SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR
GÉPÉSZMÉRNÖKI, INFORMATIKAI ÉS VILLAMOSMÉRNÖKI KAR

Gitárhangoló készítése frekvencia alapján

Mikroelektromechanikai rendszerek

(GKNB_INTM020)

Sziklai Botond

MGYJPVGIVK-Mérnökinformatikus BSc

Tartalom

1. Bevezetés

A projektmunkám egy elektromos gitárhangoló megalkotásáról szól. Egy ilyen eszköz képes arra, hogy a gitárunkat megfelelően be tudjuk hangolni a segítségével. Az eszköz mutatja a frekvenciát, illetve az éppen hangolandó húrt és hogy a hang pozitív vagy negatív irányban tér el a megfelelőtől.

Az elv a projektmunka mögött az volt, hogy minden gitár húrnak a megfelelő hangolással adott frekvenciája van, így csak azt kellett vizsgálni, hogy a megfelelő tartományba essen a frekvencia. A projektmunkám során alkalmazni tudtam a tananyagban megszerzett ismereteket és a gyakorlatban is hasznosítani tudtam.

A projektet egy Arduino Uno és egy LCD kijelző segítségével valósítottam meg.

1. Elektromos gitárhangoló

2. Elektromos gitár működése

Mielőtt bárminek is neki tudtam volna állni először meg kellett értenem egy elektromos gitár működését. Mi alapján tudom kinyerni a számomra szükséges információt? Mi az a jel? Nevéből adódóan is következik, hogy valamelyik villamos jelenség fogja közölni velünk ezt. Ez pedig a gitár kimeneti feszültsége lesz. Egy elektromos gitár alapvetően annyiban különbözik akusztikus társától, hogy nincs neki „dobja”, ami a hangot felerősíti, helyette vannak ezek az úgynevezett hangszedők. Egy gitáron egy, de akár három hangszedő is elhelyezkedhet és egy kapcsoló állításával lehet közölük választani, hogy éppen melyik „működjön”. A működjön, azért van idézőjelben, mivel ezek (javarészt) passzív eszközök. Ezek a hangszedők nem csinálnak mást, mint a gitár húrjainak rezgési frekvenciáját átalakítják azzal arányos feszültséggé.

2. Elektromos gitár hangszedője

3. Projekt felépítése

Így, hogy már tudtam mi az információ hordozó, amit fel kell dolgoznom, így már le tudtam fektetni mik az eszközzel támasztott feltételek és ötletek. A hangolónak tehát szüksége lesz egy 6,3 mm-es jack bemenetre, amelyen fogadni tudja majd a jeleket. Szükség lesz tehát egy vezérlőre, ami képes a gitár jelét feldolgozni, digitális jellé alakítani, illetve egy LCD kijelzőt is kezelni. A legkézenfekvőbb megoldás valamelyik Arduino board volt.

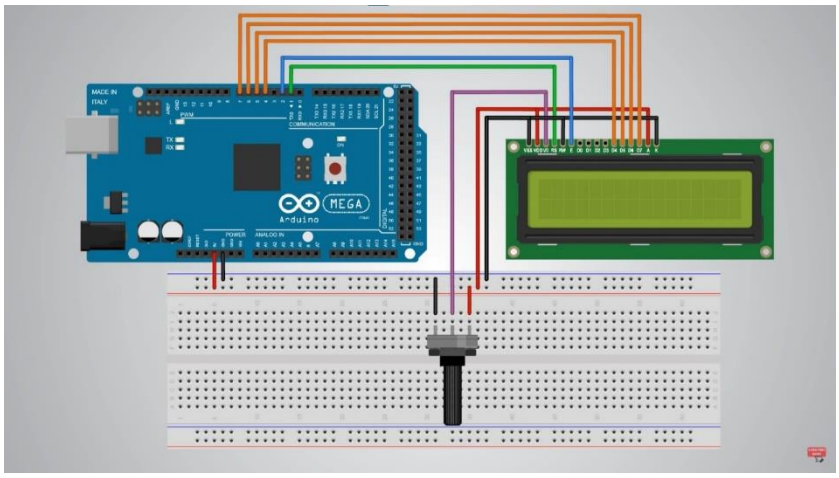
3.1 Arduino

Több ok miatt döntöttem az Uno board mellett. Az első az volt, hogy ingyenes, egyszerű fejlesztő környezete van, rengeteg különböző fórumokon található segédanyaggal, illetve könyvtárral. Az UNO esetében ez egy ATmega328/P típusú, 8 bites mikrovezérlőről beszélhetünk. A mikrovezérlő egy integrált áramkör, amiben a processzor és a memória mellett számos egyéb periféria is megtalálható.

3. Arduino Uno

3.2 LCD kijelző

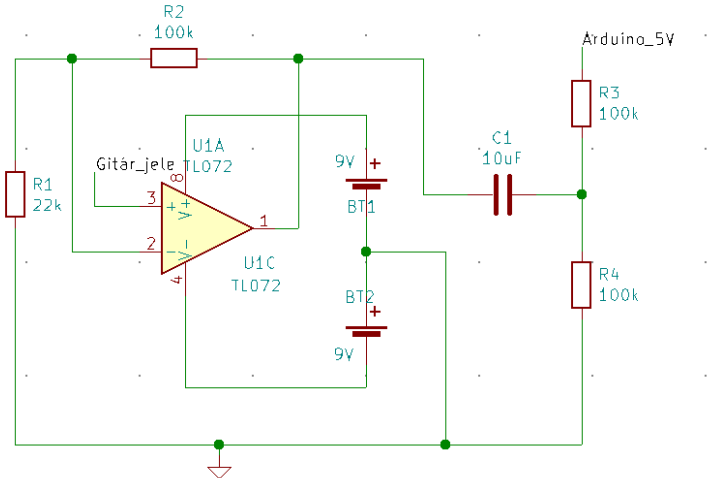
Egy kétszer tizenhatos LCD kijelzőt használtam, ugyanis a frekvencia a gitárhúr és egy ”behangolva” üzenetnek kiírására teljesen megfelelő.



4.Kijelző kapcsolási rajza

3.3 Áramkör

Az áramkör megalkotása során abba a problémába ütköztem, hogy az A0-ás pin csak 0-5V–ig érzékel, vagyis a negatív félperiódusokat levágná. Emiatt egy műveleti erősítőre volt szükség. Erre a célra a TL 072CP/DIP-et választottam, amelynek a kimenetére van csatlakoztatva egy eltolás, ami az oszcillációt 2,5V köré teszi számunkra. A szükséges 5V-os tápfeszültséget az Arduino szolgáltatja hozzá. Az áramkör 3 részegységre bontható. Előerősítő, DC Offset és Arduino részre.



5. Kapcsolási rajz

4. Program

4.1 Könyvtárak importálása

```
1   #include <LiquidCrystal.h>
```

Legelőször beimportáltam a szükséges könyvtárakat és header fájlokat. Legelső a LiquidCrystal nevezetű könyvtár volt, mely a kommunikációért felel az Arduino és az LCD kijelző között. A LiquidCrystal könyvtár lehetővé teszi az Arduino számára az LCD kijelzők vezérlését a Hitachi HD44780 (vagy egy kompatibilis) chipsettel, amely a legtöbb szöveges LCD-n megtalálható. A könyvtár 4 vagy 8 bites módban működik.

4.2 Változók létrehozása

```
3   //Creating variables
4   #define SOR 2
5   #define OSZLOP 16
6   #define PIN A0
7   int bekapcs,kikapcs;
8   float frek,periodus;
9   LiquidCrystal lcd(1,2,4,5,6,7);
```

Következő sorokban változókat hoztam létre. Három fajta változó típussal dolgoztam a programban, define, int és float. A define paranccsal olyan változókat hoztam létre, melyeknek értéke a program futási ideje alatt nem változik, konstans marad. Jelen esetben ilyen a SOR, OSZLOP és PIN változók. Az első kettő az LCD méreteit definiálja, míg a PIN változó azt a lábat adja meg, amelyen az Arduino a gitár jelét kapja. Ezek a konstansok soha nem változnak. Következő négy változó már olyan értékeket képviselnek, amelyek folyamatosan változnak. Az int és a float közötti különbség, hogy az int, mint integer, egész számokat tud csak felvenni, míg a float akár törtszám értéket is tud értelmezni

4.3 Setup

```
12  void setup(){
13    pinMode(PIN,INPUT); //A0 pin to input
14    lcd.begin(OSZLOP, SOR); //Display's size
15    lcd.print("Made by Sz. B.");
16    delay(1000);
17    lcd.clear(); //Clear display
18  }
```

A setup scope az Arduinonak az a része, amely csak egyszer fut le, mikor elindul. A void arra utal, hogy a setup függvénynek nincs semmilyen return értéke, azaz nem ad semmi visszatérési

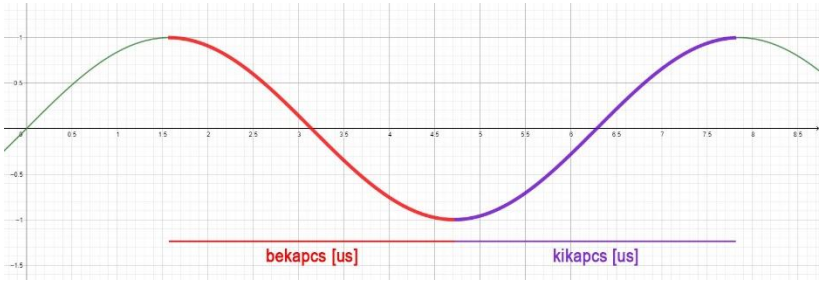
értéket. Négy dolgot állítottam be, hogy az A0-s láb legyen a bemenet, hogy mekkora a kijelzőnk mérete és hogy bekapcsolásnál jelenítse meg a kijelzőn a fejlesztő monogramját. A pinMode egy beépített függvény, ami azt jelenti, hogy az Arduino fejlesztői környezete alapból felismeri. Feladata, hogy a megadott lábat úgy konfigurálja, hogy bemenetként vagy kimenetként viselkedjen. Két paramétere van, amelyet meg kell adni. Értelemszerűen a lábat, amellyel dolgozni szeretnénk, és a viselkedési módot. Jelen esetben az A0-s lábat használtam, ezért megadtam a PIN konstanst, mivel értéke A0, és bemenetként dolgoztam vele, mivel innen kaptam a gitár jelét, ezért INPUT értéket adtam meg a második paraméter helyére. Az lcd.begin paranccsal adtam meg, hogy egy LCD-t használok, és hogy mekkora a mérete. Jelen esetben egy 16x2-es kijelzőt használtam, ezért megadtam az OSZLOP és SOR konstansokat.

4.4 Loop

```
21  void loop()
22  {
23    bekapcs = pulseIn(PIN,HIGH); //In microseconds, we get back the time until it goes from high to low
24    kikapcs = pulseIn(PIN,LOW); //In microseconds, we get back the time until it goes from low to high
25    periodus = bekapcs+kikapcs; //Period calculation
26    frek = 1000000.0/periodus; //Frequency calculation
27
28    if(periodus > 0){ //Real data
29     lcd.setCursor(0,0);
30     lcd.print(frek); //Exact frequency display
31    }
32
33    //E string
34    if(frek > 270 && frek < 360 ){
35      if(frek > 323 && frek < 336){
36        lcd.setCursor(8,0);
37        lcd.print("E-hur");
38        lcd.setCursor(3,1);
39        lcd.print("Behangolva");
40      }
41      else if(frek > 335 && frek < 360){
42        lcd.setCursor(8,0);
43        lcd.print("E -");
44      }
45      else if(frek > 269 && frek < 322){
46        lcd.setCursor(8,0);
47        lcd.print("E +");
48      }
49    }
140   delay(200); //Delay
141   lcd.clear(); //Clear display
142 }
```

Következő és egyben az utolsó részletdarabja a kódnak, a loop. Ez az a része a kódnak, mely folyamatosan fut és számolja a frekvenciát. A frekvencia kiszámításához a bekapcs, kikapcs, periodus, és frek változókat, és a beépített pulseIn függvényt használtam fel. A pulseIn függvény egy időértéket ad vissza. Két paramétert fogad el, a lábat (jelen esetben nálunk az A0), és hogy HIGH vagy LOW. Például, ha az érték HIGH, a pulsseIn() megvárja, hogy a láb LOW-ról HIGH-

ra váltson, elindítja az időzítést, majd megvárja, amíg a láb LOW-ra megy, és leállítja az időzítést. Visszaadja az impulzus hosszát mikroszekundumban, vagy feladja és 0-t ad vissza, ha az időkorláton belül nem érkezett teljes impulzus. Legegyszerűbben ezt a 6. ábra segítségével lehet szemléltetni.



6. pulseIn függvény szemléltetése.

A bekapcs és kikapcs változók mentik el ezeket az értékeket. A bekapcs változó menti el azt az időszakaszt, míg a jel magasról alacsonyra megy, míg a kikapcs alacsonyról magasra való időt. Jól látható, ha a frekvencia minél magasabb, annál gyorsabban zajlott le valami. Kérdés már csak az, hogy jelen esetünkben mi az a ”valami”. Nálam a T, a periódusidő, ami a frekvenciát adni fogja. Periódusidő alatt értem azt, hogy a jel visszatér ugyanazon állapotába. A 6. ábrát megtekintve könnyen kikövetkeztethető a periódusidőnk a bekapcs és kikapcs összege. Ennek tudatában már tudtam frekvenciát számolni, csak venni kellett a periodus változónk reciprokát. Nem egészen! Hiszen, bekapcs és kikapcs változóink értéke mikroszekundumban, azaz egy milliomod másodpercben vannak megadva, összegük is mikroszekundum lesz. Ha a frekvenciát Hz-ben akarjuk megkapni, akkor ezt a reciprokot még meg kell szorozni 1 millióval. A frekvenciát kiszámolva már kiírhattam a kijelzőre, de először meg kellett néznem, hogy egyáltalán pengetik-e a gitárt, ezt pedig egy if-es szerkezettel tudtam megvalósítani. A feltételünk az, hogy a periodus magasabb mint 0, tehát játszanak a gitárral. Majd ezután megnéztem a frekvenciát, hogy melyik tartományban volt, ami alapján be tudtam azonosítani melyik húrt akarjuk behangolni. Mivel nem volt teljesen pontos a frekvencia felismerés, ezért egy biztonsági tényező belekerült a tartományok kiválasztásánál.

String	Note	Frequency (Hz)
6	E	82.407
5	A	110.000
4	D	146.832
3	G	195.998
2	B	246.942
1	E	329.628

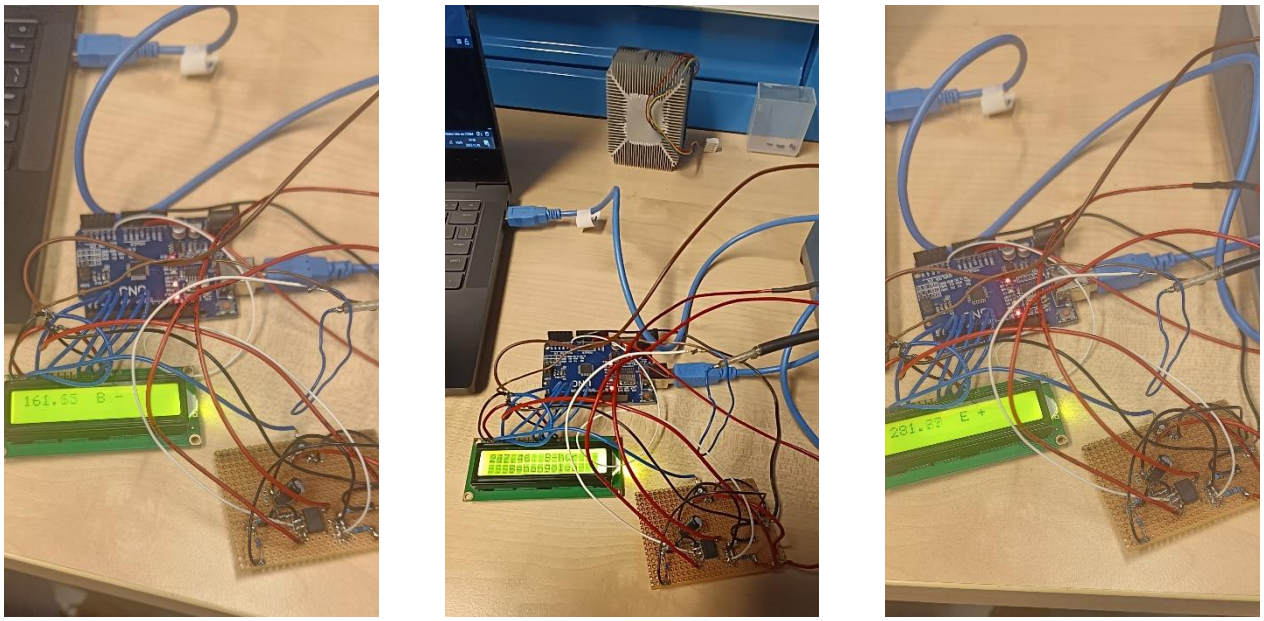
7. A frekvencia tartományok megadásához használt táblázat

A következőben a kurzort a setCursor beépített függvénnyel tudtam elérni, melynek paraméterei oszlop és sor. Jelen esetben a kijelző jobb oldalára szerettem volna tenni. Mivel C++-ban írjuk a programot a számozás 0-tól kezdődik, ezért a 8. oszlop és a 0. sort adtam meg paraméternek. A következőkben azt vizsgáltam, hogy a tartományon hova esik a frekvenciám, ami alapján meg tudtam mondani, hogy a húr alul vagy felül vagy megfelelően van hangolva. Ezt a folyamatot mind a 6 húrra megírtam. Amint az if szerkezetből kilép a kód már csak 2 dolog maradt hátra. Azt akartam, hogy az értékek a kijelzőről leolvashatóak legyenek, ezért késleltettem a programot a delay függvénnyel, mely paramétere milimásodpercben adandó meg. Majd a legvégén letöröltem a kijelzőt, hogy az új adatok hiba nélkül jelenjenek meg.

5. Tesztelés

Mivel nem rendelkezem elektromos gitárral, azonban az Arrabona Racing Team tagjaként hozzáfértem egy National Instruments Virtual Bench-hez, ami képes jelgenerálásra, így azzal le tudtam tesztelni az eszközömet. Az alábbi képeken jól látható a működése.

8. National Instruments Virtual Bench VB-8012



9. Gitárhangoló működés közben

Ábrajegyzék