Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar

Aktív zajcsökkentést demonstráló rendszer felhasználóbarát kezelői felülettel

Diplomaterv

Csofcsics Donát

2008

Nyilatkozat

Alulírott, ***Csofcsics Donát***, a Budapesti Műszaki és Gazdaságtudományi Egyetem hallgatója kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, és a diplomatervben 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.

……

Csofcsics Donát

Tartalomjegyzék

Kivonat

Az aktív zajcsökkentés napjaink érdekes és izgalmas témái közé tartozik, aminek demonstrálása igen látványos megoldásokhoz vezet. A Méréstechnika és Információs Rendszerek tanszék kitűzött célja az volt, hogy megvalósítson egy könnyen kezelhető, mobilis bemutató rendszert, amelyet a laboratóriumi mérések során is alkalmazni lehet. Ez 2007-ben egy diplomamunka keretében készült el. Az akusztikus teret egy T alakú műanyag plexi cső alkotja, amely két egymással szembefordított, zaj illetve beavatkozó hangszórót tartalmaz. A digitális jelfeldolgozást és a zajcsökkentő algoritmus futtatását az Analog Devices ADSP-BF537 EZ-KIT Lite fejlesztőkártyáján található fixpontos Blackfin processzor valósítja meg. Az így alapul szolgáló rendszer kezelői és vezérlő felületének finomítása, valamint a labortól való függetlenedés növelésének megoldása további fejlesztést igényelt.

A diplomamunkám során megismertetem az alapvető aktív zajcsökkentő problémákat, eljárásokat és az alapul szolgáló demonstrációs rendszert. Bemutatom a fent említett elvárások megoldására tervezett, általános felhasználási célú kijelzős vezérlő egységet és a megvalósításához szükséges megfontolásokat, lépéseket.

A megépített stand-alone rendszer több beviteli lehetőséget biztosít a felhasználó számára, amelynek hatása a kijelzőn jelenítődik meg. A kijelzős vezérlőegység központi egysége az ATMEL cég által gyártott ATmega16 tipusú 8 bites mikrokontroller. A megjelenítést egy 4x20-as alfanumerikus háttérvilágítással rendelkező LCD kijelző valósítja meg. A nyomtatott áramkör kialakítása olyan, hogy könnyen csatlakoztatható a DSP kártya UART kivezetéséhez, ezáltal lehetőség van a vezérlő parancsok soros továbbítására, megvalósítva az ANC rendszer vezérlését.

A megépített kijelzős vezérlő egység valóban növeli az ANC bemutató rendszer mobilitását és egyszerűsíti az ember-gép kapcsolatot, amelynek köszönhetően tovább növeltem a demonstráció színvonalát. Ezenkívül a kijelzős vezérlő eszköz általános célú fejlesztésével és a többféle kommunikációs interfészek rendszerbe való integrálásával lehetőséget biztosítottam más hasznos fejlesztéseknek.

Abstract

Active Noise Cancelling (ANC) is one of the challenging and interesting theme of nowadays engineering. Displaying the effectiveness of an ANC system for a larger audience can be an interesting demonstration of the modern signal processing algorithms and processors.

A diploma project was started in 2007 at the Dept. of Measurement and Information Systems to build an easy-to-use, mobile ANC demonstration system, which can be used for exhibitions and for student laboratory as well. The control of the resulted system was only possible by using simple buttons of the DSP development board, the user could get information about the operation by LEDs.

This Master Thesis is about a design of a user and control interface which makes the demonstration with the existing ANC system easier and more attractive.

The resulted stand-alone system offers more user input possibilities and uses an LCD screen for visualization. The software for the central unit is running on an ATMEL Atmega16, 8-bit microcontroller. A 4x20 alfa-numeric LCD screen is used for the visualization. The ANC system is controlled by serial commands from the central unit via the UART port of the original DSP development card. The build-up of the printed circuit board is optimized for easy connection with the UART port of the original DSP development card. Additional features like internal noise generation and friendlier DSP control were added to the original ANC system to improve the demonstration performance.

The designed control unit with the LCD screen improved the mobility of the demonstration and resulted a user-friendly interface for the ANC system.

1. Bevezetés

Napjainkban egyre jobban előtérbe kerül a zajcsökkentés témája, mivel az ipari fejlődés következtében egyre több és nagyobb zajt keltő motorok, ventillátorok, turbinák jelennek meg, valamint az urbanizáció hatásai miatt különösen a gépjárművek száma is növekedett, amelyek a környezetünk zajterhelését nagymértékben megnövelik. Az ezáltal létrejövő problémának a megoldása igen fontos.

Többféle megoldás létezik, előnyökkel és hátrányokkal egyaránt. A zajcsökkentés két fő kategóriája a passzív és az aktív zajcsökkentés, az utóbbi angol elnevezése: Active Noise Canceling. Azt, hogy melyiket alkalmazzuk, mindig az aktuális probléma határozza meg, de általában a kettő együttes használata biztosítja a minőségi zajcsökkentést. A technikai fejlődés, a nagyteljesítményű jelfeldolgozó processzorok és algoritmusok megjelenése miatt, elérte azt a szintet, hogy érdemben lehessen foglakozni az aktív zajcsökkentéssel is. Pár évvel ezelőttig ez több nehézségbe ütközött.

A Méréstechnika és Információs Rendszerek Tanszék már régebb óta foglakozik ezzel a témával és a hozzákapcsolódó algoritmusokkal, megoldásokkal. Ezért felmerült az igény, hogy elkészüljön egy olyan eszköz, amellyel különböző méréseket, bemutatókat lehet végezni. Egy diplomamunka keretében elkészült egy aktív zajcsökkentést demonstráló rendszer 2007-ben, amellyel a fent leírt elképzeléseket már meg lehetett valósítani, de még több fejlesztési lehetőséget rejtett magában, ami növelné a rendszer hatékonyságát.

A tanszék szerette volna kihasználni a továbbfejlesztési lehetőségeket is, ezért önálló laboratórium keretein belül foglalkoztunk a témával. Sikerült az elmúlt félév során növelnünk a rendszer kezelhetőségét. Ez magában foglalta egy színvonalas, könnyen érthető vezérlő felület elkészítését személyi számítógépre, de a rendszer mobilitását nem növeltük. Így újabb fejlesztési igények adódtak, amelyeket a diploma tervezés során próbáltam megoldani. Hiszen napjainkban egy kiemelkedő szempont, hogy eszközeink

minél mobilisabbak és felhasználóbarátabbak legyenek, minél kevesebb járulékos eszközt kelljen felhasználnunk a működtetéshez.

A mobilitás növelése a labortól való elszakadásban rejlik. Minél kevesebb laboratóriumi eszközt és kevesebb kábelt kell felhasználni, annál könnyebb a rendszert elvinni különböző bemutatókra.

1.1. Meglévő demonstrációs rendszer [1]

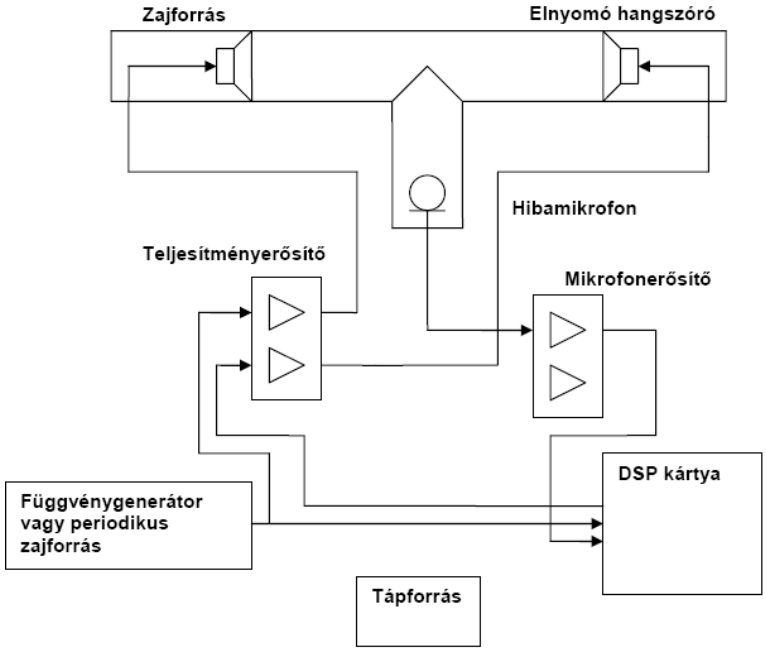
A meglévő demonstrációs rendszer alkalmas az aktív zajcsökkentés hatékony és látványos illusztrációjára, valamint a nagy sebességű jelfeldolgozó processzorok és algoritmusok bemutatására. A tervezés során felmerülő érdekes jelfeldolgozási problémák is meg lettek oldva, így más hasonló estekre is például szolgálhat. Az 1. ábra tartalmazza a rendszer blokkvázlatát.

Az akusztikus teret egy T alakú plexiüvegből elkészített átlátszó cső adja, amely majdnem zárt. A cső két végében egymással szembe fordítva helyezkedik el a két hangszóró. Az egyik a zaj előállítását szolgálja, a másik pedig az ellenzaj előállítását, vagyis a zaj elnyomását. A megoldás biztosítja, hogy különböző frekvenciájú, hangérzetű zajokkal lehessen kísérletezni. A hangszórókhoz tartozik egy teljesítményerősítő, ami potenciométerekkel szabályozható teljesítményt ad le. Így állítható be kézzel a hangerő. A T alakú cső nyitott lábánál helyezkedik el a hibamikrofon. Ehhez tartozik egy elemes mikrofonerősítő. A zajcsökkentést megvalósító szoftver az Analog Devices ADSP-BF537 EZ-KIT Lite fejlesztőkártyáján található fixpontos Blackfin processzorra készült. A fejlesztőkártyán található audió dekóder végzi a hibajel és referenciajel mintavételezését. Ezen információk birtokában a jelfeldolgozó processzor az FXLMS-en alapuló egycsatornás előrecsatolt zajelnyomó algoritmus segítségével előállítja a kioltó jelet. A

meglévő rendszer képes bármilyen más egycsatornás zajelnyomó algoritmus tesztelésére és bemutatására. A kártyán található LED-ek az éppen futó program állapotainak visszajelzését valósítják meg, a gombok pedig a felhasználói beavatkozások beviteli eszközei.
A demonstráció működéséhez szükséges zaj a laborban található függvénygenerátorral állítható elő. Az itt generált jel szolgáltatja a jelfeldolgozó algoritmus számára a referenciajelet, illetve erősítés után a zajhangszóróba kerül.

A rendszerrel tesztelése során periodikus és keskenysávú zavarjelek esetén átlagosan 14 dB, maximálisan 29 dB elnyomást sikerült elérni a tervezőnek, 150 Hz sávszélességű véletlenszerű zaj esetében átlagosan 15 dB elnyomás volt tapasztalható. Szélessávú elnyomás csak korlátozott feltételekkel, a cső akusztikai viszonyainak figyelembevételével érhető el, mivel a cső akusztikája jelentősen befolyásolja a zajcsökkenés mértékét.

A demonstráció azért hatékony, mert a zajforrás által keltett hanghullámok még azelőtt kioltásra kerülnek, mielőtt elhagynák a hangteret, így a zajcsökkentés nem koncentrálódik pusztán a berendezés közvetlen környezetére.



1. ábra Meglévő ANC rendszer blokkvázlata

1.2. Feladat bemutatása és a tervezéséhez szükséges megfontolások

Feladatom, hogy a meglévő demonstrációs rendszert továbbfejlesszem, annak érdekében, hogy növeljem a mobilitást, amely szorosan összefonódik a labortól való függetlenedéssel, valamint olyan vizuális vezérlő felületet építsek, amely a mai követelményeknek megfelel. Ezen kívül a rendszerhez szükséges személyi számítógép használatának csökkentése, hogy azt fejlesztéseknél kelljen használni, a normál üzemnél ne legyen szükség rá. Meg kell valósítanom egy olyan eszközt, amely kommunikációt végez a DSP kártyával és úgy biztosít lehetőséget a felhasználói beavatkozásokra, hogy minél kevésbé terheli a jelfeldolgozó processzort. Olyan beviteli eszközöket tervezzek a rendszerbe, amelyek segítségével könnyen vezérelhető a rendszer. A gombok funkciói jól elkülönüljenek, de mégse legyen szükség sokra, ezzel nehezítve a felhasználó dolgát. A felhasznált kijelző megfelelő megjelenítési felületet biztosítson és azon az információk ábrázolása egyértelmű legyen. A felhasznált menürendszer jól áttekinthető, logikus

legyen és illeszkedjen a rendszer bonyolultságához, struktúrájához. Ezeket az elvárásokat szem előtt tartva fogalmazhatóak meg az alábbi kérdések a tervezéssel kapcsolatban:

- Hogyan növelhető a rendszer mobilitása?
- A kijelzős egység mennyire legyen általános felhasználási célú?
- Milyen és mennyi beviteli eszközt illesszünk a kijelzős egységhez?
- Milyen kijelzőt válasszunk?
- Hogyan és hol csatlakoztassuk a DSP kártyát és a kijelzős egységet egymáshoz?
- Milyen kommunikációt alkalmazzunk a DSP kártya és a kijelzős egység között?
- Továbbfejlesztések végett, milyen más kommunikációs felületet biztosítsunk?
- Milyen legyen a menürendszer?
- Hogyan növelhető a rendszer megbízhatósága?
- Hogyan csökkenthetőek a kijelzős egység költségei?
- A felhasznált alkatrészek a későbbiek során, ha esetleg meghibásodás következik be, mennyire elérhetőek a kereskedelmi forgalomban?

A mindennapi életben folyamatosan találkozunk olyan elektronikus eszközökkel, amelyek tervezése során a fentebb említett elveket tartották szem előtt, és hasonló kérdések fogalmazódhatnak meg a termékekkel szemben. Így ezeket az eszközöket tanulmányozva választ kaphatunk néhány, az imént megfogalmazott kérdésre. Ezek a használati eszközök szintén rendelkeznek kijelzővel, valamint különböző beviteli perifériákkal (gombok, enkóderek, érintőképernyő). Ebbe a kategóriába tartoznak például a mobiltelefonok, az MP3 lejátszók, PDA-k stb.
A többi fennmaradó kérdésre, amelyek kifejezetten rendszerspecifikusak, a meglévő zajcsökkentést demonstráló rendszer tanulmányozása után kaphatunk választ.

1.3 Zajcsökkentés [1]

Ebben a fejezetben a teljesség igénye nélkül bemutatásra kerül a zajcsökkentés két alapvető megközelítése, az aktív és passzív zajcsökkentés és a legfontosabb aktív zajelnyomó algoritmusok, azzal a céllal, hogy egy átfogó képet kapjunk a megvalósított rendszer funkciójáról.

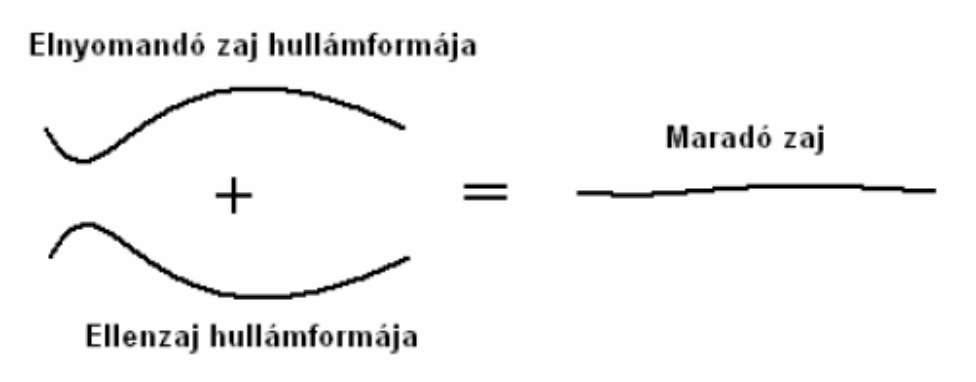
1.3.1. Zajcsökkentés elméleti áttekintés

Környezetünkben kétféle akusztikus zajt tudunk elkülöníteni egymástól:

- turbulencia által keltett zaj: a frekvenciatartományban egyenletesen széles sávban szórja szét az energiáját
- keskeny sávú zaj: energiája bizonyos frekvenciákon koncentrálódik

Zajcsökkentésnek kétféle megközelítése létezik. [2] Attól függően, hogy ellenzaj létrehozásával csökkentjük az adott térben mérhető zajterhelést, vagy valamilyen fizikai gátat, hangtompítót használunk, beszélhetünk aktív és passzív zajcsökkentő megoldásokról.
A passzív zajtompítás további két csoportra osztható: reaktív és rezisztív zajcsökkentőkre. A reaktív tompítók terelő gátak és csövek segítségével változtatják meg az akusztikus impedanciát, a rezisztív tompítók hangelnyelő anyaggal szigetelt csőben vezetik a zajt, így csökkentik annak energiáját. A passzív zajcsökkentésre használt eszközök előnye, hogy nagyfokú elnyomásra képesek a teljes frekvenciatartományban, hátrányuk viszont, hogy helyigényesek, anyagszükségletük magas lehet, így költségesebbek. A használt anyagokkal az a probléma, hogy csak addig hatékonyak, amíg a hang hullámhossza jóval kisebb a csillapító kiterjedésénél. Tehát alacsony frekvenciás csillapításhoz vastag anyag szükséges. Probléma lehet még a keresztmetszetszűkülés okozta nyomásnövekedés légáramos csőben.

Ezen problémák helyezik egyre inkább előtérbe az aktív zajcsökkentő megoldást, ami egy olyan elektroakusztikus eszközön alapszik, ami a nem kívánt hangot azonos amplitúdójú, de ellentétes fázisú ellenhang generálásával oltja ki, ami az alább ábrán látható. [1] A működés hatékonysága az amplitúdó és a fázis pontosságától függ.



2. ábra Az aktív zajcsökkentés fizikai koncepciója

A digitális jelfeldolgozó processzorok kifejezetten a valós idejű jelfeldolgozási feladatok megoldására készültek, jó hatásfokkal lehet rajtuk adaptív algoritmusokat futtatni. A 80-as években megnőtt az aktív zajcsökkentéssel kapcsolatos kutatások és fejlesztések száma, a nagy teljesítményű DSP-k megjelenésével. A 90-es évekre már több helyen alkalmazták az iparban. A technológia mára elérte azt a szintet, hogy megfizethető áron lehessen hozzájutni aktív zajcsökkentést alkalmazó fejhallgatókhoz.

1.3.2. Keskenysávú előrecsatolt zajcsökkentő rendszer

Mivel a mi esetünkben keskenysávú előre csatolt rendszer a fontos, ezért a többi megvalósításra nem térek ki. Alkalmazási területére jellemző, hogy a zajforrás keskenysávú és periodikus vagy majdnem periodikus. Nem akusztikus szenzorral helyettesíthetjük a mikrofont, ilyen mérőszenzor például a rezgésérzékelő és a

gyorsulásérzékelő. Ennek a megoldásnak az előnye abban mutatkozik meg, hogy az akusztikus visszacsatolás az elnyomó jel és a referenciajel között nem jön létre. A zajforrással szinkronban van a nem akusztikus szenzor jele és szimulálja a bemenő jel alapfrekvenciáját és felharmonikusait. Az adaptívan szűrt, szintetizált referenciajel az elnyomó jel. Sok eszközben rendelkezésre áll a fordulatszámjel, amely felhasználható referenciajelként a jelfeldolgozó processzor számára.

A keskenysávú rendszerek előnyös tulajdonságai:

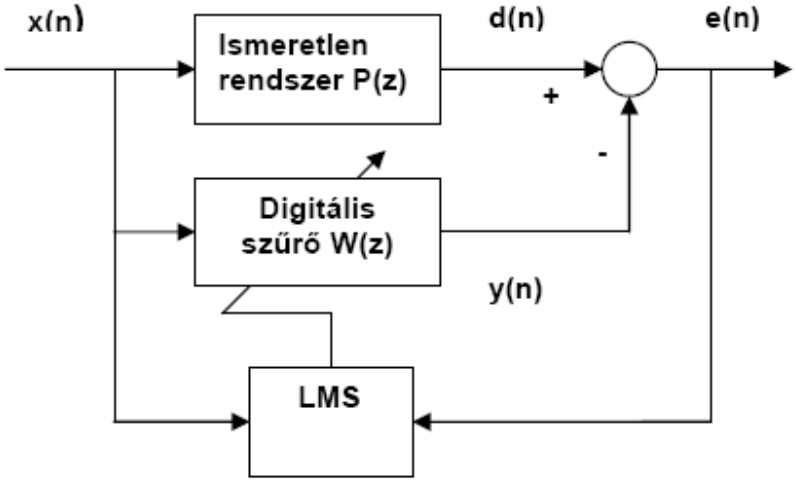
- Működéséhez nem szükséges kauzalitás. Elegendő csak a fázist és az amplitúdót állítani, mivel a zaj frekvenciatartalma konstans, emiatt viszont hosszabb lehet a szabályzó számítási ideje.
- A szintetizált referenciajel használata miatt szelektív elnyomás jöhet létre.
- Alacsonyabb fokszámú FIR szűrő használata elegendő, ez növeli a rendszer sebességét.
- Elkerülhető a káros akusztikus visszacsatolás.

1.3.3. LMS algoritmus [2]

Az LMS-algoritmus lényege a hibajel négyzetének, vagyis a teljesítményének csökkentése. Az LMS-algoritmus tulajdonságai igen kedvezőek, ennek is köszönheti, hogy elterjedt a használata:

- Könnyen megvalósítható jelfeldolgozó processzorra
- A paramétereinek kisebb változtatására stabil marad
- Viszonylag kicsi a számításigénye
- Kis maradó hibával tud működni és konvergenciasebessége viszonylag nagy

A matematikailag is igen jól leírt, könnyen kezelhető eljárások további előnyös tulajdonsága, hogy az iteratív modellillesztés elméleti levezetésében előírt, de a gyakorlatban nehezen kezelhető statisztikai paramétereket pillanatnyi értékekkel helyettesítik, ezáltal a bonyolult számítások, illetve az információhiány megkerülhető. [1]

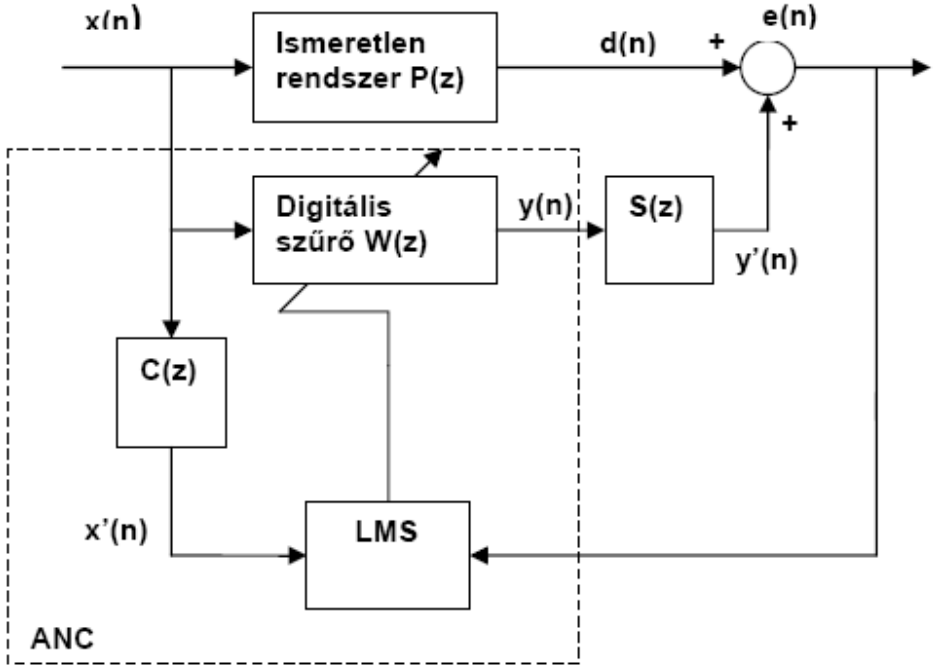


3. ábra LMS algoritmus blokkvázlata

A 3. ábraán lévő digitális szűrő és az LMS blokk együttesen egy adaptív szűrőt alkotnak. A bejövő $x(n)$ jelet szűri a $W(z)$ átvitelű digitális szűrő. A szűrő kimenete $y(n)$. Az algoritmus a szűrő súlyainak változtatásával próbálja minimalizálni az $e(n)$ hibajelet a kimenet és az elérni kívánt $d(n)$ jel között. Lényegében a szűrőegyütthatókat úgy állítja be az LMS algoritmus, hogy a hibajel négyzetének pillanatnyi értéke csökkenjen.

Az előző ábrán látható összegző csomópont a hangtérbeli akusztikus szuperpozíciót jelöli, ahol az adaptív szűrő kimenete és az elsődleges zaj találkoznak. Ez az elrendezés nem veszi figyelembe a szűrő kimenetétől ($y(n)$) a hibajelig ($e(n)$) tartó átvitelt, amelyet másodlagos útnak neveznek. Zajcsökkentésre ez az elrendezés nem alkalmas, mert stabilitási problémák lépnek fel.

A másodlagos út z-beli jelölése $S(z)$. Ez az átvitel magában foglalja a digitális-analóg (D/A) konverter, visszaállító szűrő, teljesítményerősítő, hangszóró átvitelét, valamint az akusztikus út átvitelét a hangszórótól a hibamikrofonig, a mikrofon előerősítő, az átlapolásgátló szűrő és az analóg-digitális (A/D) átalakító átvitelét. Az így kialakított rendszernél már az LMS algoritmus nem képes stabil zajcsökkentésre. Ezért szükséges kiegészíteni. Erre született az FXLMS algoritmus (4. ábra), amely hatásosan kompenzálja a másodlagos utat. Egy azonos átvitelű szűrő beiktatása a referencia jel útjába, úgy, hogy az hatással legyen az LMS algoritmust végző egység működésére, de az adaptív szűrő számára ne súlyozza a referenciajelet. A $C(z)$ tag az $S(z)$ másodlagos út becslője. Általában ez a megoldás a leghatékonyabb megközelítés.



4. ábra FXLMS algoritmus blokkvázlata

Az FXLMS stabilitásának feltétele, hogy $C(z)$ az eredeti átviteli függvény fáziskarakterisztikáját adott frekvencián legalább 90°-os pontossággal megközelítse, de 40°-os eltérésen belül már kielégítő a rendszer működése. Ez robusztusságot biztosít.

Fontos különbség a két algoritmus között, hogy az összegző csomópontnál nem kivonás szerepel az utóbbi blokkvázlatnál. Ez annak köszönhető, hogy itt akusztikus szuperpozíció van.

2. ANC bemutató eszköz bővített rendszerterve

Ahhoz, hogy a meglévő aktív zajcsökkentést demonstráló rendszer könnyen kezelhető és stand-alone legyen, egy felhasználóbarátabb kijelzős vezérlő egységet kell hozzá illeszteni. Az eddigiekben használt vezérlés a Blackfin fejlesztőkártyán található egyszerű nyomógombokkal nem felel meg az újonnan támasztott elvárásoknak.

A meglévő demonstrációs rendszernek kezdetben csak a DSP kártyán lévő gombok álltak rendelkezésre a működtetéshez. Tehát a használathoz már szükséges volt a rendszer behatóbb ismerete vagy a vezérlőtábla használata, mivel a funkciókat 4 darab gombbal lehetett kiválasztani. Valamint a rendszer állapotáról, csak a LED-ek és a hallható zaj adott visszajelzést.
Első körben önálló laboratórium keretében elkészült egy számítógépes vezérlő felület a LabView magasszintű programnyelv segítségével. Ezzel minden funkciót számítógépen, Windows felületen, kattintásokkal el lehetett érni. Az így létrejött kezelői felület sokkal magasabb szintű és színvonalú állapotvisszajelzést tett lehetővé.
A fejlesztések során sikerült a DSP szoftverét az ADSP BF537 EZ-KIT Lite kártyán található flash memóriába betölteni. Így a DSP reset után rögtön a megírt programot futatja. Ennek köszönhetően a fejlesztés után függetlenné vált a rendszer a számítógéptől, ami az egyik elsődleges célom volt a diploma munkám során. Így nem szükséges a számítógép a bemutató eszköz használatához, mivel a megírt szoftver tartósan tárolható. Ezért a már említett számítógépes vezérlő felület, továbbiakban már csak extra funkcióként élhet tovább, mivel csökkenti a mobilitást.
Az aktív zajcsökkentést demonstráló rendszer labortól való függetlenségét az imént említett flash memóriában való program tárolásán túl nagymértékben növelte a diplomamunka keretén belül még elkészített kijelzős vezérlő egység. Könnyen csatlakoztatható és rögzíthető a ADSP BF537 EZ-KIT Lite fejlesztői kártyához és magában hordozza az egyszerű kezelhetőséget valamint a magasabb szintűn vizuális állapotvisszajelzést.

A feladatom kiindulási pontja az volt, hogy egy kijelzős egységet tervezzek, amely ki- és beviteli felületekkel van ellátva és képes együttműködni a DSP kártyával egy interfészen keresztül, felhasználva a DSP kártya által biztosított tápellátást.

Az utóbbi szempont a már eddig említettekkel azonos súllyal bír, mivel külön energiaforrás esetén egy újabb eszközt kell a rendszer üzemeltetéséhez használni, ami az alapkövetelményeket nem valósítja meg és növeli a rendszer bonyolultságát, és a felhasznált kábelek számát. Egy másik megoldás lehetett volna még, ha elemeket használunk erre a célra. De ezáltal is külön gondot vettünk volna a nyakunkba, mivel oda kellett volna figyelnünk az elemek töltöttségére és a vezérlő egység külön kikapcsolására használat után, valamint növelnem kellett volna a külső fizikai méreteket. A megvalósításra került megoldásnál, ami a DSP kártyán lévő tápforrásnak a használatát jelenti, nem növelte a rendszer bonyolultságát és a fentebb említett összes problémát kiküszöbölte. A kártyán helyet adtam egy főkapcsolónak, amellyel a kijelzős vezérlő egységet áramtalanítani lehet a DSP kártyától függetlenül. Ezenkívül biztosítottam egy külső tápforrás-csatlakozást arra az esetre, ha egy másik rendszerrel szeretnénk együtt üzemeltetni a kijelzős egységet. Ennek a bemenetén egy 5 voltos feszültségstabilizáló IC-t használtam fel, így nagyobb feszültségek csatlakoztatása esetén sem léphet fel károsodás a kártyán. A megvalósítás során figyelembe vettem azt a lehetőséget is, hogy egyszerre két tápforrás csatlakozik a kártyára, ami nem megengedett, mivel ez károsodáshoz vezethet. Így egy egyszerű diódás kapcsolással kiküszöböltem ezt a problémát.

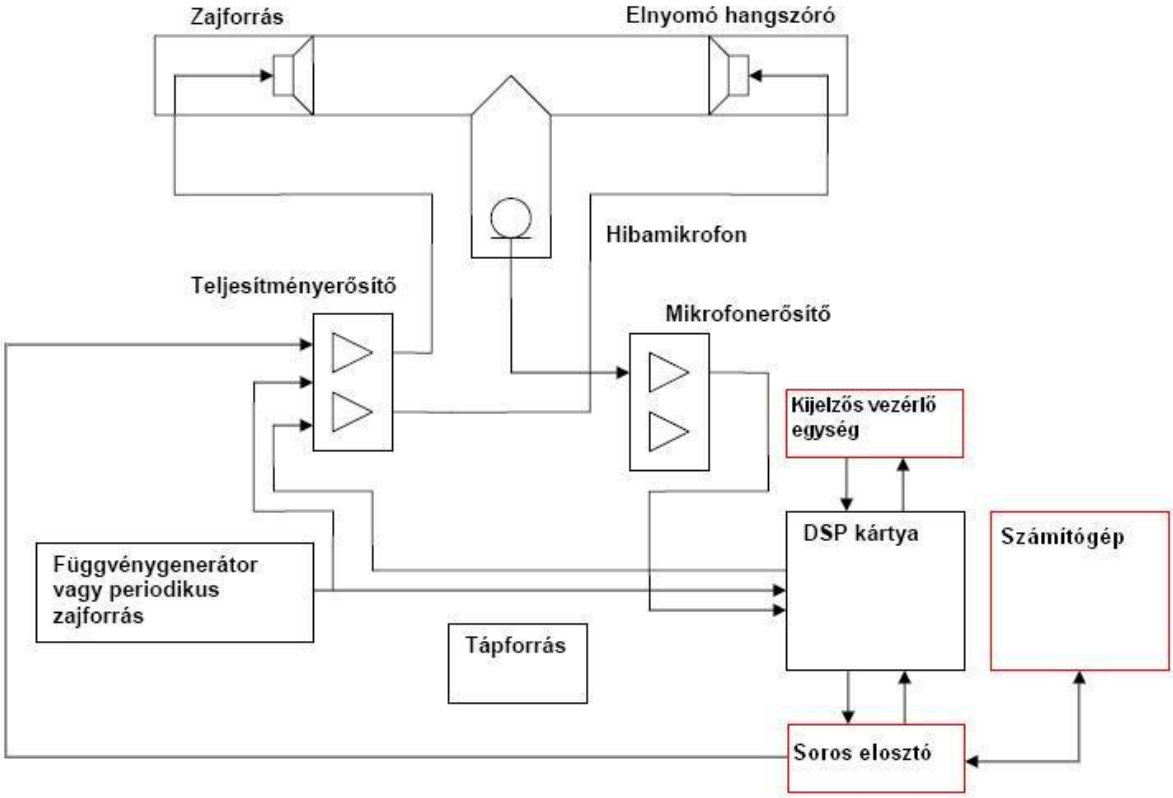
A kiviteli eszközt elsősorban a karakteres LCD kijelző valósítja meg. Itt a felhasználó számára hasznos információkat jeleníthetünk meg. A kiválasztás során felmerült a lehetőség, hogy grafikus LCD kijelzőt alkalmazzak, de nem sikerült olyan üzembiztos grafikus kijelzőt találnom, amely elérhető áron és méretben rendelkezésemre állt volna. Ezért a könnyen beszerezhető és egyszerűen használható EW20400GLY karakteres kijelző mellett döntöttem. Teljesen hagyományos megjelenítéssel, zöld háttérvilágítással

és szürke karakterábrázolással. A vezérlő IC-jének a parancsai a HD44780 karakteres kijelzőt vezérlő IC parancsaival megegyezők. Ez utóbbi IC pedig a legáltalánosabb körben elterjedt. Ennek köszönhetően a felprogramozáshoz szükséges adatok könnyen hozzáférhetőek. A kontraszt beállítására egy állítható ellenállással van lehetőség.

A beviteli eszközöket 3 darab gomb és egy enkóder képezi. A tervezés során arra a megállapításra jutottam, hogy ezek elegendőek a rendszer hatékony vezérlésére és a menüben való navigáláshoz. A két szélső gombbal a menüben való léptetést lehet elérni. Aminél egyértelműbb használat érdekében ezt a két gombot a kijelző alsó két széléhez közel helyezem el, így a funkciójuk szimbólumát megjeleníthetem a kijelző két alsó sarkában. A harmadik gomb, amely a nyugtázásra van fenntartva (azaz az OK), pedig középen kap helyet a kijelző alatt lévő vékony NYÁK lécen (9. ábra). A DSP fejlesztőkártya a gomb funkciókat képes lenne lekezelni, de azáltal, hogy külön mikrokontrollerrel dolgozom fel a felhasználói beavatkozásokat, tehermentesítem a DSP processzort. Így több erőforrás tartalékkal rendelkezik a rendszer.
Az enkóder az előbb említett három gombot egészíti ki azáltal, hogy a kiválasztott érték állítását tekeréssel érhetjük el. A tekerő kar lenyomásával a már fentebb említett OK funkciót valósítottam meg. A további fejlesztések során a két beviteli eszköz funkciói könnyen változtathatóak és bővíthetőek.

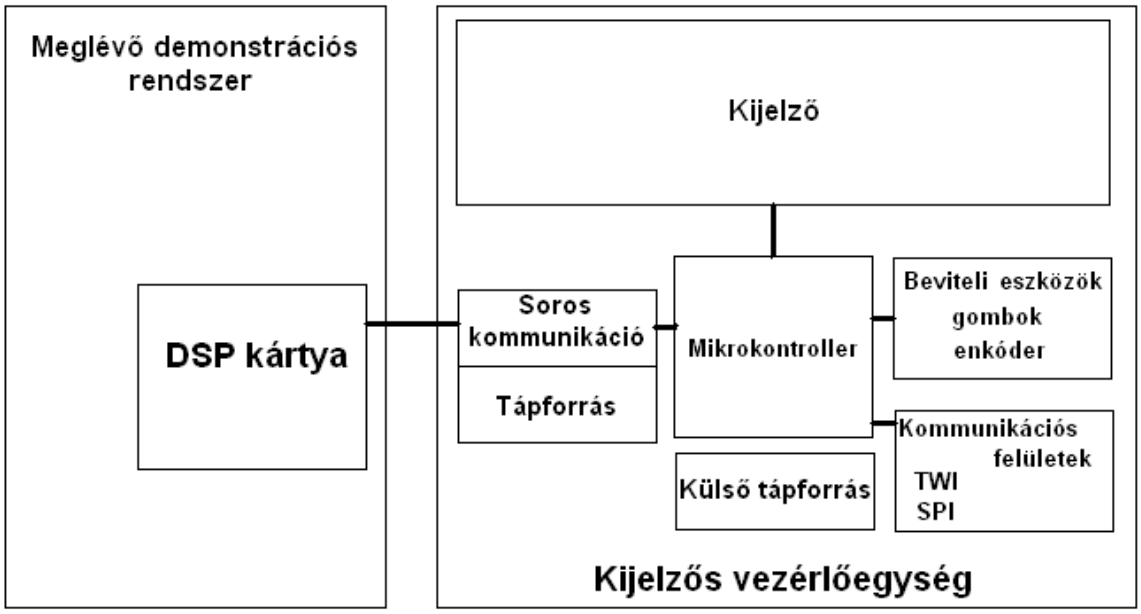
A DSP kártyával való adatok cseréjéhez a soros kommunikációt használtam fel. Választásomat az indokolta, hogy a soros kommunikáció egy már jól kiforrott hatékony és egyszerű interfész, valamint a fejlesztőkártyán elhelyezett soros kivezetés jól hozzáférhető és kényelmesen hozzátervezhető a kijelzős egység, amely az esztétikai és ergonómiai elvárásoknak is megfelelő. Biztosítottam továbbá még két kommunikációs felületet, ha az elkövetkező fejlesztések során szükség lenne rájuk. Az egyik a TWO WIRE SERIAL BUS (TWI, I2C), valamint a SERIAL PHERIPHERAL INTERFACE (SPI), aminek a felhasználásával a mikrokontroller felprogramozható. Ezek a kommunikációs felületek ki- és beviteli lehetőséget biztosítanak.

Az így létrejövő kijelzős vezérlő rendszer igen széleskörű felhasználói lehetőségeket biztosít, amely más további fejlesztéseket is támogat, ezáltal a más területeken is sikeresen felhasználható.



5. ábra A kibővitett ANC rendszer blokkvázlata

Az 5. ábra A kibővitett ANC rendszer blokkvázlatalévő blokkvázlat a demonstrációs rendszeren eddig végzett összes fejlesztést és bővítést láthatjuk. Ha a mobilitást és a látványos megjelenítést egyszerre akarjuk igénybe venni, akkor nem szükséges minden laboratóriumi eszközt használni, a fejlesztésekkel rendszerbe bejött új elemek alkalmazása variálható. A teljes rendszerből elhagyható a Soros elosztó, a Számítógép és a Függvénygenerátor vagy a periodikus zajforrás. Ekkor a hangerőszabályzás csak kézzel állítható. A zajforrást a DSP maga biztosítja az elvégzett fejlesztésnek köszönhetően. Az alább ábrán látható az így továbbfejlesztett bemutató rendszer, melyhez nem szükséges a számítógép használata.



6. ábra Diploma munka során tovább fejlesztett rendszer blokkvázlata

A következő fejezetekben ismertetem a rendszer tervezése során felmerült további kérdéseket és problémákat, az azokra adott válaszokat és megoldásokat.

3. Hardver tervezés

Ebben a fejezetben általánosan áttekintem a két fő egységet a kijelzős vezérlő egység elemei közül, majd ismertetem a tervezési megfontolásokat.

3.1. Mikrokontrollerek

A mikrokontroller olyan autonóm rendszer, amely saját programmemóriával, processzorral és perifériákkal rendelkezik. Egyetlen lapkára integrált és általában vezérlési feladatokra optimalizált számítógép. Költséghatékonyan képes ellátni egyszerű, kis számítási teljesítményt és operatív tárat igénylő műveleteket. A feladatok széles skáláját képes megoldani úgy, hogy közben kevés járulékos eszköz felhasználását igényli. A mikrokontrollerek optimalizálásának főbb irányai a fogyasztás és méret csökkentése, valamint a költségminimalizálás. Ehhez az IC lábainak multiplex felhasználása és a beépített perifériák egyre bővülő köre járul hozzá. [6]

Programozásuk a PROM-okhoz (PROM, EPROM, EEPROM) hasonló módon, a logikai magas szintnél nagyobb égetőfeszültség alkalmazásával történik. A régi típusok egyszer voltak programozhatóak, de az új eszközök gyakorlatilag mindegyike Flash-ROM alapú programtárat tartalmaz, így akár egymillió beírási/törlési ciklust is elviselnek.

A mikrokontrollerek bitszám alapján 3 családba oszthatók:

- 8 bitesek: Atmel AVR (Attiny, Atmega), PIC (16Fxxxx, 18Fxxxx, 30Fxxxx), SX
- 16 bitesek
- 32 bitesek: főleg ARM magos processzorok, általában operációs rendszerrel

A mikrokontroller ellátandó feladatától függően sok perifériát tartalmazhat, amelyeknek fogyasztásuk van (ez függ a periféria állapotától, annak beállításaitól). A perifériák fogyasztása a mikrokontroller összfogyasztását növeli. A fölösleges disszipáció elkerülése végett egyes perifériák ki- ill. bekapcsolhatóak akár a program futása folyamán is. [8]

Gyakran előforduló perifériák:

- Oszcillátor
- Operatív tár a vezérlőprogram futtatására
- Számlálók/időzítők
- Watchdog időzítők
- EEPROM memória
- Jelátalakítók
- Kommunikációs buszok
- Meghajtó egységek
- Jelgenerátorok
- Debug interface

3.2. LCD kijelzők

Folyadékkristályos, kijelző idegen névvel LCD, Liquid Cristal Display. Két üveglap között vékony folyadékkristály réteg található, amely feszültség hatására megváltoztatja az optikai tulajdonságait. A kijelző első és hátsó oldalára egy-egy polárszűrőt helyeznek, amely a fény minden irányú rezgését csak egy meghatározott síkban engedi tovább. A folyadékkristály molekulái az elektromos tér hatására elfordulnak, így bizonyos irányban átengedik a fényt. Ha tehát olyan alakú elektromos teret hozunk létre az üveglapok között, mint a megjeleníteni kívánt karakterek és rajzok, akkor ott a folyadékkristály molekulái

elfordulnak, és nem engedik át a fényt: a kijelző elsötétül. Ezt a tulajdonságot csak bizonyos hőmérsékleten belül képes produkálni (-5-től 65 C°-ig). A megjelenítés módja szerint csoportosítva 3 féle LCD kijelző létezik.

Numerikus kijelzők:

Ezek a kijelzők számok megjelenítésére alkalmasak. Egy tokban külön szegmens elemek vannak és a kivezetés csökkentés érdekében az egyik kivezetés közösítve van. Egy egyszerű változata a hétszegmenses kijelző, aminek a vezérlését hétszegmenses dekódolóval oldják meg.

Alfanumerikus kijelzők:

Az alfanumerikus kijelzők képesek már betűk megjelenítésére is. Két fajtája van, a 16-szegmenses és a mátrix. A mátrix vezérlését multiplexálással oldják meg.

Grafikus kijelzők:

Rajzok, képek megjelenítésére is alkalmas eszközök. A megjelenítés alapeleme a pixel. Ezekből a pontokból rajzolják ki a kívánt alakzatot. A grafikus kijelzőknek is több fajtáját különböztethetjük meg, különböző szempontok alapján. Nem célom azonban a kijelzők részletes bemutatása a diplomamunkám során.

Ezek alapján és a 2. ANC bemutató eszköz bővített rendszerterveemlített megfontolások alapján az EW20400GLY típusú karakteres kijelzőt választottam. Ez a kijelző 4 sorban egyenként 20 karakter megjelenítésére képes, amely elegendő információ megjelenítését biztosítja a rendszer vezérléséhez. A zöld színű LED háttérvilágítás a kijelzőbe van integrálva. A vezérlő IC-je a HD44780 szabványt használja, ami jól dokumentált, könnyen kezelhető. A kijelző lábkiosztása az 1. Táblázat olvasható.

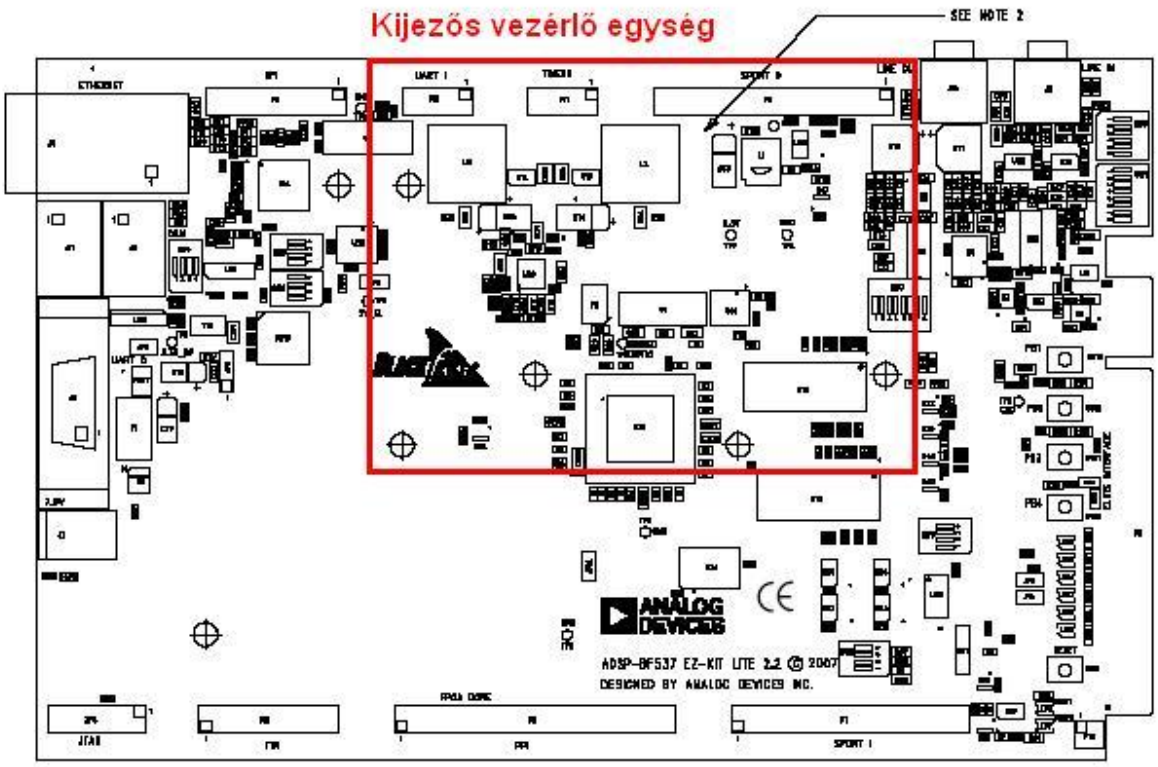
1. Táblázat A kijelző lábkiosztása

PIN NO	SYMBOL	DESCRIPTION
1	VSS	GND
2	VDD	POWER SUPPLY FOR LOGIC CIRCUIT (+5 V)
3	V0	LCD CONTRAST FOR LOGIC CIRCUIT
4	RS	INSTRUCTION/DATA REGISTER SELECTION
5	R/W	READ/WRITE SELECTION
6	E	ENABLE INPUT
7	DB0	DATA INPUT/OUTPUT LINES
8	DB1	
9	DB2	
10	DN3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	
15	VLED	POWER SUPPLY FOR LED BACKLIGHT
16	VLSS	VLSS POWER SUPPLY FOR LED BACKLIGHT (CATHODE) 0V (GND)

Az LCD 8 adatvonallal rendelkezik, de lehetőség van arra, hogy csak 4 adatvonalat használjunk (4-bit bus mode), így kevesebb lábat kell elhasználnunk a mikrokontrollertől. Ez ergonomikusabb hardvertervezést tesz lehetővé. Emiatt a vezérléshez elegendő 7 pin, ezek a következők: RS, R/W, E, DB4, DB5, DB6, DB7. [9]

3.3. Kijelzős vezérlő egység

A rendszertervben leírtak alapján és az említett elvárásokat figyelembe véve terveztem meg a nyomtatott áramkört. A kialakítás során figyelembe kellett venni, hogy a létrejövő NYÁK-ot az Analog Devices ADSP-BF537 EZ-KIT Lite fejlesztőkártyájához kell illeszteni. A DSP kártyát áttanulmányozva arra a megállapításra jutottam, hogy a kártya fölső középső részén lehet a legjobban mechanikailag és elektronikailag is csatlakoztatni, mint azt az alább is mutatja.

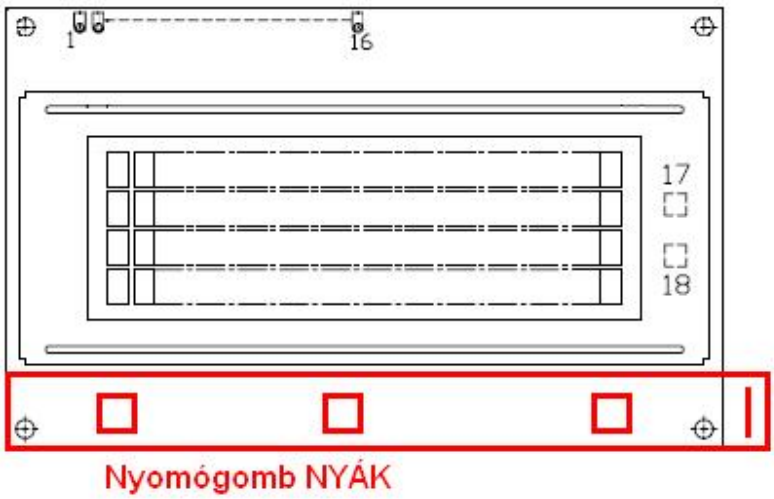


7. ábra Kijelzős vezérlő egység helye a DSP kártyán

A rögzítéshez csavarokat és távtartókat használtam fel. A rögzítési pontok a 7. ábra által mutatott letakart területen lévő furatok. Az elektromos csatlakozáshoz a DSP UART1 tüskesoros kivezetéshez illesztettem a vezérlő egység NYÁK alsó részén elhelyezett csatlakozó aljzatot. Mivel a két NYÁK között legalább 2 cm-nek kellett lennie a DSP-n lévő alkatrészek miatt, ezért két egymással toldott csatlakozó aljzatra volt szükségem. Egy másik tüskesor csatlakozó aljzat csatlakozás is helyet kapott alulról a NYÁK jobb felső részén, de ez elektronikai szempontból nem bír jelentőséggel, csupán mechanikai funkciót lát el.

A tervezés során az is szempont volt, hogy a DSP-re rögzített NYÁK ne tűnjön robosztusnak, de a mérete kellő helyet biztosítson a felhasznált alkatrészek számára.

Egy másik fontos illesztést is figyelembe kellett venni a tervezés során. Szükség volt a karakteres kijelző illesztéséhez is csatlakozási pontokat biztosítani. Ez a vezérlő kártya fölött helyezkedik el, részben eltakarva a felületét. A nyitott rész a vezérlő kártya felső és jobboldalán elhelyezkedő L alakú terület. A mechanikai rögzítését szintén csavarokkal és távtartókkal oldottam meg.

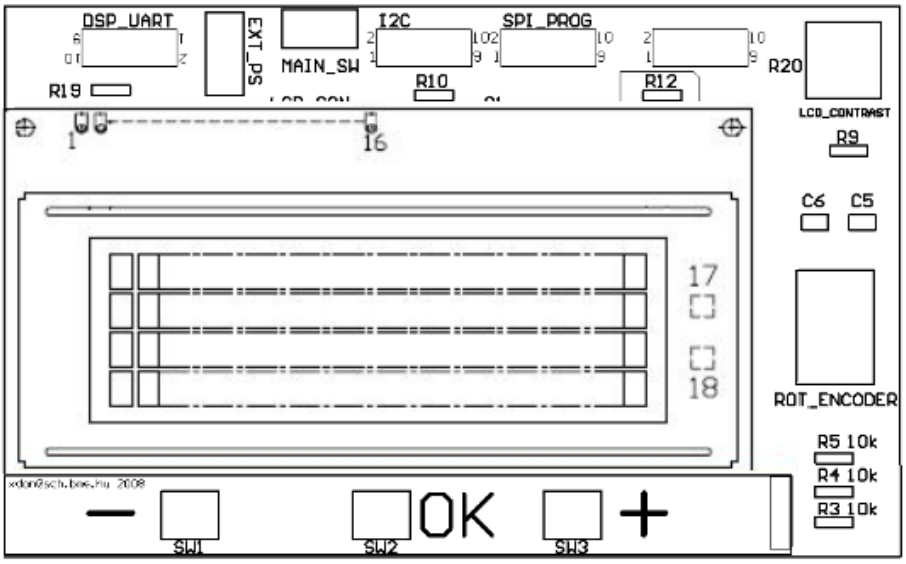


8. ábra LCD kijelző fölé elhelyezett nyomógomb NYÁK

Célszerű volt a nyomógombokat minél közelebb elhelyezni a karakteres kijelzőhöz, ezért egy hosszúkás NYÁK darabon kaptak helyet, amely a kijelző nyomtatott áramköre fölé lett illesztve.(8. ábra) Mechanikailag a kijelzőn túl nyúló csavarokkal rögzítettem. Az elektromos kapcsolatot a vezérlő kártya jobb oldalán található csatlakozási felület biztosítja, amelyre a gomb NYÁK-ról egy hosszúkás tüskesor nyúlik le.

Az így létrejövő kártyarendszer emeletes struktúrájú (9. ábra), ami jó szellőzési elrendezése miatt az alkatrészek melegedésére nézve kedvező. A vezérlő NYÁK fedett részein helyeztem el a felhasználó számára lényegtelen részeket, például: mikrokontroller, egyéb IC-k. Az L alakú fedetlen rész jobb oldalán helyezkedik el az enkóder és a kijelző kontrasztját állító ellenállás. Az enkóder a magasságából és helyzetéből adódóan a kijelzőhöz közel helyezkedik el, de elég távol ahhoz, hogy a felhasználót tekerés közben

ne zavarja. A NYÁK felső fedetlen részén helyeztem el a különböző interfész kivezetéseket, a külső tápforrás csatlakozóját és a vezérlő NYÁK főkapcsolóját.



9. ábra Kijelzős vezérlőegység (felülnézet)

Az IC-k beültetésénél IC foglalatot használtam, biztosítva a könnyebb cserélhetőséget meghibásodás esetén.

4. A felhasznált hardver és szoftver eszközök

Ebben a fejezetben bemutatom a felhasznált szoftver és hardverkörnyezetet, valamint a rendszerbe ágyazott interfészeket.

4.1. ADSP BF537 EZ-KIT Lite [3]

A meglévő ANC demonstrációs rendszer az Analog Devices ADSP BF537 EZKIT Lite nagyteljesítményű DSP kártyájára lett kifejlesztve. A kártyán az Analog Devices Blackfin BF537, fixpontos, 600 MHz-es processzora található. [3] A kártya Universal Serial Bus (USB) porton csatlakozik a számítógéphez. A kártyán különböző perifériák kaptak helyet. Az egyik a jelfeldolgozáshoz nélkülözhetetlen AD1871 sztereo DA-átalakító és AD1854 sztereo AD-átalakító, összefoglaló nevén CODEC, ami 48 kHz mintavételi frekvencián üzemel. A bemenő jeleket 24 bites kettes komplemens kóddá konvertálja a szigma-delta elven működő AD-átalakító. Az így előállított kódot további feldolgozásra a processzor soros vonalon kérdezheti le. A kimenethez szükséges analóg jelet a DA-átalakító a processzortól sorosan küldött 24 bites számokból, szintén szigma-delta elven működő modulációval alakítja át. Mindkét konverter jelei az audio technikában általánosan elterjedt sztereó jack aljzatokra vannak kivezetve. A kártyán 4 db általános célú nyomógomb található, amelyekkel lehetőség nyílik a processzor számára megszakítást generálni. A kártyán, állapotok jelzésére alkalmas 6 db LED kapott helyet, melyek a processzor általános I/O lábaira vannak csatlakoztatva. A futó program állapotainak monitorozására a LED-ek és nyomógombok használata a legegyszerűbb lehetőség, ami elég korlátozott információmegjelenítést biztosít. A kártyára integráltak még egy 4 MB-os flash memóriát, aminek köszönhetően képes a felhasználó által megírt programot tárolni és futtatni, úgy hogy áramtalanítás utáni újra bekapcsolás során az általunk elkészített

szoftver fusson. Meg kell még említeni a kártya funkciói közül a különféle kommunikációs eszközöket, ezek közül is a diplomamunka szempontjából fontosak a következők: a jelfeldolgozó processzorban két UART kapott helyet. Az egyik egy RS-232 jelszintillesztő után a szabványos csatlakozóra van kivezetve. Míg a másikat közvetlenül a processzorról vezették ki, a kártyán található tüskesorok egyikére. Az előbbire a számítógépet csatlakoztatjuk az általunk használt rendszertervben, míg az utóbbi a kijelzős vezérlő egység interfészéül szolgál. A kártya sok más funkcionális egységet és kommunikációs lehetőséget tartalmaz, de ezek a feladatom szempontjából nem lényegesek.

4.2. A BF537 jelfeldolgozó processzor [3],[4]

A BF537 a Blackfin processzorcsalád tagja. A gyártó ezt a családot úgy fejlesztette ki, hogy kód- és lábkiosztás kompatibilis legyen a BF534 és BF536 típusokkal. Fixpontos processzorcsalád, amely egyesíti a DSP és a beágyazott processzorok előnyös tulajdonságait, átmenetet képezve így köztük. A jelfeldolgozó processzor további főbb jellemzői:

- Módosított Harvard-architektúra: a memória két részre van osztva. (adatmemória és programmemória). Előnye az, hogy az adatok memóriába írásával vagy olvasásával egy időben már be is tölthetjük a következő utasítást a programmemóriából, felgyorsítva a rendszer működését, vagy egy utasításciklus alatt egyszerre két adatszóhoz férünk hozzá. Az utóbbi annak köszönhető, hogy az adatmemória csak adatokat tartalmazhat, a programmemória a programkód mellett adatokat is tartalmazhat. Ez a tulajdonság igen hasznos FIR szűrés esetén. A bemenő adatokat és a szűrőegyütthatókat célszerű külön memóriaterületen tárolni.

- Összesen 132 kbyte on-chip SRAM, amely a processzor órajelén működik. A Harvard-architektúrának megfelelően két részre van osztva. 48 kbyte programmemóriát és 64 kbyte adatmemóriát tartalmaz.
- 600 MHz órajelfrekvencia.
- Párhuzamosan működő, 2 db 16 bites MAC műveleti egység, 2 db 40 bites akkumulátor. A MAC jelentése Multiply and Accumulate. Egy utasításciklus alatt egyszerre szorzást végez, és az előző szorzás eredményét hozzáadja az eredményhez. Ezzel a művelettel elsősorban a transzverzális szűrők megvalósítását és más jelfeldolgozó algoritmusokat (pl. Fourier transzformáció) tehetünk gyorsabbá.
- 8 db 32 bites általános regiszter, melyek mindegyike kezelhető 2*16 bitesként.
- 2 db független címaritmetikai egység
- 4 db cirkuláris buffer kezelése
- 8 db általános pointer regiszter.
- 8, 16, 32 bites adatok.
- 16, 32 bites utasítások.
- Debug funkciók, teljesítménymonitorozás és nyomkövetés lehetősége a JTAG interfészen keresztül.
- RISC utasításkészlet, csökkentett utasításkészletű számítógép

4.3. VisualDSP++

A DSP programozása a VisulalDSP++ integrált fejlesztőkörnyezettel történt. Az elindítása után a program ablak fő részét a projekt editor képezi, ahol a forrásfájlok szerkesztése valósítható meg. Ezek a forrásfájlok együtt összerendelve alkotnak egy projektet, amit a fordító linkel össze a fordítás során és generál belőle a DSP kártyára

letölthető fájt. Ez az integrált fejlesztői rendszer C, C++ és assembly nyelven biztosítja a forrásfájlok szerkesztését. Ezek a nyelvek a használat során egymásba ágyazva is használhatóak. [4]

A fejlesztői környezet további része még a debugger, amely segítségével kipróbálhatóak és tesztelhetőek a felhasználó által a környezetre írt programok. A debugger tartalmaz egy szimulátort, amellyel a processzor működését lehet szimulálni számítógépen. Ennek köszönhetően a megírt forráskód várható viselkedése könnyen tesztelhető. A szimulátor használhatóságát az korlátozza, hogy a perifériákat nem tartalmazza, így bizonyos esetekben nem támaszkodhatunk csak a szimulációs eredményekre. A debugger egy másik, a fejlesztéshez igen hasznos része az emulátor. Természetesen ennek használatához szükségünk van a DSP-re is. A Blackfin JTAG interfészen keresztül érhetjük el a kártyán lévő regisztereket az emulációhoz. Ez is, mint ahogy a futatható program rátöltése is USB porton keresztül érhető el. A program futása megszakítható, töréspontok helyezhetők el, lehet léptetve végrehajtani az utasításokat, és az egyes regiszterek állapota folyamatosan nyomonkövethető, valamint megszakítás után átírhatóak. Megszakítás után a futtatás folytatható a változtatott adatokkal. Lehetőségünk van a memória fájlba történő kimentésére különböző formátumokban, ami elősegíti a további feldolgozást. Léptetve futtatva a processzort lehetőség van például különböző aritmetikai műveletek vizsgálatára, a túlcsordulások könnyebben észrevehetőek.

További hasznos része a fejlesztői környezetnek a beépített Flash Programer, amelynek segítségével hozzáférhetünk a DSP kártyán lévő flash memóriához. Ezáltal lehetőségünk van az általunk megírt programok boot programként való betöltésére. Így a kártya áramtalanítása után sem veszik el a betöltött program. Ennek a hasznos funkciónak a segítségével, loader fájlként lefordítva tölthetjük bele a flash memóriába az elkészített projektünket. A projekt betöltése előtt szükség van azonban egy bizonyos driver fájl flash memóriába való elmentésére, amit a fejlesztői rendszer alkönyvtárában találhatunk meg.

A DSP számábrázolásából adódóan érdemes még kiemelni a fejlesztőkörnyezet által nyújtott segítséget, amellyel lehetőségünk nyílik a C-ben történő egyszerű fejlesztésre. A 16 bites fixpontos processzorhoz leginkább a 16 bites előjeles egész, a short típus használata illeszkedik. Jelfeldolgozási feladatoknál az A/D-D/A átalakítók tartományán belül lévő jelet legcélszerűbb előjeles törtszámmal ábrázolni, mivel ez illeszkedik legjobban a fizikai képhez. Az előjeles számokat kettes komplemens kódban ábrázolják. A legmagasabb helyértékű bit előjelbitként viselkedik, az ezt követő bit értéke ½, a legalsó bit helyi értéke pedig 2^{-15} . Mivel a szabványos C-ben ez a fajta törtszámábrázolás nem létezik, ezért az Analog Devices definiálta a fract16 és fract32 típusokat, amelyek 16 és 32 bites törtszámok ábrázolását teszik lehetővé. Ezt a két új típust és a hozzátartozó műveleteket a fejlesztőkörnyezet biztosítja C könyvtári függvényekkel.

4.4. ATMEGA16/32 mikrokontroller [7]

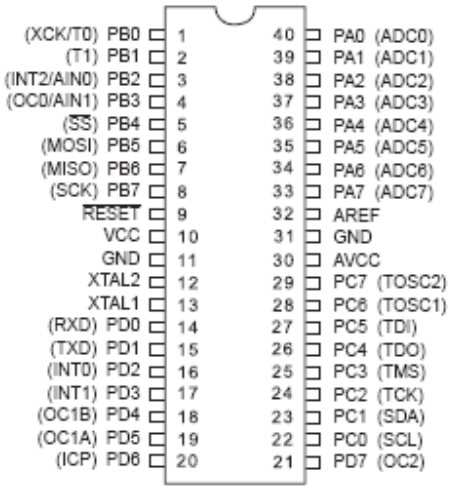
A választásom azért esett erre a mikrokontrollerre, mert ez már megfelelő programmemória mérettel rendelkezik, így egy összetettebb feladat megvalósítására is alkalmas. A felhasználói beavatkozásokat maradéktalanul, kellő gyorsasággal képes kiszolgálni. Több kommunikációs interfészt tartalmaz, így képes több különböző külső rendszerekkel adatcserét végezni, anélkül hogy bármiféle megszorítást kelljen alkalmazni. Ezenkívül könnyen illeszthető más periféria a processzorhoz, például egy kijelző. Könnyen beszerezhető és különböző kivitelben lehet megvásárolni. Én furatszerelt DIP-40 tokozásban használtam fel, mivel a kezdeti tesztelések során ezt tudtam könnyen illeszteni a próbapanelba. A lábkiosztása a 10. ábra látható.

A 8-bites Atmel AVR processzor fontosabb tulajdonságai a következők:

- RISC felépítés, egy utasítást átlagosan 1 órajel alatt hajt végre, (kb. 8 MIPS), szétválasztott kód- és adatmemória
- 16 kbyte on-chip flash (10 000 írás-törlés), opcionális Boot code szekcióval, önprogramozási lehetőséggel
- 1 kbyte on-chip adat SRAM
- 512 byte on-chip EEPROM (100 000 írás-törlés)
- In-System Programming (ISP), In-Application Programming
- JTAG támogatás (programozás, debuggolás, Boundary-scan)
- UART, SPI, I2C kommunikációs interfészek
- 2 db 8-bites timer, 1 db 16-bites timer (compare, capture, PWM üzemmód), független timer előosztók, real-time clock külön kvarccal, watchdog timer
- Kisfogyasztású üzemmódok (hatféle)
- Maximum 16 MHz-es órajel, programozható rendszeróra-osztó (fogyasztáscsökkentés céljából)
- Egyes perifériák tápellátása külön ki/be kapcsolható, ezzel biztosítva további fogyasztás-optimalizálási lehetőségeket
- Bekapcsolási RESET áramkör, programozható Brown-out detektor
- 8-csatornás, 10-bites A/D-átalakító
- External and Internal Interrupt Sources

Az ATMEGA16 és az ATMEGA32 megegyező tulajdonságokkal rendelkeznek. Egyetlen lényeges különbségüket a nevükben is hordozzák. Az ATMEGA16 16 kbyte on-chip flash memóriával rendelkezik, míg a másik 32 kbyte on-chip flash memóriával. Itt említeném meg, hogy ezeknek a mikrokontrollereknek is több verziójuk látott napvilágot. A tervezés során felmerült egy alacsonyabb, 3 voltos feszültségű eszköz alkalmazásának a lehetősége, amit a név végén elhelyezett L betűvel szokás jelölni. A DSP kártya felhasznált tüskesorán az 5 V-os kivezetés mellett az alacsony feszültségű is megtalálható.

Mivel az alfanumerikus kijelző 5 V-ról üzemel, ezért jobbnak láttam, ha az egész kijelzős vezérlő egység azonos tápfeszültséget használ. [7]



10. ábra ATMEGA16/32 DIP tokos lábkiosztása

4.5. Soros kommunikáció [8]

A soros adatátvitelnél nagyon fontos megemlítenünk az univerzális aszinkron adóvevőt, rövidebben UART-ot. Ez egy hardver, mely olyan tulajdonsággal rendelkezik, hogy képes a soros és párhuzamos interfészek közötti fordításra. Esetünkben mikrokontrollerből érkező 8 bites párhuzamosan érkező adatokat sorosít. Ha az átvitel soros, akkor az UART aszinkron bitfolyammá alakítja át a bájtokat, majd ezeket elektromos impulzusokkal vezeti tovább. Ezt a fajta adóvevőt gyakran kombinálva használják az RS-232-vel és más különböző kommunikációs szabványokkal. Leggyakrabban olyan közegben találkozhatunk vele, ahol szükséges a számítógéppel vagy más, soros porttal rendelkező eszközökkel történő kommunikálás. Ezek integrált áramkörök, amelyek a ma gyártott mikrokontrollerekbe már be vannak építve. Léteznek olyan chipek, amelyekbe két UART

is található, ezek a Dual UART-ok, DUART-ok. Ilyet találhatunk például az ATMEGA128-as mikrokontrollerben is. A legmodernebbek már szinkron módon is képesek kapcsolatot teremteni, ezeket USART-nak nevezik.

Az adatátvitel során valamiféle közegen keresztül kell eljuttatni a biteket a célállomáshoz. A távolság és a kiépítés költségei egymással egyenes arányban változnak, tehát, ahogy nő a kiépítendő távolság, úgy nőnek a kiépítés költségei is, így bizonyos esetekben a lehető legkevesebb csatornán kell átvinni az adatokat. A diplomamunkám során távolságból jelentkező problémák nem jelentettek tervezési szempontot, mivel a két kártya egymáshoz közel, fixen rögzítve helyezkedik el. A soros kommunikáció folyamán, mint azt már említettem, az eszközök nem párhuzamosan, hanem időben egymás után juttatják át az adatokat, egy csatornán keresztül. A fentebb tárgyalt UART alakítja át egymásba oda és vissza a párhuzamos és soros adatfolyamot. Az ehhez szükséges soros-párhuzamos átalakítást egy alapvető digitális technikai eszköz végzi, a léptetőregiszter, ami minden UART-ba be van építve.

Az UART-ra nem jellemző, hogy közvetlenül fogadna, vagy generálna olyan jeleket, amiket a csatornán keresztül átviszünk. Ezért általában egy külön áramkör felelős, hogy az UART folyamán használt logikai szinteket – ilyen például a TTL – átkonvertálja az átvitel jelszintjére. A jelszintekből megmondható, hogy minek alapján különböztethető meg számunkra a bit 0 vagy 1 értéke (ez nagyrészt egy adott feszültség vagy átmenet). Az RS-232, az RS-422 és az RS-485 például jelszint-szabványok. Az összekapcsolódás végbemehet „full-duplex” (küldés és fogadás is végbemehet egy időben) vagy „half-duplex” (csak vétel vagy adás lehet egy azonos időben) módon. Vezetéken kívüli más adatátviteli csatornák is léteznek: infravörös port, optikai kábel, Bluetooth (SSP). A modemeknél és a rádióvezérelt eszközöknél a jeleket sokszor modulálják, ami vezetéknél, vagy vezeték nélkül is megoldható.

A mikrokontrollerekben, napjainkban sokszor használják a RS-232 szabvánnyal beépített rendszerekben az UART-ot. Ilyenkor az UART TTL szintű jeleit MAX232 elnevezésű IC-vel konvertálják RS-232 szintű jelre.

Az aszinkron adatátvitelnél az UART-ok beállítás szerint először egy „start” bitet küldenek, azután a legkisebb helyértékűvel kezdve 5-8 adatbitet, majd egy opcionális paritásbitet és végül egy, másfél vagy két „stop” bitet. Ellentétes polaritású lesz a startbit az alapállapothoz képest, a stopbit azonos polaritású. A paritásbit feladata, hogy az egyesek száma mindig páros vagy páratlan legyen, de ez a bit el is hanyagolható. A páratlan paritás a megbízhatóbb, ami azzal magyarázható, hogy biztosítja azt, hogy legalább egy adatátmenet legyen, ami segíti az UART újraszinkronizálását.

A szinkron adatátvitelnél start-stop bitek helyett órajelet használnak a szinkronizálásra. Ez azért előnyös, mert így megnövekszik a bitfolyam információtartalma, viszont az órajelhez külön kábel szükséges. Ez nagyobb hatékonysággal is jár, mivel a bitek nagyobb része szállít információt és kisebb az adatvesztés esélye.

A bitrátával (másodpercenként átvitt bitek száma) szokás jellemezni az UART sebességét. Gyakori, hogy ezt a meghatározást összetévesztik Bauddal, ami az átvitt szimbólumok számát jelenti másodpercenként. Van átfedés a két mennyiség között, de nem használhatók egymás meghatározására. Az USB átalakítók akár 1 Mbps adatátvitelre is képesek, a legújabb számítógépek 115200 bps-ig terjedő sebességet valósítanak meg, míg a régebbi modemek és távírók 110-300 bps sebességre alkalmasak.

4.6. Serial Peripheral Interface Bus (SPI)

A Motorola által elnevezett Soros Periféria Interface Bus vagy más néven SPI Bus egy full duplex módban működő szinkronizált soros adatkapcsolati séma. master/slave módban kommunikálhatnak az eszközök, ahol a master eszköz határozza meg a data frame-et. Amennyiben több slave eszközt szeretnénk üzemeltetni, megtehetjük, ha használjuk a Slave Select PIN-ek megfelelő állását. Az SPI-t sokszor nevezik “négy vezetékes” soros busznak, ellentétben a többi három, kettő vagy egyvezetékes megoldásokkal. [8]

Az SPI négy különböző logikai jellel dolgozik:

SCLK — Serial Clock (Soros órajel)

MOSI/SIMO — Master Output, Slave Input (Master kimenet, Slave bemenet)

MISO/SOMI — Master Input, Slave Output (Master bemenet, Slave kimenet)

SS — Slave Select (Slave kiválasztó)

A kommunikáció megkezdéséhez a Master beállítja az órajelet, a Slave maximális órajelével azonos vagy annál kisebb frekvenciára. Ezek általában 1 és 70 MHz között mozognak. Ezután a Master a Slave Select-et nullára állítja. Amennyiben várakozási időre is szükség van, mint ahogy az analóg-digitális átalakításnál, akkor a Masternek várnia kell legalább addig, amíg ez a várakozási idő letelik, és akkor indíthatja az órajelet. Minden egyes SPI órajel kibocsátással együtt full duplex adatközlés is történik:
A Master küld egy bitet a MOSI csatornán, a Slave innen kiolvassa.
A Salve küld egy bitet a MISO csatornán, a Master kiolvassa innen.
Nem minden átvitelhez szükséges feltétlen ez a négy művelet, a gyakorlatban azonban ezt használják. Az átvitel bármennyi órajelciklust tartalmazhat. Ha már nincs átvivendő adat, akkor a Master leállítja az órajelet, majd lezárja a Slavet. Az átvitel általában 8 bites szavakban történik, amiket természetesen lehet többszörözni, ahogy ezt az átvitel megköveteli.

4.7. Inter-Integrated Circuit (I²C) [7]

Az I²C (Inter-Integrated Circuit) egy soros multi-Master computer busz, melyet a Philips vezetett be a kis sebességű perifériákhoz. 2006 októbere óta nem kell licensz díjat fizetni, hogy az I²C protokollt használjuk. Az SMBusz egy az I²C része, mely a szigorúbb elektronikai és protokoll szabályok betartásáért felelős. Ennek folyományaképp, a modern I²C rendszerek szabályokat és folyamatokat használnak az SMBusz révén.

Az I²C-nek mindössze két darab kétirányú Open-Drain csatornája van: Soros Adat (SDA) és Soros Órajel (SCL), melyek szintjét ellenállásokkal emelik meg. A leggyakrabban használt feszültség értékek a 3.3V és az 5V, bár a rendszer ezektől eltérő akár alacsonyabb, akár magasabb értékeket is elfogad.

Az I²C referencia alapján a 7-bites címmező 16 lefoglalt cím esetén összesen maximum 112 Node-dal képes kommunikálni ugyan azon buszon.

A legáltalánosabb I²C busz módok a 100kbit/s-os “Standard mode”, illetve a 10kbit/sec-es “Slow Mode”, mindamellett, hogy az órajelet akár az egyen szintig csökkenthetjük. A nemrégiben végzett vizsgálatok alapján az I²C több host-ot is ki tud szolgálni. Fast Mode-ban akár a “ 400kbit/sec sebességgel, vagy “Fast mode plus”-ban 1 Mbit/sec illetve High speed mode-ban 3,4 Mbit/sec sebességgel. Továbbá lehetőség van egyéb opciók alkalmazására is, mint a 10 bites címzés.

A megengedhető legtöbb node-ok száma természetesen a cím mező nagyságától függ illetve a busz kapacitásától,(400pF) ami a kommunikációs csatorna hosszát korlátozza általában néhány méterre.[8]

4.8. AVR Studio

A mikrokontrollerek alkalmazásának alapvető feltétele egy jól használható fejlesztői környezet, amelynek hiányában még a legkiválóbb áramkör is használhatatlan. A fejlesztői környezetek ma már kizárólag PC alapúak, a korábban használt célhardverek teljesen korszerűtlenné váltak és kiszorultak a piacról. Az egyszerűbb kontrollereknél assembler az inkább használatos, míg a nagyobb processzoroknál C nyelvben történik a fejlesztés.

A 8-bites RISC AVR mikrokontrollerek könnyebb fejlesztéséhez és használatához az Atmel cég az AVR Studio, ingyenes, grafikus szoftver fejlesztőkörnyezetet adta ki, amely a következő operációs rendszereken használható: Windows 9x/Me/NT/2000/XP. Én a 4.13-as verzióját használtam a munkám elkészítése során, Windows XP alatt. A forráskódot C nyelven írtam meg. A környezet tartalmaz egy assemblert és C-t támogató forrásfájlszerkesztőt szintaktikai kiemeléssel, projekt menedzsment támogatást, chip szimulátort és in-circuit emulátort. Az utóbbi kettő funkcióra a fejlesztés során nem volt szükségem. Az AVR Studio szoftver integráltan tartalmazza az AVRprog segédprogramot is, amely a mikrokontroller felprogramozásánál játszik szerepet. Ezzel a beépített programmal tudjuk a már hex fájlba lefordított programot rátölteni az eszközünkre, egy erre a célra kialakított programozó (TavIRisp STK500) segítségével. Az önálló működés miatt - és kompatibilitási okokból - a modul 19200 bps sebességgel kommunikál a PC-vel. Ekkor az AVRprog önállóan detektálja a programozót.

4.9. LabVIEW grafikus fejlesztői környezet [10]

A LabVIEW egy mérési, tesztelési és szabályozási feladatokhoz, valamint beágyazott rendszerek fejlesztéséhez használt grafikus fejlesztő környezet. A méréstechnikában népszerű szoftverként emlegetik, kiterjesztve a fent említett képességeket az ipari területekre is.
A LabVIEW-ban történő fejlesztés során egy grafikus Windows felületen drag&drop módszerrel építhetjük meg, gyorsan és egyszerűen a virtuális műszer kapcsolási rajzát, blokkdiagramját. Az így létrejövő virtuális műszerrel a számítógép perifériáin keresztül férhetünk hozzá a külvilághoz. Előnye, hogy a felhasználó által egyszerűen módosítható és olcsó az eszközök megvalósítása. Ezenkívül nincs szükség a teljes fejlesztői környezetre a már megvalósított projektekhez, mivel lehetőséget biztosít *.exe kiterjesztésű indítható fájl készítésére, amit bármely más PC-n futtathatunk.

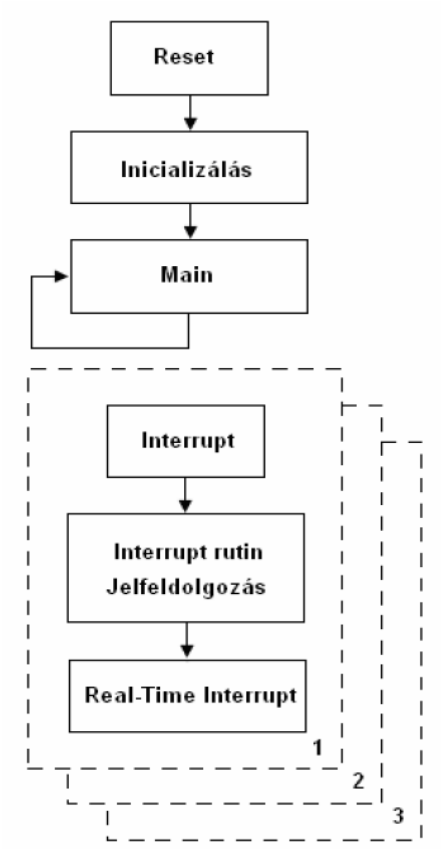
A fent említett szempontok miatt a meglévő rendszer számítógépes kezelői felületének fejlesztése LabVIEW környezetben történt, az önálló laboratórium keretén belül, ami a grafikus felület elkészítését egyszerűen teszi lehetővé, és számos, a feladathoz jól illeszkedő felhasználóifelület-elemet tartalmaz. A megvalósítás során vezérlő felületet biztosítottunk a felhasználó számára. A felhasználó által kiadott utasításokat feldolgozzuk, és eljuttatjuk a DSP kártyára, illetve a digitális potenciométer kártyára a megvalósított rendszernek megfelelően (5. ábra), így vezérelve a zajcsökkentést és a hangerőt.
Mivel ebben a fejlesztői környezetben a soros port kezelése is átgondolt és könnyen elsajátítható, ezért a kommunikációt soros porton keresztül oldottuk meg egy általunk kidolgozott és megvalósított egyszerű protokollal. A kommunikáció kétirányú. A számítógép parancsokat küld a kártyáknak. A DSP kártya pedig, az együtthatók lekérdezésénél visszaküldi azokat. Mivel nincs folyamatos beszéd a csatornán, a soros kábel levehető és később újra csatlakoztatható anélkül, hogy a rendszerben bármiféle zavart okozna.

5. A rendszer szoftverei

Ebben a fejezetben a rendszerben különböző architektúrákon futó programok struktúráját, lényeges működési mechanizmusát mutatom be. Először a meglévő rendszer programfelépítését és főbb elemeit írom le a tervező diplomamunkája alapján. Majd az általam elvégzett változtatásokat és a mikrokontrollerre megírt programot mutatom be.

5.1.1. DSP programok felépítése [1]

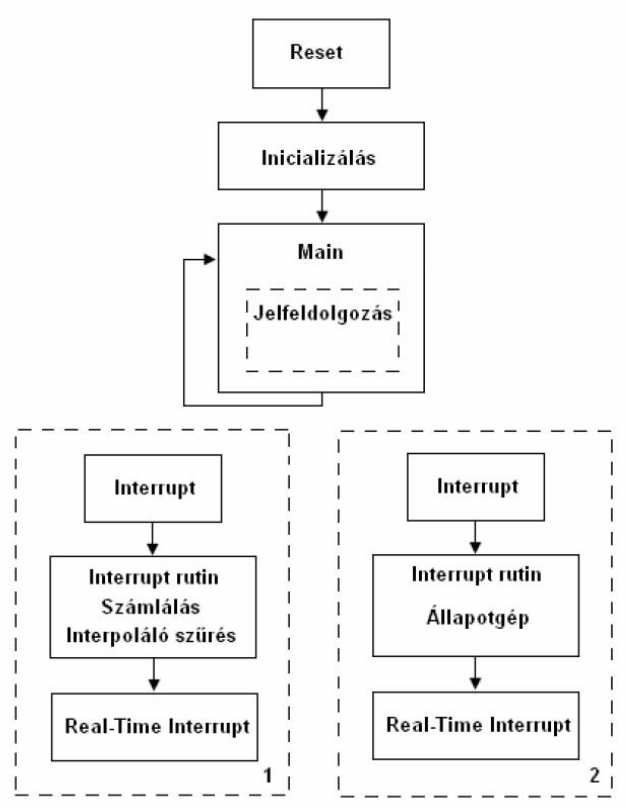
A jelfeldolgozó programok általános felépítése az 11. ábraalább ábrán látható a [5] alapján.



11. ábra Jelfeldolgozó programok általánosan alkalmazott struktúrája

A program futása reset után az inicializálással kezdődik. Ennek a lefutása alatt a külső megszakítások letiltásra kerülnek. Itt történik meg a processzor beállításainak rögzítése, a flagek, IO portok beállítása, az interrupt prioritások beállítása, a változók kezdőértéket kapnak. A processzor kezdeményezi külső áramkörök, mint például az A/D-D/A konverter elindítását. A megszakítások engedélyezése és a beállítások után elindul a főprogram. Általában a főprogramban csak egy végtelen while ciklus van, amelyben nincsenek műveletek. A processzor várakozik a megszakításra. Megszakítás esetén a processzor a kérést kiszolgálja, majd visszatér a főprogramba.

A fentebb írt általános struktúra alapján valósította meg a rendszer tervezője a DSP szoftvert, amit én a későbbiek során kiegészítettem. A felépítése a alábblátható:



12. ábra Zajcsökkentő program struktúrája

A tervező leírása alapján a program a következő fájlokból áll:

ANC.h: a felhasznált függvények prototípusait, a globális változók deklarálását tartalmazza.

Initialize.c: a processzor indulásakor szükséges függvények definícióját tartalmazó fájl. Beállítja a flag-eket, a megszakításvektorokat és elindítja az audio kodeket. Inicializálja a DMA-t. A DMA azért kell, hogy az AD/DA átalakítók közvetlenül hozzáférhessenek a memóriához.

ISR.c: ez a fájl tartalmazza a megszakításokat kiszolgáló rutint. Az A/D felől érkező megszakítás meghívja a Decimate() függvényt. A nyomógombok felől érkező megszakítás kezelését ebben a fájlban oldottam meg. A függvény megvizsgálja, hogy melyik gombot nyomta le a felhasználó és ettől függően léptet az állapotgépben. A főprogrambeli funkciókat flageken keresztül aktiválja. Ezek a következők:

- ident_mode: másodlagos út mérése
- fxlms_mode: zajelnyomás aktiválása
- coef0_mode: adaptív szűrő együtthatóinak nullázása

Main.c: a főprogram, reset után ez fut le először. A jelfeldolgozáshoz szükséges függvény- és változódeklarációkkal kezdődik. A változódefiníciónál feltölti a $W(z)$ adaptív szűrő tömbjét a coefs_w.dat, a másodlagos út $C(z)$ szűrőjét pedig coefs_c.dat fájlból. Ezeket a fájlokat a projekt könyvtárából olvassa be. Az identifikálás hangos és zajos folyamat, ha a demonstrációnál nem kívánjuk elindítani, lehetőség van a korábban identifikált és elmentett becslő betöltésével kezdeni a zajcsökkentést. A beolvasandó identifikálás eredményét a coefs_c.dat fájlban fract16 formátumban kell tárolni. A főprogram először meghívja az inicializáló függvényeket, majd belép egy végtelen while ciklusba. A cikluson belül három egység van, egyszerre csak egy indulhat el. Az egységeket elindító flagek állapotát a megszakításrutin állítja. A főprogram felépítése:

```
#Include
Változó és függvény deklarációk
Változó definíciók
Szűrőegyüttthatók betöltése
Void main(void) {
        Inicializálás
        while(1) {
                if(ident_mode) //identifikálás
                {
                        identifikáló fehér zaj generálás
                        identifikáló zaj szűrése átlapolódás ellen
                        tesztelt rendszer kimenetének beolvasása
                        az adaptív modell válaszának számítása
                        különbség képzés a rendszer és a modell között
                        új koefficiensek számítása LMS-el
                }
                if(coef0_mode){ szűrőegyüttthatók nullázása }
                if(fxlms_mode)
                {
                        input beolvasás
                        ellenzaj generálása
                        ellenzaj kiadása
                        referenciajel szűrése a másodlagos út becslőjével
                        új koefficiensek számítása LMS-el
}}}
```

Conv_asm.asm: assembly nyelven készült, annak érdekében, hogy kihasználja a processzor párhuzamosítási lehetőségeit a számítás során. FIR szűrést megvalósító assembly függvény. Paramétereit a főprogramtól programtól kapja.

Lms_asm.asm: assembly nyelven készült, hasonló okokból, mint a conv_asm függvény. LMS algoritmust megvalósító assembly függvény. Paramétereit a főprogramtól kapja. Az assembly függvényeket a C program egyszerű függvényként látja. A fejlesztőkörnyezet előírja, hogy a C függvény argumentumait az assembly függvény a processzor R0, R1, R2 regiszterekben kapja meg. Háromnál több argumentum átadása mutatókkal valósítható meg.

5.1.2. DSP program bővítése

Ahhoz, hogy hatékonyan tudjam a meglévő programrendszert bővíteni, először át kellett tanulmányoznom és meg kellett értenem a rendszer működését. A fejlesztés során nem volt szükség az összes projektben meglévő fájl bővítésére. Ezért ebben a fejezetben csak a változtatásokra térek ki.

A demonstráció során nem mindig áll módunkban a zajforráshoz függvénygenerátort használni. Ez egy különálló labor eszköz, használata jelentősen csökkenti a bemutató rendszer mobilitását. A DSP függvénygenerátorként való kiegészítése jelentette a probléma megoldását. Ehhez három változtatásra volt szükség:

1. Zajjel függvény megvalósítása

   Mivel a zajcsökkentő algoritmus a meglévő tesztek alapján a szinuszjelet nyomja el a legsikeresebben, ezért a tanszéki laborban már használt sin2pi_fr16 függvényt illesztettem be a programba. Ez a függvény egy külön fájlban kapott helyet (**sin2pi_fr16.c**).

2. A szinuszfüggvény felhasználása és engedélyezése

   A **Decimate.c** fájlban hívódik meg a sin2pi_fr16 függvény. Ezért itt kell egy feltételes utasítással engedélyezést vizsgálnunk, hogy külső vagy belső zajforrást használunk. Ezt a beállítást a kijelzős vezérlő egységgel változtathatjuk. Engedélyezés esetén mindig, amikor mintát olvasunk be az AD átalakítóról, egyúttal a folytonos szinusz következő értékét is kiszámítjuk.

3. Az előállított szinuszfüggvény felhasználása

   A DSP számára a referencia jelet a szinuszfüggvény értékeiből adjuk meg. Ezenkívül a DSP RIGHT OUT csatornájára kapcsoljuk a zajjelet, így azt a zajhangszóróra vezetve megoldottuk a zaj forrását.

A rendszer továbbfejlesztése során soros kommunikáció használata elengedhetetlenné vált. A DSP kártya két UART-jára egyaránt szükség volt. Mind a két interfészhez

szükséges inicializáló függvényt a **main.c** fájlban helyeztem el. Az adatok fogadását a soros portról interrupt segítségével oldottam meg, aminek a handler függvényét az **ISR.c** fájlban helyeztem el. Ebben, a receive handler függvényben switch-case szerkezettel megvalósított állapotgép segítségével dolgozom fel a bejövő parancsokat és állítom be a megfelelő változókat.

Az általam végzett bővítéseket C nyelvben írtam meg. A forráskód elkészítése során mindvégig ügyeltem arra, hogy megfelelő kommentekkel lássam el, ezáltal biztosítva a gyorsabb továbbfejleszthetőséget.

5.2. Mikrokontroller programok felépítése

A mikrokontrolleren futó program struktúráját hasonlóképpen valósítottam meg, mint a DSP programét. Ügyeltem arra, hogy itt is a jól elkülöníthető funkciók külön eljárásban és külön fájlokban legyenek. Így a következő fájlokból épül fel a megvalósított projekt.

Main.c: a mikrokontroller bekapcsolás után a main() függvényt indítja el. Itt először az inicializálási rész fut le, ahol beállításra kerülnek az alábbiak:

- Lábak ki- vagy bemenetbe állítása (init_labak()).
- Interrupt módok beállítása (init_IT()).
- Soros kommunikáció inicializálása (UART_init()).
- LCD kijelző inicializálása (lcd_c_init4()).
- Változok kezdőértékének beállítása.

Az inicializálási rész alatt a megszakítások tiltásra kerülnek. Ezek után a mikrokontroller programokra jellemzően egy végtelen ciklusban fut tovább a processzor, várva a megszakítások érkezését.

A végtelen ciklusban különböző feltételes utasítások vannak. A feltételek állapotai határozzák meg, hogy a menü melyik részében vagyunk. A beviteli eszközök hatására érkező megszakítások által állított globális változók hatására, végrehajtja a kiválasztott parancsot. Frissíti a kijelzőn megjelenő menüt a kívánt módosítások alapján. Vezérlő parancsot küld a soros vonalon a DSP felé. Törli a megfelelő globális változók értékét, ezzel jelezve, hogy a kívánt parancs már lefutott, majd tovább várja a megszakítások érkezését.

Init.c: A main elején lefutó inicializáló függvények egy részét tartalmazza. A mikrokontroller megfelelő regisztereinek beállításával érhető el a lábak irányának megadása. Valamint a megszakításvektorok beállítását tartalmazza.

Interrupt.c: Ez a fájl tartalmazza a megszakításokat kiszolgáló rutint. Az ATMEGA16 mikrokontrollernek 3 külső megszakítása van. Ebből kettőt az enkóder állapotának változása juttat érvényre. Az enkóder elfordulása esetén, ugyanazon az enkóder lábon lefutó vagy felfutó él generálódik. Az egyik IT rutin a lefutóra, a másik pedig a felfutó élre van beállítva. Mind a két rutinban lekérdezésre kerül az enkóder második lába is, ami alapján eldöntjük, melyik irányban is fordult el valójában. A harmadik külső megszakítást a gombok lenyomása váltja ki. Minden gomb lenyomás esetén, beleértve az enkóderen találhatót is, egy VAGY kapun keresztül megszakítást generál. A lekezelő rutinban szintén lekérdezve a mikrokontroller lábainak állapotát, megmondható, hogy melyik gomb került lenyomásra. Miután eldöntöttük, hogy melyik esemény következett be, a megfelelő flagek állításával az érvényre jut a főciklusban.

Lcd.c: A karakteres LCD használatához szükséges függvényeket tartalmazza. Inicializálás, pozíciómegadás, karakterkiíratás, várakoztatás.

Uart.c: A soros kommunikáció használatához szükséges függvényeket tartalmazza. Inicializálás, karakterküldés.

Altalanos.h: A felhasznált könyvtári függvények bejegyzését, az általam megírt függvények prototípusát tartalmazó header fájlok bejegyzését, a globális változók deklarálását, konstansok deklarálását tartalmazó header fájl.

Lcd.h: A felhasznált LCD függvények prototípusait tartalmazza.

Uart.h: a felhasznált soros kommunikációhoz szükséges függvények prototípusait tartalmazza.

A fájlok kiterjesztéséből is jól látható, hogy a program fejlesztését C nyelven valósítottam meg.

6. A kijelzős egység menüje

6.1. Menüvel szembeni elvárások

Az alapvető tervezési szempontok a következők:

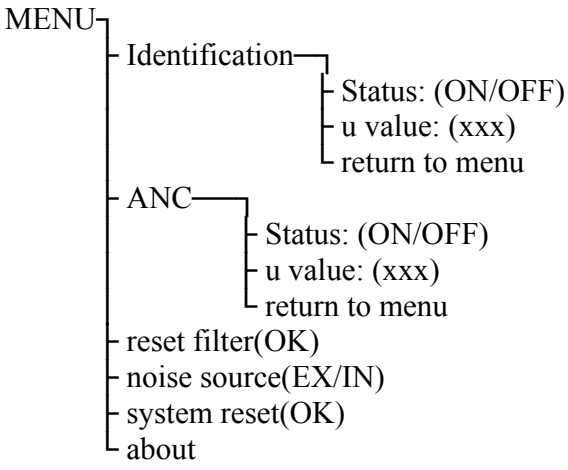
- Egyszerű legyen.
- Felhasználónak mindig tudnia kell, mikor és merre léphet tovább.
- Könnyen lehessen benne navigálni.
- Könnyen áttekinthető legyen.
- Minden funkció elérhető legyen.
- Megfelelő esztétikai megjelenés valósuljon meg.
- Képernyő mindig ugyanolyan struktúrájú legyen.
- A menü illeszkedjen az elérésének módjához (nyomógombokhoz).
- Használata egyértelmű legyen.
- A mindennapi életben használt menükhöz hasonló legyen, ezáltal gyorsabban elsajátítható a felhasználó számára.
- Nyelvtanilag helyes, érthető kifejezéseket használjunk.
- Helyet kell biztosítani a további fejlesztések számára.

Mindezeket és a kijelző fizikai méreteit figyelembe véve kell megtervezni a menüt.

6.2. Menü megvalósítása

Az 6.1. Menüvel szembeni elvárásokösszefoglaltak alapján az alábbi menüt valósítottam meg.

A rendszer állapotainak vezérlésére nem szükséges nagyszámú menüelem, ezért a megvalósítás során kerültem a hierarchikus felépítés túlzott elmélyítését. A menüelemek sorrendje használati sorrendben vannak elhelyezve. A rendszerben használt alfanumerikus kijelző elegendő megjelenítési felületet biztosít ahhoz, hogy a menüelemek feliratából asszociálhassunk az általuk nyújtott funkcióikra. Ezek alapján a menü a következőképpen épül fel:



Az első kettő és az utolsó menüelem használatánál jelenik meg egy új menü, amelyekben a főmenühöz hasonlóan navigálhatunk. A többi főmenü elem használata során az elemnév mellett jelenítjük meg a kívánt változtatást. A menü tetszőlegesen bővíthető új elemek felvételével, teret adva a későbbi fejlesztéseknek.

A menüben való barangoláshoz két nyílból összeállított kurzor nyújt segítséget, amelyek közrefogják az aktuális menüelemet. A kurzor mozgatásához használhatjuk a kijelző alatt elhelyezett nyomógombokat. A menüben megjelenő számértékekre állva (μ értékének változtatása), azt az enkóder segítségével változtatathatjuk. Az OK gomb lenyomásával juttatjuk érvényre az értékeket, ekkor továbbítódnak a DSP kártya felé.

Három fő részre osztottam az információ megjelenítés szempontjából a kijelzőt. Legfelül, középen található az éppen aktuális menü elnevezése. Alatta helyezkedik el a kijelző nagy részét elfoglaló, menü elemeket megjelenítő terület. A fennmaradó két oldalsó sáv alkotja a harmadik részt. Ennek a területnek a bal oldalán felül jelenítem meg a kurzor által birtokolt menü elem számát. Alul pedig a bal oldali gomb funkcióját (-). A jobb felső részben jelenítem meg, hogy az aktuális menüben hány elem van összesen. Alul pedig a jobb oldali gomb funkcióját (+). Ez a fajta elrendezés biztosítja, hogy a menü több elemet is tartalmazhat, mint amennyi aktuálisan elfér a kijelzőn egyszerre. Erre például szolgál a főmenüt tartalmazóalább, ami szintén görgethető menü.

13. ábra Megvalósított főmenü

7. Különböző rendszerek és programok közötti kommunikáció

A teljes rendszerben két helyen valósul meg a sorosporti kommunikáció. Az egyik a számítógépen futó LabVIEW és a DSP kártya között, amelyet az önálló laboratórium keretein belül valósítottunk meg. Ennek analógiájára készítettem el a másik kommunikációt a DSP kártya és a kijelzős vezérlő egység között. Mindkét esetben az alábbi követelmény fogalmazódott meg:

- Legyen egyszerűen bővíthető a fejleszthetőség érdekében.
- Minél kevésbé terhelje a DSP-t.
- Legyen „önszinkronizáló”: ha valamilyen hiba történik az átvitel során, ne kelljen újból indítani a rendszert, hanem a kommunikáció lehetőleg minél kevesebb parancs elvesztése árán álljon helyre.

Ennek megfelelően az alábbi protokoll került megvalósításra:

1. szinkron byte
2. parancs byte
3. 2 adat byte

Kötelezően minden üzenetnek a szinkron byte-tal kell kezdődnie: amennyiben nem így történt, megvárjuk, míg jön egy ilyen karakter, és az utána következő karaktereket fogjuk csak parancsként értelmezni. Természetesen előfordulhat, hogy a szinkron byte valamiért elveszik, és egy adatot, mely megegyezik annak értékével, szinkronkarakterként értelmezünk, és így a többi ezután beérkező adatot rosszul értelmezzük. Ez azonban csak akkor jöhet létre, hogyha az átvitel során két byte is elveszett, mivel a szinkron után következő értelmezhető parancs byte-ok között nem szerepel a szinkronkarakter. Tehát csak akkor jöhet létre ez az állapot, hogyha a parancs byte után következő 2 adat byte közül az első pont a szinkron byte, a másik pedig az egyik parancs byte értékével egyezik

meg, melynek valószínűsége kb.: (9/256)*(1/256)*(1/5) = 0,00003. (9 értelmes parancs byte van, és 1/5-öd a valószínűsége, hogy a 4 byte-ból az első kettő veszik el) Így emiatt, illetve a feladat nem kritikus jellege miatt nem szükségszerű az átvitel hibatűrőbbé tétele.

A parancs byte határozza meg az adat byte értelmét, illetve vannak olyan parancs byte-ok, melyek után nem következik értelmezett adat byte, csak 2 „dummy” karakter. (pl. Zajcsökkentés/Identifikáció, ki/bekapcsolás, együtthatók nullázása). A számítógép és a DSP közötti kommunikáció esetében parancs byte dönti el azt is, hogy az adott parancs a potenciométer kártyának, vagy a DSP-nek szól.

8. Mérések és eredmények

Látványos mérési eredményeket a diplomamunkám jellegéből kifolyólag nem tudok bemutatni. A megtervezett nyomtatott áramkör az alkatrész beültetése után készenállt a tesztek elvégzésére.

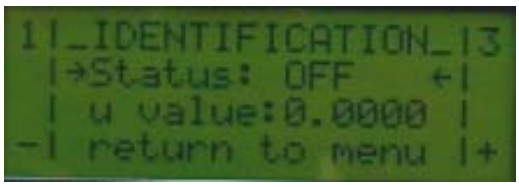
Először a DSP kártyához való illesztést próbáltam ki. A terveknek megfelelően illeszkedett a két kártya egymáshoz, valamint a külön álló kijelző kártya és a gombsor is jól csatlakozott a vezérlőegységhez. Így elvégezhettem a további teszteket, amelyek már a működésre irányultak. A rendszer bekapcsolása után az SPI interfészen keresztül sikerült rátöltenem a mikrokontrollerre a vezérlő szoftvert. A program futtatásakor sikeresen kirajzolódott a menü a kijelzőre, amiben a gombok segítségével navigálni is tudtam.

14. ábra Kijelzős vezérlőegységgel ellátott DSP kártya

A továbbiakban bemutatom a rendszer kezdőállapotból (15. ábra) való elindulását, egészen a zajcsökkentés megvalósításáig és a rendszer nullázásáig. Az ehhez szükséges lépéseket a menü ábráival illusztrálom.

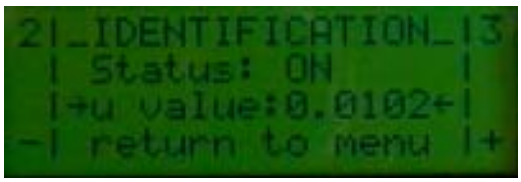
15. ábra Menü kezdőállapota

A kezdőállapotból a középső OK gomb lenyomása után az identifikációs almenübe lépünk bele, amit az alábbi ábra mutat. Itt elindíthatjuk az identifikációt és beállíthatjuk az identifikáló μ-t.



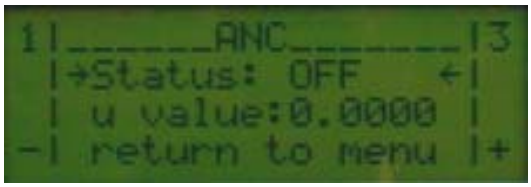
16. ábra Identifikációs almenü

A státuszon állva az OK gomb lenyomásával elindíthatjuk az identifikációt. Ekkor az OFF felirat átvált ON-ra. (17. ábra) A menü elemek váltását a két szélső gombbal érhetjük el. Az identifikáló μ-t a második sorban állíthatjuk az enkóder segítségével. A kívánt érték elérése után az OK gomb lenyomásával jutathatjuk érvényre a DSP kártyán.



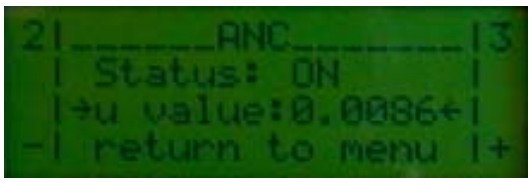
17. ábra Identifikációs almenü ON

Identifikáció után már kész a rendszerünk a zajcsökkentés bemutatására. Ebben az almenüben is kikapcsolhatjuk az identifikációt, vagy a zajcsökkentést elindítva automatikusan megtörténik. Az almenükből való visszatérés a főmenübe a legalsó menüelemmel érhető el. (return to menu)



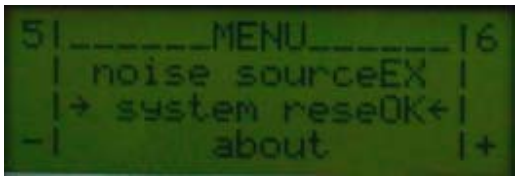
18. ábra ANC almenü

A zajcsökkentés elindítását az ANC almenüben tehetjük meg. Itt az előzőhöz hasonló módon van lehetőségünk elindítani a zajcsökkentést és állítani a μ-t. (19. ábra)



19. ábra ANC almenü ON

Amennyiben a rendszert az eredeti állapotba kívánjuk visszaállítani, a főmenüben található system reset elemet kell kiválasztanunk. (20. ábra) Az OK gomb megnyomása után leállít mindenféle műveletet a DSP-n és nullázza a felhasznált μ-k értékeit.



20. ábra System reset

A tesztek során megállapítottam, hogy a beviteli illetve kiviteli eszközök mindegyike megfelelően működik. A mikrokontroller az általam kiadott utasításokra az elvárt módon reagál. A kijelző megfelelő gyorsasággal, hibamentesen ábrázolja a menürendszert. A megtervezett menü egyértelműen, jól kezelhető. A menüelemek mindegyike elérhető. A kommunikáció során a vezérlőegység által kiadott utasítások mindegyike megjelent a DSP kártyán, de még is valami oknál fogva a zajcsökkentés bizonyos funkciói nem megfelelően viselkedtek. Egyenlőre úgy tűnik, hogy a kommunikációs kapcsolatban lehetnek hibák, de a hibakeresés jelen fázisában nem azonosítható be egyértelműen a hiba, amit természetesen a továbbiakban szeretnék megoldani.

9. Összefoglalás

Diplomamunkám kereteiben sikerült továbbfejlesztenem a meglévő zajcsökkentést demonstráló rendszert. Megterveztem és megépítettem egy olyan általános célú eszközt, amellyel látványosabb és kényelmesebb vezérlés valósítható meg. Sikerült csökkentenem a demonstrációhoz szükséges eszközök számát, ezáltal növeltem a mobilitást.

A feladat megoldása során megismerkedtem az aktív zajcsökkenés elméleti hátterével és a meglévő bemutató rendszerrel, továbbá a mikroprocesszor családok, valamint a felhasználható interfészek és protokollok általános ismereteivel. Megszereztem a kiválasztott jelfeldolgozó processzor és mikrokontroller programozásához szükséges ismereteket, majd elvégeztem a DSP programban szükséges változtatásokat, valamint implementáltam a mikrokontrolleren futó programot.

Dolgozatomban bemutattam az aktív zajcsökkentés felhasználási területeit és alapvető struktúráit. Ezek után rátértem a meglévő rendszerben alkalmazott FXLMS algoritmusra. Bemutattam még három, a rendszerbe beletervezett kommunikációs lehetőséget. A kijelzős vezérlőegység tervezéshez szükséges megfontolások ismertetése után a megtervezett hardver eszköz működését mutattam be. Ismertettem a mikroprocesszor programok működését és a tervezésükhöz szükséges megfontolásokat, valamint a megvalósított mikrokontroller szoftvert és a jelfeldolgozó szoftver módosításait.

Elmondható, hogy sikerült olyan rendszert létrehozni, amely működése a kiírásban megfogalmazott céloknak megfelel.

További fejlesztési lehetőségeket illetően szóba jöhet más DSP-n futó algoritmus kipróbálása, esetleg az algoritmusok közötti váltás biztosítása, amit a kijelzős vezérlőegységgel lehetne kiválasztani.

A DSP által generált szinuszos zajjel frekvenciájának változtatása, esetleg más hullámformájú jelek előállítása és ezek közötti váltás szintén a vezérlőegységgel.

Az identifikáció és az elnyomás során kívánt különböző hangerőt a program automatikusan állítsa be, illetve lehetőség lenne arra is, hogy a mikrofon túlvezérlése esetén (FS minták egymás utáni érkezése az AD-ról) a DSP kártya automatikusan hangerőcsökkentést eszközölhessen az erősítőn, illetve a jobb jel/zaj viszony beállítása érdekében a maximális hangerőt állítsa be, ahol még nincs túlvezérelve a mikrofon. Ez utóbbi fejlesztési lehetőség a DSP és a digitális potméter kártya közvetlen összeköttetésével érhető el vagy a számítógép ismétlőként való felhasználásával. Megvalósítható lehetne ebben az esetben az is, hogy a vezérlőegységgel kézi állításban változtathassuk a hangerőt.

Irodalomjegyzék

[1] Horváth András, „*Aktív zajcsökkentést demonstráló rendszer tervezése,* Diplomamunka”, 2007

[2] S. M. Kuo – R. Morgan, „Active Noise Control: A Tutorial Review”, *Proceedings of thee IEEE*, Vol.87, No. 6, June 1999, 944-954. o.

[3] Analog Devices, „*ADSP-BF534 ADSP-BF536 ADSP-BF537 Blackfin Embedded Processor Data Sheet (Rev. C).pdf”*

[4] Analog Devices, „*C/C++ Compiler and Library Manual for Blackfin Processors”*

[5] Molnár Károly, Sujbert László, „*Jelfeldolgozó processzor alkalmazása, Mérési útmutató*”, 2006.

[6] Dr. Kovács F. Ferenc; „*Az informatika VLSI áramkörei*”; Pázmány Egyetem Elektronikus Kiadó 2004

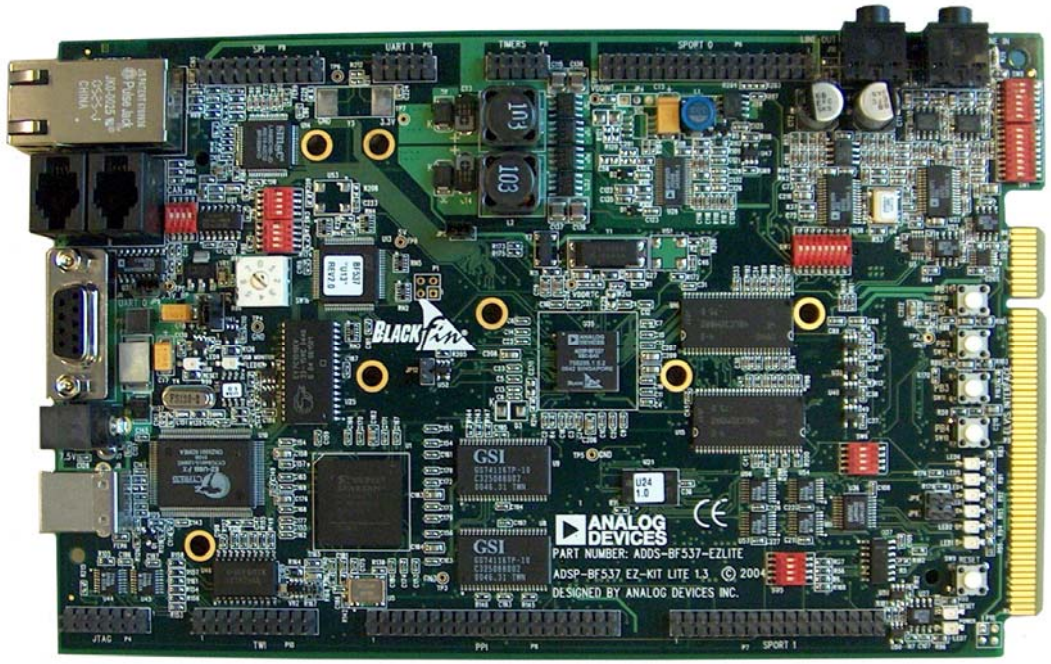
[7] Atmel® AVR ***ATmega32L Datasheet***: *www.atmel.com*

[8] Mikrokontrollerek és interfészek: http://wikipedia.org

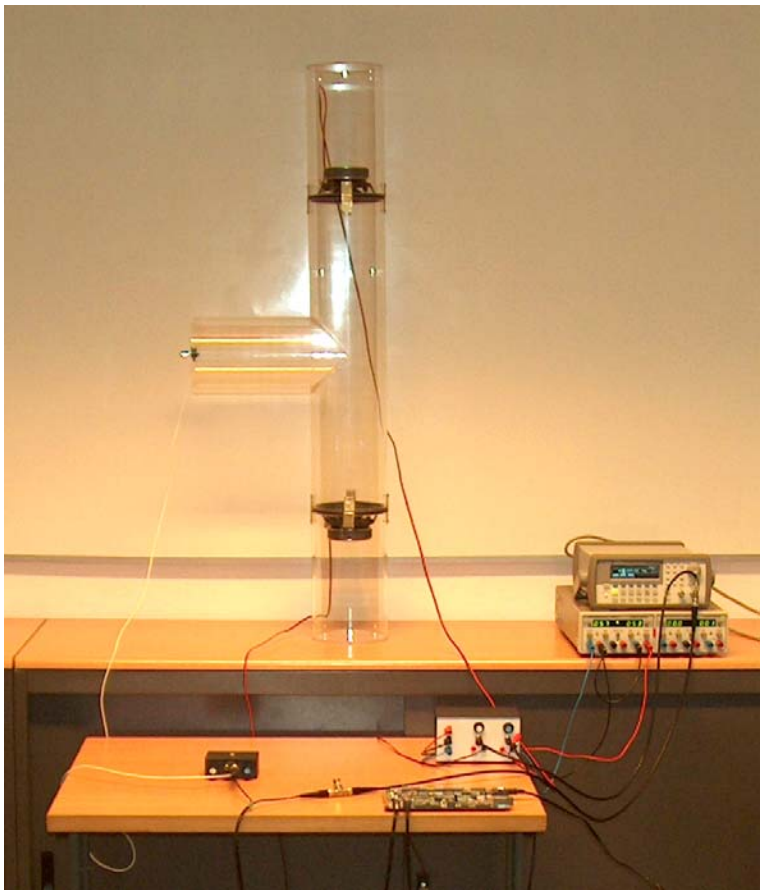
[9] TávIR mikrokontroller programozás és információk: http://avr.tavir.hu/index.php

[10] National Istruments, LabVIEW grafikus fejlesztői környzet: http://www.ni.com/

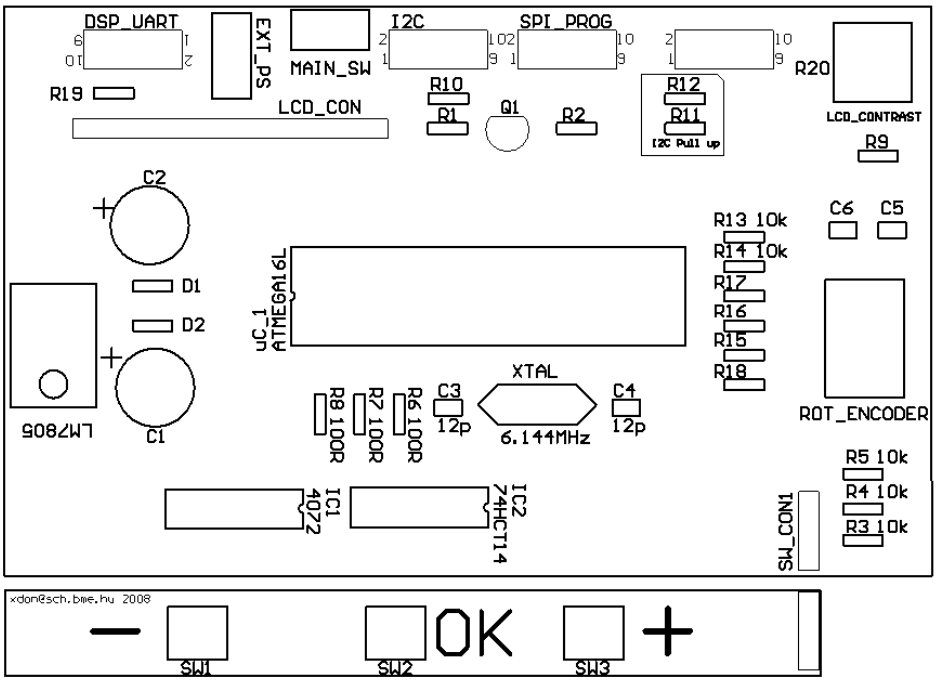
Függelék



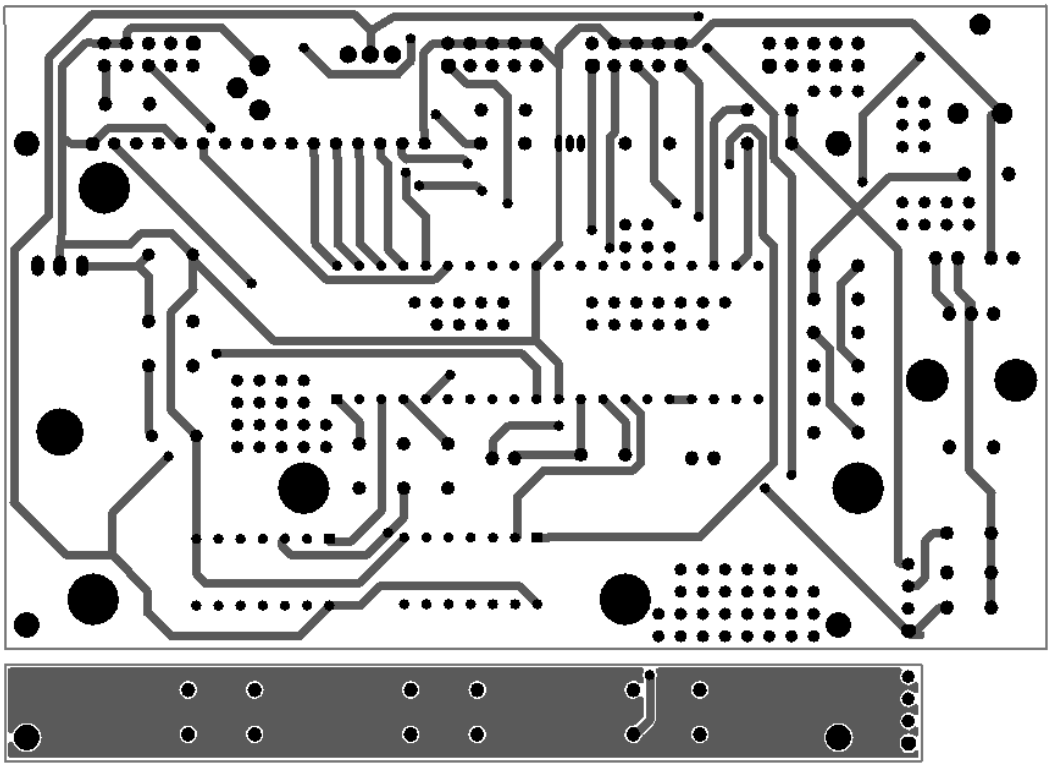
F1. ábra. BF537-EZLITE fejlesztőkártya



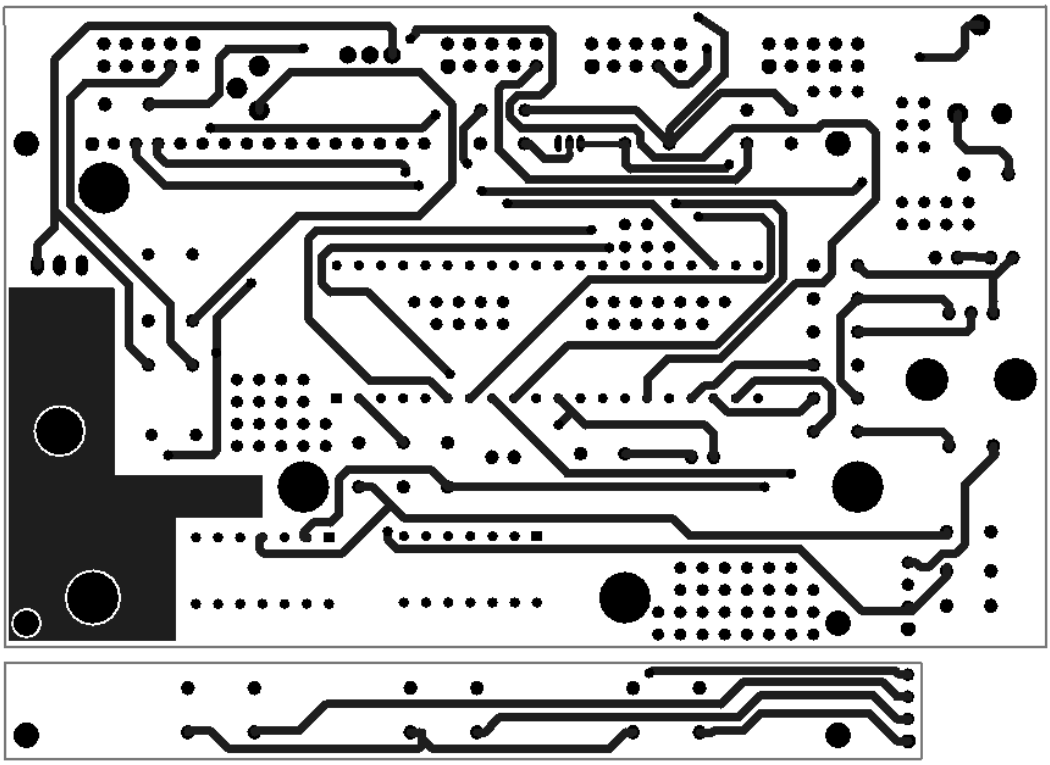
F2. ábra. Meglévő demonstrációs rendszer



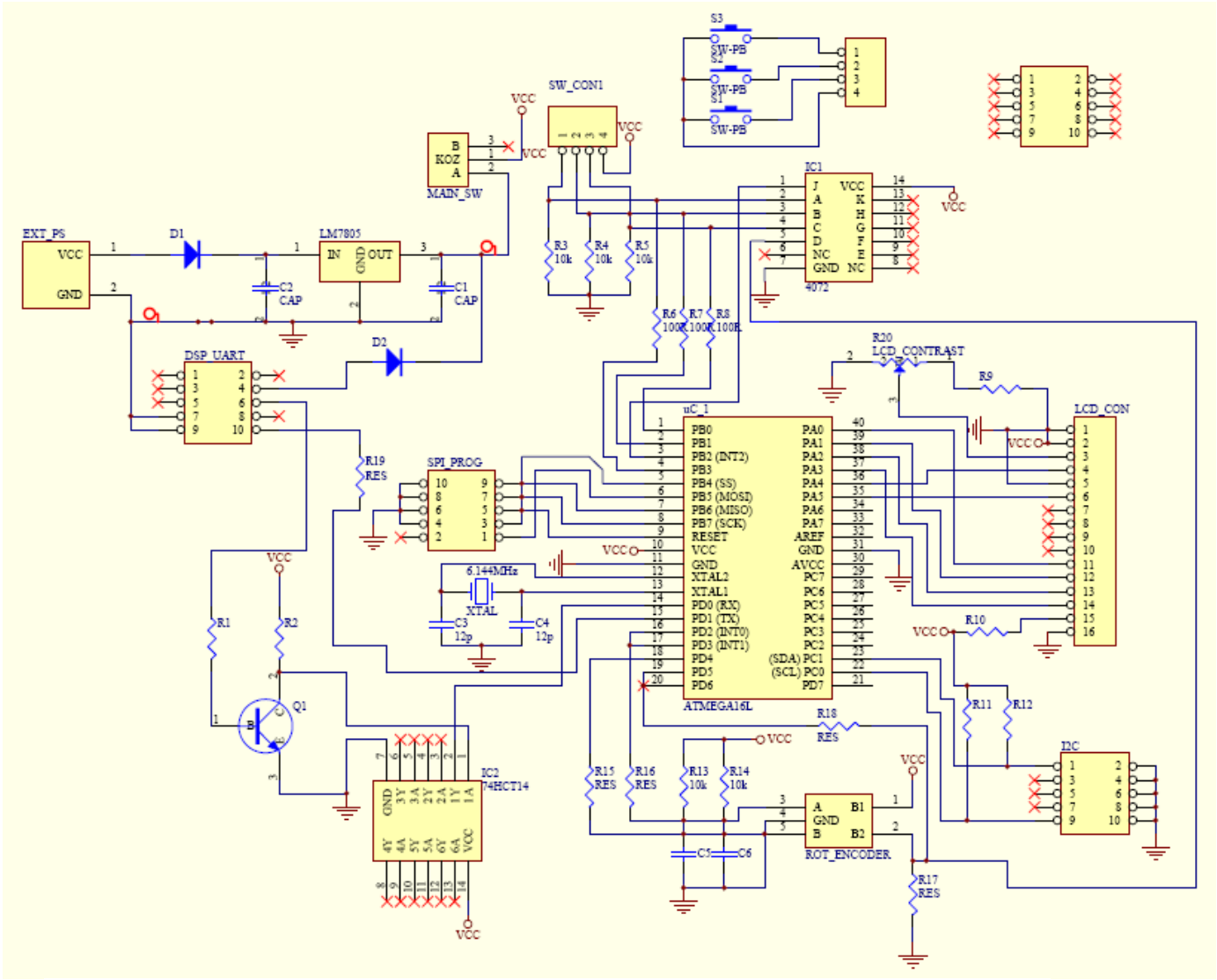
F3. ábra. Top overlayer (kijelzős vezérlőegység)



F4. ábra. Bottom layer (kijelzős vezérlőegység)



F5. ábra. Top layer (kijelzős vezérlőegység)



F6. ábra. Kijelzős vezérlőegység kapcsolási rajza

Alkatrészlista a kijelzős vezérlőegységhez

Sorszám	Mennyiség	Alkatrész	Érték/Típus
1	2	C1,C2	u
2	2	C3,C4	12p
3	2	C5,C6	130p
4	1	R1	6,12k
5	1	R2	5k
6	6	R3,R4,R5,R13,R14,R17	10k
7	2	R11,R12	2,1k
8	1	R9	5
9	1	R10	50
10	6	R6,R7,R8,R15R16,R18	47k
11	2	D1,D2	(dióda)
12	1	R20	10k (potméter)
13	1	BC182	(tranzisztor)
14	1	XTAL	6.144MHz
15	1	MAIN_SW	(kapcsoló)
16	1	EXT_PS	(külső tápcsatlakozás)
17	4	2soros tüskesor	2x5
18	1	1soros csatlakozó aljzat	1x16
19	1	1soros csatlakozó aljzat	1x4
20	1	forgatható enkóder	()
21	1	LM7805	5V-os
22	1	4072	(vagykapu)
23	1	74HCT14	(inverter)
24	1	ATMEL16	(mikrokontroller)
25	1	EW20400YLY	(4x20-as kijelző)
26	3	SW1,SW2,SW3	(nyomógomb)