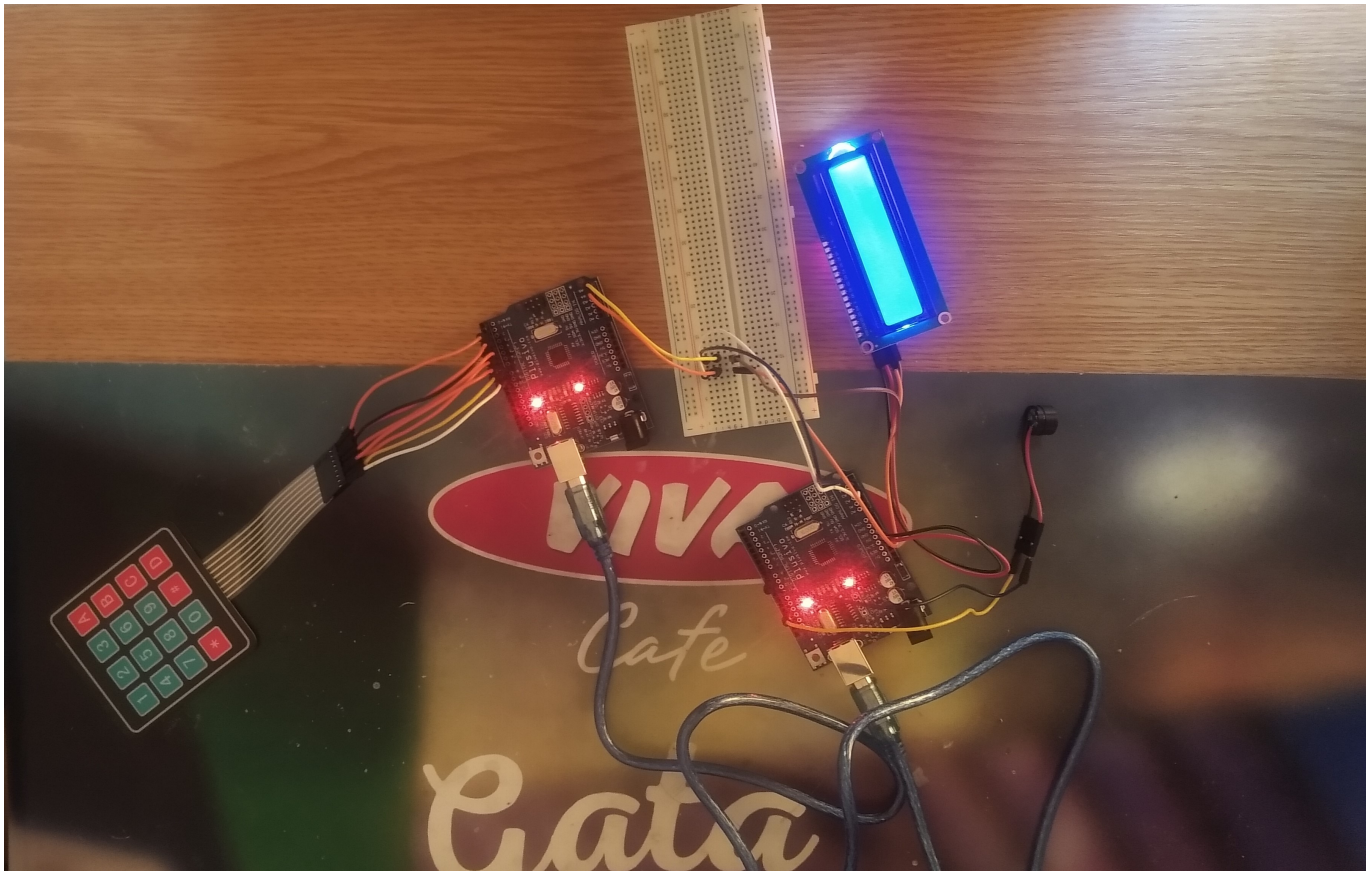
# Poza 3

Aici apare in prim plan Slave-ul care foloseste cate 2 pini pentru buzzer si pentru LCD pentru a le alimenta si a le oferi un Ground.




# Poza 4

In ultima poza se poate vedea tot proiectul in ansamblu.



## Concluzii

Experienta m-a facut sa apreciez mai mult lucrul cu microcontrolere si nevoia acestora pentru a implementa aplicatii care n-au nevoie de componente de uz general precum PC-ul. Sunt recunoscator ca tehnologia a avansat atat de mult incat n-a fost nevoie sa pun deloc mana sa lipesc, totul mai nou avand o varianta cu pinuri.

Aplicatia in sine este completa cand vine vorba de redarea unei note, aceasta continand octava, timbrul si durata notei respective. Ar putea fi folositoare pentru studiu didactic al notelor, observand astfel diferenta dintre note. O persoana putand verifica octave diferite pentru aceeasi nota pentru a observa diferentele sonore dintre acestea, acelasi lucru fiind valabil pentru timbrul superior si cel inferior. De asemenea, persoanele care nu au un sens bun al timpului ar putea incerca durate diferite pentru a se obisnui cu acestea.

## Download

iancu_alex_sound_keyboard.zip

# Bibliografie/Resurse

## Resurse Software

- DIY Arduino LCD-Keypad Maths Game
- Piano Keyboard Diagram - Layout Of Keys With Notes
- Playing Melodies using Arduino Tone() Function
- Using Keypad 4x4 with Arduino
- Keypad Interfacing in Tinkercad
- Interfacing LCD With Arduino on Tinkercad
- Arduino Communication Using I2C
- Keypad I2C Display

## Resurse Hardware

- Use 16x2 LCD With I2C
- I2C 1602 LCD Datasheet
- Buzzer Datasheet
- 4x4 Matrix Membrane Keypad Datasheet
- ATmega328P Datasheet

Export to PDF