Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Digitális gitárhangoló tervezése

Szakdolgozat

Készítette:

Stolczenberger Róbert

Konzulensek:

dr. Sujbert László, docens

Scherer Balázs, tanársegéd és doktorandusz

2008

Feladatkiírás

Az elektronikus gitárhangoló a zenész munkáját könnyíti meg: a készülék a megpendített húr által adott hang frekvenciáját méri és jelzi ki. Egyes típusok a húr alaphangját felismerik és a hangolás irányát is jelzik. A gitárhangoló tipikus beágyazott rendszer, amelynek érzékelési és jelfeldolgozási feladatokat kell ellátnia, miközben a felhasználóval nagyon egyszerű kezelői felületen kommunikál. Gitárhangoló kereskedelmi forgalomban is kapható, de egy korszerű eszköz tervezése tipikus mérnöki feladat.

A gitárhangoló megvalósítására mikrokontroller alapú rendszer alkalmas, a megfelelő érzékelő áramkörökkel, kijelzővel, illetve kezelőszervekkel. A szakdolgozat keretében a hangoló rendszertervét kell elkészíteni, illetve a valós idejű működést megfelelően kiegészített mikrokontrolleres fejlesztőrendszeren kell demonstrálni.

Fentiek alapján a szakdolgozat-készítés keretében az alábbi konkrét feladatokat kell megoldani:

- Ismertesse a megoldandó jelfeldolgozási problémát, és adja meg a jelfeldolgozó rendszer blokkvázlatát!
- Végezze el a szükséges tervezési feladatokat, a helyes működést tesztelje MATLAB környezetben!
- Készítse el a gitárhangoló általános rendszertervét!
- Implementálja gitárhangolót az STM32 processzort tartalmazó IAR KickStart Kit fejlesztőrendszer segítségével! A helyes működést mérésekkel ellenőrizze!

.....

dr. Sujbert László
docens

Nyilatkozat

Alulírott, ***Stolczenberger Róbert***, a Budapesti Műszaki és Gazdaságtudományi Egyetem hallgatója kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, és a szakdolgozatban csak a megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

.....
Stolczenberger Róbert

Tartalomjegyzék

Kivonat

Szakdolgozatom célja egy kompakt digitális gitárhangoló megvalósítása, mely kevés analóg elemet tartalmaz, a legtöbb funkciót (pl. szűrés) digitálisan valósítja meg.

A gitár által kiadott hanghullámot egy mikrofon feszültségjellé alakítja. Ezt a feszültségjelet adott mintavételi frekvenciával egy 32 bites mikrokontroller belső AD-átalakítója mintavételezi. A mikrokontroller folyamatosan, az adott húrnak megfelelően szűri a mintavételezett jelet, illetve figyeli a nullátmenetek között eltelt időt. Ennek az időnek a reciproka az adott hang alapharmonikus frekvenciája. Az alapharmonikus frekvenciája kölcsönösen egyértelmű függvénye a kiadott hang magasságának.

Mindezt egy 32 bites, ARM magos, STM32 processzort tartalmazó STM32F10B-EVAL próbakártya segítségével valósítottam meg. A kártyán található mikrovezérlő korszerű, nagy teljesítményű, alacsony fogyasztású, real-time feladatok elvégzésére alkalmas, így beágyazott rendszerek fejlesztéséhez ideális eszköz. Mindemellett nagyon kedvező ára teszi népszerűvé a fejlesztők körében. Kiemelkedő tulajdonságai és alacsony ára alapján nyugodtan kijelenthetjük, hogy felveszi a versenyt a 8 bites kontrollerekkel.

A megvalósított gitárhangoló működik, elvárásainknak megfelelően számítja a frekvenciát, és kijelzi, hogy a húr által kiadott hang alapharmonikus frekvenciája megfelelő-e, illetve hogy milyen irányban tér el a kívánt értéktől. Pontossága bizonyos esetekben eltér a tervezett 1 centtől, melynek egyrészt a szűrőspecifikáció, másrészt a frekvenciamérés pontatlansága lehet oka. Továbbfejlesztési lehetőségek a következőek: Pontossági követelmény (1 cent) megvalósítása minden húrra; automatikus húrfelismerés; valamint a gitárhangoló termékké fejlesztése.

1. Abstract

The purpose of my BSc Thesis is to design a digital guitar tuner, that contains only a few analog parts, the most of the functions are implemented with digital units.

A microphone converts the sound of the guitar to voltage signal. The signal is sampled with the analog-to-digital conversion unit of a 32-bit flash microcontroller. The sampled signal is being filtered continuously according to the selected string, while the controller measures the time between the zero crossings. The reciprocal value of this time is the fundamental frequency of the sound. It is a one-to-one onto function between the fundamental frequency and the tone pitch.

I have implemented the digital guitar tuner, using an STM32F10B evaluation board. The microcontroller on the board is a 32-bit Flash microcontroller, that based on the breakthrough ARM Cortex-M3 core. This is an advanced microcontroller with high-performance, real-time, low-power and low-voltage operation, it is specifically developed for embedded applications. On the other hand it has a very low prize. The outstanding features of the STM32 microcontroller and its low prize is the reason for that, it is entered into competition with the 8-bit microcontrollers.

The realized guitar tuner works in pursuance of our expectation. It is measuring the fundamental frequency, and it is displaying, when the pitch of the sound is too high, too low or it is good. The accuracy of the tuner differs from our expectation (1cent). The reason for that is the inaccurate specification of the filters or the inaccurate measuring of the frequency. Further development tasks are the following: improvement of the accuracy; automatic string detection; production schedule of the guitar tuner.

2. Bevezetés

2.1. A gitár

A gitár közkedvelt pengetős hangszer. Népszerűségét sokoldalúságának köszönheti. Sok fajta típusa létezik, melyek mind felépítésüket, mind használatukat tekintve nagyon hasonlítanak egymáshoz. A gitár kivitele szerint lehet klasszikus, akusztikus, elektronikus vagy elektroakusztikus. Gitáron ugyanúgy előadhatók a klasszikus darabok, mint napjaink modern popzenéi. Egyesek a zeneiskolában szerzik meg első tapasztalataikat, mások autodidakta módon tanulnak meg gitározni. A gitár egyszerre egyszerű és bonyolult hangszer. Könnyen meg lehet tanulni az akkordjátékot: hogyan kell lefogni az akkordokat, hogyan lehet ezeket bontani, esetleg kicsit cifrázni. Valamivel nagyobb kihívás dallamot játszani rajta.

2.1.1. A gitár felépítése

Minden gitár alapvetően három fő részből áll: hangszertest, nyak és fej. A hangszertestnek fontos szerepe van a hang kialakításában, erősítésében. A gitár nyakán található a fogólap és a bundok (érintők). A fejen helyezkednek el a hangolókulcsok. A húrok a húrláb és a nyereg között feszülnek, feszességüket a hangolókulcsokkal állíthatjuk be, így hangolhatjuk a gitárt. (2.1. ábra) [1]

2.1. ábra - A gitár felépítése

2.1.2. Hangolás fontossága

A zene különböző hangok kombinációjából születik. A kiadott hangok sorozata dallamot alkot. A dallamot az egymást követő hangok magassága, dinamikája (erőssége) és ritmikája adja. A zene dinamikája és a ritmikája tanulható, gyakorolható, szubjektív tényező. Ezzel szemben lejátszott hangok pontossága alapvetően határozza meg, mennyire élvezhető a dallam. Ha a játékos hamis hangokat játszik, a zenei harmónia nem jön létre. A gitár fogólapján található bundok határozzák meg a relatív pontosságot, így az csak a gitár minőségétől függhet. Amennyiben jó minőségű gitárról van szó, a hamis játék csak a rossz behangolásból következhet. Fontos, hogy az adott hangszer be legyen hangolva. A gitár különösen érzékeny a külső fizikai hatásokra, gyakran elhangolódik. Ennek oka a hangszer mérete, ebből következően mobilitása, így az időjárás, a páratartalom mind hozzájárulnak az elhangolódáshoz. Mindezek mellett az is elmondható, hogy a gitáron való játék is megerőlteti a húrokat, így azok egy idő után elhangolódnak.

Összességében elmondható, hogy a gitár hangolása nagyon fontos a játék szempontjából, mivel a megszólaltatott hangok tisztasága szükséges feltétele a zenei harmónia kialakulásának.

2.2. Hangolási módszerek

2.2.1. Gitár behangolása

A húr megpendítéskor a kiadott hang magasságát annak anyaga, hossza és feszítettsége határozza meg. Hangoláskor általában a le nem fogott húrt pendítjük, hangoljuk. Ekkor a húr anyaga és hossza állandó, csak a feszítettségén tudunk változtatni, ezzel hangolhatjuk a gitárt. A gitár fején kialakított hangolókulcsok segítségével feszíthetünk, illetve engedhetünk az adott húron.

A húr hosszának változtatása a játék közben kap szerepet, mivel úgy lehet kiadni a gitár többi, kiadható hangmagasságú hangjait.

Egy klasszikus gitárnak hat húrja van, ezeket egyenként kell behangolni. Az alábbiakban áttekintjük a hangolási módszereket. [2]

2.2.2. Relatív hangolás

Ha csak egy gitáron szeretnénk játszani, más hangszerek kísérete nélkül, elegendő a relatív hangolás. Ennek lényege, hogy először csak az egyik húrt hangoljuk be valamilyen referencia magasság szerint, utána pedig megfelelő módon – fülünk segítségével – hozzá igazítjuk (relatíve) a többi húr által kiadott hang magasságát. A referencia hang lehet egy hangvilla, egy „A”-síp vagy egy vonalas telefon „A” hangon búgó vonalhangja. Ezek segítségével behangolhatjuk az „A2” húrt. A relatív hangolás kétféleképpen történhet: húrok lefogásával, vagy üveghangok segítségével.

A húrok lefogásával a következő elv alapján hangolhatjuk be a többi húrt. A már behangolt „A” húrt lefogjuk az 5. bundnál, megpendítjük, így az azon a hangon fog szólni, mint ahogy az eggyel alatta lévő „D3”-húrnak kellene szólnia. A „D3”-húr hangmagasságát az 5. bundnál lefogott „A2”-húr által kiadott hangnak megfelelően állítjuk. A következő lépésben a „D”-húrt fogjuk le az 5. bundnál, és pendítjük meg. Az így a következő húrral egy magasságban szól. Ezzel a módszerrel folytatjuk a hangolást a legmagasabban szóló „E4”-húrig. Ez a módszer viszonylag egyszerű, viszont a gyenge vagy közepes minőségű hangszereknél a bundozás nem elég pontos, a hangolási hiba minden egyes húr hangolásánál azonos előjelű hibát eredményez. Mivel a többi húrt egymáshoz viszonyítva, „rekurzív” módon hangoljuk, a hangolási hiba mértéke a hangolás előrehaladtával növekszik.

A másik módszer az üveghangokat használja ki, mely a húrelméletből adódik. Ez viszonylag pontos relatív hangolást tesz lehetővé, viszont szükséges feltétele egy jó minőségű gitár.

2.2.3. Abszolút hangolás

Amennyiben a gitárral több hangszer együtt szól, fontos, hogy mindegyik hangszer azonos alaphangra legyen hangolva. Ilyenkor abszolút hangolás szükséges. Ennek is több változata lehetséges: Segítségül szolgálhat egy másik referencia hangszer, például egy zongora. Fülünk segítségével a zongora hangjaihoz igazítjuk a gitárok egyes húrjai által megszólaltatott hangot. Másik megoldás a hangoló síp, mely lényegében 6 síp egyben, és a sípok a gitár hat húrjának hangjainak megfelelő magasságú hangokkal szólnak.

Az eddig felsorolt hangolási módszerek mind azt használták ki, hogy a gitárosnak jó hallása van. Meghallja, hogy az azonos magasságú hangok tényleg azonos magasságúak, vagy ha nem, akkor korrigálja a hangolandó húrt. Ennek szükséges feltétele a jó hallás, de nem elégséges. Emellett egyéb tényezők is befolyásolják a hangolás pontosságát.

A digitális technika fejlődésével lehetőség nyílt a digitális gitárhangolók kialakítására. Ezek abszolút pontosságú hangolást tesznek lehetővé. Alapelvük az, hogy egy mikrofonnal mintavételezik a megpendített húr által kiadott hangot, jelkondicionálás után feldolgozzák a beolvasott jeleket, és az eredményt kijelzőn, vagy LED-ek segítségével jelenítik meg. A kijelző egységek általában hibajelet mutatnak. A hibajelet többnyire az ember mint szabályozó egység dolgozza fel, és addig húzza, illetve ereszti az adott húrt, míg a hibajel el nem tűnik.

2.3. A hang fizikája

2.3.1. A hang mint hullámmozgás

Általában senkinek sem kell elmagyarázni, hogy mi a hang. Hang az, amit hallunk. Kissé pontosabb, ha a következőképp definiáljuk: A hang egy olyan mechanikai rezgés, amely az emberi fülben hangérzetet vált ki. A hang kialakulásához szükség van *hangforrásra, hangtérre* és *hallószervre.* A hangforrás bocsátja ki a rezgést, így a hangot. A hallószerv dolgozza fel a hangérzetet. E kettő között pedig a hangtér teremt kapcsolatot, amely nem más, mint egy rugalmas közeg, rendszerint a levegő.

A hang keletkezése a hangforrás mechanikai rezgésével magyarázható. A keletkezett hanghullám a hangtérben terjed tovább. A hangforrás által kiadott hang a közegben hullámként terjed tovább. Úgy, hogy annak részecskéi szintén rezgőmozgást végeznek. A részecskék rezgésének iránya megegyezik a hanghullám terjedési irányával, tehát a hang terjedése *longitudinális hullámmozgás*. [3]

A hang kísérletileg kapott terjedési sebessége 20 °C-os hőmérsékletű levegőben:

$$v = 344 \frac{\mathrm{m}}{\mathrm{s}}$$

A hang csak rugalmas közegben terjed, a kevésbé rugalmas anyagokban elnyelődik.

2.3.2. Hangmagasság, hangszín, hangskála

A hang rezgőmozgás, így annak frekvenciája fontos fizikai mennyiség, amely alapvetően meghatározza az adott hang magasságát. Egy adott hang több frekvenciakomponensből áll össze. Az emberi fül kb. 20 Hz-től 16 kHz-ig hallja a hangokat.

Amennyiben eltekintünk a zajoktól, ideális körülmények között egy melodikus hangszer által kiadott zenei hang spektruma egy *alapharmonikus* frekvenciakomponensből, illetve ennek egész számú többszöröseinél megjelenő, úgynevezett *felharmonikus* frekvenciakomponensekből áll. Az alapharmonikus frekvencia határozza meg a hang zenei értelemben vett magasságát. A felharmonikusok csökkenő intenzitással jelentkeznek, más-más hangszerek esetén különbözőképpen. Fontos, hogy ezek intenzitása időben hogyan változik. A hangszínt tehát az alaphang és felharmonikusainak aránya, illetve időbeli helyzete, fázisa határozza meg. Valójában a természet nem ennyire ideális, a hangszerek felhangjai nem mindig az alapharmonikus egész számú többszörösei, megjelenhetnek egyéb komponensek is.

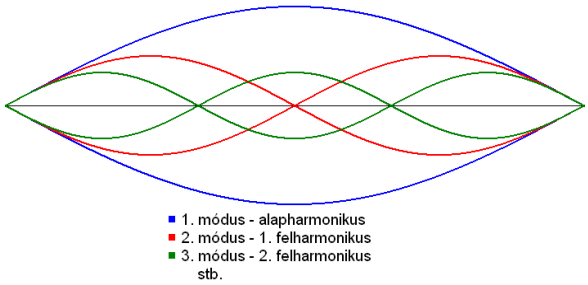
1936-ban egy zenei konferencia ajánlotta, hogy az egyvonalas „A” hang frekvenciája 440 Hz legyen. 1955-ben a Nemzetközi Szabványügyi Szervezet szabványosította ezt az értéket (ISO 16). Azóta a hangszerek nagy többségét ennek megfelelően hangolják – pl. hegedű, zongora, gitár stb.

A hangskála adott alaphangok közötti távolság. Két zenei hang alapharmonikus frekvenciájának hányadosával arányos mennyiség. Egy oktáv távolság azt jelenti, hogy az egyik hang alapharmonikus frekvenciája kétszerese a másik hang alapharmonikus frekvenciájának. A kromatikus skálán a zeneileg tiszta hangokat értjük, méghozzá abban az értelemben, hogy azok nem hamisak. A kromatikus hangsorban fél hangonként követik egymást az adott hangok. Egy fél hang távolság $\sqrt[12]{2}$ nagyságú frekvenciaaránnyal határozható meg. [3]

2.3.3. Ideális húrmodell

Gitár esetén a hangot a húr rezgése kelti. A húr rezgését a gitár teste mint rezonátor erősíti. A húr hullámegyenletének numerikus megoldása a húrmodell. Az ideális húr hullámegyenlete a klasszikus egydimenziós hullámegyenlet. Ugyanígy modellezhető pl.

az ideális távvezeték vagy a levegőoszlop hullámterjedése (pl. fúvós hangszerek esetén). A hullámegyenlet megoldására több módszer létezik. Számunkra a legkényelmesebb módszer a húr rezgésének módusokra bontása. A húr pillanatnyi alakját állóhullámok szuperpozíciójaként közelítjük. A kitérés módusait a 2.2. ábrán láthatjuk. A húr két végén mindig csomópont található. [4]



2.2. ábra - A húr kitérésének módusai

A megfeszített húr nyugalmi állapota az, amikor az a legrövidebb. Megpendítésekor a húr arra törekszik, hogy visszakerüljön nyugalmi állapotába. A megpendítés pillanatában a húr helyzeti energiája megnövekszik. Amint elengedjük a húrt, elindul nyugalmi állapota felé, miközben helyzeti energiája csökken, viszont sebessége megnövekszik. A húr a nyugalmi állapotnál nem áll meg, mivel ott a legnagyobb a sebessége, így mozgási energiája tovább lendíti az alsó helyzetig, majd ismét elindul felfelé. Egyéb fizikai hatások miatt kis idő elteltével az előbb leírt rezgőmozgás csillapodik, így végül a nyugalmi helyzeténél megáll a húr. A csillapodás oka főként a környezet felé történő kicsatolás. A húr rezgésének hatására a hangszertest rezonátorként viselkedik. Felerősíti a húr által kiadott hangot, majd a levegőnek átadja a rezgőmozgást. Ennek energiáját a húr fedezi, így az felerősített hanghatás hamarabb lecsillapodik, mintha csak egy hangszertest nélküli monokord rezegne.

A harántrezgéseket végző húr sajátfrekvenciái:

$$f_n = \frac{n}{2l}\sqrt{\frac{F}{q\rho}} \quad (n = 1, 2, \ldots) . \tag{2.1}$$

Ha $n = 1$, a (2.1) formula annál magasabb hangot ad, minél rövidebb a húr, minél nagyobb a húr feszítőereje (F) és minél kisebb a húr egységnyi darabjának tömege ($q\rho$). [3]

3. A digitális gitárhangoló alapelve

3.1. Megoldási lehetőségek

A gitárhangoló tipikus beágyazott rendszer. Megvalósításához érzékelési és jelfeldolgozási feladatok elvégzésére van szükség. A felhasználóval egyszerű kommunikációt valósít meg: az esetleges beállításokat a nem túl bonyolult kezelői felületen lehet elvégezni; kimeneti egysége egy kijelző, illetve adott esetben elegendő néhány LED. A hangmagasság a bemenő hang digitalizálása és szűrés után frekvenciaméréssel vizsgálható. Ha nem elég pontos a kiadott hang, elvégezhető az esetleges korrekció. A frekvenciamérést többféleképpen valósíthatjuk meg, a különféle megoldási lehetőségek között azonban nagyon sok a közös vonás: Minden esetben a hang elektromechanikus átalakító eszköze a mikrofon. A mikrofon jelét AD-átalakítóra vezetjük. A beérkezett elektromos jelet célszerű szűrni. A szűrés lehet az AD-átalakító előtt, vagy után, tehát analóg vagy digitális. A digitális jel feldolgozása is többféleképpen valósítható meg. Célunk a frekvencia mérése. Az alábbiakban bemutatok két módszert, melynek segítségével megmérhetjük egy adott hang alapharmonikusának frekvenciáját.

3.1.1. Sávszűrő alkalmazásával

A húroknak megfelelő hangot feszültségjellé alakítjuk, majd digitalizálás után szűrjük. A szűrő, egy a húrnak megfelelő alapharmonikus frekvencia körüli sávszűrő. A szűrés után viszonylag tiszta szinuszjelet kapunk, melynek nullátmenetei között eltelt idő alapján könnyen becsülhetjük a frekvenciát. A mért frekvenciát összehasonlítjuk a referenciaértékkel, így pl. egy LED-en kijelezhetjük, hogy a húr által kiadott hang alacsonyabb, vagy magasabb a referenciamagasságnál.

Hat húr esetén hat szűrőt kell terveznünk. A zenész hangoláskor mindegyik húrt be szeretné hangolni. A digitális gitárhangoló intelligenciájától függ, hogy meg tudja-e különböztetni a különböző húrokat a megpendítés pillanatában. Néhány gomb és egy egyszerű kijelző segítségével megkerülhető a probléma, ekkor a zenészre bízzuk a húr kiválasztását.

3.1.2. Diszkrét Fourier-transzformáció alkalmazásával

Frekvenciamérésre alkalmas lehet a Fourier-transzformáció is. Ennek hátránya, hogy a megfelelő pontosság eléréséhez nagyon nagy pontszámú DFT-t kell végrehajtanunk.

A DFT felbontóképessége a mintavételi frekvencia és a pontszám hányadosa:

$$\Delta f_{DFT} = \frac{f_s}{N} . \tag{3.1}$$

Ezek szerint a felbontóképesség javulását a mintavételi frekvencia csökkentésével és a Fourier-transzformáció pontszámának növelésével érhetjük el. A mintavételi frekvencia nem csökkenthető le nagyon, mert a digitalizált jel az átlapolódás miatt torzulhat. Ha mégis ezzel a megoldással élnénk, átlapolásgátló szűrők, és decimálás alkalmazásával csökkenthetjük a mintavételi frekvenciát, az alapharmonikus frekvenciáknak megfelelően. A legnagyobb mérendő frekvencia 329,63 Hz.

Legyen a lecsökkentett mintavételi frekvencia egységesen, minden húr esetén 400 Hz. A legmélyebb (E2) húr alapharmonikus frekvenciája 82,41 Hz, ehhez kell igazítani a felbontóképességet. Ennél 1 centtel alacsonyabban lévő hang frekvenciája a következő módon számítható:

$$\Delta f_{min} = 82{,}41\ Hz \cdot (1 - 2^{-\frac{0{,}01}{12}}) \approx 0{,}04759\,\text{Hz} .$$

A DFT osztásköze a fenti értéknek kb. a fele legyen, tehát: 0,02379 Hz.

Ha átrendezzük a (3.1) egyenletet, a DFT minimális pontszámára a következő eredményt kapjuk:

$$N = \frac{2 \cdot f_s}{\Delta f_{min}} \simeq 16811 .$$

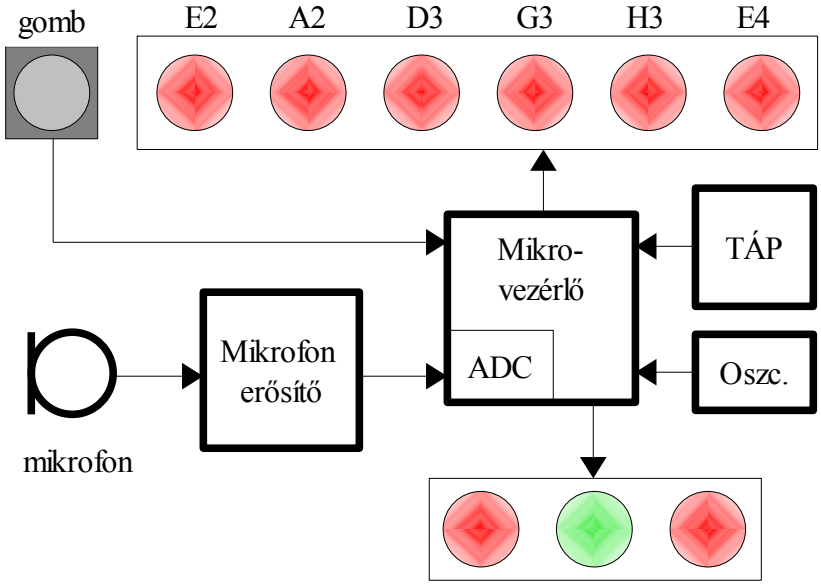
Ez a DFT pontszám egy mikrokontroller számára túl nagy számítási igényt támaszt.

3.2. Gitárhangoló általános rendszerterve

A korábban részletezett frekvenciamérési módszerek közül a sávszűrővel megvalósított frekvenciamérést választottam. Előreláthatólag ez a megoldás jobban illeszkedik egy mikrokontrolleres megvalósításhoz, mint a frekvenciamérés diszkrét Fourier-transzformáció alkalmazásával.

A tervezési feladatok közé tartozik a gitár által kiadott hangok digitalizált jelének analízise. A tervezési feladatokat MATLAB program segítségével végezhetjük el, az analízist wav formátumú felvételeken valósíthatjuk meg. Az egyszerűség kedvéért érdemes először csak egy húrt vizsgálni, később azonban minden húrra el kell végezni a következő lépéseket: Idő- és frekvenciatartományban részletesen meg kell vizsgálni az adott húr által kiadott hang digitalizált jelét. A digitalizált jel analíziséből következik a szűrőspecifikáció szigorúsága. Választanunk kell, hogy végtelen, vagy véges impulzusválaszú (IIR vagy FIR) szűrőt szeretnénk tervezni. A választás és a szűrőspecifikáció alapján végezhető el a kívánt sávszűrő tervezése. A megtervezett szűrővel szűrjük a digitalizált jelet. A szűrő kimenetén megjelenő jelet is megvizsgáljuk idő- , illetve frekvenciatartományban, ezzel ellenőrizhetjük, hogy megfelelő szűrőspecifikációt alkalmaztunk-e a szűrőtervezéskor. Amennyiben a szűrés hatékonysága megfelel elvárásainknak, folytathatjuk a tervezést. Az eddig említett módszereket minden húrra elvégezzük, és a legszigorúbb specifikációnak megfelelően választjuk meg a szűrő fokszámát. Erre azért van szükség, mert a megvalósítás szempontjából jó volna, ha a tervezett szűrők együtthatóinak száma megegyezne. A következő lépés a frekvenciamérés algoritmusának megtervezése. Fontos, hogy a frekvenciát a kezdeti tranziens lezajlása után mérjük. Továbbá nem szabad megfeledkezni a frekvenciamérés pontosságáról sem, célunk az, hogy megfelelően pontos legyen. Ha ezzel is végeztünk, akkor következhet a megvalósítás.

A 3.1. ábrán látható, milyen blokkokból épül majd fel a gitárhangoló. A digitális gitárhangoló megépítéséhez szükség van egy processzorra, egy AD-átalakítóra, tápra, egy kondenzátormikrofonra, egy mikrofonerősítő fokozatra, egy nyomógombra, egy bekapcsológombra, 9 LED-re, néhány ellenállásra, egy pár kondenzátorra és egy nagy pontosságú kvarc kristályra. A nyomógombbal lehet beállítani, hogy épp melyik húrt szeretnénk hangolni, ehhez a kijelzést 6 LED segíti. A maradék 3 LED arra szolgál, hogy a megpendített hangról megmondja, alacsonyabb, magasabb vagy pontosan jó a hangolása. A hangolás tervezett pontossága 1 cent. Ez a tervezés szempontjából a szűrő fokszámára és a frekvenciamérés pontosságára vonatkozó követelményeket támaszt.



3.1. ábra - Gitárhangoló sematikus blokkvázlata

Fontos, hogy a felhasználó számára egyszerű kezelőfelületet biztosítsunk, és az eszköz használata magától értetődő legyen. A gitárhangolót be kell kapcsolni. Alapértelmezésben az E2 húrnak megfelelő LED világít, amely azt jelenti, hogy az E2 húrt lehet behangolni. Egy nyomógomb segítségével adhatjuk meg, melyik húrt kívánjuk hangolni. A megpendítés után a készülék három LED-en jelzi a hangolás állapotát. Ha a baloldali LED világít, akkor még húzni kell a húron a hangolókulcs segítségével, mert alacsonyabb a húr által kiadott hang, mint a referenciamagasság. Ha a jobb oldali LED világít, akkor ereszteni kell a húron. Ha a középső, zöld LED világít, akkor az azt jelenti, hogy 1 cent pontossággal sikerült behangolni az adott húrt.

A megtervezett szűrőegyütthatók száma alapvetően meghatározza a szükséges számítási igényt, tehát azt, hogy milyen processzort kell majd alkalmaznunk. A felhasznált STM32F103VBT6 egy 32 bites, nagy teljesítményű, alacsony fogyasztású ARM magos mikrovezérlő. Tudásához képest nagyon olcsó. A gitárhangoló megvalósításához nagy előnye, hogy nagy pontosságú belső AD-átalakítóval rendelkezik, így a rendszerterven említett AD-átalakító nem foglal külön helyet a megtervezendő nyomtatott áramkörön. Nagy sebessége, és alacsony fogyasztása szintén nem elhanyagolható tényező, hiszen a real-time szűrésnek nagy a számítási igénye.

A tervezést egy STM32F10B-EVAL fejlesztőkártyán végeztem el. A megvalósításhoz hozzátartozik a szükséges ki-, bemenetek igénye is. Sajnos a fejlesztőkártyán csak négy általános célra felhasználható LED található, amely leszűkítette az általános rendszertervben megfogalmazott igényeket.

A gitárhangoló megvalósítása után oszcilloszkóppal megvizsgálom a timer modul pontosságát. Majd leellenőrzöm, hogy a szűrés belefér-e a két mintavételezés közötti időintervallumba. Ezt a kontroller egyik lábának billegtetésével mérhetjük meg: A timer interrupt után közvetlenül 0-ra állítom a mikrovezérlő egyik lábát, majd a kritikus műveletek elvégzése után visszaállítom 1-be. Így a timer modul időzítésnek megfelelő periodikus impulzusjel kitöltési tényezőjéből következtetni lehet a processzor kihasználtságára. A megvalósított gitárhangoló pontosságát az AP Guitar Tuner számítógépes alkalmazás segítségével ellenőrzöm.

4. Tervezési feladatok

4.1. Hangolás frekvenciaméréssel

A digitális hangoló alapelve, hogy megmérjük az adott húr által kiadott hang alapharmonikus frekvenciáját, és azt összehasonlítjuk a kívánt hang referencia frekvenciájával. Egy klasszikus, akusztikus vagy elektromos gitárt általában az I. táblázatban található hangokra hangolnak.

Gitár húrjai	Alapharmonikus frekvencia
E2	82,4069 Hz
A2	110,0000 Hz
D3	146,8324 Hz
G3	195,9977 Hz
H3	246,9417 Hz
E4	329,6276 Hz

I. táblázat - A gitár húrjainak hangjai és alapharmonikus frekvenciái

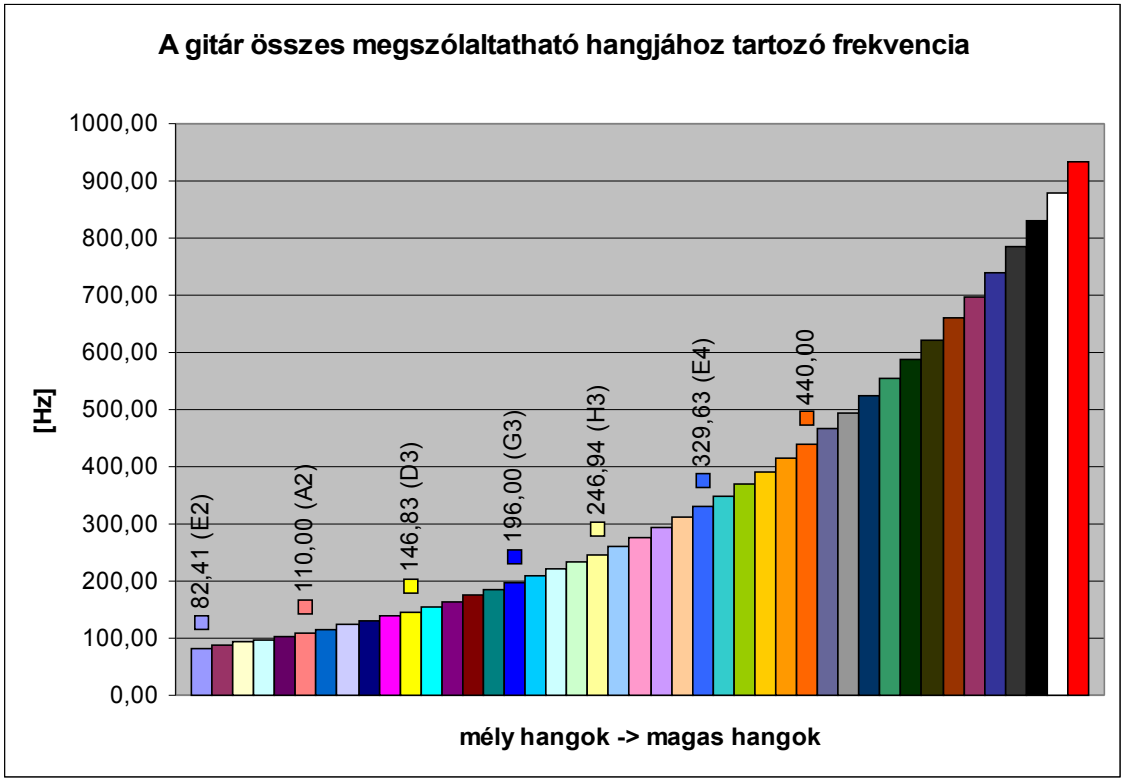
Az I. táblázatban megadott hangok a gitár húrjainak megfelelő legmélyebb hangok, a bundoknál való lefogással már csak magasabb hangokat adhat ki a hangszer. A 4.1. ábrán a gitár által kiadható hangokhoz rendelt alapharmonikus frekvenciákat láthatjuk.

Korábban említettem, hogy a zenében egy fél hang távolság azt jelenti, hogy a két hang alapharmonikus frekvenciájának aránya $1:\sqrt[12]{2}$. A 4.1. ábrán jól látszik, hogy a frekvencia exponenciális függvénye a hangmagasságnak. Ez egyben azt is jelenti, hogy a megfelelő hangolási pontosság elérésének érdekében az alacsonyabb hangok frekvenciáját pontosabban meg kell határozni. A pontosság itt abszolút pontosságot jelent. A relatív pontosság minden hang esetén azonos mértékű. Az exponenciális összefüggésből adódóan az abszolút és relatív pontosság is aszimmetrikus.

A hangmagasság pontossága legyen x, centben megadva. Az abszolút pontosság ekkor:

$$\Delta f_- = f_x \cdot \left(1 - \sqrt[12]{2^{-\frac{x}{100}}}\right), \quad \Delta f_+ = f_x \cdot \left(\sqrt[12]{2^{\frac{x}{100}}} - 1\right).$$

A relatív pontosság az abszolút pontosság osztva a mérendő mennyiséggel, tehát f_x-szel.



4.1. ábra - A gitárral kiadható kromatikus hangsor

Amíg a legmélyebben szóló „E2”-húrnak megfelelő alapharmonikus frekvencia távolsága az alatta lévő félhangétól 4,63 Hz, addig a lefogás nélkül megpendített „E4”-húr 19,60 Hz-re van a fölötte egy félhanggal szóló hang frekvenciájától. A frekvenciamérés pontosságát ezek szerint a legmélyebb hanghoz kell igazítani. A hangolás abszolút pontosságát „cent”-ben szokták megadni: 1 cent egy félhang egy század része, frekvenciában megadva $\sqrt[12]{2^{0,01}}$ szorzót jelent.

4.2. Mintavételezési frekvencia megválasztása

A mikrofon jelét az erősítő fokozat után mintavételezni kell, hogy azon a digitális jelfeldolgozást elvégezhessük. A mintavételi frekvencia megválasztása sok tényezőtől függ, illetve egyéb következményekkel jár.

Egyrészt azt mondhatjuk, hogy a mintavételi törvény szerint a mintavételezett jel spektruma a mintavételi frekvencia szerint periodikus. A spektrum átlapolódását el szeretnénk kerülni, mivel nem szeretnénk, hogy a mérést elrontsa egy hibás frekvenciájú, átlapolódott komponens. Minél nagyobb frekvenciát választunk, annál inkább elkerüljük az átlapolódást.

Másrészt a mintavételi frekvencia nagysága alapvetően befolyásolja a frekvenciamérés pontosságát. Ez abból ered, hogy digitális periódusidő-mérést alkalmazunk. Ennek működésére részletesen ki fogok térni. Elöljáróban csak annyit, hogy minél nagyobb a mintavételi frekvencia, annál pontosabb a periódusidő, illetve a frekvenciamérés.

Harmadrészt azért sem mindegy, hogy mekkorára választjuk a mintavételi frekvenciát, mert annak nagysága a digitális FIR szűrő specifikációjára is nagy hatással van. Adott frekvenciaintervallumokra tervezünk sávszűrőket. Nem mindegy, hogy az adott szűrő mennyi szűrőegyüttható val valósítja meg a specifikációnak megfelelő szűrést. Minél kisebb a mintavételi frekvencia, annál kevesebb együtthatóra van szükség.

A mintavételi frekvencia megválasztásához utolsó szempontként azt említem, hogy a real-time működés miatt minden mintavételezés között el kell végezni a szűrést és a nullpontkeresést. Ilyen szempontból nem mindegy, milyen sűrűn érkeznek a mintavételezett adatok, mert két minta közötti idő alatt adott számú utasítást el kell tudni végezni. Ha ez a sok utasítás nem fér bele két mintavétel közé, hibásan fog működni a megírt program.

4.3. Tervezés előkészítése

A tervezési feladatok megoldásához a következő eszközöket használtam fel:

- gitár,
- számítógép, és annak hangkártyája,
- mikrofon.

A következő szoftvereket használtam:

- MATLAB,
- MS Hangrögzítő,
- AP Guitar Tuner 1.0.

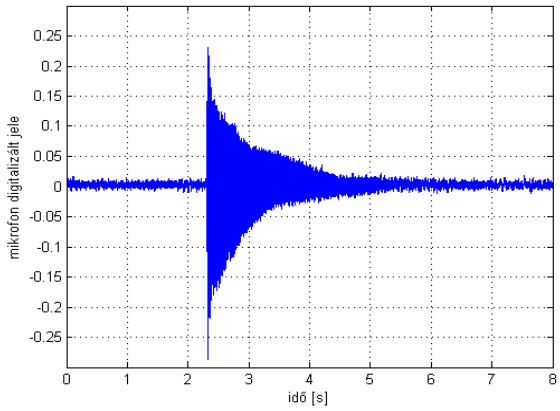
Hangrögzítő segítségével egyenként felvettem a hat húr által kiadott hangokat. A felvételt először 44100 Hz-es wav fájlban tároltam el, később elegendőnek bizonyult a 8 kHz-es formátum is. Ezeket a MATLAB `wavread` függvényével olvastam be. A beolvasott értékeket időtartományban, illetve frekvenciatartományban analizáltam. Ennek megfelelően elkészítettem a szükséges sávszűrőket, illetve megterveztem a gitárhangoló frekvenciaméréséhez szükséges algoritmust.

A gitárom behangolásához referencia hangolásként az AP Guitar Tuner 1.0. nevű PC-n futó programot használtam. Ennek pontosságát a MATLAB-ban nagy pontszámú FFT segítségével ellenőriztem, illetve összehasonlítottam a megvalósított hangoló algoritmussal is. Meglepően jó eredményeket kaptam.

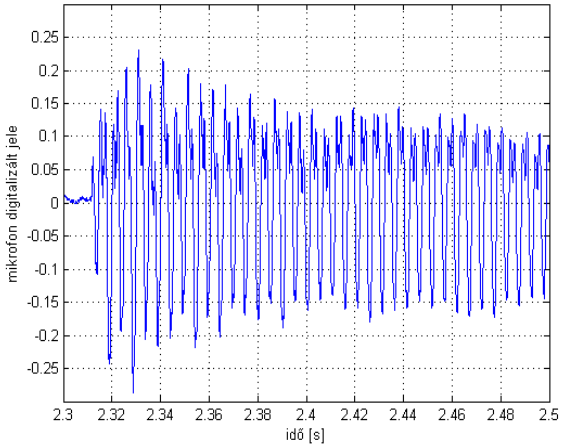
4.4. Jelanalízis

4.4.1. Időtartománybeli analízis

Az időtartománybeli analízis eredményeként azt kaptam, hogy egy kezdeti tranziens után viszonylag szép, periodikus jel következik, amely egy idő után lecseng (4.2-4.3. ábra). A jelenség teljesen érthető: közvetlenül a megpendítés után még érintkezik az ujj és a húr, amely akadályozza a húrt a mozgásában, ez indokolja a kezdeti tranzienst. A hang lecseng, egyre kisebb intenzitással szól, végül teljesen elhalkul. A periodikusság pedig teljesen érthető, hiszen a melodikus hangszerek hangja időtartományban periodikus.



4.2. ábra - G3 húr által kiadott hang időbeli lecsengése



4.3. ábra - G3 által kiadott hang kezdeti tranziense

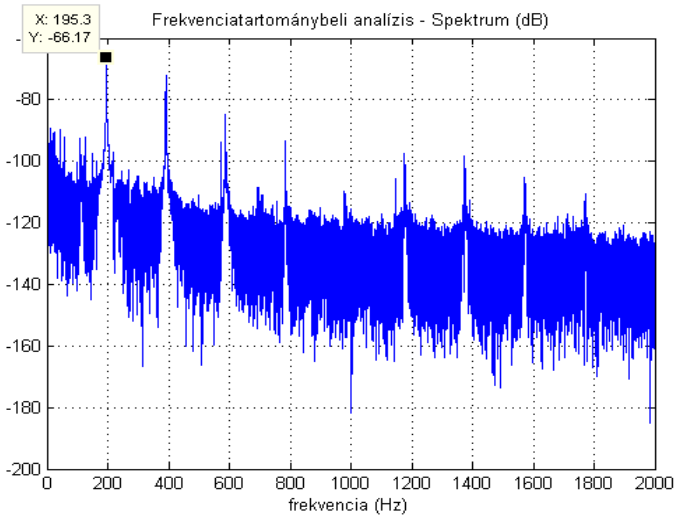
A kezdeti tranziens és a periodikusság nagyon különböző volt a hat húr esetén. Ez egyrészt azzal magyarázható, hogy a gitárom mély hangot kiadó három húrja fém bevonatú damil, a többi pedig damil húr. Másrészt az egyik húr megpendítésekor a többi húr berezonálhat, ami így mérési hibát okozhat. Ezzel a jelenséggel a kész hangoló berendezés használatakor is számolnunk kell. Ezért indokolt az olyan sávszűrő használata, amely csak a kívánt alaphangnak megfelelő frekvenciát ereszti át, a nemkívánatos komponenseket elnyomja.

Az időtartománybeli analízist később a tervezett szűrő verifikálására használtam. Mind a hat húrra leellenőrizzem, elegendő-e a szűrő fokszáma, hogy nullpontátmenet figyelésével frekvenciát mérhessünk.

4.4.2. Frekvenciatartománybeli analízis

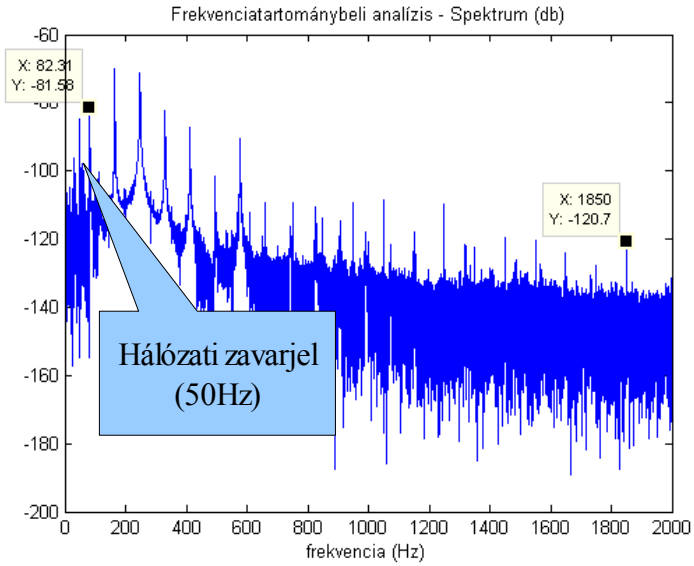
Minden esetben meghatároztam a digitalizált jel spektrumát. Ez egyrészt arra volt jó, hogy a mintavételi frekvencia értékét megfelelően megválaszthassam, illetve hogy nagy pontszámú Fourier-transzformációval leellenőrizzem az AP Guitar Tuner PC-s gitárhangoló alkalmazás pontosságát. A szoftver pontossága megfelelő volt, referencia hangolásként használhatjuk.

A 4.4. ábrán látható a G3-húr által kiadott hang spektruma. A hang periodikussága a vonalas spektrumban is megmutatkozik. A többi hang frekvenciatartománybeli analíziséből következik, hogy a jelek sávkorlátja kb. 2 kHz. Ennek megfelelően a mintavételi frekvenciát – a mintavételi törvény alapján – 4 kHz-nek választottam.



4.4. ábra - G3 húr spektruma

A hálózati 50 Hz minden esetben jelentős zavarjelet eredményezett. Később kiderült, hogy ez leginkább a legmélyebb, E2-húr esetén okozott mérési hibát, mivel annak alap-harmonikus frekvenciája 82,41 Hz, ez pedig nagyon közel van az 50 Hz-hez (4.5. ábra).



4.5. ábra - E2 húr spektruma

A vonalas spektrum elemzésekor megállapítható, hogy E2-húr esetén az alapharmonikus frekvenciának megfelelő komponens intenzitása a felharmonikus komponensekhez képest elég csekély. Ez további mérési hibát eredményezhet, ezért ez esetben a szűrőtervezéskor szigorú specifikációjú sávszűrőt kell tervezni.

4.5. FIR szűrő tervezése

4.5.1. Szűrőválasztás

Az időtartományban periodikus, de viszonylag nagy harmonikus tartalommal rendelkező jelet célszerű szűrni. Amennyiben digitális szűrést alkalmazunk, kétféle megoldás jöhet szóba: végtelen impulzusválaszú, vagy véges impulzusválaszú (IIR vagy FIR) szűrő.

Az IIR szűrő előnye, hogy alacsony fokszámú szűrő elég hatékony szűrést tesz lehetővé, egyszerűen tervezhető és fizikai rendszereket jól modellez. Kézenfekvő lenne ezt a megoldást alkalmazni. Viszont mivel abban a kimenet visszacsatolódik, és a felhasznált processzor véges szóhosszúságú, és véges pontosságú, alkalmazásával hibát halmozhatunk fel: túlcsordulás léphet fel, illetve stabilitási problémáink lehetnek. Ezek mellett színes kvantálási zaj jellemzi, sokszor nagy a paraméter-érzékenység és nemlineáris a fázismenete.

A FIR szűrő ezzel szemben mindig stabil. Annak kimenete nincs visszacsatolva, így a kimenet csak a bemenet, és annak késleltetett értékeinek függvénye. Nincs túlcsordulás, a kvantálási zaj fehér zaj formájában jelentkezik, kicsi a paraméter-érzékenység és lineáris fázismenet tervezhető. A FIR szűrés hátránya a nagy együtthatószám. Ez egyrészt nagy tárolási kapacitást igényel, másrészt mikrokontrolleres alkalmazás esetében nagy számítási igénnyel kell számolnunk. [5]

4.5.2. Működés

A digitális szűrőket a következő diszkrét átviteli függvénnyel írhatjuk le:

$$H(z)=\frac{B(z)}{A(z)},$$

ahol a *B(z)* és *A(z)* polinomok. Az átviteli függvény *z*-ben racionális törtfüggvény, a mintavételi frekvencia (f_s) szerint periodikus. Az átviteli függvény alapján felírható a rendszeregyenlet:

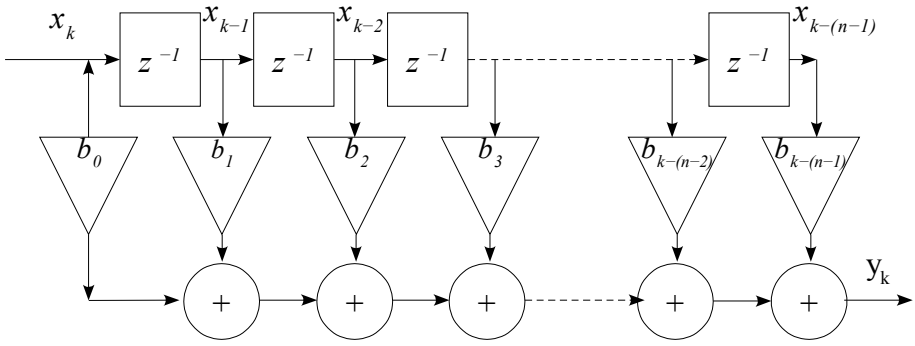
$$y(k)=\sum_{i=0}^{n-1} b_i \cdot x(k-i)-\sum_{i=0}^{m-1} a_i \cdot x(k-i),$$

ahol *x(k)* és *y(k)* a gerjesztés és a válasz mintái a *k.* időpillanatban, az a_i és a b_i konstansok, és rendre az *A(z)* és a *B(z)* együtthatói.

FIR szűrők esetén a rendszeregyenlet e következőképpen egyszerűsödik:

$$y(k)=\sum_{i=0}^{n-1} b_i \cdot x(k-i). \qquad (4.1)$$

A FIR szűrő elvi működése a 4.6. ábrán látható.



4.6. ábra - FIR szűrő

Az ábrából, illetve a (4.1)-ből is jól látszik, hogy a FIR szűrő impulzusválasza megegyezik a szűrő együtthatókészletével.

4.5.3. FIR szűrő tervezése MATLAB segítségével

A FIR szűrő tervezéséhez a MATLAB `fir1(n,[w1 w2])` függvényét használtam fel, amely ablakozásos módszerrel készíti el a kívánt szűrőt. A paraméterek a következő jelentéssel bírnak: az `n` a terveett fokszámnál eggyel kisebb szám, a `[w1 w2]` interval-lummal adhatjuk meg a sávszűrő alsó és felső határfrekvenciáját, méghozzá úgy, hogy a mintavételi frekvencia feléhez viszonyítjuk, tehát az adott határfrekvenciát elosztjuk a mintavételi frekvencia felével, és ezt adjuk meg a függvénynek. A `fir1()` MATLAB függvény kimenete egy vektor, amely a tervezett FIR szűrő együtthatóit tartalmazza. A szűrést a `filter()` függvénnyel végezhetjük el.

Az egyes húroknak megfelelő frekvenciatartományú sávszűrőket terveztem. A szűrők a kívánt frekvencia környékén 1-et erősítenek. Az áteresztő tartományokat először úgy definiáltam, hogy a szomszédos húrok alapharmonikus frekvenciái között a hangmagasság szerint feleztem a tartományt, és az annak megfelelő frekvenciát adtam meg határfrekvenciának. (II. táblázat)

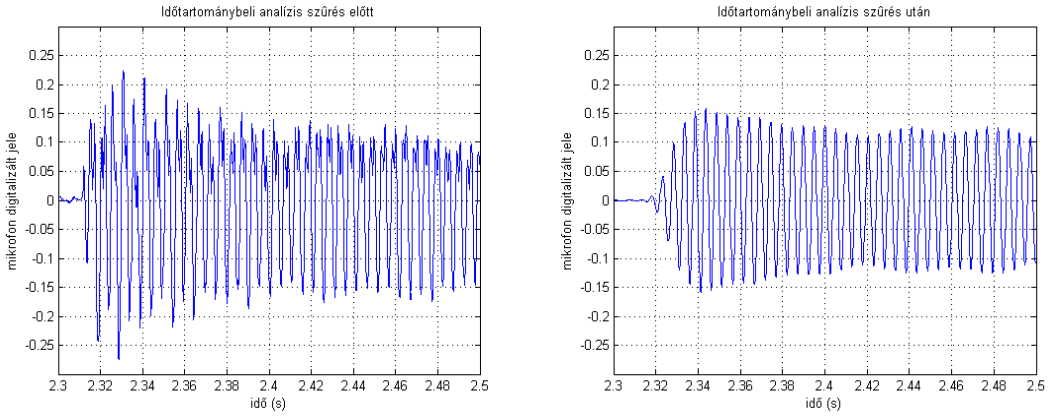
húr	$f_{alsó}$ [Hz]	$f_{felső}$ [Hz]	$f_{alsó}'$ [Hz]	$f_{felső}'$ [Hz]
E2	72,33	95,21	72	94
A2	95,21	127,09	94	126
D3	127,09	169,64	126	167
G3	169,64	200,00	172	292
H3	200,00	285,30	210	292
E3	285,30	380,34	284	379

II. táblázat - Frekvenciaértékek a sávszűrők tervezéséhez

A tervezés során arra jutottam, hogy minden húr esetén megfelelő frekvenciamérési pontosságot kapunk, ha a szűrő fokszámát 100-nak választjuk. A megtervezett szűrőket a `freqz()` MATLAB függvénnyel ellenőriztem, és korrigáltam a kezdetben megadott határfrekvenciákat, ezeket feltüntettem a II. táblázatban.

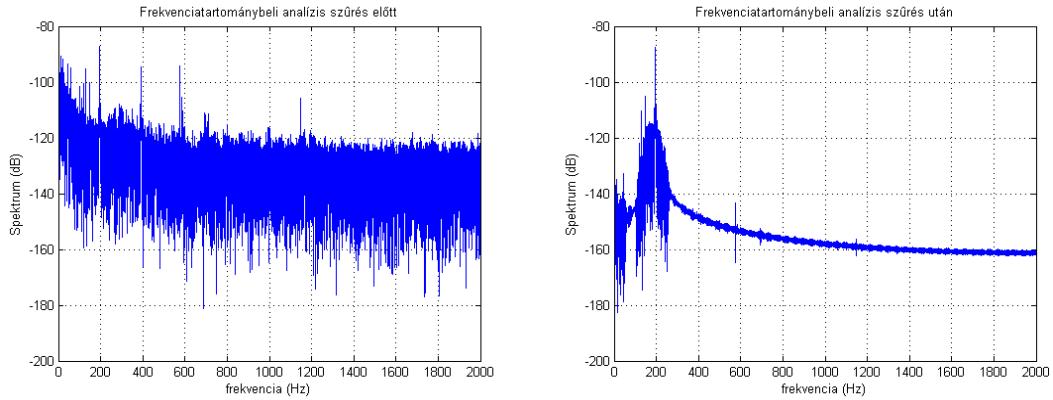
4.5.4. FIR szűrő hatása a idő- és frekvenciatartományban

A jól megválasztott FIR szűrő a 4.7. ábrának megfelelően alakítja a jelet az időtartományban. Jól látszik, hogy a szűrés után eltűnnek a nem kívánatos zavarjelek, és közelítőleg szinuszos jelet kapunk.



4.7. ábra - szűrés hatása az időtartományban

A frekvenciatartományban a 4.8. ábrának megfelelő hatást fejti ki a FIR szűrő. A nagy frekvenciás komponenseket szépen levágja, ellenben az alacsony frekvenciás komponenseket bizonyos mértékben átereszti. Ennek ellenére elmondhatjuk, hogy a szűrő fokszáma – az előbb említett időtartománybeli analízis alapján – megfelelő nagyságú.



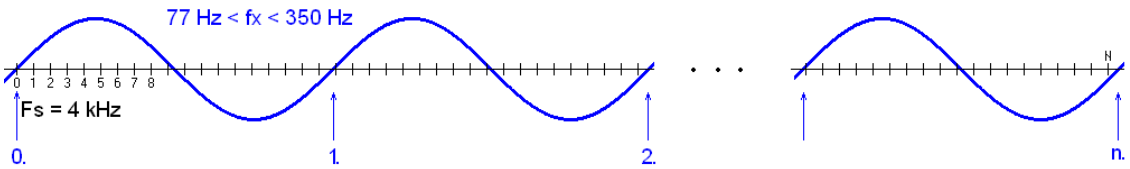
4.8. ábra - szűrés hatása a frekvenciatartományban

4.6. Nullátmenetek számítása

A szűrt jel frekvenciáját a nullátmenetek között eltelt idő alapján mértem. Mivel a mintavételi frekvencia 4 kHz, a mérendő frekvenciák pedig kb. 77–350 Hz tartományban mozognak, ezért a frekvencia méréséhez a digitális periódusidő-mérő elvét használtam fel.

4.6.1. Digitális periódusidő-mérő

A digitális periódusidő-mérés működése a 4.9. ábra segítségével szemléltethető.



4.9. ábra - Digitális periodusidő-mérés szemléltetése

A mérendő jelből n számú periódus alatt megszámoljuk a nagy frekvenciájú órajelek ütésének számát, ezt jelöljük N-nel. A mérendő jel periódusa T_x, ennek reciproka a mérendő frekvencia f_x. Az órajel frekvenciája adott, jelöljük f_s-sel.

Ha megszámoljuk n periódus alatt hány mintánk volt, tehát ismerjük N értékét, kiszámítható a mérendő jel periódusideje és frekvenciája:

$$T_x = \frac{N}{n \cdot f_s}, \quad \text{és}$$
$$f_x = \frac{n \cdot f_s}{N}. \tag{4.2}$$

A periódusidő-mérés hibája a (4.3) képlettel számítható. A mérendő periódusidő reciproka a frekvencia, így a mért frekvencia hibájának nagysága megegyezik a periódusidő mérés hibájával.

$$\left|\frac{\Delta T_x}{T_x}\right| = \frac{\Delta f_s}{f_s} + \frac{f_x}{f_s n} + \frac{1}{\pi n} \cdot \frac{U_{zp}}{U_{xp}} \simeq \left|\frac{\Delta f_x}{f_x}\right|. \tag{4.3}$$

A mérés relatív hibája három komponensből áll össze: az órajel pontosságából, a kvantálási hibából és a triggerhibából. Amennyiben az órajel elég pontos, elhanyagolhatjuk az általa okozott hibát. A triggerhiba jel-zaj viszony és n függvénye. A FIR szűrés után a

jel-zaj viszony jobb lesz, és mivel az *n* értékét a kvantálási hiba miatt nagyra választjuk, a triggerhibát is elhanyagolhatjuk. A kvantálási hiba mindenképpen befolyásolja a mérést. Nem mindegy, hogy hány periódust mérünk meg, mekkorának választjuk az *n*-et. [6]

4.6.2. Pontosság meghatározása

A hangmagasság mérésének pontosságát 1 cent-re szeretnénk beállítani, amely 1% relatív pontosságot jelent. A frekvenciamérés pontosságát ennek megfelelően kell beállítani:

$$\left|\frac{\Delta f_x}{f_x}\right| = \frac{f_x - \frac{f_x}{2^{\frac{0.01}{12}}}}{f_x} = 1 - 2^{-\frac{0.01}{12}} \approx 557{,}5\,\text{ppm} = h_{qmax} \,. \tag{4.4}$$

A kvantálási hiba a (4.2) képlet alapján így számítható:

$$h_q = \frac{1}{N} = \frac{f_x}{n \cdot F_s} \le h_{qmax} \,. \tag{4.5}$$

A kvantálási hiba maximuma 557,5 ppm. A (4.5) képlet átrendezéséből becsülhetjük *n* minimális értékét:

$$n \ge \frac{f_x}{F_s \cdot h_{qmax}} \,.$$

A méréshez szükséges minimális *n* periódusszám megválasztása frekvenciafüggő. Ennek mértékét a III. táblázatban foglaltam össze.

Húrok	f_x	n_{min}
E2	82,41 Hz	36
A2	110,00 Hz	48
D3	146,83 Hz	64
G3	196,00 Hz	85
H3	246,94 Hz	107
E4	329,63 Hz	143

III. táblázat - A periódusszám frekvenciafüggése

A III. táblázat alapján elmondhatjuk, hogy az *n* értékét legalább 143-ra kell választani, hogy a hangmagasság mérését a kívánt pontossággal el tudjuk végezni. Ez egyben azt is jelenti, hogy az alacsonyabb mérendő alapharmonikus frekvencián szóló hangokat pontosabban be tudjuk hangolni. Ez nagyon jó, mert a hangmagasság és a frekvencia közötti logaritmikus összefüggésből adódóan pontosan így kell eljárni (lásd. 21. oldal 4.1. ábra).

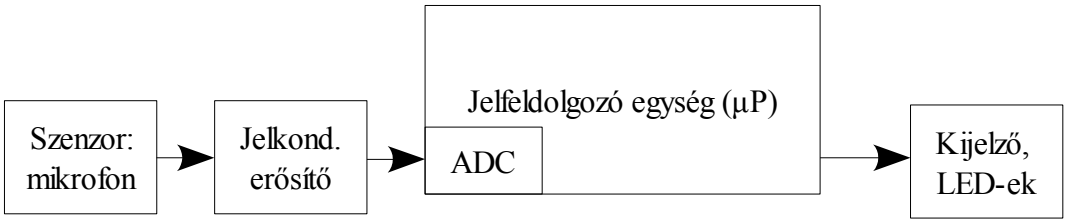
4.7. Tervezési feladatok összesítése

Az analízis során elvégzett a szűrőtervezési, és a frekvenciamérési feladatokat összesítettem egy – a hangolást megvalósító – MATLAB scriptben, amely a mellékelt CD-n megtalálható (`gitarhangolo.m`). A MATLAB script tetszőleges megpendített gitárhúr 4 kHz-es wav formátumú felvételéről képes megállapítani, hogy mennyi annak az alapharmonikus frekvenciája. A tervezési feladatok elvégzése után következhet a megvalósítás.

5. Megvalósítás

5.1. Hardver rendszerterv

A gitárhangoló blokkvázlata a 5.1. ábrán látható. Főbb részei: a szenzor, a jelkondicionáló, a jelfeldolgozó egység és a kijelző. A szenzor egy kondenzátormikrofon, a jelkondicionáló egy erősítő fokozat, a jelfeldolgozó egység egy processzor, amely egyrészt elvégzi a szűrést, valamint a frekvenciamérést és vezérli a kijelző egységet.



5.1. ábra - Digitális gitárhangoló blokkvázlata

Célom, hogy a jelfeldolgozó processzor a jelfeldolgozást és a kijelzést real-time módon valósítsa meg, hiszen a frekvenciamérés bizonyos esetekben idő-, illetve amplitúdófüggő. Továbbá, ha a beérkezett nagy mennyiségű adatot rögtön feldolgozzuk, nem kell egy egész felvételt eltárolnunk a memóriában. Helyette az egyes mintavételi időpontok között végzi el a processzor az összes szükséges utasítást. Ehhez elegendő egy cirkuláris buffer implementálása, amely egyszerre csak a szűrő fokszámának megfelelő adatmennyiséget tárol el a processzor memóriájában. A szoftveres megvalósításra később külön kitérek, most nézzük meg közelebbről a hardvert.

5.2. Hardver áttekintés

A digitális gitárhangolót a 5.2. ábrán látható, STMicroelectonics által gyártott fejlesztőkártyán valósítottam meg. Eltértem a feladatkiírásban szereplő STM32 processzort tartalmazó IAR KickStart Kit fejlesztőkártyától, mivel a tervezés korai szakaszában tönkrement a kártyán található mikrovezérlő. Az STM által gyártott fejlesztőkártya kontrollere szintén STM32-es processzor. Annyi a fő különbség, hogy az IAR kártyán

64, az STM kártyán pedig 100 lábszámú STM32-es mikrovezérlő található. Ez azt is jelenti, hogy az STM fejlesztőkörnyezet több modullal rendelkezik, illetve egy időben több modul használható rajta.

Felhasznált eszközök:

- Hardver: STM32F10B-EVAL evaluation board (5.2. ábra),
- Fejlesztő szoftver: IAR Embedded Workbench with a 32KB edition of IAR C/C++ Compiler,
- Programozó: IAR J-LINK JTAG debugger.

5.2. ábra - STM32F10B-EVAL evaluation board

A fejlesztőkártya a gyártó szerint tipikus beágyazott feladatok elvégzésére alkalmas eszköz. Ehhez elegendő ki-, bemeneti interfésszel, illetve beépített áramkörrel rendelkezik. A kártya tervezésre, fejlesztésre, integrálásra és tesztelésre használható. A korszerű fejlesztőkörnyezet lelke egy STM32F103VBT6 típusú mikrovezérlő egység.

5.2.1. STM32 processzor

Az STM32 egy 32 bites mikrovezérlő család, melynek tagjai a korszerű ARM Cortex-M3 mag köré épülnek. Ez egy szabványosított mikroprocesszor mag, amely egy 32 bites processzort, egy speciális busz architektúrát, fejlett megszakításkezelő egységet, debug rendszert és szabványos memória layoutot tartalmaz. A Cortex-M3 mag Harvard architektúrájú, tehát külön program-, és adatbusszal rendelkezik, és tipikus RISC processzor, tehát egy utasítást kb. egy gépi ciklus alatt végez el, pipeline műveletvégzésre képes. Az ARM processzorokat alacsony fogyasztás jellemzi. Ezek a processzorok beágyazott rendszerek körében igen népszerűek. A Cortex processzorok az ARMv7 architektúrán alapulnak. A Cortex-M3 az alacsony költségű feladatok elvégzésére alkalmas processzor mag, amely az ARM Ltd. által fejlesztett Thumb-2 egyszerűsített utasításkészletettel is rendelkezik. Az ARM Ltd. csak fejlesztéssel foglalkozik, mikrovezérlőket nem gyárt. A fejlesztett architektúrák licencjogát adja el nagyobb gyártó cégeknek. Így készült az STMicroelectronics által gyártott STM32 mikrovezérlő család. A Cortex-M3 mag adottságai: 72 MHz-es működés, szorzás egy gépi ciklus alatt, hardver osztás, fejlett megszakításkezelés (43 maszkolható megszakítási csatornával).

Az STM32 mikrovezérlők kihasználják az ARM mag architekturális adottságait: A Thumb-2 egyszerűsített 32 bites utasításkészlettel jobb teljesítmény és kódsűrűség érhető el, kiemelkedően gyorsan reagálnak a megszakításokra és nagyon alacsony fogyasztás jellemzi őket. Az STM32 család egy teljes 32-bites sorozatot kínál, amely kombinálja a nagy teljesítményű, real-time, alacsony fogyasztású és alacsony feszültségű működést, továbbá teljes integráltságú, illetve egyszerű fejlesztési lehetőséget nyújt a fejlesztők számára. Egy konkrét mikrovezérlőnél a felhasznált a lábkiosztás, a perifériák és a szoftver kompatibilis az STM32 család minden tagjával. Az STM32 család különböző tagjai lábszámban, FLASH és RAM méretben különböznek egymástól. [7]

Az STM32 korszerű mikrovezérlő család, melynek tudása mellett az ára is igen kedvező. Tudása és ára miatt nyugodtan kijelenthetjük, hogy felveszi a versenyt a hagyományos 8 bites kontrollerekkel. Az STM fejlesztőkártyán található STM32F103VBT6 típusú kontroller árait tartalmazza a IV. táblázat.

Darabszám	Egységár (USD)	Össz ár (USD)
1	15,86000	15,86
10	14,04300	140,43
100	11,32500	1132,50
250	10,41900	2604,75
500	9,51300	4756,50
1000	7,24800	7248,00
2500	6,56850	16421,25

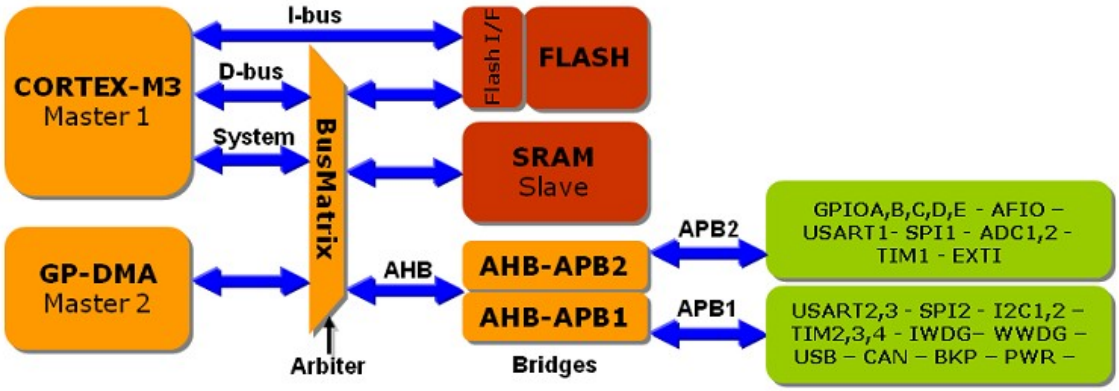
IV. táblázat: STM32F103VBT6 árai [1]

A megvalósításhoz elegendő egy kisebb lábszámú, és kevesebb modullal rendelkező mikrovezérlőt választani, amely így még olcsóbb lehet (egy darab rendelése esetén is található 10 USD .alatti kontroller).

A mikrovezérlő tápfeszültsége 2,0-3,6 V között kell, hogy legyen, a Cortex-M3 mag tápfeszültsége 1,8 V. A mikrovezérlő belső reset-áramkörrel rendelkezik, melynek minimum felszültsége 2,0 V, 40 mV hiszterézissel. Belső RC oszcillátort tartalmaz, amelyet a beleintegrált PLL-lel legfeljebb 72 MHz-re szorozhat fel. A megfelelő időzítési pontosság elérésének érdekében külső oszcillátort kell alkalmazni. 128 KB FLASH memóriával, és 20 KB SRAM-mal rendelkezik. 2 db 12 bites szukcesszív approximációs AD-átalakító található benne, melyek 1 MHz sebességűek. 7 csatornás DMA vezérlő található benne. Soros portról debuggolható, valamit JTAG interfésszel is rendelkezik. 80 általános célú I/O lába van, melyek közül tetszőlegesen maximum 16 megszakítást okozhat, és 5 V-ot is kibírnak. 7 időzítő egységet (timert) tartalmaz, illetve 9 kommunikációs interfész található rajta: 2 I^2C interfész, 3 USART, 2 SPI, illetve egy CAN és egy USB interfész.STM32 architektúrája

A 5.8. ábrán látható a STM32-es processzorok architektúrája. A Cortex-M3 mag köré bonyolult buszrendszeren keresztül csatlakoznak a különböző modulok.

1 Forrás: http://www.digikey.com/



5.3. ábra - STM32 belső architektúrája

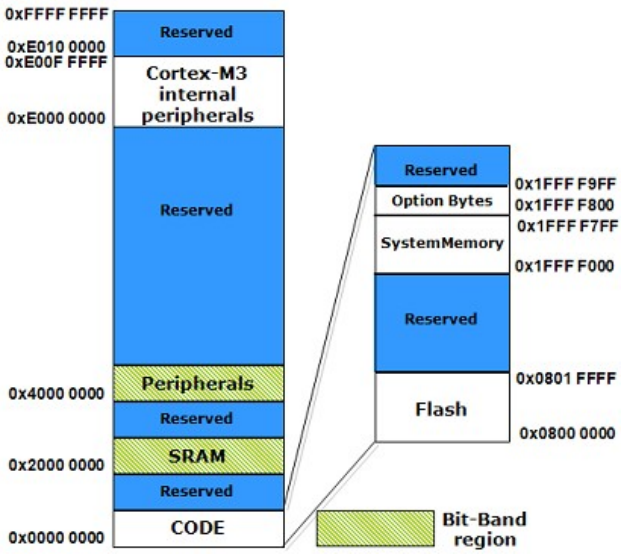
A fő buszrendszer, melynek vezérlését a BusMatrix segíti, maximum 72 MHz sebességgel működik. Ezen keresztül éri el a Cortex-M3 processzor a Flash memóriát, az SRAM-ot, a további buszokat vezérlő hidakat, valamint a DMA vezérlő egységet. Az egyéb modulokat, perifériákat az AHB buszon keresztül érhetjük el, melynek sebessége ugyanúgy maximum 72 MHz. A perifériabuszok közül az APB1 legfeljebb 36 MHz sebességű lehet, az APB2 sebessége max. 72 MHz. Az APB2-n keresztül érhetők el az STM32 mikrovezérlő általános célú lábai, illetve egyéb nagy sebességű perifériák (pl. ADC, TIM1 modul). Az APB2 perifériabuszon keresztül érhetjük el pl. az USB- és a CAN-vezérlő áramkört stb.

Cortex-M3 processzor és a DMA vezérlő áramkör lehet a buszon master. Arbitráció csak akkor van, amikor egyszerre használnák az SRAM-ot, az APB1-et vagy az APB2-t. A busz arbitrációs egysége 2/3 utasítás-végrehajtási időt garantál a DMA-nak, és 1/3-ot a processzor magnak.[8]

5.2.2. STM32 címkezelése

A 32 bites cím 4 GB területet képes direkt címzéssel megcímezni. A 5.4. ábrán látható a processzor címkezelése. A 0x00000000 címen található a programmemória, a FLASH. A chipen található belső SRAM a 0x20000000 címen kezdődik, amely bitcímezhető terület. A Boot Mode lábak beállításától függően RAM-ból is futtathatjuk a megírt prog-

ramot. A 0x40000000 címtől kezdődően helyezkednek el a periféria címek. Ez szintén bitcímezhető terület. A 0xE0000000 területtől kezdődően találhatók a Cortex-M3 processzor belső perifériái.



5.4. ábra - címkezelés

A programmemória három részre osztható: a felhasználói FLASH-re, rendszer memóriára és beállító bájtokat tartalmazó területre. A rendszermemória bootloader segítségével az USART1-en keresztül programozható.

5.2.3. Órajelek

Amennyiben a beépített RC oszcillátornál pontosabban szeretnénk előállítani az órajelet, külső oszcillátort kell csatlakoztatni a mikrovezérlőhöz. A fontosabb külső oszcillátor a HSE (nagy sebességű külső oszcillátor), melynek frekvenciája maximum 25 MHz lehet. Ez az órajel látja el a Cortex-M3 processzor és a perifériák szükséges órajeleit. A másik külső oszcillátor, az LSE (kis sebességű külső oszcillátor), amely a real-time órajel és a watchdog timer időalapját adja. Ajánlott sebessége: 32,768 kHz. A bonyolult buszrendszer miatt minden egyes perifériának külön engedélyezni kell a megfelelő órajellel vezérelt buszt.

5.2.4. AD-átalakító

A beépített AD-átalakítók 12 bites szukcesszív approximációs elven működnek. 18 csatornát képes multiplexelten mintavételezni: két belső forrást, illetve 16 külsőt. Az AD-átalakítás lehet: egyszeri, folyamatos, ellenőrzött és szakaszos.

Főbb tulajdonságai:

- 12 bites eredmény
- megszakításgenerálás egy vagy több átalakítás után
- egyszeri, vagy folyamatos átalakítás
- önkalibrálás
- csatornánként programozható mintavételi idő
- maximum 1 MHz sebesség
- két AD-átalakító együttes használata

5.3. Fejlesztőkártya áttekintés

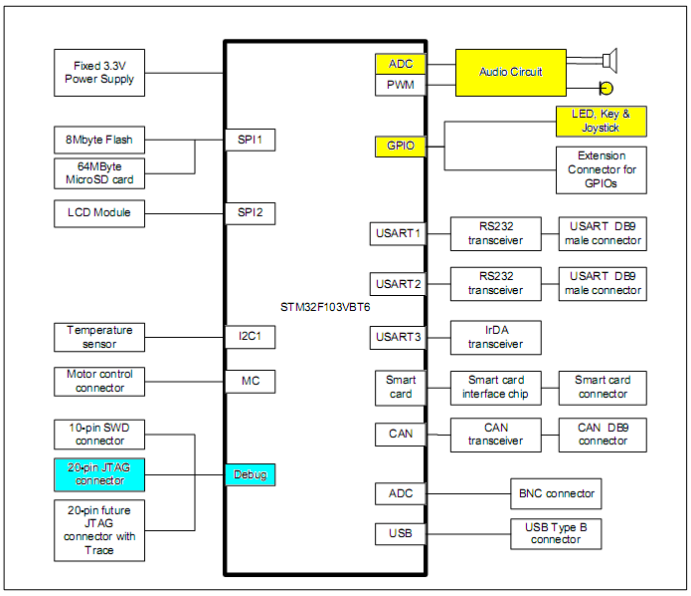
A fejlesztőkártyán a STM32F103VBT6 típusú mikrovezérlő található. Ennek fő tualjdonságai: 72MHz sebesség,128K FLASH, 20K SRAM memória, beépített UART, ADC, USB, CAN modul. Tokozása: LQFP100, -40-től +85 °C-ig garantáltan működik, és gyártástechnológiailag megfelel az RoHS szabályozásnak, amely az elektromos és elektronikus berendezésekben használatos ólom, és más potenciálisan veszélyes anyagok korlátozására vonatkozik.

A fejlsztőkártya fő tulajdonsságai: [9]

- Három 5 V-os táp ellátási lehetőség: stabilizált DC-táp, USB csatlakozó vagy kiegészítő kártya 5 V-ja
- Bootolás FLASH-ből, tesztelés FLASH-ben vagy SRAM-ban
- Audio lejátszás és felvételi lehetőség
- 64 MB-os MicroSD kártya
- Smartcard támogatás (Type A és Type B)
- I^2C/SMBus kompatibilis soros interfészű hőmérő szenzor

- Két RS232 kommunikációs csatorna (az egyik handshake jeleket is támogat)
- IrDA adó egység
- USB 2.0 (full speed)
- CAN 2.0A/B csatlakozó
- csatlakozó indukciós motor vezérléshez
- JTAG, SWD
- 240x320 színes TFT LCD
- négy irányú joystick
- Reset, wakeup, tamper and user push buttons
- 4 LED
- Real Time Clock
- tüskesor kiegészítőkártya csatlakoztatásához

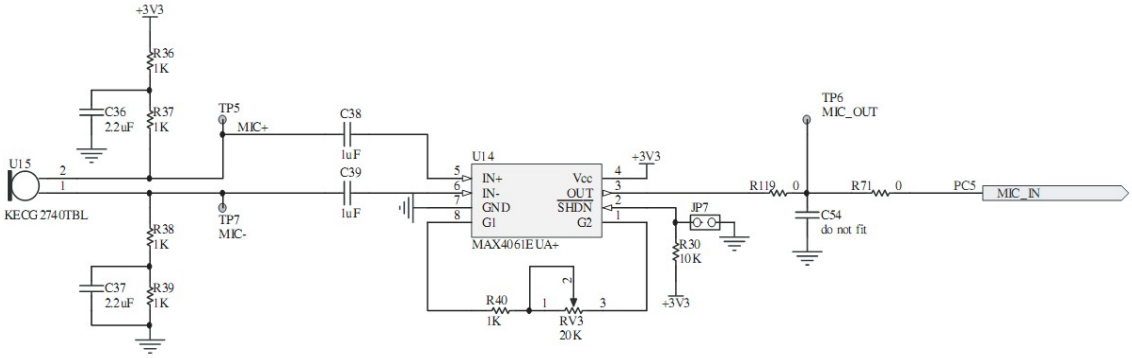
A fejlesztőkártya blokkdiagramja az 5.5. ábrán látható. A fejlesztőkártyán a következő modulokat használtam. A processzoron belül a TIM1 modult és a GPIO modult, a processzoron kívül az audió erősítőt, és a kondenzátormikrofont. Mindezek mellett szükség volt tápellátásra, amelyet egy 5 V-os stabilizált táppal oldottam meg. A kártya programozását az IAR J-LINK JTAG debugger végeztem el.



5.5. ábra - fejlesztőkártya blokkdiagramja

5.3.1. Mikrofon erősítő fokozat

Az STM32 fejlesztőkártyán realizált mikrofonerősítő fokozat látható az 5.6. ábrán.



5.6. ábra - Mikrofon erősítő fokozat

A hangjelet egy kondenzátormikrofon alakítja át elektromos jellé. A kondenzátormikrofon működésének alapelve, hogy az egy a hang rezgését felvevő, mozgó membrán okozta kapacitásváltozást alakítja át váltakozó feszültségű jellé. Ezt a jelet egy kiszajú erősítő fokozatra vezetik. A fejlesztőkártyán egy speciális mikrofonerősítő IC-t helyeztek el (MAX406). Az IC kimenete egy aluláteresztő szűrőn keresztül kötötték a mikrovezérlő lábához. [9]

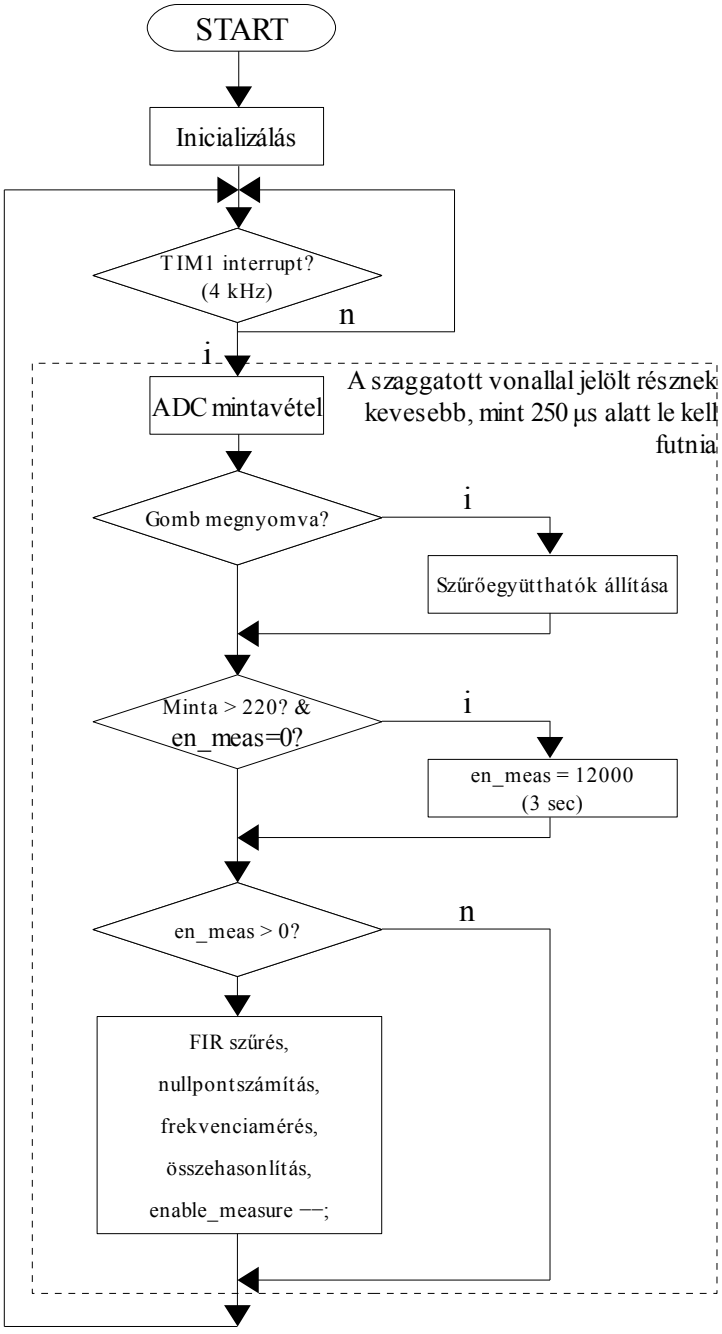
5.4. Szoftver

Az STM32 fejlesztőkártya fejlesztőkörnyezete az IAR Embedded Worckbench IDE nevű program, mellyel C, illetve C++ nyelven programozhatjuk az eszközt. A fejlesztés mellett ezzel a programmal végezhetjük el a programletöltést a kártyára, amely történhet RAM-ba, illetve FLASH-be is. A letöltést a J-Link nevű eszköz végzi, amely egy USB-JTAG Debugger, tehát az egyik végén USB-n keresztül a számítógéppel kommunikál, a másik végén pedig a JTAG-en keresztül csatlakozik a kártyához.

A gyártó jóvoltából a programozás alapjainak elsajátításához számos példaprogram rendelkezésünkre áll. Magas szintű szintaktikára utalnak a rendelkezésre álló gyári függvények. Mind a függvények, mind a példaprogramok nagyon jól kommentezettek. A megvalósítást az IAR Worckbench fejlesztőkörnyezethez mellékelt plédaprogram segítette.

5.4.1. Szoftver terv

A digitális gitárhangolónak real-time működést kell megvalósítania. Ez szoftver rendszerterv esetén ennyit jelent, hogy az AD-átalakító két mintavételezése között el kell tudni végezni az összes szükséges jelfeldolgozási műveletet, tehát a szűrést, a frekvenciamérést. Az 5.7 ábrán látható a szoftver felépítése.



5.7. ábra - Folyamatábra

A szűrőegyütthatókat sorfolytonosan egy statikus tömbben tároltam. Egy nyomógomb segítségével lehet beállítani, hogy melyik húrt akarjuk hangolni. Ez alapvetően meghatározza, hogy melyik húrnak megfelelő szűrőegyütthatókat használja a program, illetve milyen frekvenciatartományban vizsgálja az eszköz, hogy jó-e a hangolás.

A szükséges időzítést timer modul (TIM1) használatával valósítom meg, ezzel állítom be viszonylag pontosan a 4 kHz mintavételi időt. A processzorba integrált AD-átalakítót nagy sebességű, autonóm mintavételezésre állítom, viszont az AD-átalakító eredmény regiszteréből csak 4 kHz-enként olvassa be a program az értéket. Erre azért van szükség, mert az AD-átalakító mintavételezési frekvenciáját nem lehet elegendően alacsonyra állítani. Az AD regiszter kiolvasása után korrigálom a nullpontot hibát. Amennyiben a mintavételezett érték bizonyos amplitúdó fölé kerül, 3 másodpercig engedélyezem a jelfeldolgozást: a szűrést, és a frekvenciamérést és az összehasonlítást.

A főprogram elején el kell végezni egy-két inicializációs lépést. Fel kell éleszteni a buszokat, be kell állítani az órajeleket, inicializálni kell az I/O lábakat, be kell állítani a timer interruptot, illetve le kell kezelni azt. A `while(1)` ciklusban a 4 kHz-re állított timer interruptban bebillentett flag segítségével futtatunk minden további programot.

5.4.2. Órajelek beállítása

Minden buszt inicializálnunk kell, meg kell adnunk mindegyiknek az órajelét. A következő pár sorban bemutatom az órajelek inicializálását.

Először a rendszer órajelének a belső 8 MHz-es RC oszcillátort választjuk (1.):

```
// 1. Clocking the controller from internal HSI
RCClockingClocking the controller from internal HSI RC
// (8 MHz)
RCC_HSICmd(ENABLE);
// wait until the HSI is ready
while(RCC_GetFlagStatus(RCC_FLAG_HSIRDY) == RESET);
RCC_SYSCLKConfig(RCC_SYSCLKSource_HSI); // 8 MHz
```

Ezek után engedélyezzük a külső 8 MHz-es oszcillátort (2.), a belső PLL-lel felszorozzuk 72 MHz-re (3.). Később ez lesz a rendszer órajele . Ennek megfelelően állítjuk be a két perifériabuszt: PCLK2-t 8 MHz-re, a PCLK1-et 1 MHz-re. Ezek után az USB órajelét 48 MHz-nek, az AD-átalakító sebességét pedig 1 MHz-re (4.) választjuk.

```
// 2. Enable ext. high frequency OSC (8 MHz)
RCC_HSEConfig(RCC_HSE_ON);
// wait until the HSE is ready
while(RCC_GetFlagStatus(RCC_FLAG_HSERDY) == RESET);
// 3. Init PLL
RCC_PLLConfig(RCC_PLLSource_HSE_Div1,RCC_PLLMul_9); // 72MHz
RCC_PLLCmd(ENABLE);
// wait until the PLL is ready
while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET);
// 4. Set system clock dividers
RCC_PCLK2Config(RCC_HCLK_Div1);   // 8 MHz
RCC_PCLK1Config(RCC_HCLK_Div2);   // 4 MHz
RCC_USBCLKConfig(RCC_USBCLKSource_PLLCLK_1Div5); // 48 MHz
RCC_ADCCLKConfig(RCC_PCLK2_Div8); // 1 MHz

RCC_HCLKConfig(RCC_SYSCLK_Div1);  // 8 MHz
```

Ezután inicializáljuk a FLASH késleltetést:

```
#ifdef EMB_FLASH
 // 5. Init Embedded Flash
 // Zero wait state, if 0 < HCLK 24 MHz
 // One wait state, if 24 MHz < HCLK 56 MHz
 // Two wait states, if 56 MHz < HCLK 72 MHz
 // Flash wait state
 FLASH_SetLatency(FLASH_Latency_2);
 // Half cycle access
 FLASH_HalfCycleAccessCmd(FLASH_HalfCycleAccess_Disable);
 // Prefetch buffer
 FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);
#endif // EMB_FLASH
```

Mindezek után már csak azt kell beállítani, hogy a rendszer órajele a PLL által felszorzott 72 MHz legyen.

```
// 6. Clock system from PLL
RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);
```

5.4.3. Timer inicializálás

A timer modult önmagában is inicializálni kell, viszont emellett az interrupt vezérlést is meg kell oldani (NVIC), még akkor is, ha ez esetben nem túl bonyolult interrupt rendszerről van szó. Az inicializáláshoz definiálnunk kell két struktúrát. Ennek paraméterei határozzák a timer modul, illetve az NVIC működését.

```
TIM1_TimeBaseInitTypeDef TIM1_TimeBaseInitStruct;
NVIC_InitTypeDef NVIC_InitStructure;
```

Elsőként engedélyezzük a modulnak az APB2 buszt, és letiltjuk az APB1-et. A timerhez tartozó struktúra paraméterei alapján elmondhatjuk, hogy a timer modul órajelén 72-szeres előosztást végeztünk; beállítjuk, hogy a timer feléfelé számláljon; az interrupt függvény meghívási periódusa 250; a fő órajel pedig a 72 MHz legyen. A számláló kezdeti értékét 0-ra állítottuk. Ezek után meghívható a TIM1_TimeBaseInit() függvény.

```
// Timer1 Init
// Enable Timer1 clock and release reset
RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM1,ENABLE);
RCC_APB1PeriphResetCmd(RCC_APB2Periph_TIM1,ENABLE);
RCC_APB1PeriphResetCmd(RCC_APB2Periph_TIM1,DISABLE);

// Set timer period 0.2 sec
TIM1_TimeBaseInitStruct.TIM1_Prescaler = 71;  // 1us resolution
TIM1_TimeBaseInitStruct.TIM1_CounterMode = TIM1_CounterMode_Up;
TIM1_TimeBaseInitStruct.TIM1_Period = 249;  // 4kHz
TIM1_TimeBaseInitStruct.TIM1_ClockDivision = TIM1_CKD_DIV1;
TIM1_TimeBaseInitStruct.TIM1_RepetitionCounter = 0;
TIM1_TimeBaseInit(&TIM1_TimeBaseInitStruct);
```

Ezek után fel kell konfigurálni a timer modult.

```
// Clear update interrupt bit
TIM1_ClearITPendingBit(TIM1_FLAG_Update);
// Enable update interrupt
TIM1_ITConfig(TIM1_FLAG_Update,ENABLE);
```

Az interruptokat kezelő modul (NVIC) belállításához tartozó struktúra elemeit a következőképp kell beállítani: Az interrupt csatornájának meg kell adni a timer modul interrupt csatornáját. A prioritását tetszőlegesen megválaszthatjuk, mert csak a timer modul lehet interrupt forrás. Végül engedélyezni kell az interruptokat. Ezek után meghívhatjuk az NVIC_Init() függvényt.

```
NVIC_InitStructure.NVIC_IRQChannel = TIM1_UP_IRQChannel;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);
```

Fontos, hogy a timer modult mint perifériát engedélyezni kell.

```
// Enable timer counting
TIM1_Cmd(ENABLE);
```

5.4.4. Timer interrupt kezelő függvény

A timer modul és az NVIC inicializálása után meg kell írni az interrupt kezelő függvényt. Ebben törlöm az interrupt flaget, illetve bebillentek egy változót, melyet utána a főprogramban figyelek.

```
void Timer0IntrHandler (void)
{
  // Clear update interrupt bit
  TIM1_ClearITPendingBit(TIM1_FLAG_Update);
  Update = TRUE;
}
```

A főprogram vételen ciklusában figyelem az `Update` változót. Ha timer interrupt érkezett, törlöm az `Update` flaget. majd kiolvasom az AD-átalakító értékét. Ezek után végzem el a többi szükséges műveltet.

```
while(1)
{
  //4000Hz
  if(Update)
  {
    Update = FALSE;

    //sampling
    Sample = (Int16S)(ADC_GetConversionValue(ADC1) - 1883);

    /*(...)*/

  }
}
```

5.4.5. Általános célú I/O lábak inicializálása

Először definiálni kell egy struktúrát, amely az előre megadott függvény GPIO_Init() függvény paramétere lesz. A struktúra egyes elemei a láb kívánt sebességére, módjára és számára vonatkoznak. Az adott láb sebességét 10, 20 vagy 50 MHz-re állíthatjuk be. Módjai a következők lehetnek: ki-, illetve bemenet, pull-up vagy pull-down ellenállás, analóg vagy digitális stb. A GPIO_Init() függvény másik paramétere a port neve.

A következő kódrészletben látszik, hogyan definiáltam a az egyes ki-, bemeneteket:

```
GPIO_InitTypeDef GPIO_InitStructure;

// Configure PC.06...09 as output
```

```
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7 | GPIO_Pin_8
| GPIO_Pin_9;
GPIO_Init(GPIOC, &GPIO_InitStructure);

// Configure PB.09 as input (button)
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
GPIO_Init(GPIOB, &GPIO_InitStructure);

// Configure PC.05 as input - analog input (ADC_in)
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
GPIO_InitStructure.GPIO_Speed = (GPIOSpeed_TypeDef)0;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
GPIO_Init (GPIOC, &GPIO_InitStructure);
```

Mindezek után engedélyezni kell a megfelelő priférіabuszt:

```
//GPIOB and GPIOC clock enable
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
```

A fenti kódrészlettel állítottam be az általános célú ki-, bemeneteket úgy, hogy az a gitárhangoló rendszertervének megfeleljen. Definiáltam egy analóg bemenetet az AD-átalakítónak, egy bemenetet a a nyomógombnak, illetve négy kimenetet a LED-ek vezérléséhez.

5.4.6. AD-átalakító inicializálása

Az AD-átalakító inicializálásához ismét definiálnunk kellett egy struktúrát. Ez a struktúra, illetve az két ADC közül az egyik az ADC_Init() függvény paramétere. A struktúra a következő fontos beállításokat tartalmazza: a mintavételezés módját, folyamatos mintavételezést, külső interrupt forrás beállítását, a12 bites adat igazítását a 16 bites eredmény regiszterben, illetve a csatornák számát.

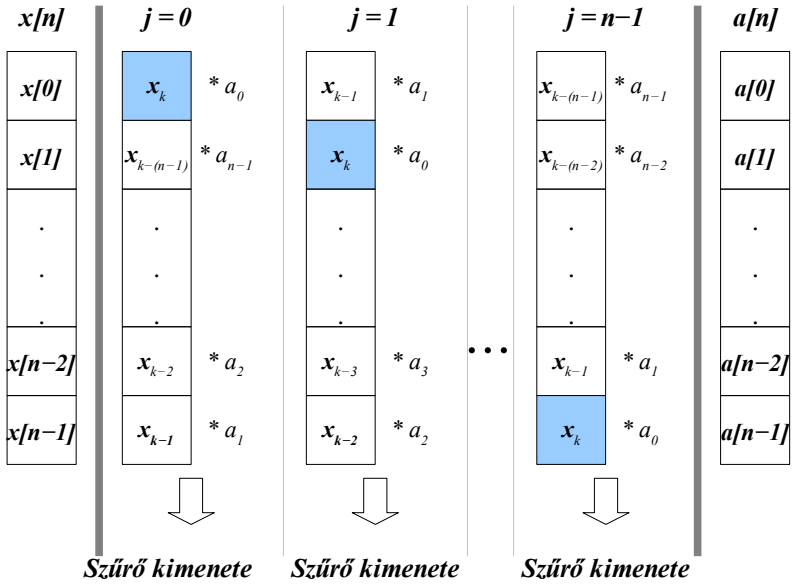
```
// ADC Structure Initialization
ADC_StructInit(&ADC_InitStructure);

ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
ADC_InitStructure.ADC_ScanConvMode = DISABLE;
ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
ADC_InitStructure.ADC_ExternalTrigConv =
                        ADC_ExternalTrigConv_None;
ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
ADC_InitStructure.ADC_NbrOfChannel = 1;
ADC_Init(ADC1, &ADC_InitStructure);
```

```
// Enable the ADC
ADC_Cmd(ADC1, ENABLE);

// ADC calibration
ADC_StartCalibration(ADC1);
while(ADC_GetCalibrationStatus(ADC1) == SET);

// Configure channel
ADC_RegularChannelConfig(ADC1, ADC_Channel_15, 1,
                         ADC_SampleTime_55Cycles5);

// Start the conversion
ADC_SoftwareStartConvCmd(ADC1, ENABLE);
```

Ezek után engedélyezhetjük az AD-átalakítást, kalibrálhatjuk az átalakítót, majd be kell állítani a mikrovezérlő megfelelő csatornáját. Végül szoftveresen engedélyezzük az AD-átalakítót. Nem szabad megfeledkezni az órajel beállításáról sem, viszont ezt az órajelek inicializálásakor már megtettük.

5.4.7. FIR szűrés

A FIR szűrés elve a 4.5.2. fejezet 4.6. ábrán látható. Az ott ábrázolt késleltető láncnak megfeleltethetünk egy cirkuláris buffert, az `x[n]`-et. A késleltetést egy cirkuláris buffer segítségével valósíthatjuk meg. Először feltöltjük a buffert, majd az új beérkezett mintát mindig a legrégebbi minta helyére írjuk. Egy `n` hosszú buffer megfelel egy `n-1` hosszú késleltető láncnak. A szűrést a 5.8. ábrán szemléltettem. A szűrőegyütthatókat az `a[n]` tömbben tárolhatjuk.



5.8. ábra - FIR szűrés szoftveres megvalósítása

A szűrés az 5.8. ábra alapján így programozható le:

```
// Sampling and make in the "x" cicular buffer
x[j] = Sample;

// FIR filtering
sum = 0;
for(i=0; i<n; i++)
{
      sum += x[(i+j)%n] * a[string+(n-i)%n];
}

// Increment buffer index for "x"
j++;
j %= n;
```

Az új mintavételezett értéket mindig az `x[j]`-ben tároljuk el. A szűrő kimenete a 4.5.2 fejezet 4.1. képlete alapján számítható. A programban ezt egy `for()` ciklussal valósítottam meg. A leírt szorzó képlet annyiban más, hogy a bejövő jelek fix tömbben vannak eltárolva, így azokat a `j`. ütem szerint korrigálni kell, az `a[]` tömbben pedig az adott húrnak megfelelő helyen kell lennünk, ebben segít a `string` változó. A `for()` ciklus végén a `sum` változó tartalmazza a szűrt értéket. Ha végeztünk a szűréssel, `j` értékét növelni kell, hogy a következő mintavételezés után megfelelően legyenek indexelve a tömbök. A tömbök indexeit minden esetben modulo `n` szerint kell figyelembe venni, nehogy kifussunk a definiált statikus tömbökből.

Ez a folyamat minden 4 kHz-nek megfelelő periódusban lefut, és mivel a fix, 4 kHz-es bejövő adatokkal számolunk, a sum kimenet szintén 4 kHz-enkénti jeleket fog eredményezni.

A szűrőegyüttható k számát 100-nak, a mintavételezett értékek és a szűrőegyütthatók formátumát a fejlesztőkörnyezetben előre definiált Int16S-nek választottam. A formátum elnevezése magától értetődő: 16 bites előjeles egész szám. A szűrő kimenete így biztosan bele fog férni a `sum` változóba, mely egy 32 bites előjeles egész szám (Int32S).

5.4.8. Frekvenciamérés

Adott egy `counter` változó, mely minden 4 kHz-es periódusban eggyel növekszik. Ha nullátmenet van, kiolvasom ezen számláló értékét, majd lenullázom. A kiolvasott értéket eltárolom egy 300 elemű cirkuláris bufferben. A buffer értékeit összeadom, és

megkapom a 300 periódus digitális értékét. Az előbb kapott összegből meghatározható a jel frekvenciája. Ez elvileg a 4.6.2. fejezetben tárgyalt digitális periódusidő-mérő pontosságágának megfelelő frekvencia, illetve hangmagasságmérést eredményez.

A `buffer1, buffer2` változók a nullpontkeresést segítik. A FIR szűrés utolsó két eredményének előjelét veszik csak figyelembe, ezek segítségével könnyen mérhetjük periódusszámot. A minták 4 kHz-enként követik egymást, így a `counter` számláló a mintavételi frekvenciának megfelelő gyakorisággal változik.

A periódusidő összeadását nem kell minden egyes mintavétel után elvégezni, hiszen elég a felülírt, régi értéket levonni a SumPeriod változóból, majd hozzáadni az újat.

```
buffer2 = buffer1;
buffer1 = (sum > 0) ? 0: 1;

// Measure frequency
// low to high edge
if( (buffer2 == 1) && (buffer1 == 0) )
{
  OldPeriod = PeriodBuf[PeriodBuf_Index];
  NewPeriod = counter;
  PeriodBuf[PeriodBuf_Index] = NewPeriod;
  PeriodBuf_Index++;
  PeriodBuf_Index %= N;
  SumPeriod = ((SumPeriod - OldPeriod) + NewPeriod);

  counter = 0;
}
counter++;

Frequency = (4000.0/SumPeriod)*N ;
```

Az átlag prióduszidőt a következő képlet alapján kapjuk:

$$T_{átl} = \frac{\frac{1}{f_s} \cdot SumPeriod}{N}$$

Ennek reciproka adja a mért frekvenciát:

$$f_x = \frac{N \cdot f_s}{SumPeriod}$$

5.4.9. Kijelzés

Ha a nyomógomb segítségével beállítottuk a megfelelő húrt, a kapott frekvencia, és a határértékek alapján kijelezhetjük az eredményeket. A négy LED közül egyen jelezzük, hogy épp engedélyezve van-e a jelfeldolgozás. A másik három LED-en pedig azt, hogy a húr még alacsonyabb hangot ad ki a megfelelő hangmagasságnál, hangolása pontos, vagy túl meghúztuk az adott húrt, és magasabb a kiadott hang a kívántnál.

```
if(enable_measure < 10000)
{
  Too_Low  = (Frequency < Low_Frequency ) ? 1: 0;
  Too_High = (Frequency > High_Frequency) ? 1: 0;
  Its_Good = ((Low_Frequency<=Frequency) &&
              (Frequency<=High_Frequency)) ? 1: 0;
}
```

Az első sor azt jelenti, hogy kijelzés vezérlését csak fél másodperccel a jelfeldolgozás kezdete után engedélyezem, ezzel elkerüljük a kezdeti tranziens okozta esetleges mérési hibáját.

Az adott LED jelzőbitjének megfelelően minden egyes timer interrupt után kigyújtjuk, illetve eloltjuk a LED-eket,:

```
// LED: too high
GPIO_WriteBit(GPIOC,GPIO_Pin_6 ,(Too_High)?Bit_SET:Bit_RESET);
// LED: it's good
GPIO_WriteBit(GPIOC,GPIO_Pin_7 ,(Its_Good)?Bit_SET:Bit_RESET);
// LED: too low
GPIO_WriteBit(GPIOC,GPIO_Pin_8 ,(Too_Low )?Bit_SET:Bit_RESET);
```

5.4.10. Állapotváltás gombnyomás hatására

Amennyiben a timer interrupt befutása után gombnyomást észlelünk, belekerülünk egy ciklusba, amely addig fut, míg el nem engedjük a gombot. A gomb nyomásra szoftveresen pergésmentesített, elengedése után pedig állapotot váltunk, ami annyit jelent, hogy a `string` változót megnöveljük 100-zal, és ha az eléri a 600-at, kinullázzuk. Ez arra jó, hogy az `a[]` tömbből kiválasszuk a megfelelő szűrőegyütthatókat, mivel a hat húrnak megfelelőeket egymás után helyeztük el abban.

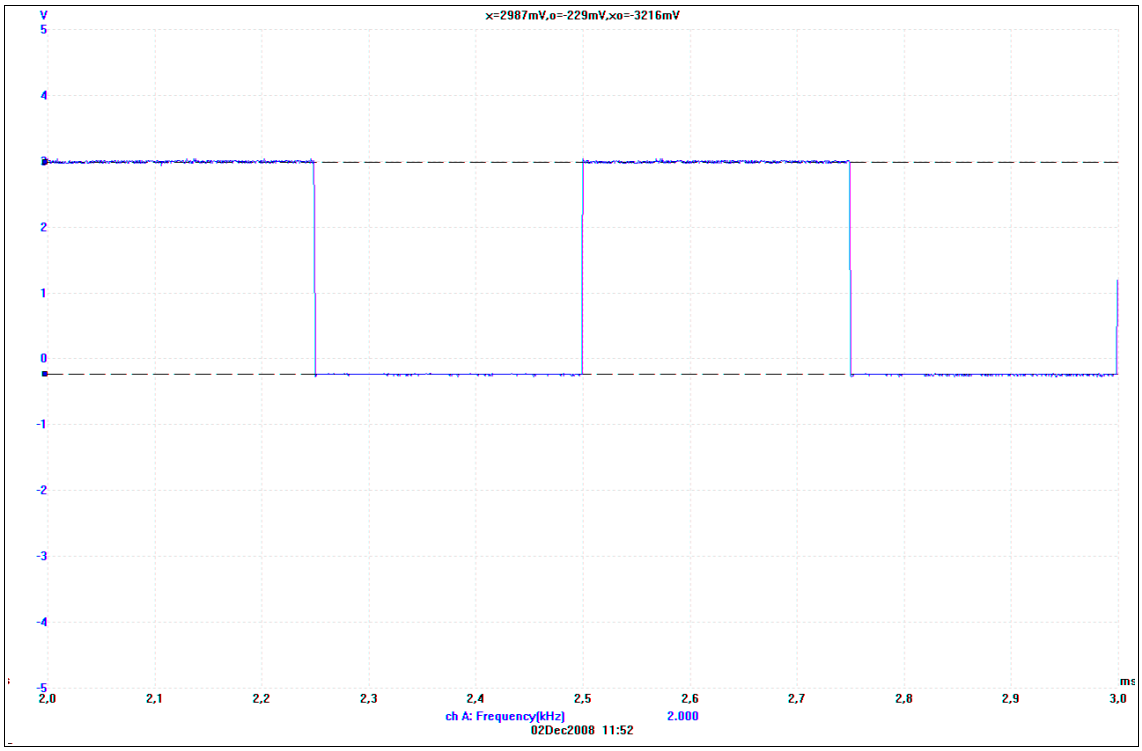
```
// button is pushed
if(GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_9)==0x00)
{
  enable_measure = 0;
  Dly100us((void*)500);
  if(GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_9)==0x00)
  {
    GPIO_WriteBit(GPIOC,GPIO_Pin_6,Bit_RESET);
    GPIO_WriteBit(GPIOC,GPIO_Pin_7,Bit_RESET);
    GPIO_WriteBit(GPIOC,GPIO_Pin_8,Bit_RESET);
    GPIO_WriteBit(GPIOC,GPIO_Pin_9,Bit_RESET);

    // next string
    string += 100;
    if(string==600) string = 0;

    // wait until release button
    while(GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_9)==0x00);
  }
}
```

6. Eredmények

A megtervezett gitárhangoló programot a RAM-ban futtattam, így megfelelően működött. A FLASH-ben nem futott rendesen a program, mivel a FLASH hozzáférés lassúsága miatt a real-time működést nem sikerült megvalósítanom. A szoftver működését debuggolással és oszcilloszkóppal vizsgáltam. A hangoló működését egy gitár behangolásával teszteltem, majd annak pontosságát az AP Guitar Tuner PC-s alkalmazással ellenőriztem. A frekvenciamérés pontosságát debuggolás közben többször leellenőriztem, és a kívánt eredményt kaptam. A futtatáshoz szükséges kód megtalálható a mellékelt CD-n.

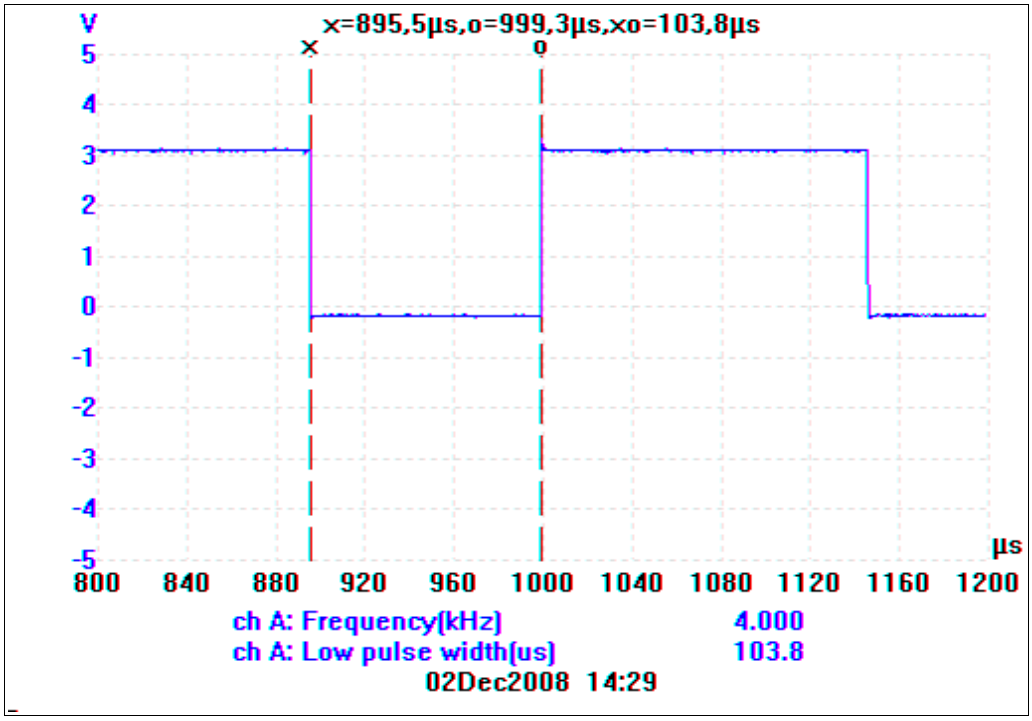
6.1. Szoftver működésének vizsgálata

A timer modul pontosságát oszcilloszkóppal vizsgáltam. Az interrupt kezelő függvényt kiegészítettem egy paranccsal, így abban az egyik LED-et negáltam, villogtattam. Ezzel fele akkora sebességet kaptam, mint a timer interrupt befutási ideje. A PicoScope-pal mért eredmény a 6.1. ábrán látható. A kapott jel 2 kHz-es 50%-os kitöltéi tényezőjű négyszögjel.



6.1. ábra - timer interrupt beütésének vizsgálata

A következő mérés arról szólt, hogy leellenőriztem, hogy a real-time működés megvalósul-e, tehát a timer interrupt hatására lefutó programrésznek van-e elég ideje két interrupt befutása közben. Ebből következtetni lehet a processzor kihasználtságára. A mérés eredménye a 6.2. ábrán látható.



6.2. ábra - processzor kihasználtsága

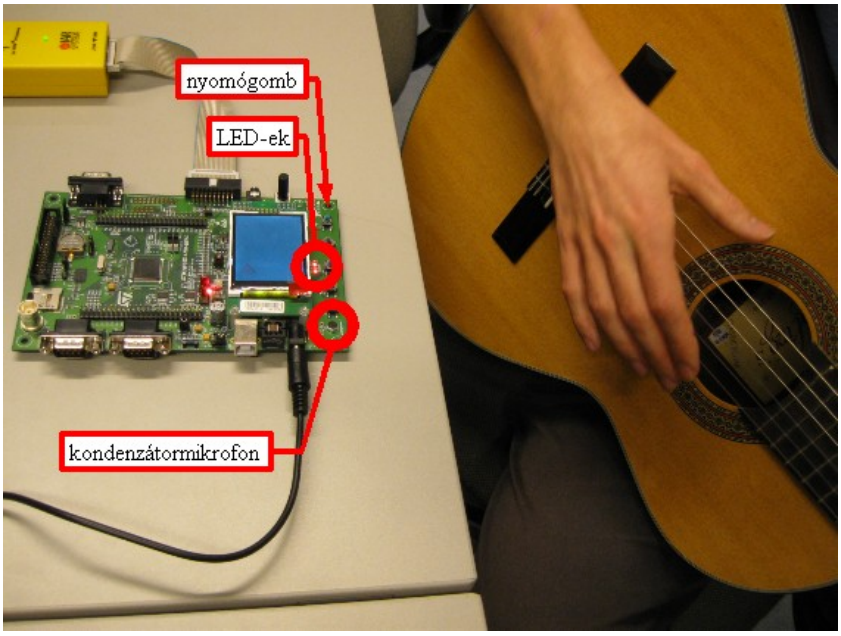
A 6.2. ábrán jól látható, hogy a timer interrupt frekvenciája 4 kHz. Ennek reciproka annak befutási periódusideje, amely 250 µs. A processzor kihasználtsága ezek szerint:

$$\frac{103{,}8\,\mu\mathrm{s}}{250\,\mu\mathrm{s}} = 41{,}52\,\%.$$

Mindezek alapján elmondhatjukm hogy a 72 MHz sebességgel működő processzor hozzávetőlegesen 7500 gépi ciklust hajt végre két timer interrupt befutása közben.

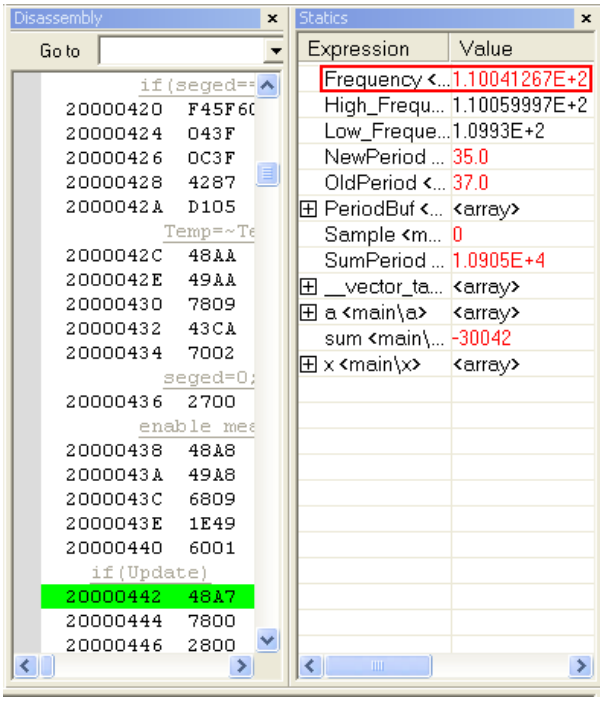
6.2. Tervezett hangoló értékelése

A mérési elrendezés a 6.3. ábrán látható. A tervezett gitárhangolót a következő módszerrel teszteltem: Rátöltöttem a programot az STM32-es fejlesztőkártyára majd futtattam. A nyomógomb segítségével beállítottam a megfelelő húrt. A fejlesztőkártya mikrofonját és a gitár hanglyukját egymáshoz közel helyeztem el. A nyomógombbal beállítottam a mérni kívánt húrt, amelyet aztán megpendítettem Így megtudtam az adott húrról, hogy az általa kiadott hang túl magas, vagy túl mély. Addig tekergettem finoman az adott hangolókulcsot, míg a megfelelő magasságot jelző LED ki nem gyulladt. A tesztelés során azt tapasztaltam, hogy a gitár által kiadott hang magassága időben változik, így viszonylag nehéz 1 cent pontossággal beállítani a húrok által kiadott hang hangmagasságát. Ha a megpendítés közben a három hangolást segítő LED közül a középső, és valamelyik szélső váltakozva villogott, befejeztem a hangolást.

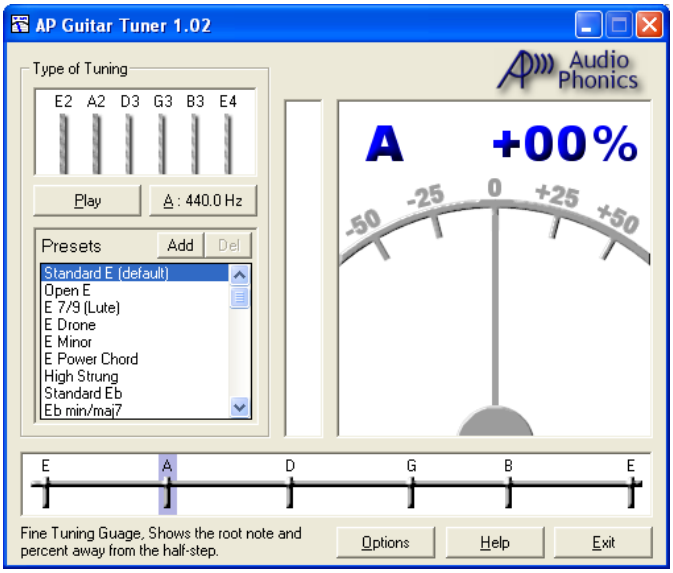


6.3. ábra - mérési elrendezés

A behangolt húrok pontosságát az AP Guitar Tuner PC-s alkalmazás segítségével ellenőriztem. A kapott eredmények elfogadhatóak, két húr esetén kaptam kissé eltérő értéket, tehát a specifikáción lehet még finomítani. Ez azt jelenti, hogy a kívánt 1 cent hangolási pontosság nem érhető el teljes mértékig. A mért frekvenciáját debuggolással határoztam meg (6.4. ábra) és a kapott eredményt összevetettem az AP Guitar Tuner program által jelzett pontossággal (6.5. ábra).



6.4. ábra - A gitárhangoló program által mért frekvencia



6.5. ábra - AP Guitar Tuner által mért pontosság

7. Összefoglalás

A szakdolgozat készítése során megoldottam a feladatkiírásban szereplő feladatokat. MATLAB-ban elvégeztem a gitárhang szükséges analízisét, illetve megterveztem a hangoló megvalósításához szükséges mérési eljárást. A megvalósítás kezdetén megismerkedtem egy korszerű beágyazott rendszerek fejlesztésére alkalmas fejlesztőkártyával. Megírtam rá a hangoló programot, amely real-time működést valósít meg. Végül több méréssel teszteltem a megvalósított eszközt.

Az STM32 fejlesztőkártyán megvalósított gitárhangoló működésének alapelve röviden így foglalható össze: A kondenzátormikrofon a hang rezgését elektromos jellé, feszültséggé alakítja. Ez a feszültség egy erősítő fokozaton keresztül csatlakozik az STM32-es mikrokontroller egyik AD-lábához. A mikrovezérlő 12 bites AD-átalakítója 4 kHz-cel mintavételezi a jelet. A beérkező mintavételezett jelet egy cirkuláris bufferben tároljuk el, melynek nagysága megegyezik a FIR szűrő fokszámával. A processzor két mintavételezés között mindig elvégezi a következő műveleteket: a FIR szűrő kimenetének számítását, a periódusidő-mérésen alapuló frekvenciamérést, az összehasonlításokat és a kijelző segédváltozók állítását. Az előbb említett folyamat a tényleges megvalósítástól annyiban különbözik, hogy a jelfeldolgozást – tehát a szűrést, a frekvenciamérést, a komparálást, illetve a kijelző bitek állítását – valójában csak akkor végezzük el, ha az AD-átalakító értéke egy adott szintet meghalad. Az eddig leírt folyamat csak egy húr hangolására alkalmas, tovább kell lépnünk. A többi húr hangolását egy nyomógomb segítségével állíthatjuk be. A gomb megnyomása után húrváltás következik be, amely azt jelenti, hogy a processzor másik szűrőegyütthatókat használ a FIR szűréshez, illetve a hangmagasság értékelésekor más komparálási szintekkel dolgozik. A kijelzést LED-ek segítségével valósítottam meg. A kijelző biteknek megfelelően minden mintavételezés után kigyulladnak, illetve elalszanak a LED-ek. A kijelzés fő funkcióját három LED végzi. Ezek közül a baloldali jelzi, hogy alacsonyabb, a jobboldali, hogy magasabb, a középső pedig azt, hogy megfelelő a mért alapharmonikus frekvencia, tehát a hangmagasság. A tervezett digitális gitárhangoló abszolút hangolást tesz lehetővé.

A gitárhangoló a tervezett módon méri a frekvenciát. A frekvenciamérés pontossága nem minden esetben érte el a kívánt 1 cent hangolási pontosságot. Továbbfejlesztési lehetőségként először ezt említeném meg. Második továbbfejlesztési lehetőség a nyomógomb elhagyása az eszközből. Ez jelfeldolgozási szempontból azt jelenti, hogy a real-time hangoló a megpendítést követően a jelfeldolgozó eszköz ismerje fel, hogy a felhasználó melyik húrt pendítette meg. A szakdolgozat folytatása lehet a megtervezett gitárhangoló gyártási terve, illetve legyártása.

Irodalomjegyzék

[1] Tony Bacon, *Nagy gitárkönyv*, Budapest: Panamex, 1998

[2] Muszty Bea, Dobai András, *Gitáriskola*, Budapest: Múzsák Kiadó, 1990

[3] Dr. Budó Ágoston, Dr. Pócza Jenő, *Kísérleti fizika I.*, *III. rész - Rezgések és hullámok; hangtan*, Budapest: Tankönyvkiadó, 1962, pp. 331-336

[4] BME Méréstechnika és Információs Rendszerek Tanszék, Digitális Jelfeldolgozás Laboratórium, 2008. június, [Online] Elérhető: http://dsp.mit.bme.hu, [Hivatkozva: 2008.12.06.]

[5] Simonyi Ernő, *Digitális szűrők, A digitális jelfeldolgozás alapjai*, 1984

[6] Zoltán István, *Méréstechnika*, *5. A frekvencia és az idő mérése*, Budapest: Műegyetemi Kiadó, 1997, 143-153

[7] STMicroelectronics, STM32 (CORTEX M3) - 32-bit Microcontrollers, 2008, [Online] Elérhető: http://www.st.com/mcu/inchtml-pages-stm32.html, [Hivatkozva: 2008.12.05.]

[8] HITEX, *The Insider's Guide To The STM32 ARM Based Microcontroller*, 2008

[9] STMicroelectronics, *STM3210B-EVAL evaluation board*, 2005

Függelék

A MATLAB-ban megírt gitárhangoló script, illetve készített IAR projekt megtalálható a mellékelt CD-n. Az F.3. Függelékben olvasható a megtrvezett eszköz használati utasítása.

F.1. Magyarázat a CD-n mellékelt Matlab kódhoz

A CD gyökérkönyvtárában található a `gitarhangolo.m` MATLAB script. A program tetszőleges gitárhang 8 kHz-es wav formátumú felvételéről képes megállapítani, hogy mennyi az alapharmonikus frekvenciája.

A scriptben meg kell adni a wav fájl elérési útját, illetve azt, hogy melyik húrról készült a felévtel:

```
(...)

[y1,Fs,bits] = wavread('<elérési út>');

(...)

char = '<húr>'; % e2 * e2 * d3 * g3 * e4

(...)
```

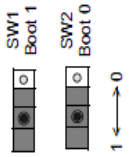
A szükséges változtatások után futtatható a program. A MATLAB Command Window részében fog megjelenni a felvételből számított alapharmonikus frekvencia értéke.

Megjegyzés: A CD gyökérkönyvtárában lévő „hangolás” nevű mappában található néhány 8 kHz-es wav formárumú felvétel.

F.2. STM3210B-EVAL evaluation board programozása

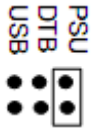
Végezze el a következő lépéseket!

1. Állítsa be a bootolási kapcsolókat úgy, hogy az SRAM-ból bootoljunk! (F.1. ábra)



F.1. ábra - Bootolás

2. Le kell ellenőrizni, hogy a tápfeszültséget beállító jumperek jól vannak-e beállítva (F.2. ábra).



F.2. ábra - Táp jumper beállítása

   Utána csatlakoztathatjuk az 5 V-os stabilizált tápfeszültséget.

3. Indítsa el az IAR Embedded Workbench nevű programot, majd nyissa meg a következő CD-n található projektet.
4. Nyissa meg a main.c fájlt!
5. Csatlakoztassa a J-Link Debuggert az eszközhöz, illetve a számítógéphez!
6. A Workspace ablakban állítsa a legördülő menüt a „RAM Debug” feliratra!
7. Kattintson a menüsoron a Projekt > Make menüpontra!
8. Kattintson a menüsoron a Projekt > Debug menüpontra!
9. Indítsa el a programot, a Go ikonra kattintva!

Ezek után működik a gitárhangoló program. A program megállítható, így kiolvasható a hangoló által mért frekvencia értéke.

F.3. Rövid használati utasítás

Az F.2. Függelék segítségével töltse rá az STM3210B-EVAL fejlesztőkártyára a gitárhangoló programot! Indítás után az E2 húrt hangolhatja. Amennyiben másik húrt szeretne hangolni, nyomja meg a B3-as jelzésű General Pourpose Key nevű nyomógombot. Az elengedése után megjelenik a húrnak megfelelő kód (lásd. V. táblázat), majd miután kigyulladt, és elaludt az összes LED, az A2 húr hangolható. A nyomógombbal ugyanígy válthatunk a további húrok között.

húr	**LED9**	**LED8**	**LED7**	**LED6**
E2	világít			
A2		világít		
D3			világít	
G3				világít
H3	világít	világít		
E4			világít	világít

V. táblázat - Húroknak megfelelő felvillanó kód

A hangolási folyamat úgy zajlik, hogy a gitár hanglyukját a kondenzátormikrofonhoz közel kell helyezni. Ezután pengesse meg a hangolandó húrt! Ha a pengetés hangereje megfelelő volt, a készüléken található LED9 villogni kezd, a maradék három LED-en pedig a hangolás állapota látható. Ha a középső LED világít, akkor a hangolás megfelelő volt, ha a baloldali, akkor még húzni kell a húron, ha a jobboldali, engedni. Mivel a hangmagasság időben váltakozhat, megfelelő hangolási eredményt érünk el, ha a középső és az egyik szélső LED váltakozva villan fel.