Diplomaterv

Dolgozat címe: Beszéddetektor algoritmus felhasználása taktilis kijelzőben

Konzulensek neve: Tihanyi Attila, Takács György

Intézmény neve: Pázmány Péter Katolikus Egyetem Információs Technológiai Kar

Dézsi István András, 2008

PÁZMÁNY PÉTER KATOLIKUS EGYETEM

INFORMÁCIÓS TECHNOLÓGIAI KAR

DIPLOMATERV-TÉMA BEJELENTÉS

Név: **Dézsi István András**

Tagozat: **nappali** Szak: **Műszaki Informatika**

Témavezető neve: **Dr Takács György, Tihanyi Attila**

A dolgozat címe: **Beszéddetektor algoritmus felhasználása taktilis kijelzőbe**

A dolgozat témája:

A beszéddetektorral kapcsolatos konferencia, folyóirat és szakkönyvben megjelent irodalmak tanulmányozása. A fellelt beszéddetektor algoritmusok osztályozása, műszaki megvalósíthatóság alapján. Az algoritmusok osztályozásánál elsődleges szempontként vegye figyelembe a taktilis kijelzőben történő közel real-time alkalmazás megvalósíthatósági szempontjait. A kiválasztott algoritmusok számítógépes szimulációs tesztkörnyezetben történő implementálása, a diplomaterv tárgyát képező kijelzőbe történő illesztés megvalósíthatóságának tanulmányozása. Az implementált beszéddetektorokkal végezzen méréseket, az összegyűjtött és rendszerezett mérési eredmények alapján válassza ki a tárgy megvalósíthatóságának legmegfelelőbbet, és azt taktilis kijelző környezetében valósítsa meg. Az algoritmusokat a gyorsaságuk és erőforrás igényük alapján osztályozza, ez alapján legyen kiválasztva a kijelző számára megvalósítandó beszéddetektálási eljárás. Erre a feltételre a taktilis kijelző kis méretű, hordozható mivolta révén van szükség.

A témavezetést vállalom:

.....
(a témavezető aláírása)

Kérem a diplomamunka témájának jóváhagyását.

Budapest, 2007. ……

.....
(a hallgató aláírása)

A diplomamunka-témát az Információs Technológiai Kar jóváhagyta.
Budapest, 2007. ……

.....
Nyékyné dr. Gaizler Judit
Dékán

A diplomatervet átvettem:

Budapest, 2007…….

.....
(a témavezető aláírása)

Nyilatkozat

Alulírott Dézsi István András, a Pázmány Péter Katolikus Egyetem Információs Technológiai Karának hallgatója kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, és a diplomamunkában 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. Ezt a Diplomamunkát más szakon még nem nyújtottam be.

……..……..
(a hallgató aláírása)

Tartalomjegyzék

Kivonat

A Pázmány Péter Katolikus Egyetem Információs Technológiai karán indult évekkel ezelőtt egy olyan projekt, amely a siket és halláskárosult embereknek hivatott a mindennapi életüket könnyebbé tenni. Olyan eszközök és módszerek fejlesztése a célunk, melyekkel elősegítjük ezen emberek számára a külvilággal való kommunikációt azáltal, hogy pótoljuk a kiesett érzéküket - a mi esetünkben a hallást - valamely más érzék útján, például a látással vagy a tapintással.

Az én feladatom egy ilyen eszköz kibővítése, illetve hatékonyabbá tétele. A szóban forgó eszköz egy úgynevezett vibrotaktilis kijelző, amely a szájról olvasást segíti elő azáltal, hogy olyan információkat szolgáltat, amelyek ilyen módon nem láthatóak, konkrétan a hangszalagok rezgését, amely a zöngés-zöngétlen hangokat képzi. A taktilis kijelző úgy valósítja meg ezen adatok átvitelét, hogy a hangjeleket tapintással érzékelhető taktilis rezgőmozgások ingerévé alakítja, tekintettel arra, hogy az alapvető érzékelési csatornát, a látást ne zavarjuk. Ezek az ingerek a beszédhang zöngés és zöngétlen hangok közötti akusztikus átmeneteit közvetítik, ezáltal az eszköz viselője könnyebben tudja szegmentálni a beszédfolyamot, ami elősegíti a szájról olvasást.

A kijelző a hordozhatóság érdekében egy saját processzorral rendelkezik, amelynek memóriája foglalja magába a vezérlőprogramot. Egy USB csatlakozón keresztül van lehetőség a PC környezetben megírt programok mikrokontrollerre való felöltésére, és futtatására. A chip firmware kódját már kibővítettem az USB kapcsolat felvételét, és a kétirányú kommunikációt megvalósító rutinokkal.

A gyakorlati alkalmazás során felmerülő, a kijelzővel szembeni alapvető követelmény az, hogy zajos környezetben is megbízhatóan közvetítse a használó számára fontos információt, amihez viszont szükség van a beszédhangok érzékelésére. Ezt egy gyors és hatékony beszéddetektor algoritmus integrálásával lehet elérni.

A vonatkozó irodalom áttanulmányozásával sikerült egy megfelelő algoritmust találni, amely a mel-cepstrum modulácós energiát (MCME) használja fel arra, hogy a beszédet megkülönböztesse a zajtól. Ezzel a módszerrel a mel frekvencia skálán elemezzük a beérkező, szegmentált hangjeleket, amely egy rendkívül hatékony eljárásnak bizonyult a real-time alkalmazhatóság terén.

Az algoritmust először szimulációs környezetben valósítottam meg, majd többféle mintavételi frekvenciájú, és bitábrázolású bemeneti hangfájlokon teszteltem. Ezek között szerepelt a tiszta, zajmentes beszédhang, a különféle hangszerekkel játszott zene, a forgalmas helyeken

jellemző zajok (pl autó, metró), valamint az ezekkel a zajokkal terhelt beszédhang. Más kipróbált algoritmusokkal szemben, a mérési eredmények meggyőzőek voltak, és indokolták a kijelző natív vezérlőprogramjába való integrálását.

A célhardverbe való implementálás előtt azonban szükség van egy PC környezetbe való C nyelvű megírásra is, amely már közelebb áll a mikrokontroller architektúrájához, így lemérhető a futási idő, és a memóriaigény.

Abstract

At the Pázmány Péter Catholic University, a project started a few years ago, which aims to help persons in they everyday life, who are deaf, or have an aggrieved hearing potential. Our goal is to develop devices and methods, which helps these persons to communicate, in the way of replacing they missing senses - in our case, the hearing - with another one, for example the vision or the feeling.

My task is to extend such a device, and turn it more effective. The device in question is a tactilic display, which helps in the process of reading from the mouth, in a way that it provides information which is not readable in that way. With the vibrancy of the voice bands, an important information, the voiced-unvoiced sound generation is intangible. The tactilic display carries out the transferring of this data by converting the incoming voice signal into a stimuli of tactilic motion, that can be sensed by the feeling, while not disturbing the primary channel of sensation, the vision. This stimuli transfers the various characteristics of the speech sound, in particular the acoustic transitions of the voiced-unvoiced sounds, hereby it helps to the user by segmenting the speech flow, which helps in turn in the reading from the mouth.

In the favor of mobility, the tactilic display device has its own processing unit, which memory contains the program of control. Programs, which are written in a PC environment, can be uploaded to the microcontroller, and can be run through an USB port. The firmware code of the chip was previously extended by myself, with such routines, that establishes the USB connection and the two-way communication.

The basic requirement against the display device in the case of practical application is to dependably provide the important information to the user in a noisy environment, which in turn needs the detection of speech signals. This can be achieved by the integration of a fast and effective speech detector algorithm.

With the studying of the referring literature, I managed to find a suitable algorithm, which uses the mel-cepstrum modulation energy to discriminate speech from noise. With this method, we analyzed the incoming segmented voice signal in the mel scale, which turned out to be an exceedingly effective method in the scope of real-time application.

First, I implemented the algorithm in a simulation environment, then I ran test on it with sound files with different sampling frequency and bit representation. Among them was clear, noiseless speech, music played by various instruments, noises which are common in beehive places (e.g. metro, car), and speech sounds loaded with such noises. In contrast to other

algorithms, the results of measurement were convincing, and made its integration reasonable into native control program of the device.

However, before the integration to the hardware, it is necessary to make a C language implementation in PC environment, which is closer to the architecture of the microcontroller, enabling to make measurements on the execution time, and the memory demand.

1. Bevezetés, célkitűzés

Mindennapjaink során sokféle információval találkozunk, ezekből igyekszünk a számunkra hasznosakat kiválogatni. Ez az általunk, a fülünkkel érzékelt hangjelekkel sincs másképp. Az agyunk automatikusan megkülönbözteti a jelentést hordozó beszédet a zajtól. A siket vagy halláskárosodott embertársainknak azonban a technikai lehetőségek által nyújtott segítségre van szükségük, hogy nemcsak a hangokat észleljék, hanem a számukra releváns információkat kinyerjék.

A siketek hallását más érzékekkel szükséges pótolni, a választás a tapintásra esett, az érzékelendő hangokat pedig mechanikai ingerekkel helyettesítjük. Ennek megvalósítására egy vibrotaktilis kijelző bizonyul a megfelelő eszköznek. A műszer képes a beérkező hangjelek mechanikai ingerré való átalakítására, azonban a műszer tökéletesítésére van szükség, hogy segítségével csak a lényeges információ, a beszéd jusson el a felhasználóhoz.

Célunk egy beszéddetektor algoritmus integrálása a kijelző real-time alkalmazásába, amellyel meg tudja különböztetni a beszédet a csendtől, zajtól és zenétől. Feladatunk olyan algoritmus keresése, implementálása, tesztelése és célkörnyezetbe való illesztése, amely egyszerű, gyors, megbízható, és képes a kijelző lehetőségeihez képest nagy biztonsággal való beszéddetektálásra. Az integrálást követően a zöngés és frikativ hangok megkülönböztetése a cél.

Munkánk első szakasza az utóbbi években megjelent, a beszéddetektor algoritmusokkal foglalkozó folyóiratok, konferencia kiadványok, szakkönyvek és egyéb irodalmak tanulmányozása, az általános beszéddetektálás koncepciójának és jól bevált algoritmusoknak a megértése. Ezután vizsgáljuk a már meglévő megoldásokat, valamint saját implementációkat hozunk létre. Ezt követően teszteljük az algoritmusokat, a mérési eredményeket összehasonlítjuk és rendszerezzük, ez alapján választjuk ki a megvalósítandó eljárást.

Miután a beszéddöntésre megfelelő algoritmust leteszteltük szimulációs környezetben, a célhardverbe való ágyazás a feladat. A programot úgy igazítjuk a taktilis kijelző mikrokontrollerének architektúrájához, hogy az optimálisan használja ki annak korlátozott erőforrásait, és emellett hatékonyan végezze el a feladatát.

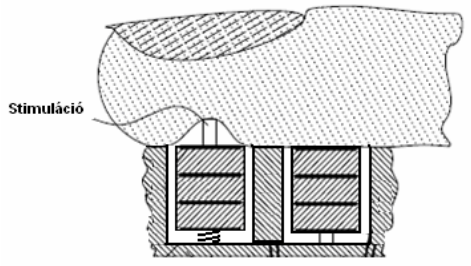
2. Előzmények

Mielőtt a beszéddetekció módszereit, és az általam megvalósított megoldásokat ismertetném, bemutatom a használt eszközöket és az ezeken végzett korábbi munkáimat, amelyek a kijelző vezérlésével, és a külvilággal való kommunikációval kapcsolatosak.

2.1. A taktilis kijelző

A vikrotaktilis kijelző [1] feladata az, hogy a beérkező vizuális vagy hanginformációt átalakítsa megadott paraméterek felhasználásával mechanikai ingerré.

Az egyik alapérzéket, a tapintást használja ki a kijelző, amely a bőrben elhelyezkedő érző receptorokat stimulálja. A kijelzés frekvenciája, dinamikája, amplitúdója, nyomása és a pontok távolsága határozza meg az átvihető információ jellegét és mennyiségét.



1. ábra Vibrotaktilis ingerlés

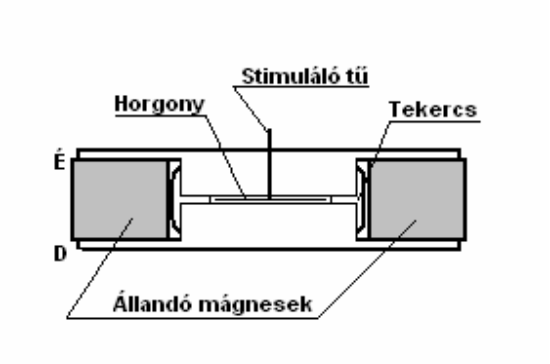
A legérzékenyebb taktilis felvevő pont az újjak vége. Itt 1-2 mm távolságra lévő rezgő tüskék jól elkülöníthetőek egymástól, függetlenül attól, hogy a felületre merőleges vagy oldalirányú az erőhatás.

Egy ponton rezgésként érzékelhető legmagasabb frekvencia 1000Hz, a bőr a kb. 250Hz frekvenciájú rezgésekre a legérzékenyebb.

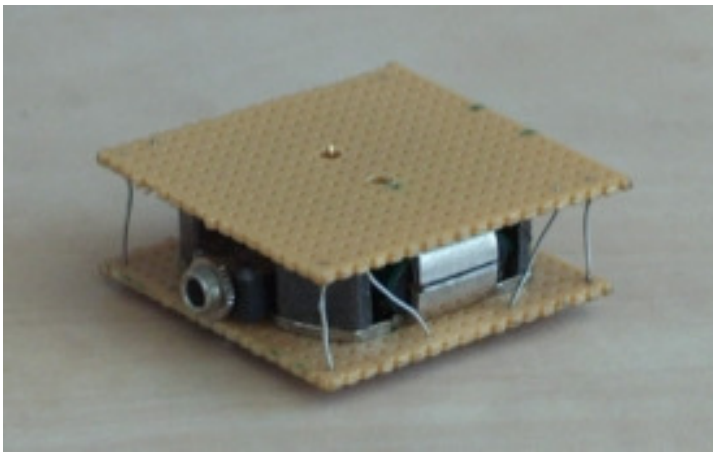
A mi szerkezetünk egy úgynevezett. elektromágneses stimulátor. Az elektromágnes a motor rotációs mozgását lineáris mozgássá alakítja át. Ennél a megoldásnál széles a megvalósítható rezgetési frekvencia. A kijelző mozgási tartománya akár 10 mm is lehet. Nincs meghatározva a kijelző mérettartománya és magas a felületre gyakorolt nyomás értéke.

Az elkészített és kipróbált kijelző vezérlésére egy számítógépes környezetben felépített szimuláció szolgál. A szimuláció a környezetből érkező hanghullámok digitalizálása után a hang jellemzőinek kinyerését valamint a kijelző vezérlését végzi. Az emberi beszédhang két paramétere lett kiválasztva a mechanikai megjelenítésre: a mássalhangzók kétféle típusa, a zöngés hangok és a frikatív hangok. A hangképzés ennél a két típusnál vizuálisan nem mindig határozható meg könnyedén, így esett a választás ezekre a beszédjellemzőkre.

A zöngés hangok frekvenciatartománya 200 – 300 Hz közti érték, a frikatív hangok ennél szélesebb frekvenciaspektrummal rendelkeznek, az 1000 Hz-től 3500 Hz-ig terjedő tartománya esnek bele leginkább. Az első lépés a vezérlés megvalósításánál az analóg jel digitalizálása. Így valósítható meg a kívánt frekvenciaértékek kiszűrése a 0-5000 Hz-ig terjedő emberi beszédjelből. A kívánt szűrt értékek előállítása után frekvenciatartomány modulációt kellett alkalmazni, mert míg a zöngés hangok frekvencia értékei a legérezhetőbb rezgési tartományba esnek, addig a frikatív hangok ötször, hatszor magasabb értékekkel rendelkeznek. A megfelelő moduláció alkalmazása után a jel visszaalakítását kell alkalmazni, olyan formában, hogy, a két különböző paraméter megjelenítése külön-külön jelenik meg ugyanabban az időben a vibrotaktilis szerkezeten. Erre a megoldásra a sztereo hangcsatorna tökéletesen elegendő, így egy egyszerű modulációval a kívánt jel a csatorna egyik, illetve másik oldalába vezérelhető. Ilyen formában tökéletesen adja vissza a taktilis kijelző a megjelenítésre szánt beszédhangot.



2. ábra Az elektromágneses elven működő kijelző felépítése



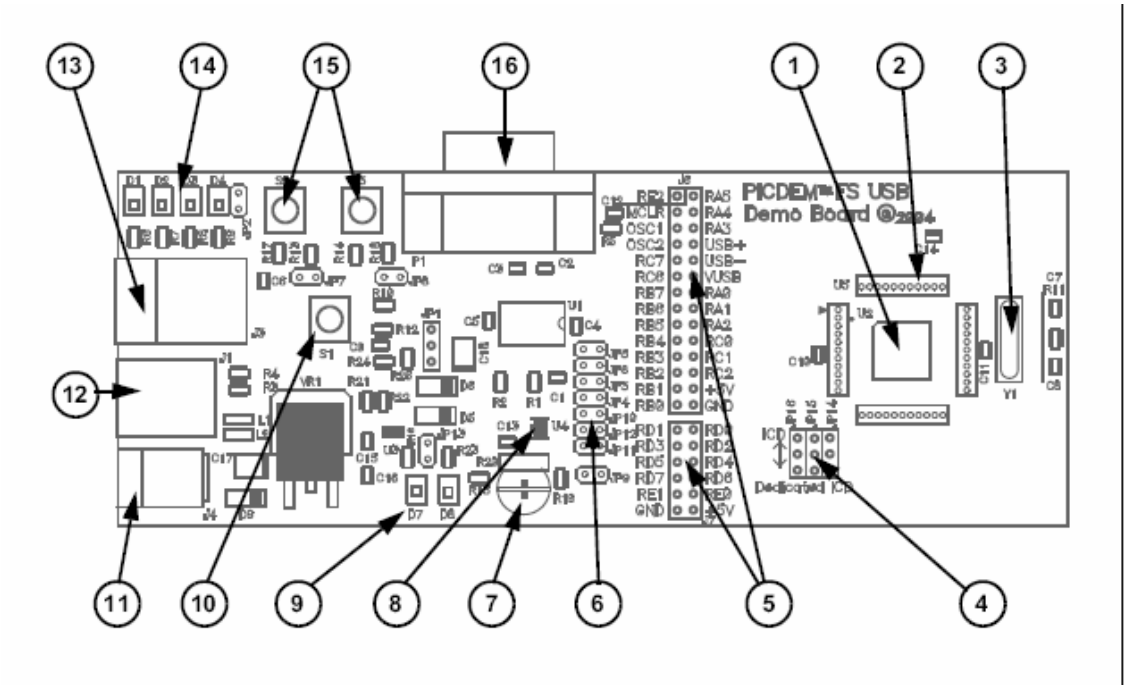
3. ábra A taktilis kijelző

A taktilis kijelző által szolgáltatott információk abban az esetben segítik a beszédmegértésben a felhasználót, ha az megtanulta a taktilis jelek értelmezését. Siketek esetében nem természetes a zöngés/zöngétlen fogalom ezért ennek, illetve az ebből képzett taktilisan átvitt jellemző megtapasztalása csak tanulás után segíti a szájról olvasott beszéd megértését.

A kijelzővel kapcsolatos alapvető tervezési szempont, hogy saját vezérlési egységgel rendelkezzen. Az önálló processzoron való kísérleteket egy erre a célra tervezett demonstrációs kártyán végeztük.

2.2. A PICDEM FS USB demonstrációs kártya bemutatása

A Microchip által gyártott PICDEM FS USB demonstrációs kártya [2], amelyet a feladatunkhoz használtunk, egy könnyen használható tesztplatform a 2.0 Full Speed USB képességeinek kipróbálására. A lap egy PIC18F4550 típusú mikrokontrollert tartalmaz, amellyel a taktilis kijelzőt vezéreljük.



4. ábra A PICDEM FS USB demonstrációs kártya

A lap alkatrészei a következők:

1. Mikrokontroller: A 44 lábú, TQFP PIC18F4550 mikrokontroller a demonstrációs lap „szíve”, és valamennyi USB funkciót szolgáltatja egy chipen.
2. ICE interfész riser: A mikrokontroller egy 44 bemenetű blokkal van körülvéve, négyfelé osztva, 11 bemenet a négy oldalon. Ezeket a riserek mountolására használhatjuk, ha egy In-circuit debuggert akarunk csatlakoztatni.
3. Oszcillátor: A demonstrációs lap egy 20Mhz-es kristály oszcillátort használ a fő órajel szolgáltatására. A lap ezt az oszcillátort használja mind az USB soros interfész motor (SIE), mind a processzor órajelének generálásához.

4. ICD konfigurációs jumperek: Ezek a használaton kívüli jumper pozíciók lehetővé teszik a felhasználók számára a megfelelő ICSP és ICD portok kiválasztását.
5. Bővítő és PICtail headerek: Ezek a blokkok lehetővé teszik, hogy a felhasználó közvetlenül hozzáférhessen a mikrokontroller I/O jeléhez. Továbbá a 14 számozott bemenet a jobb oldalon egy interfészként szolgál további csatlakoztatható lapok számára.
6. Konfigurációs jumperek: Összesen 13 szabad jumper pozíció áll rendelkezésre a lapon; ezek lehetővé teszik a lap hardverének konfigurálását az egyéni szükségleteknek megfelelően. Alapból mindegyik jumper engedélyezve van.
7. Potenciométer: A potenciométer az analóg bemenetet szimulálja a lapon. Ezt a feladatunknál is felhasználtuk, erről a későbbiekben bővebben kitérünk.
8. Hőmérő: A Microchip TC77 digitális hőmérője folyamatosan méri a lap környezetének hőmérsékletét. Az adat a 3 szalagos SPI interfészen keresztül továbbítódik a kontroller felé.
9. Power LED-ek: Ezek a lámpák a lap áramellátását és ennek mértékét jelzik. A D7-es LED azt jelzi, hogy a lap a buszról kap-e áramot, míg a D8 azt, hogy valamely más forrástól.
10. Reset gomb: Ez a kapcsoló a kontroller MCLR bemenetére van csatlakoztatva, a megnyomása egy hard resetet idéz elő.
11. Táp bemenet: Egy 9 VDC tápfeszültség szolgáltatható a lap számára egy külső forrástól. A használata opcionális, mivel a lap USB kábelen keresztül is kap áramot.
12. USB bemenet: Ez egy szabványos USB „B” típusú foglalat. Az USB port a fő csatorna a demonstrációs lappal történő kommunikációban és vezérlésében.
13. ICD bemenet: A 6 drótos RJ11 konnektor szolgáltatja a szabványos bemenetet a mikrokontroller MPLAB ICD 2 segítségével történő programozásához és debugolásához.
14. Állapot LED sor: A négy LED-ből álló sor a lap műveleti állapotát mutatják. Két LED-et (D1 és D2) az alkalmazás firmware használ, hogy az USB kapcsolat státuszát mutassa. A többi LED (D3 és D4) a felhasználó által állítható be.
15. A felhasználó által beállítható gombok: Ez a két kapcsoló (S2 és S3) a digitális bemenet szimulációját szolgálják. Megnyomva bármelyiket a hozzájuk tartozó port „0”-t értelmez.

16. RS-232 (DB9F) port: Szabványos D konnektor, egy shifterrel együtt soros kapcsolatot biztosít a demonstrációs lap számára.

A tesztprogramjainkat az USB Firmware Framework keretrendszert felhasználva írtuk. A keretrendszer egy fájlrendszer konstrukció, amely felhasználható USB alkalmazások készítésére, az MPLAB IDE integrált fejlesztői környezetben használjuk. Lehetővé teszi, hogy C nyelven írt programokat írhassunk és futtathassunk a mikrokontolleren. Egy referencia projektként is tekinthetünk rá, amely az USB műveletekhez szükséges firmware kódokat és általánosságban a felhasználó kódjait tartalmazza. A keretrendszer továbbá olyan moduláris interfészeket szolgáltat, amely elvégzi az USB kommunikáció implementálásával járó munka nagy részét.

2.3. A tesztprogramok

A legelső programunk felkapcsolta és villogásra késztette a demonstrációs lap általunk kiválasztott LED-jeit. Ehhez először a LED-eket is kezelő, a mikrokontrollerhez tartozó portokat és a regisztereket kellett megismerni. A LED-ek be- és kikapcsolásához regiszterek megfelelő bitjeit kellett beállítani.

A következő feladat a LED-ek villogtatása volt, ehhez a Timer modult, annak interrupt funkcióját, és a kisebb processzor terhelés érdekében a sleep módot használtuk. Az időzítő a 0000h értékről a FFFFh értékig inkrementálódott ciklikusan, ahogy elérte, a program egy interrupt műveletet hajtott végre, amelynek következtében felvillant a LED. Az inkrementumok között a mikrokontroller sleep módba kapcsolt.

A LED villogás intenzitásának módosíthatósága is felmerült. Ezt a demonstrációs kártyába integrált potenciométerrel kívántuk elérni.

Először is a potméterhez tartozó csatornát kívántuk analóg bemenetként beállítani. Ezt a megfelelő adatirány regiszterek bitjeinek beállításával értük el. Ezután ezt az analóg bemenetet szerettük volna egy A/D átalakító segítségével digitális kimenetté konvertálni és ezzel a kimenettel befolyásolni a LED-ek villogásának frekvenciáját.

Az A/D átalakítás lépései a következőek:

1. Az A/D modul konfigurálása:
   - Az analóg pinek, feszültség referencia és digitális I/O beállításai Az A/D bementi csatorna kiválasztása
   - A mintavételi idő kiválasztása
   - A konverziós óra kiválasztása

- Az A/D modul bekapcsolása

2. Várni a kiválasztott mintavételi ideig
3. Az átalakítás megkezdése:
4. Várni az A/D átalakítás befejezéséig
5. Az A/D eredményregiszterek kiolvasása

Miután az A/D átalakítást elvégeztük, a LED-ek villogásának intenzitását a potméterrel tetszőlegesen lehetett módosítani.

Az A/D átalakítót egy későbbi fejezetben mutatom be részletesen.

A demoboard alkatrészeit tehát módunkban állt vezérelni, a következő céljaink között a mikrokontroller és a PC közti kapcsolat felvétele, és a kétirányú kommunikáció megvalósítása szerepelt. Ezért egy USB protokollt írtunk, az MPUSBAPI PC oldali keretrendszer segítségével.

2.4. Az MPUSBAPI keretrendszer bemutatása

Az MPUSBAPI [3] egy DLL modul, amely wrapper funkciókat szolgáltat a Microchip Windows alá tervezett, általános USB illesztő programjához, a mchpusb.sys fájlhoz és egyszerűsít a Win32 API funkcióinak komplexitásán. A keretrendszert C++ nyelven írták meg. Az új USB protokollunk erre a keretrendszerre épül.

A munkánk idején lévő kiadás hét alapvető funkcióval rendelkezett:

- **DWORD *MPUSBGetDLLVersion(void)**
  Lekérdezi az MPUSBAPI.DLL revíziós dátumát, amely egy 32 bites szám MMMMmmmm alakban. Csak a szoftver kódját adja vissza, USB műveletet nem végez.
- **DWORD *MPUSBGetDeviceCount(PCHAR pVID_PID)**
  Megadja az érvényes VID és PID kódokkal rendelkező eszközök számát.
- **HANDLE *MPUSBOpen(instance, pVID_PID,pEP,dwDir,dwReserved)**
  Megadja az érvényes VID és PID kódokkal rendelkező végpontok útvonalához tartozó kezelőt. Mindegyik útvonal a FILE_FLAG_OVERLAPPED attribútummal nyílik meg. Ez lehetővé teszi az MPUSBRead, MPUSBWrite és MPUSBReadInt számára, hogy Time-out értékkel rendelkezzenek.
- **DWORD *MPUSBRead(handle,pData,dwLen,pLength,dwMilliseconds)**
  Olvas a kezelőről

- **DWORD *MPUSBWrite(handle,pData,dwLen,pLength,dwMilliseconds)**
  Ír a kezelőre
- **DWORD *MPUSBReadInt(handle,pData,dwLen,pLength,dwMilliseconds)**
  Egész értékeket olvas a kezelőről
- **BOOL *MPUSBClose(handle)**
  Lezárja a megadott kezelőt.

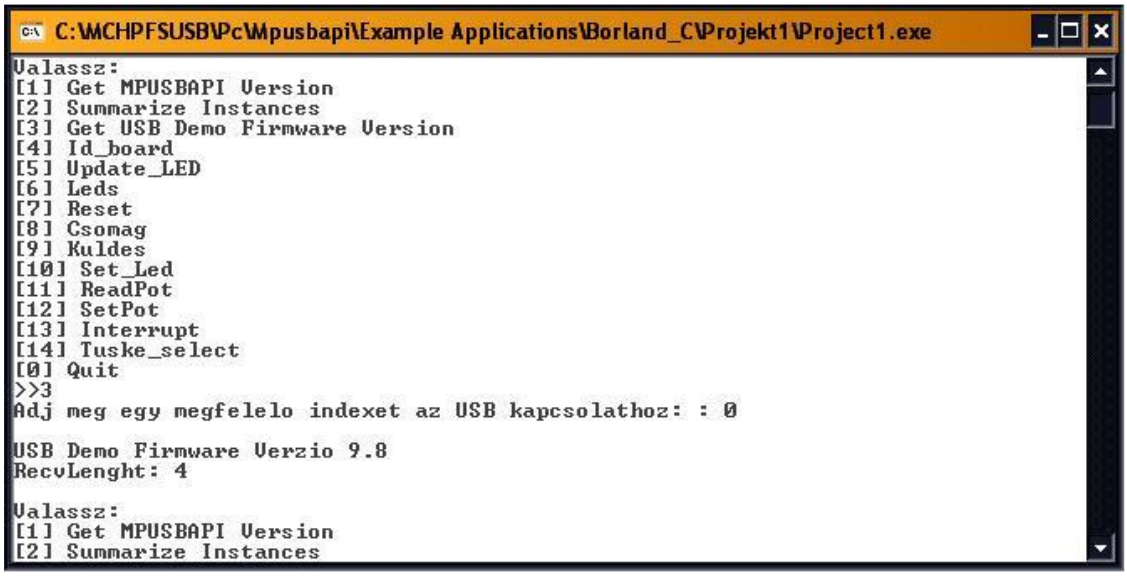
2.5. Az USB protokoll működése

A PC oldali vezérléshez az MPUSBAPI keretrendszerhez mellékelt demoprogramot használtuk fel. Az _mpusbapi.c forrásfájlban találhatóak a fent szereplő kulcsfontosságú funkciók megvalósításai. A program belépési pontjában a console.cpp-ben ezeket a funkciókat használjuk fel a protokoll alapvető műveleteinek megvalósítására: mikrokontrollerrel való kapcsolat felvételére, kapcsolat megszakítására és a kommunikációra. A fájlban különféle függvényeket írtunk, amelyek számos vezérlési és kommunikációs funkciót látnak el. Ahhoz, hogy a kapcsolat létrejöjjön a PC és a mikrokontroller között, a számítógép és mikrokontroller oldali programoknak párhuzamosan kell futniuk. A PC program minden funkció esetében egy hexa formátumú parancsot küld el a demonstrációs lapra, amelyet a mikrokontrolleren futó program egy case szerkezetben dolgoz fel, és a parancsnak megfelelően reagál, például megadja a verziószámát, felkapcsol egy LED-et, vagy adatokat küld és fogad.

A protokoll az alábbi lépésekből áll:

1. A PC oldalon megnyitja az oda-vissza adatcsöveket a mikrokontrollerhez,
2. létrehozza a küldött és fogadott adatok számára a puffereket (**BYTE send_buf[64]** és **receive_buf[64]**),
3. meghatározza a fogadott csomag elvárt méretét (**DWORD RecvLength**),
4. definiálja az elküldött parancsot és a küldött csomag hosszát, ezt a két adatot átadja a **send_buf[]** első két elemének,
5. a mikrokontroller oldalon létrehoz egy **DATAPACKET** típusú csomagot
6. a **DWORD SendReceivePacket(BYTE *SendData, DWORD SendLength, BYTE *ReceiveData, DWORD *ReceiveLength, UINT SendDelay, UINT ReceiveDelay)** wrapper függvény ellenőrzi a csomagok méretét, majd elküldi az adatot a **send_buf[]** által és fogadja a **receive_buf[]** segítségével,

7. a mikrokontroller oldalon a **void ServiceRequests(void)** eljárásban az **USBGenRead(*DATAPACKET dataPacket, int size)** fogadja az adatcsomagot, melynek a CMD komponensét egy case struktúrában feldolgozza, és esetenként adatot küld a PC-re,
8. A PC lezárja az oda-vissza utat



5. ábra A program futás közben

A példaprogramban az alábbi függvényeket írtuk meg:

- **void Leds(void)**

  LED-ek kiválasztása, be- és kikapcsolása
- **void Reset(void)**

  A demonstrációs lap resetelése
- **void Csomag(void)**

  Numerikus, karakter és sztring típusú adatok fogadása
- **void Kuldes(void)**

  Numerikus, karakter és sztring típusú adatok küldése
- **void ReadPot(void)**

  A potenciométer értékének kiolvasása
- **void Interrupt(void)**

  Az interrupt be- és kikapcsolása
- **void Tuske_select(void)**

A taktilis kijelző tüskéinek kiválasztása, be- és kikapcsolása, rezgésük intenzitásának PC-ről történő beállítása

Összefoglalva: megismertük a PC és az USB eszköz közötti kapcsolat részleteit, felépítését és annak működését. Ezt a tudást felhasználva megírtuk a PC-t és a kijelzőt összekötő kommunikációs protokollt, amelynek segítségével felvehetjük a kapcsolatot a kijelzőt vezérlő mikrokontrollerrel, parancsokat küldhetünk a számítógépről a mikrokontrollerre, amelyet értelmez és végrehajt, tetszőleges típusú adatot küldhetünk és fogadhatunk, lehetővé téve a real-time interaktív kommunikációt, képesek vagyunk a kijelző tüskéit egymástól függetlenül rezegtetni, és a PC-ről beállítani a rezgés intenzitását. Ez lehetővé teszi, hogy mikrofonból beérkező hangjelek esetén a zöngés és frikatív beszédhangok egymástól elkülönülve, két különböző csatornán jelenjenek meg.

Ennek bemutatására egy demonstrációs programot hoztunk létre, amely ezeket a funkciókat szemléletesen mutatja be. A program alapjául szolgálhat számos további cél megvalósításához, például egy komplett vezérlő program, vagy egy, a taktilis kijelzőt meghajtó driver megírásához.

A kijelző immár egy önálló, független, kompakt rendszernek fogható fel, amely készen áll arra, hogy egy szűrőhöz csatlakoztassuk, amely egy beérkező hangjelből a zöngés és frikatív beszédhangokat szűri ki. A kijelző működhet akár egy számítógépes perifériaként, amely egy mikrofonból érkező valódi hangjellel működik, összekapcsolható mobiltelefonnal, és akár önálló mikrofonnal rendelkező, saját teleppel működő változat is megvalósítható.

A kijelző a külvilágból érkező hangjelek valamennyi részét feldolgozza, így a zajt is. A műszert használó ember számára azonban fontos, hogy csak a beszédet, illetve annak zöngés és zöngétlen komponenseiről szóló információt jelezze ki, mivel ez a két jellemző az, amit szájról nem tud leolvasni. Ahhoz, hogy a kijelzőt valamennyi, így a zajterhelt környezetben is használni lehessen, egy olyan beszéddetektor algoritmust kellett a rendszerbe integrálni, amely megkülönbözteti és kiemeli a beszédet a háttérzajból. Maga az algoritmussal szembeni elvárás az, hogy legyen gyors, alacsony komplexitású, és biztosítsa a kis méretű, hordozható eszköz megbízható és hatékony működését.

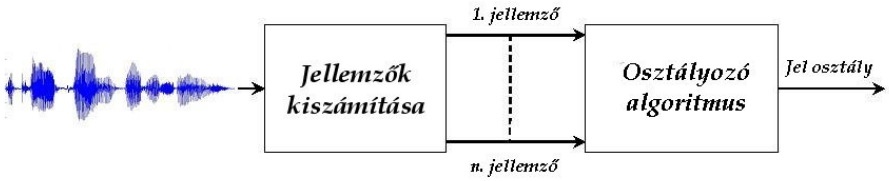
A következő lépés tehát a beszéddetektor, illetve beszéd/zene megkülönböztető algoritmusok irodalmának tanulmányozása volt, annak érdekében, hogy a fenti elvárásoknak megfelelő eljárást kiválasszam, és megvalósítsam.

3. Beszéddetektor algoritmusok általános ismertetése

A beszéddetektor (VAD-Voice Activity Detection) [4] algoritmusokat az audió jelekben fellelhető beszédhangok észlelésére alkalmazzák. Fontos szerepet játszanak a hangfeldolgozás elő-feldolgozási szakaszában. Például a VoIP és a mobil kommunikáció alkalmazásában csökkenteni tudják a sávszélesség és hálózat terhelést, mivel csak akkor lesz egy csomag elküldve, ha előtte beszédet detektáltak. Továbbá növelni lehet a beszédfelismerés, beszélő felismerés és hangforrás lokalizáció teljesítményét, ha az audió jel csak azon részeire alkalmazzák az algoritmusokat, melyek beszédként lettek azonosítva. Zajok elnyomására szintén használják, hallókészülékekben és audió konferencia lebonyolításakor.

A beszéddetektálás egy osztályozási probléma, melyben az hangjelet a jellemzői alapján különítik el különböző osztályokba. Az osztályozási eljárás során, a hangjelet kisméretű, meghatározott hosszúságú keretekre osztják. Ezután minden keretre kiszámítják a jellemzők értékeit, és az osztályozási algoritmusnak továbbítják bemenetnek. A keretek általában 20 ms hosszúságúak.

Az osztályok mennyisége és típusa nagyban függ az alkalmazás feltételeitől. Például, ha a beszéd és egy más fajta jel között kell különbséget tenni, mint a VoIP alkalmazások esetén, akkor az audió jelet két osztályba lehet sorolni: beszéd és nem beszéd.



6. ábra Egy VAD rendszer sruktúrája

Ahogy azt már korábban leírtam, a beszédhangokat tovább lehet osztályozni zöngés és frikativ hangokra. A zöngés hangok általában determinisztikus hullámformák, amelyek magán- és

mássalhangzókat tartalmaznak. A frikativ hangok ezzel szemben mássalhangzókat tartalmazó sztochasztikus hullámformák. Ez a klasszifikáció azoknál a forrás lokalizációs eljárásoknál hasznos, melyeket csak a zöngés beszéd szakaszokon alkalmaznak.

Az osztályozáshoz szükséges jellemzők kiválasztása függ az egyes osztályoktól. Az egyik használható jellemző a beszéd észlelésben a jel energiája, amely a minták amplitúdójának keretenkénti négyzetösszege. Egy másik fontos jellemző a nullátmenetek száma. Ez azt mutatja, hogy a jelnek hányszor változott az előjele egy keretben. A különböző típusú audió jelek különböző mennyiségű nullátmenettel rendelkeznek. A zöngés beszédhangoknak általában kevesebb nullátmenetei vannak,, mint a zöngétleneknek. Ennek a két jellemzőnek a segítségével különbséget lehet tenni a két hangtípus között.

A zöngés beszédjelek általában alacsony, míg a zöngétlen jelek magas frekvenciájúak. Azzal számolunk, hogy a zöngés hangok energiája nagyobb hányadban van jelen az alacsony frekvencia sávokban (2kHz-nél alacsonyabb), míg a frikativ hangok energiája egyenletesebb eloszlású. Az alacsony frekvencia sáv aránya a teljes energiához képest szintén egy megkülönböztető jellemző.

A lineáris prediktiv kódolással (LPC) kapcsolatos jellemzők is a beszéddetektálás szolgálatába állíthatók. Az LPC predikciós együtthatói egy bemeneti jel LPC elemzésével nyerhetők ki. Ezek az együtthatók a rendszer modellt reprezentálják; segítségükkel rekonstruálni lehet az eredeti jelet. Beszédjelek osztályozásában is hasznosak, mivel az első néhány LCP együttható a jel információinak nagy részét tartalmazza. Az egyik jellemző az LCP hibája, amely az LPC együtthatókat használó visszaállított jel és az eredeti bemeneti jel különbségeként definiálható.

Az LPC jel perdikciójának pontossága a véletlenszerűségéből mérhető. Ennek következtében a zöngés hangokra vonatkozó LPC hiba alacsonyabb, mint a zöngétlen hangoknál, így ez a jellemző segít a zöngés beszéd és a véletlenszerű jelalakkal bíró jelek közötti megkülönböztetésben.

Magasabb szintű statisztikai jellemzőket, mint például a torzulást és a csúcsosságot, szintén használhatják osztályozásra. A Gaussi jelek az átlaguk és az eltérésük alapján jellemezhetőek. Ezeknek a jeleknek nincs magasabb szintű statisztikai jellemzőjük. A beszéd jelek nem Gaussi jelek, így magasabb szintű jellemzőket használnak a beszéd és a Gaussi zajok közti megkülönböztetésre.

Beszéddetektáló rendszerek tervezésénél az egyik legnagyobb kihívást azoknak a jellemzőknek az azonosítása jelenti, amely alapján osztályozhatóak a hangjelek. A megfelelő jellemzők csoportja biztosítja, hogy ne legyen átfedés az osztályok között, és ha igen, akkor az

nagyon kis mértékű legyen. Ha a jelek szorosan összefüggnek, akkor ezeket magasan korreláltnak mondjuk. Ezt célszerű elkerülni, mert hátrányosan befolyásolhatják egymás részvételét az osztályozásban. A jellemzők kiválasztását célzó algoritmusok a vonások egy tetszőleges részhalmazát választják ki a jellemzők teljes halmazából.

A Fisher F aránya az egyik leggyakrabban használt algoritmus a jellemzők kiválasztására. Egyszerre csak egy jellemzőre alkalmazható, hogy meghatározzák a klasszifikációban való fontosságát. Ha a jellemzők nem korreláltak, akkor valamennyi jellemzőre alkalmazzák és a legalkalmasabb lesz kiválasztva beszéddetektálásra. Az egyik hátránya ennek az algoritmusnak az, hogy nem képes észlelni az osztályok közötti átfedést. Ha a jellemzők korreláltak, akkor az F arány variációját az összes jellemzőre alkalmazzák a fontosságuk megbecsüléséhez.

A jellemzők optimális részhalmaza úgy határozható meg, hogy a kiválasztásos algoritmust valamennyi részhalmazon alkalmazzák, és ezután választják a klasszifikációra legalkalmasabb részhalmazt. Ez egyszerűnek hangzik, azonban ha egy nagy halmazról van szó, akkor ennek a módszernek túl nagy lehet a számításigénye, ezért nem célszerű alkalmazni. Az elő- és utó-kiválasztásos algoritmusok jelentik a megoldást erre a problémára.

Az elő-kiválasztásos módszer első lépéseként azt feltételezzük, hogy semelyik jellemző sem fontos. Az első iterációban figyelembe veszi valamennyi tulajdonság alkalmasságát az osztályozásban, és a klasszifikáció teljesítményéhez leginkább hozzájáruló jellemzőt választja ki. Az egymást követő lépésekben egyszerre veszi a kiválasztott vonást a többivel együtt és ezeknek azon kombinációját választja ki, mely a legnagyobb szerepet tölti be az osztályozásban. Ezt addig ismétlik, amíg egy újabb tulajdonság hozzáadásának nincs, vagy lényegtelen a hozzájárulása az osztályozás teljesítményéhez, vagy egy bizonyos feltétel nincs kielégítve.

Az utó-kiválasztás az elő-kiválasztásos módszer fordítottja. Először azt feltételezzük, hogy valamennyi jellemző fontos. Az első iterációval eltávolítjuk az osztályozáshoz legkevésbé hozzájáruló tulajdonságokat, majd az egymást követő lépésekben elvetjük azokat a vonásokat, amelyek nem, vagy nem lényegesen járulnak hozzá a klasszifikáció teljesítményének növeléséhez.

Általában az elő- és az utó-kiválasztás a jellemzőknek ugyanazt a részhalmazát adja vissza. Mindkét megközelítés mohó kereső algoritmus, mivel nem veszik figyelembe az összes részhalmazt, az elő-kiválasztás iterációinak során nem vizsgálják felül a jellemzők hasznosságát, hasonlóan az utó-kiválasztás során elvetik, és nem vizsgálják újra a szükségtelennek leírt jellemzőket. Így a tulajdonságok halmaza nem lesz optimális.

A jellemzők kinyerése is fontos szerepet játszik a klasszifikációban. Míg a jellemzők kiválasztása egy részhalmazt választ a teljes halmazból, addig a tulajdonságok kinyerésével a

bemeneti adatoknak csökkentjük a dimenzióit, úgy, hogy az eredeti vonások kombinációiból formálunk új jellemzőket. Ez azokban az esetekben rendkívül hasznos, amelyekben az eredeti jellemzőkkel nem állapíthatóak meg az osztályok határai. A jellemzők egy új tere alakítható ki, amelynek segítségével különbséget tehetünk az osztályok között.

Az elsődleges komponensek elemzése a legegyszerűbb és legelterjedtebb módja a jellemzők kinyerésének, amely egy lineáris transzformációt végez a bemeneti jellemzőkön. Egy másik módszer a diszkriminációs analízis.

A jellemzők kinyerésének előnye a tulajdonságok tárolására kijelölt hely méretét csökkenti, hátránya viszont a megnövelt számítási igény, mivel a kinyert jellemzők mellett az eredeti vonások kiszámítására is szükség van.

Az klasszifikációhoz szükséges jellemzők kiválasztása mellett, az osztályozási algoritmusok kiválasztása is egy lényeges elem a beszéddetektor rendszerek tervezésében. A már meglévő algoritmusok két fő kategóriára oszthatók: a felhasználó által definiált küszöbszint alapú, és a gépi tanulási algoritmusra.

A felhasználó által definiált algoritmusok egy vagy több jellemzőhöz számítják ki a küszöbszintet, az osztályok közötti döntés meghozatalához. Ezeket a küszöb értékeket a felhasználó által kijelölt szabályok szerint választják ki. Ennek a megközelítésnek az előnye, hogy a felhasználó teljes mértékben definiálhatja a klasszifikációs döntési funkciót, mivel saját maga dönt a küszöb értékeket kiszámító szabályokról. A legfőbb hátránya azonban az értékek megtalálásának nehézségében rejlik, különösen akkor, ha nagyméretű a jellemzők halmaza.

A felügyelt gépi tanulásos algoritmusok ezzel szemben az ismert osztályok adatait használják tanulási adatként, hogy létrehozzák a döntési funkciót. Ez az eljárás egy előre definiált osztályba sorolja az audió jelet. A beszéddetektálásban használt gépi tanulási algoritmusok diszkriminánsanalízist (DA), neurális hálózatot (NN), és segéd vektoros gépeket (SVM) foglalnak magukba. Ezeket a technológiákat a jellemzők kiválasztására is használják.

A paraméteres osztályozó már meglévő ismeretet feltételez az adatok eloszlásáról, szemben a nem paramétereresekkel. A paraméteres osztályozókba tartozik a diszkriminánsanalízis, amely az adatok normális (Gaussi) eloszlását feltételezi. Ha az adat nem normálisan elosztott, a DA így is megbízható marad, mivel az adathalmaz nem tartalmaz kirívó elemeket. A legegyszerűbb DA eljárás a Mahalanobis távolság osztályozás (MDC). A Mahalanobis távolság (MD) az Euklideszi távolsághoz hasonló, és az adatok az osztályok súlypontjától való távolságot mérjük vele. A bemenő adatot a legkisebb MD távolságú osztályhoz rendeljük. Az osztályok

súlypontját a gyakorló adatokból nyerjük. A lineáris és négyzetes DA a további példák a DA osztályozókra

A DA-val szemben, az SVM és az NN nem paraméteres osztályozók. Az NN számítási egységként használt neuronokat tartalmaz, amelyek több bemenő jelet fogadnak, és egy kimenő 1 értéket generálnak abban az esetben, ha a bemenő jelek súlyozott összegének értéke egy küszöb érték felett van. A neuronok küszöb értékeit a gyakorló adatokon futtatott algoritmusokkal határozhatók meg.

Az SVM szintén gyakorló adatokat használ a különböző osztályokat szétválasztó metszeti sík megtalálásához. Az optimális metszeti sík saját maga és az általa elválasztott osztályok gyakorló adatai közti távolságot maximalizálja, ami megnöveli az osztályozó általánosítási képességét. Ezután egy ismeretlen osztály bemeneti adatának saját osztályába való klasszifikációjára használják a síkot. Ha a gyakorló adat nem lineárisan szétválasztható, akkor az adatmintákat egy magasabb dimenziójú térbe transzformálják, hogy lineárisan szeparálhatóak legyenek. Meg kell jegyezni, hogy mód van az adatok transzformációjának elkerülésére, ha magukon a bemeneti adatokon dolgozunk.

A statisztikai osztályozó algoritmusok, mint a rejtett Markov modellek és a Gaussi kevert modellek szintén alkalmazhatóak beszéddetektálásban. A Markov modell esetében annak a valószínűsége, hogy egy jelenlegi keret egy bizonyos osztályhoz tartozik az előző keret osztályától függ.

A Gaussi kevert modell a jellemzők vektorának Gaussi eloszlását feltételezi. Ezt a modellt a gyakorló adatokhoz tartozó jellemzőkön alapuló osztályok valószínűségi eloszlásának megállapítására használják. A valószínűségi eloszlás becslése a többdimenziós Gaussi eloszlás súlyozott összegén alapszik. Az eloszlást az adat osztályozására használják.

Egy jó klasszifikációs algoritmus elvárt jellemzője a magas találati arány. Az osztályozó ismeretlen adatnak a megfelelő osztályába való sikeres besorolásának képessége a generalizáció. Bizonyos esetekben az osztályozó a gyakorló adatot megfelelően sorolja be, azonban az új adatokat nem. Ez a túlilleszkedés problémája. Általában azt várjuk el, hogy az osztályozót úgy tanítsák, hogy a generalizációs képessége kedvező legyen, és alacsony túlilleszkedést produkáljon.

Ahhoz, hogy megállapítsák egy osztályozó generalizációs képességét, a kihagyásos módszer alkalmazható. Az osztályozó egy kivételével az összes gyakorló adat mintán végez tanítást, és megjósolja a kihagyott minta osztályát. Ez az eljárást valamennyi adatmintán elvégzik, és a teljes klasszifikációs arányból kikövetkeztethető az osztályozó teljesítménye.

Egy további megközelítés egy másik, a tanító adattól független adathalmaz használata. Ezeknek az algoritmusoknak a gyakorló adatok halmazán végzett osztályozások teljesítményeinek

összehasonlítása a legegyszerűbb módja a megfelelő algoritmusok kiválasztásának. Ebben az esetben nem beszélhetünk valós találati arányról. Statisztikus szignifikancia tesztekkel állapíthatóak meg a találati esélyek.

	Lépések	Kihívások
1	Jellemzők halmazának kiválasztása, majd a vonások kinyerése	• Ne legyen csoportok közti átfedés • Ne legyen korreláció • Csak a fontos jellemzők megtartása
2	Osztályozó algoritmus kiválasztása	• Magas találati arány • Alacsony hiba arány
3	Gyakorló adat kiválasztása	• A legtöbb jelforma lefedése • Megfelelő generalizáció • Kis mértékű túlméretezés

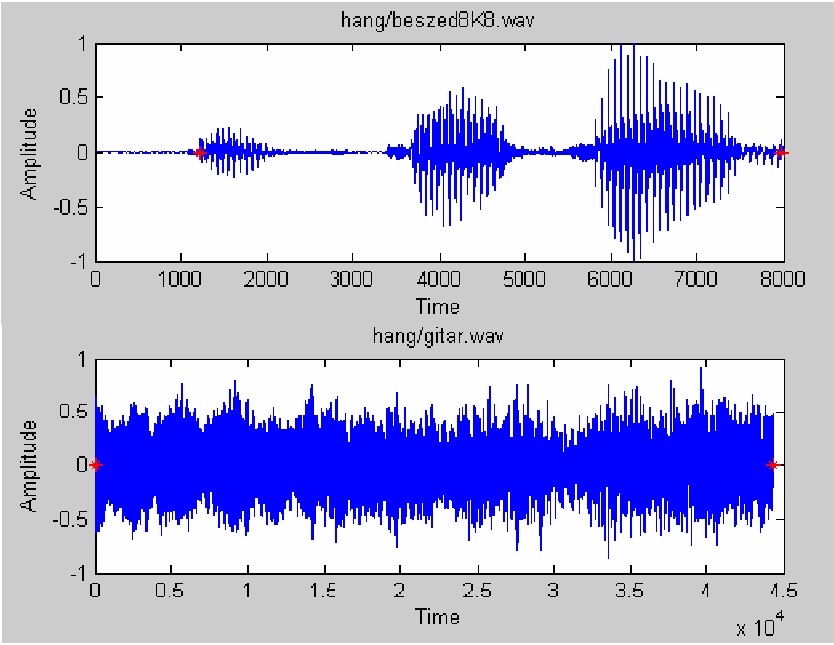
A megfelelő gyakorló adat kiválasztása szintén egy fontos tényező. SVM algoritmusokban a gyakorló adat kiválasztása befolyásolja a döntési funkció ismeretlen adaton végzett hatékony besorolási képességét, ezért érdemes nagyszámú gyakorló adatot bemenetként venni. A gyakorló adatoknak a legtöbb olyan jel forgatókönyvét kell lefednie, amin beszéddetektálást végzünk.

Miután ebben a fejezetben bemutattam a beszéddetektor algoritmusok jellemzőit és tervezési kihívásait, a következő pontban a beszéd a háttérzenéből való kiemelésének módszereit ismertetem.

4. Csúcsosság alapú megkülönböztetés

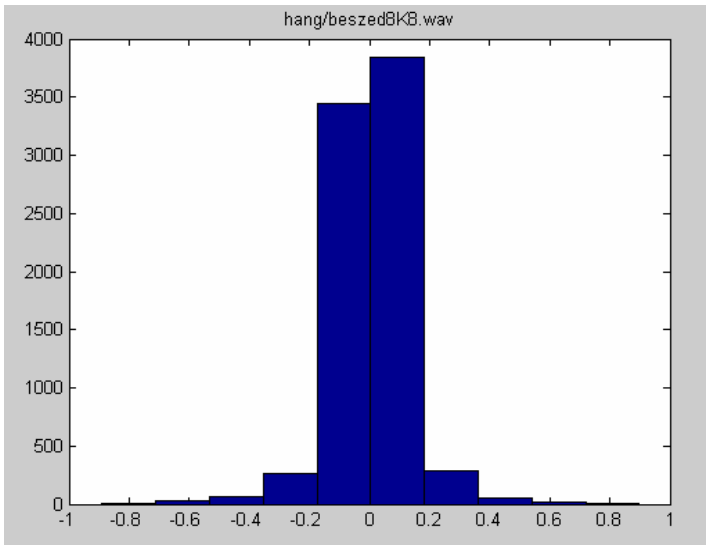
Első közelítésben a beszéddetektálás talán legkézenfekvőbb megoldását, a hangjel csúcsosságának vizsgálatát [5] tartottam megvalósításra és tesztelésre alkalmasnak. Ez egy igen egyszerű módszer, és a példából jól lehet szemléltetni, hogy milyen hiányosságai lehetnek a nem real-time alkalmazásra optimalizált megoldásoknak.

Annak vizsgálatára, hogy a jel csúcsossága mennyire tér el a Gaussi eloszlástól, a beszéd és a zene egymástól eltérő jelalakjai szolgáltattak ötletet. Míg a beszédnél a jel amplitúdója intenzíven változik, és tartalmaz csendes szakaszokat, addig a zene a beszédnél jóval egyenletesebb és folytonosabb jelalakot szolgáltat, ahogy az a 7. ábra is mutatja. Az ábra felső felében egy 1 másodperces beszéd, míg az alsó részén egy 1 másodperces gitár szóló szekvencia jelalakja látható.

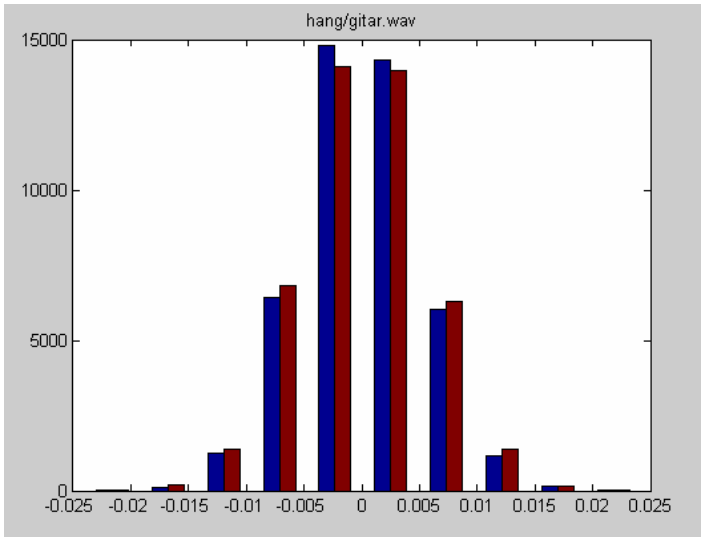


7. ábra A beszéd és zene jelalakja

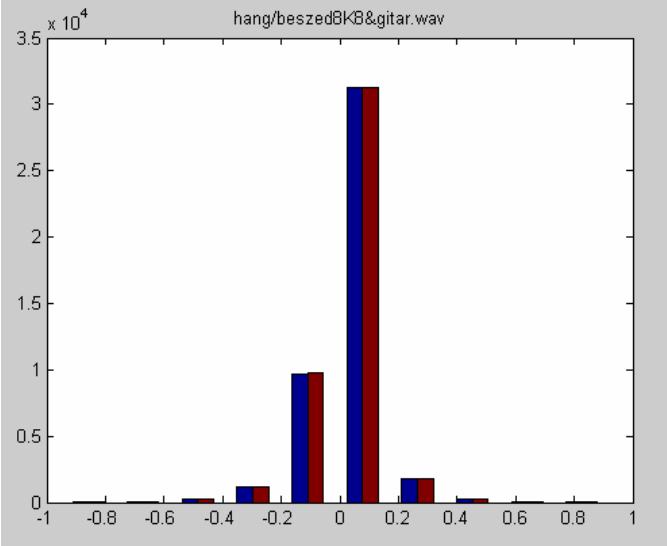
A két jelalak közti különbség, a jelek csúcsosságain is tetten érhető, a beszéd szekvenciának rendre nagyobb lesz a csúcsossága, mint a zenének, függetlenül az audió jelek mintavételi frekvenciájától, illetve bit felbontásától. Ennek a jelenségnek szemléltetésére egy Matlab funkciót írtam. Az alábbi ábrák, amelyek a Matlab függvény kimenetei, a beszéd, zene, valamint a zenével aláfestett beszéd hisztogrammjait mutatják be:



8. ábra A beszéd hisztogrammja



9. ábra A zene hisztogrammja



10. ábra A beszéd és zene hisztogrammja

Látható az ábrákon, hogy a beszéd csúcsossága nagyobb a zenéénél, és a beszéd és a zene együttes hisztogrammjának jelalakja a beszédét veszi fel, vagyis a csúcsosságát a beszéd csúcsossága határozza meg.

A két hangtípust egy küszöbértékkel kategorizáltam. A Matlab programban threshold értéknek a 10-es egész számot választottam, mivel ennél az értéknél az összes mintavételezési frekvencia és bitfelbontás esetében megbízhatóan dönti egy bemeneti audió jelről, hogy az tartalmaz-e beszédet, vagy sem. Ha a jel csúcsossága 10-nél nagyobb, akkor tartalmaz észlelhető beszédet, ha 10-nél kisebb, akkor nem.

A fenti eredményekből látható, hogy ez a módszer jól osztályoz, azonban nagyon sok érv szól az ellen, hogy ezt a meglehetősen triviális megoldást válasszam. A legfontosabb érv a real-time alkalmazhatóság terén merült fel. Az algoritmus egy már előre felvett hangjel összes szegmensét vizsgálja, majd végzi el az elemzést és a döntést a teljes jelre. A keretekre osztást nem végzi el. Ezzel szemben egy olyan megoldásra van szükség, amely egy folyamatosan beérkező hangjelet szegmentál, és végzi el azokon a szegmenseken egyenként az elemzést, valamint figyelembe veszi a beszéd időben változó természetét. Erre a csúcsosság vizsgálatán alapuló módszer nyilvánvalóan alkalmatlan.

5. További módszerek vizsgálata

Miután a fenti módszert nem találtam alkalmazhatónak, az irodalom és több konkrét megvalósítás rendszer tanulmányozásával korábban említett algoritmusokat vizsgáltam meg, figyelembe vettem erőségeit és gyengéit, a gyakorlati alkalmazhatóság és a kijelzőbe ágyaztathatóság szempontjából.

Először a lineáris prediktív kódolást (LPC), mint beszéddetekcióban részt vevő algoritmus [5] alkalmazhatóságát vizsgáltam. Ahogy már korábban leírtam, az LPC egy hatékony módszernek tűnik a beszédhangok elemzésében, valamint az osztályozásban is használható. Jól alkalmazható zajos környezetben, mivel az LPC hibajel segítségével csökkenthető a zaj. A predikciós együtthatókkal leírható, jellemezhető a beszédjel. Az együtthatók meghatározása azonban számításigényes feladat, ugyanis egy mátrix egyenlet megoldása szükséges, amihez invertálni kell a mátrixot. Több megközelítés ismert a műveletek számának és bonyolultságának csökkentésére, például a kovariancia vagy az autókorrelációs módszer [6], melyben szimmetrikussá alakítják a mátrixot. Egy másik módszer a PARCOR modell [7], amelyben nem csak az előre irányuló, hanem a visszafelé irányuló hibával is számolnak. A hibákból számítják ki a reflekciós együtthatókat rekurziv formulával. A mikrokontroller várhatóan nem tudna ilyen műveleteket hatékonyan és gyorsan elvégezni, ezért annak ellenére, hogy az LPC nagy pontosságú beszéddetekciót tenne lehetővé, erőforrás gazdálkodási szempontból nem lenne jó választás.

Egy lehetséges másik megoldás a rejtett Markov modell használata volt, amelyre többféle példát találtam [8]. A Markov modellek diszkrét osztályokkal dolgoznak, rendszerint csak a beszéd meglétét tudják megállapítani, és nem használhatóak hatékonyan zajos környezetben. Az ilyen rendszereknek problémát okozna a beszéd és a zene egymástól való megkülönböztetése. Továbbá az osztályozáshoz használt Viterbi dekódoló megvalósítása egy igen komplex feladat.

A neurális hálók is szóba jöttek lehetséges eszközként. Kevés konkrét példát találtam a megvalósításra, az egyik érdekesebb az úgynevezett konvolúciós hálózat [9] alkalmazása volt. Ez a hálózat egy olyan előrecsatolt neurális hálózat, amelynek osztott súlyai vannak. A hálózat rétegei a zaj, valamint a rövid és hosszú távú spektrális jellemzői állapítják meg, továbbá a klasszifikációt is elvégzik. A jellemzők kiemelése és az osztályozása tehát egyetlen hálózatba van foglalva. A módszer jól alkalmazható olyan környezetben, amire nincs betanítva. A nyilvánvaló előnyök ellenére, nincs mód a taktilis kijelzőben való alkalmazására, mivel neurális hálózat implementálásához nem kedvez a mikrokontroller architektúrája.

6. A mel-cepstrum modulációs energia

Az imént említett eljárásokkal szemben olyan beszéddetektor algoritmust kerestem, amely spektrális vagy cepstrális jellemzőket vesz figyelembe, nem végez komplex, időigényes műveleteket, valamint megbízható osztályozást hajt végre, nem csak a beszédhangok megállapítása, hanem a beszédet a zenétől való megkülönböztetés terén is. Hosszas keresés után végül rátaláltam a 2007-es Pilseni Text, Speech and Dialoge konferencia kiadványában [10] bemutatott mel-cepstrum modulációs energián alapuló algoritmusra, amelyet alkalmasnak találtam komolyabb tanulmányozásra

Az audió jelek elemzésében lényeges paraméterek lehetnek a mel-frekvencia cepstrális együtthatók (MFCC), amelyeket a hangjel cepstrális reprezentációjából származtatnak. A hagyományos cepstrum és a mel-frekvencia cepstrum közti különbség abban nyilvánul meg, hogy a mel-frekvencia cepstrumban (MFC), a frekvencia sávok logaritmikusan helyezkednek el, amelyek alkalmasabban közelítik meg az emberi hallórendszer válaszát, mint a fourier transzformációból származtatott, lineárisan elhelyezkedő frekvencia sávok. Az MFCC-t az alábbi módon származtatjuk:

1. Az (ablakozott) jel Fourier transzformációját vesszük,
2. Leképezzük a fenti művelettel kapott spektrum logaritmikus amplitúdóját, háromszögletű egymást átlapoló ablakokkal a mel-skálára,
3. A mel logaritmikus amplitúdók listájának vesszük a diszkrét cosinus transzformációját,
4. Az MFCC-k lesznek az eredményül kapott spektrum amplitúdói

Lehetséges megkülönböztető paraméterként a modulációs energia (ME), és ennek a cepstrális tartománybeli kiterjesztése, a mel-cepstrum modulációs energia jöttek még szóba. Ezt a két jellemzőt vizsgáltam a későbbiekben.

6.1. Az ME

Tegyük fel, hogy X[n,k] az audió jel n-edik keretének diszkrét Fourier transzformációja (DFT). Ebből tudjuk kinyerni a nagysági modulációs spektrumot (MMS), úgy, hogy a k-dik DTF együttható idő szekvenciájának DFT-jét vesszük:

$$MMS[n,q] = \sum_{p=0}^{P-1} |X[n+p,k]| e^{-j2\pi qp/P}$$

ahol n az audió jel keretindexe, k az első, q a második DFT frekvencia tengelyének indexe, és P a második DFT mérete. Kisebb q érték lassabb, míg a magasabb q gyorsabb spektrális változásokat jelez. A beszédnek gyorsabban változó spektruma van, mint a zenének, a zöngés és zöngétlen hangok váltakozása miatt, ezért a modulációs spektrum alkalmas jellemző a beszéd és a zene megkülönböztetésére.

Az ME kiszámításához általánosságban nem használják fel közvetlenül az első DFT eredményét, hanem a mel-filterbank elemzés eredményét adják meg a második DFT bemeneteként.

Az ME-t a következőképen definiáljuk:

$$ME[n,q] = \frac{\frac{1}{M}\sum_{m=0}^{M-1} |FMS[n,m,q]|^2}{\frac{1}{P}\sum_{p=0}^{P-1} \log(E[n+p])}$$

, ahol FMS[n,m,q] a filterbank kimenetéből számított modulációs spektrum, M a filterbank sorszáma, és E[n] az audió jel n-dik keretének négyzetösszege.

6.2. Az MCME

Tegyük fel, hogy C[n,l] az X[n,k] DFT-jének valós cepstruma. Mivel a C[n,l] egy valós szimmetrikus szekvencia, egy négyszögjeles alacsony querfrenciás emelővel, amely cepstrális együtthatóknak csak az alacsony komponenseit tartja meg, ezért a spektrumnak egy cepstrálisan kiegyenlített becslését kaphatjuk meg.

$$\log S[n,k] = C[n,0] + \sum_{l=1}^{L-1} 2C[n,l]\cos(2\pi lk/K)$$

Ezt a képletet felhasználva, az MMS egy cepstrálisan kiegyenlített becslését számíthatjuk ki, ha a log S[n,k]-t vesszük P pont felett, és a q-dik együtthatókat a következőképen figyelembe véve:

$$MMS'[n,k,q] = \sum_{p=0}^{P-1} \log(S[n+p,k]) e^{\frac{-j2\pi pq}{P}} =$$

$$\sum_{p=0}^{P-1} C[n+p,0] e^{\frac{-j2\pi pq}{P}} + \sum_{l=0}^{L-1} \frac{\cos(2\pi kl)}{K} \sum_{p=0}^{P-1} 2C[n+p,l] e^{\frac{-j2\pi pq}{P}}$$

Az utolsó operandust felhasználva, a mel-cepstrum modulációs spektrumot (MCMS), a következőképen határozzuk meg:

$$MCMS[n,l,q] = \sum_{p=0}^{P-1} C[n+p,l] e^{\frac{-j2\pi pq}{P}}$$

Az előző két képletből kiindulva arra a következtetésre juthatunk, hogy az MMS'[n,k,q] az MCME[n,l,q] lineáris transzformációja. Az MFCC együtthatók kölcsönösen nem korreláltak, ezért az MCME-től jobb teljesítményre számítunk, mint az MMS-től.

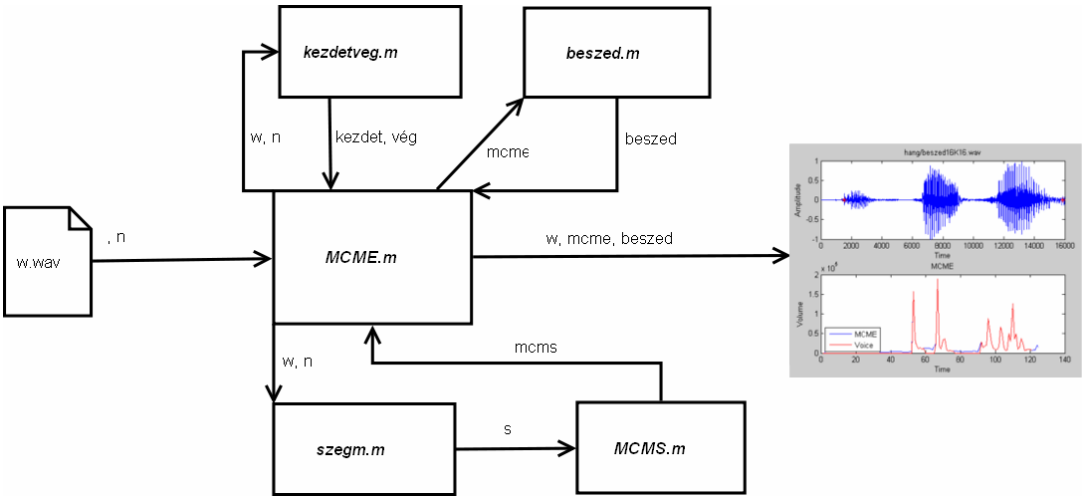
Az MCMS-en alapuló modulációs energiát ezért a beszéd-zene megkülönböztetés egy szignifikáns jellemzőjének tekintjük, az MCME-t a pedig következőképen jelöljük:

$$MCME[n,q] = \frac{\frac{1}{L}\sum_{l=0}^{L-1} |MCMS[n,l,q]|^2}{\frac{1}{P}\sum_{p=0}^{P-1} \log(E[n+p])}$$

E képlet alapján valósítottam meg – a csúcsosságot kiszámító programhoz hasonlóan - Matlab szimulációs környezetben a beszéddetektor algoritmust, amely rendelkezik a beszéd-zene megkülönböztetés képességével.

7. Tervezés és megvalósítás Matlab környezetben

A programot az egyéni funkciókat ellátó modulokra osztás elvét szem előtt tartva írtam meg. Egy .wav fájlt, és egy keret méretet kap bemeneti paraméterként, míg kimenetként az MCME vektor reprezentációját adja vissza, valamint a .wav fájl és az MCME-t, és annak beszédet tartalmazó szakaszait jeleníti meg a **plot()** funkció segítségével. A futása során más funkciókat is meghív, amelyek a szegmensekre osztás, a .wav fájlban jelen levő beszéd kezdetének és végének, valamint a beszédet tartalmazó szakaszok kinyerését végzik.



11. ábra A beszéddetektor program blokk diagrammja

7.1. Modulok

Megoldásom az alábbi funkcionális egységekből tevődik össze:

- **MCME.m**: A program központi modulja, egy .wav fájl elérési útvonalát, és a keretek méretét fogadja bemeneti paraméterként. A hangfájlt vektor alakba konvertálja, majd normálja a vektort. Kiírja a szabványos kimenetre a vektorban észlelt beszéd szakasz kezdetének és végének indexét. A vektornak a szegmens mátrixát veszi, majd kiszámítja az MCMS-t, amelyet szintén mátrix alakban kap vissza. Ezután, az MCMS-t felhasználva, a már bemutatott képlet alapján kiszámítja az MCME-t, amelyből kinyeri a beszédet tartalmazó részeket, amelyeket egy vektorba ment. Végül a plot() függvénnyel kirajzolja a hangfájt és az MCME-et, amelyen kijelzi a beszéd szekvenciákat. A hangfájl grafikonján a beszéd kezdetének és végének helyét jelzi. Visszatérési értéke az MCME vektor.
- **szegm.m**: Ez a modul a szegmentálást végzi el. Egy vektort, egy egész számot, és választható paraméterként egy eltolási tényezőt fogad. A vektort a második paraméterként megadott számú hosszúságú szeletekre bontja, majd ezeket a szeleteket egy mátrix soraiba helyezi. Minden szegmensen egy Hamming ablakozást végez, az ablakok hossza a paraméterként megadott egész szám. Módunk van egymást átlapoló szegmenseket létrehozni, a harmadik paraméterként megadott eltolási tényezőt megadva. Visszatérési értéke a szegmenseket tartalmazó mátrix.
- **MCMS.m**: Ez a funkció az egyetlen bemeneti paraméterként megadott szegmens-mátrixon az MCMS-t számítja ki, amelyet kimenetként ad vissza.
- **kezdetveg.m**: A első bemenetként megadott hangfájl-vektoron, a második paraméterként megadott küszöbértéket felhasználva ad vissza két egész számot: a vektorban található beszéd szakasz kezdetedének és végének indexét.
- **beszed.m**: Az inputként adott MCME vektornak adja meg a beszédet tartalmazó szekvenciáit. Az MCME mellett további két paramétert, egy egész számként megadott küszöbértéket és egy időintervallumnak hosszát fogadja. Az MCME vektor minden elemére megvizsgálja, hogy az a küszöbértéket átlépi-e, ha igen, annak az elemnek az indexétől kezdve a megadott hosszig bezárólag az összes elemet egy vektorba menti. Ezután újra megvizsgálja az index+hossz indexű elemet, és ha annak az értéke is átlépi a küszöböt, akkor a további szekvenciát és hozzáfűzi a vektorhoz. Ez a művelet addig folytatódik, amíg a küszöbértéket át nem lépő elemet talál. Kimenetként a kinyert beszéd-

vektort és az első MCME első azon indexét adja vissza, amelyhez tartozó elem átlépte a küszöbértéket, vagyis a beszéd kezdetének indexét.

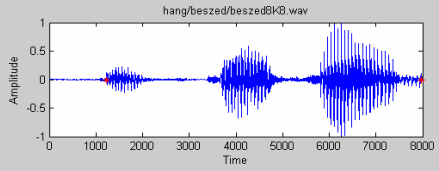
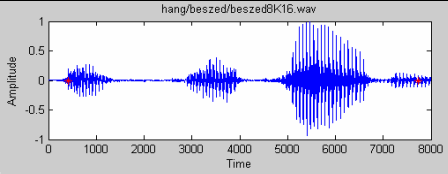
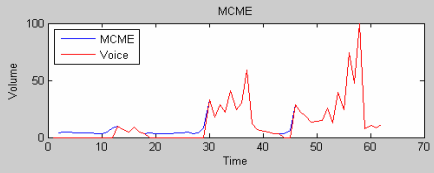
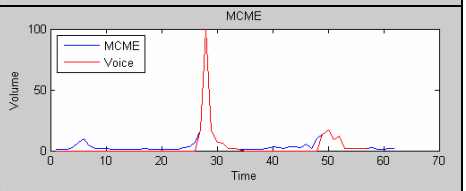
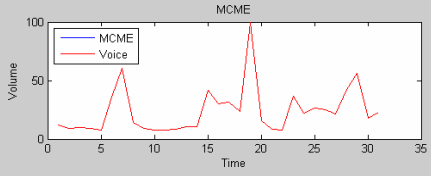
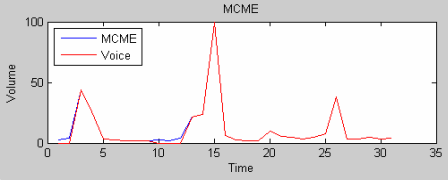
7.2. A program működésének részletes leírása

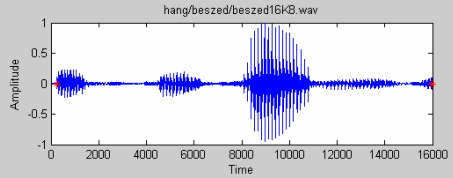
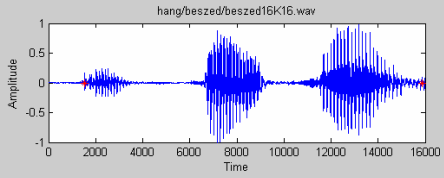
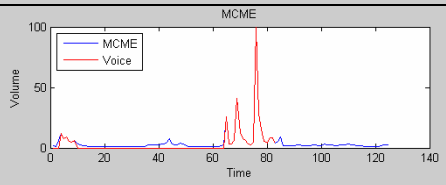
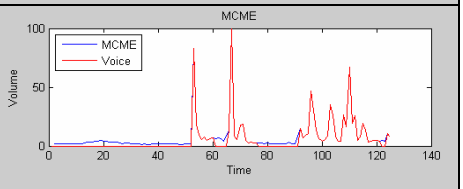
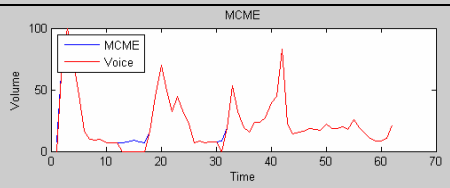
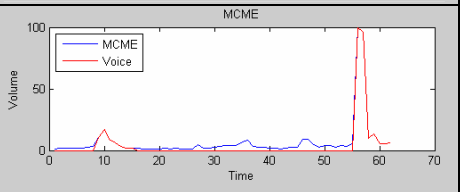
A program működésének lépései a következők:

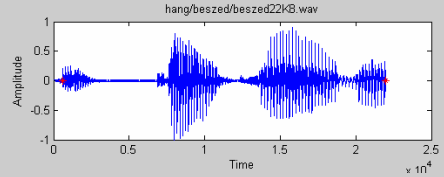
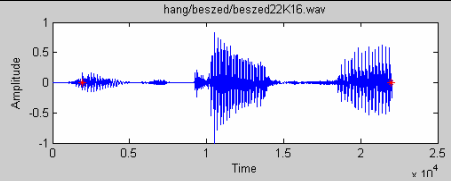
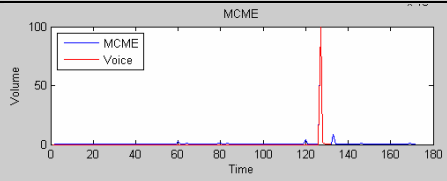
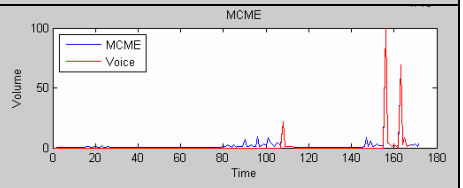
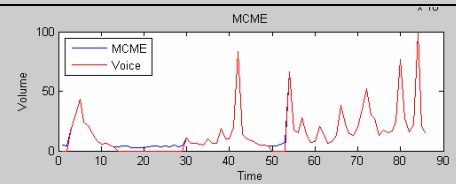
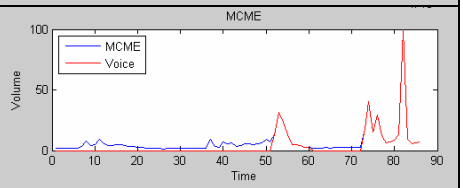
1. Belépési pontja az MCME.m fájl. Egy .wav hangfájl elérési útvonalát, és egy egész számot, a szegmenshosszt kapja bemenő paraméterként.
2. A hangfájt a **wavread()** függvény segítségével vektorrá konvertálja, majd egy normalizálást végez el rajta.
3. A **kezdetveg()** függvénnyel a vektor reprezentációban a küszöbérték szerinti beszéd elejének és végének az indexét szerzi meg, és írja ki a Matlab szabványos kimenetére.
4. A **szegm()** függvény elvégzi a vektoron a szegmentálást. Egy mátrixot épít, a mátrix sorai a vektor szegmensei lesznek. A szegmenshossz a szegm() függvénynek bemenetként adott egész szám. Alapértelmezésként nem generál egymást átlapoló szeleteket. Minden keretet egy Hamming ablakkal lát el, az ablakok hossza azonos a szegmensek hosszával.
5. Az **MCMS()** funkcióval a szegmens mátrix minden sorára kiszámítja az MCMS-t, a korábban erre vonatkozó képlet alapján. Valamennyi sornak veszi a valós cepstrumát, annak a Fourier transzformáltját, abszolút értékét, majd decibelbe átszámítja a kapott eredményt. Az így kapott mátrix sorai a hangfájl kereteinek MCMS értékei.
6. Kiszámítja az MCME-t. A már ismertetett képlet szerint, a nevező a keretek MCMS-einek abszolút értékeinek négyzetösszegeinek, míg a nevező a keretek abszolút értékű négyzetösszegének logaritmusainak átlaga.
7. A **beszed()** függvénnyel kinyerjük a beszédet tartalmazó részeket. Az MCME minden elemére megvizsgálja, hogy az a küszöbértéket átlépi-e, ha igen, annak az elemnek az indexétől kezdve a megadott hosszig bezárólag az összes elemet egy vektorba menti. Ezután újra megvizsgálja az index+hossz indexű elemet, és ha annak az értéke is átlépi a küszöböt, akkor a további szekvenciát és hozzáfűzi a vektorhoz. Ez a művelet addig folytatódik, amíg a küszöbértéket át nem lépő elemet talál.
8. Megjeleníti a hangfájl-t, és az MCME-t. A beszédet az MCME grafikonján piros színnel ábrázolja. A hangfájl grafikonján a beszéd kezdetének és végének helyét jelzi.

7.3. A Matlab implementáció teszteredményei

A Matlab programot számos bemeneti hangfájlra teszteltem. A lehetséges variációk a hangfájlok tulajdonságaira és a program paramétereinek beállításaira terjedtek ki. A beszéd a saját magam által mondott „egy, két, há” mondat. A jeleket 8, 16, és 22 kHz-es mintavételi frekvencián, valamint 8 és 16 bit felbontásban rögzítettem, a fájlokat 128-as és 256-os keret mérettel szegmentáltam. Az alábbi táblázatok az egyszerű, zajmentes beszédre kapott teszteredmények ábráit tartalmazzák.

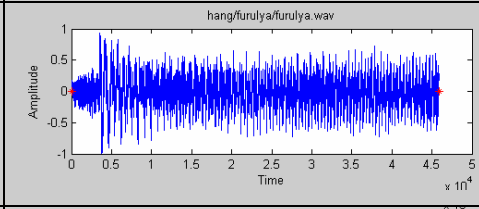
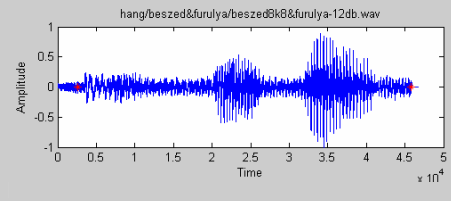
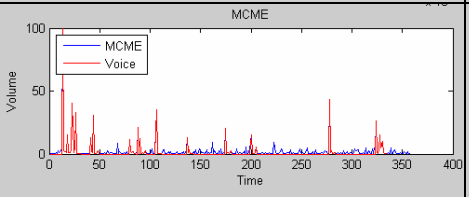
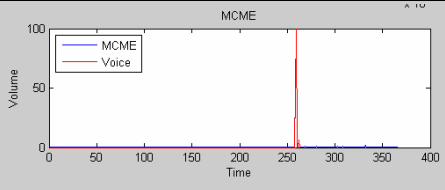
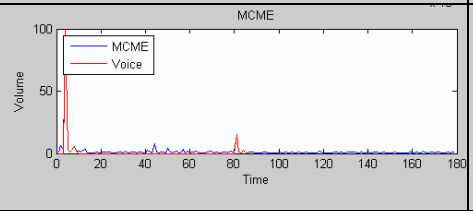
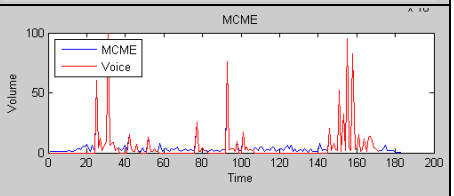
	8 kHz, 8 bit	8 kHz, 16 bit
Eredeti jel		
MCME, 128-as keret		
MCME, 256-os keret		

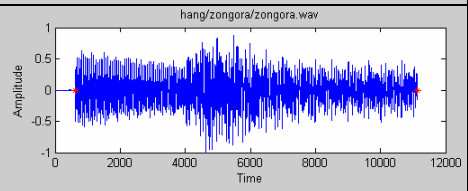
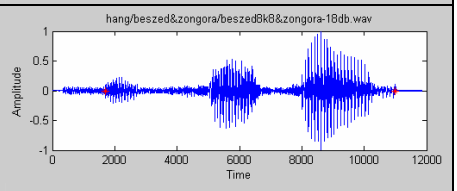
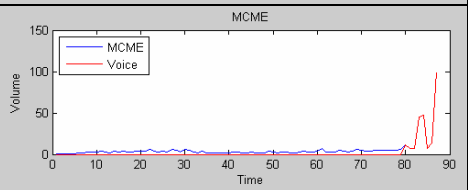
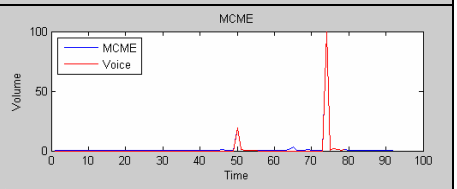
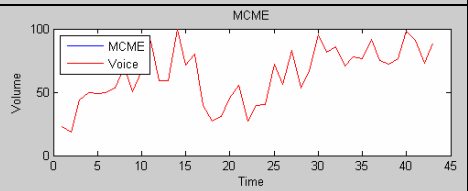
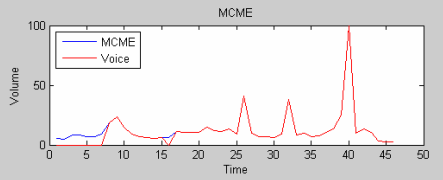
	16 kHz, 8 bit	16 kHz, 16 bit
Eredeti jel		
MCME, 128-as keret		
MCME, 256-os keret		

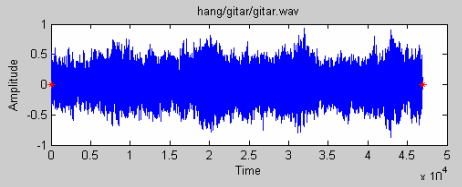
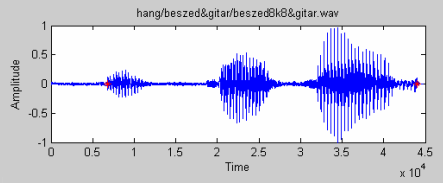
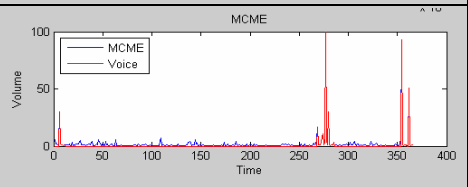
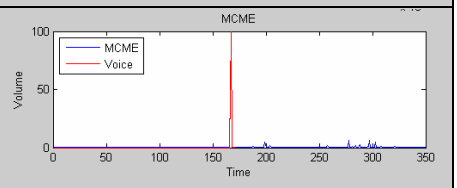
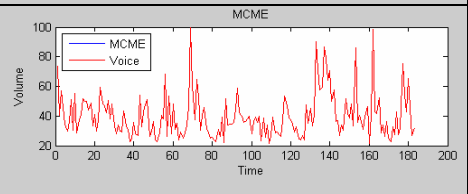
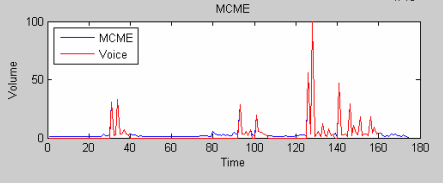
	22 kHz, 8 bit	22 kHz, 16 bit
Eredeti jel		
MCME, 128-as keret		
MCME, 256-os keret		

Megfigyelhető, hogy az alacsony felbontású jelek esetében, a 128-as keretméret alkalmazásával detektálhatjuk pontosabban a beszédet, míg a magasabb frekvenciájú jeleknél a 256-os méretet érdemes alkalmazni. A legpontosabb eredményt a 8-16 kHz-es frekvenciával, és 8 bites felbontással kaphatjuk.

Az következő táblázatokban háromféle hangszer által adott dallamokon teszteltem az algoritmust, furulyán, zongorán és gitáron. Az MCME jelalakja nem függ a hangerősségtől, hanem kizárólag az eredeti hang jelalakjától. A táblázatok második oszlopai a tiszta zenét, harmadik oszlopai a beszéd és a zene együttes jelenlétét ábrázolják. A furulya hangerejét 12, a zongoráét és a gitárét 18 decibellel csökkentettem.

	Furulya	Beszéd és furulya
Eredeti jel		
MCME, 128-as keret		
MCME, 256-os keret		

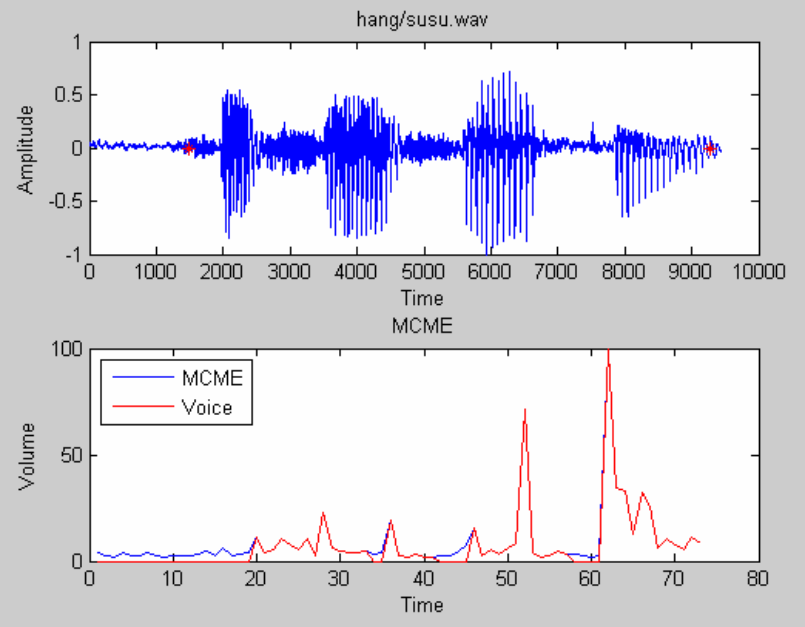
	Zongora	Beszéd és zongora
Eredeti jel		
MCME, 128-as keret		
MCME, 256-os keret		

	Gitár	Beszéd és gitár
Eredeti jel		
MCME, 128-as keret		
MCME, 256-os keret		

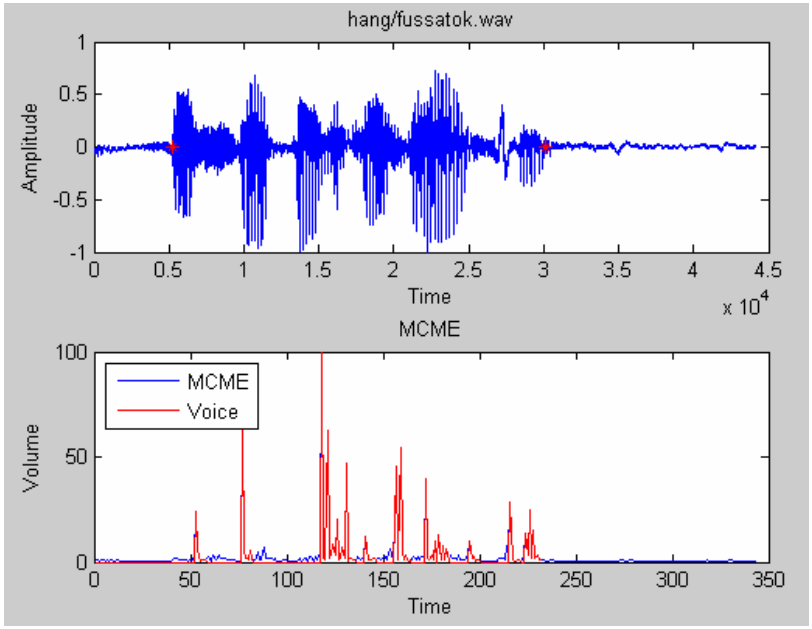
A program nagy pontossággal dönti el 128-as ablakmérettel, hogy a zenei jelek nem tartalmaznak beszédet. A furulya esetében még a 256-os kerethosszúság is pontos találatot ad, ez

összefügghet a jel alacsony frekvenciájával. Magasabb frekvenciájú bemeneteknél azonban ez a méret már nem alkalmas a beszéddetektálásra. A beszéd és a zene együttesével azonban a 256-os hosszal nagyobb pontosságot értünk el, mint a 128-assal.

Az algoritmus olyan beszéd jeleken is alkalmazható, amelyeknen nehezen különböztethetőek meg a zöngés és frikativ hangok. Az alábbi ábrákon a „Süsü a sárkány”, és a „Fussatok, dzsidások” mondatokra generált MCME kimenetek láthatóak, 128-as ablakmérettel:

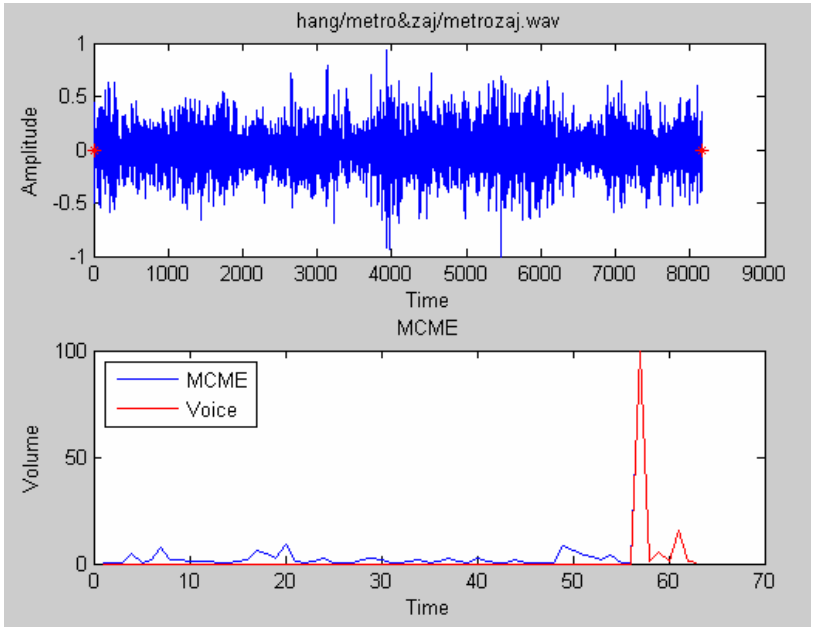


12. ábra Süsü a sárkány



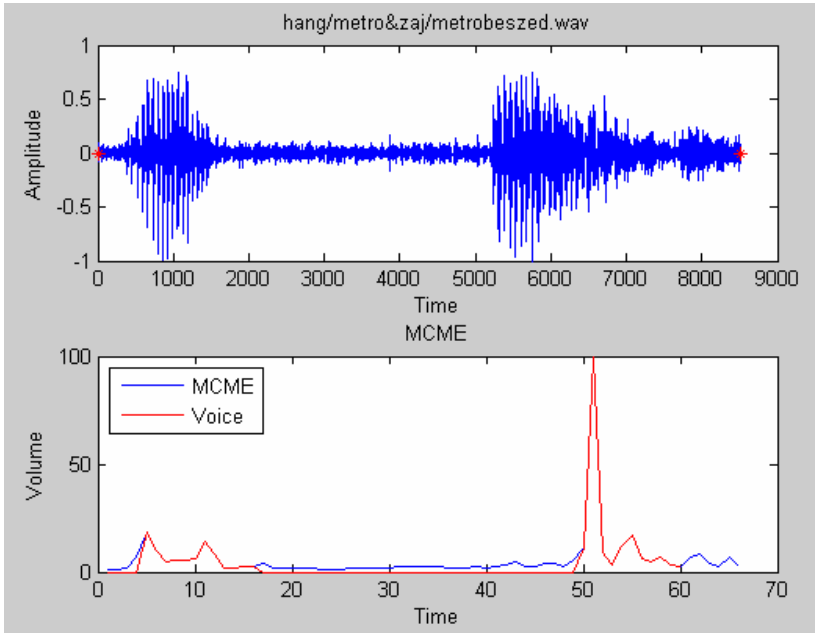
13. ábra Fussatok, dzsidások!

Egyszerű metrózajra a program kis hibától eltekintve nem detektál beszédet:



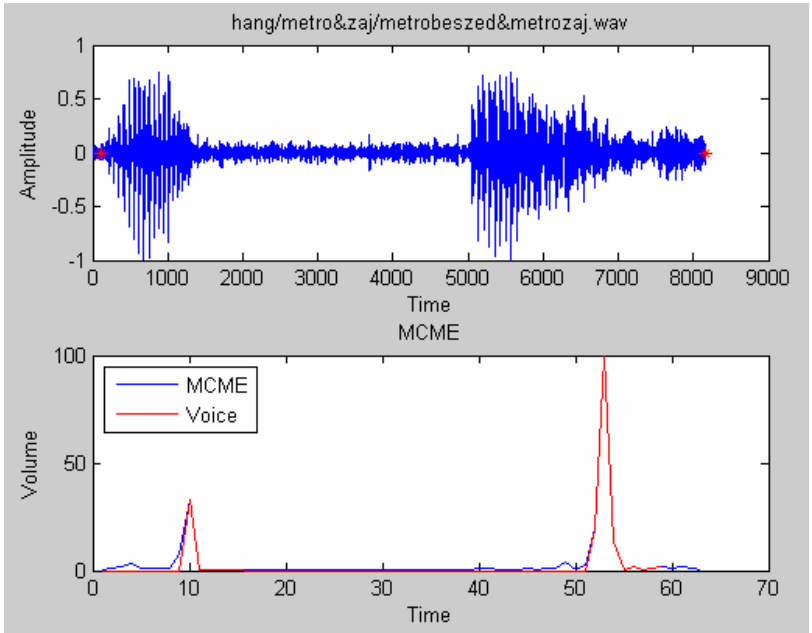
14. ábra Metrózaj

A metrón való beszédet viszont észleli a detektor:



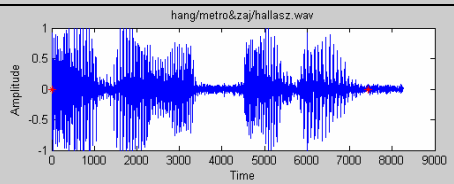
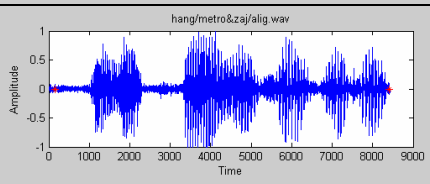
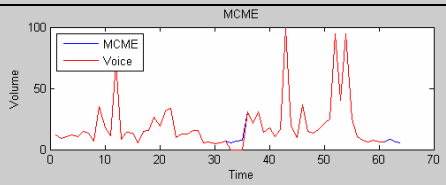
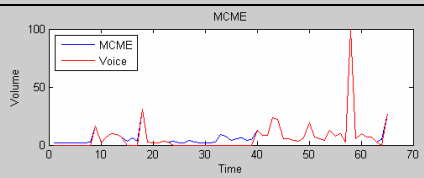
15. ábra Beszéd a metrón

Az előző jelre rákevertem egy kis metrózajt, az eredmény hasonló lett.

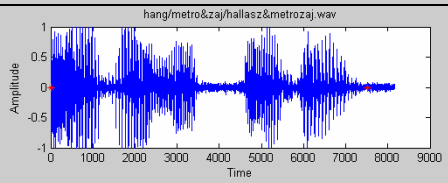
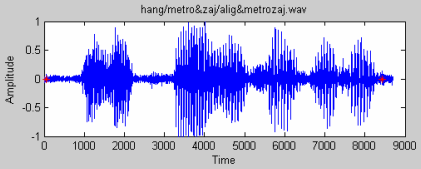
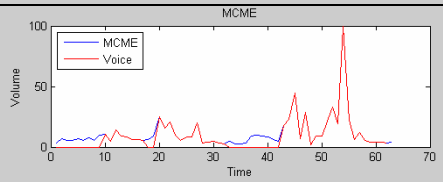
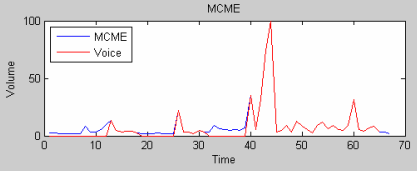


16. ábra Beszéd és metrózaj

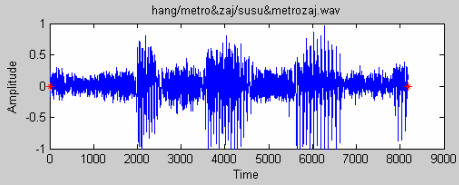
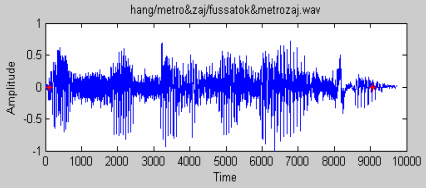
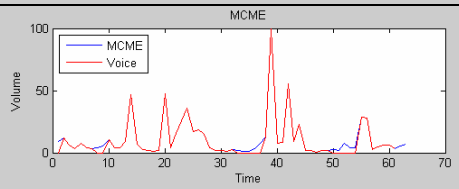
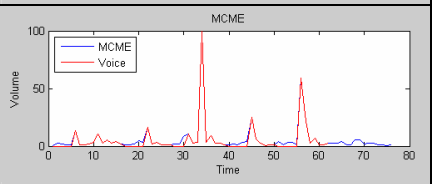
A főváros egy forgalmas pontján, a Ferenciek Terén is felvettem egy beszédet, a „Szia, hallasz engem?” és az „Alig hallom magamat” mondatokat:

	„Szia, hallasz…”	„Alig hallom…”
Eredeti jel		
MCME		

Ezekre a jelekre később metrózajt kevertem:

	„Szia, hallasz…”, metrózaj	„Alig hallom…”, metrózaj
Eredeti jel		
MCME		

Végül a „Süsü a sárkány”, és a „Fussatok, dzsidások” mondatoknál is hasonlóképpen jártam el:

	„Süsü…”, metrózaj	„Fussatok…”, metrózaj
Eredeti jel		
MCME		

Megállapítható, hogy az MCME-en alapuló algoritmus nagy pontossággal, alacsony hibaszázalék mellett képes észlelni és megjeleníteni a beszédet tartalmazó szekvenciákat erősen zajos környezetben is.

8. Tervezés és megvalósítás PC környezetben

A Matlabos implementációval rendelkezésünkre áll egy konkrét megvalósítás, amely képes hangfájlokon végrehajtani a beszéddöntést, így megbizonyosodhatunk az algoritmus működőképességéről. Lépéseket tehettem hát arra vonatkozóan, hogy az algoritmust a taktilis kijelző vezérlő programjába integráljam. Ennek megfelelően egy C nyelvű újraírás szükséges, hogy lefordíthassam a Microchip MPLAB C18 fordítója segítségével, amellyel lehetővé válik a célkörnyezetbe való integrálás. Előtte azonban a C nyelvű megvalósíthatóságról kellett megbizonyosodnom, mivel ebben az esetben nem állnak rendelkezésünkre a Matlabhoz tartozó eszközkészletek, kiegészítések, valamint a gyakori részműveletek automatizálása, például a .wav fájlok szegmentálása, jelfeldolgozási algoritmusok, tömbökön végzett műveletek, információ grafikus megjelenítése, stb. Ezeket részben saját, illetve már fellelhető megoldásokkal kell pótolni.

Elsőként az MCME algoritmus alapműveletét, az FFT (Fast Fourier Transform) algoritmus C nyelvű megvalósítását tűztem ki célul. Tehát a kutatási munkám következő fejezete a lehető legoptimálisabb FFT változat megkeresése, és alkalmazásának vizsgálata.

8.1. Az FFT elméleti ismertetése

A digitális jelfeldolgozásban alkalmazott Fourier analízis talán legtöbbet felhasznált számítása a Fourier transzformáció [11], amelynek következtében egy függvényt időtartomány helyett frekvenciatartományban ábrázolunk. Tekintsük h(t)-t, egy függvénynek az időtartományban, amely tehát t-től függ. Legyen H(f) a frekvenciatartomány szerinti ábrázolás, amely az f frekvenciának függvénye, ahol $-\infty < f < \infty$. A h(t) és H(f) ugyanannak a függvénynek két különböző reprezentációja. Az ábrázolási formák között a Fourier transzformációval léphetünk át:

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f t} dt$$

$$h(t) = \int_{-\infty}^{\infty} H(f) e^{-2\pi i f t} dt$$

Ha t-t másodpercekben mérjük, akkor az f-et Hertzben.

A legtöbb esetben a h(t) egy egyenletes időközökkel mintavételezett jel. Legyen Δ az minták közötti időköz, ahol a minták sorozata:

$$h_n = h(n\Delta) \quad n = ...,-3,-2,-1,0,1,2,3,...$$

A Δ időköz reciproka a mintavételezési idő; ha Δ-t másodpercben mérjük, akkor a mintavételezési idő a minta/másodperc.

Becsüljük egy véges számú mintát tartalmazó jel Fourier transzformációját. Tegyük fel, hogy N darab mintával rendelkezünk:

$$h_k \equiv h(t_k), \quad t_k \equiv k\Delta, \quad k = 0,1,2,...,N-1$$

Ekkor a hk diszkrét jel Fourier transzformációja:

$$H_n \equiv \sum_{k=0}^{N-1} h_k e^{2\pi ikn/N}$$

, ahol hk és Hn is komplex számok, és mindegyikükből N darab van. A diszkrét inverz Fourier transzformáció:

$$h_k \equiv \frac{1}{N} \sum_{n=0}^{N-1} H_n e^{-2\pi ikn/N}$$

A képletek közötti különbség mindössze a kitevő előjele és egy N-el való leosztás. Ezért a kis módosítástól eltekintve ugyanazzal a képlettel számolható az eredeti és az inverz transzformáció.

Szeretnénk tudni, hogy mennyi műveletet igényel egy N pontos diszkrét Fourier transzformáció kiszámítása. Legyen W egy komplex szám:

$$W \equiv e^{2\pi i/N}$$

Ebben az esetben a diszkrét Fourier transzformáció így írható fel:

$$H_n \equiv \sum_{k=0}^{N-1} W^{nk} h_k$$

A hk értékelet tartalmazó vektort a W értékeket tartalmazó mátrixszal szorozzuk össze, melynek eredménye a Hk értékeket tartalmazó vektor lesz. Ez a mátrixművelet N^2 darab komplex szorzást igényel, ezen kívül a W hatványait előállító műveleteket. Ezért a diszkrét Fourier transzformációt O(N^2) számításigényűnek mondjuk, amely meglehetősen sok. Létezik azonban egy lényegesen gyorsabb algoritmus, az úgynevezett gyors Fourier transzformáció (FFT), amely O(N log N) műveletet igényel. Az N log N és az N^2 idők közötti különbség számottevő. Például, ha N=10^6, akkor az eltérés körülbelül 30 másodperc és 2 hét processzor idő közötti különbség. Az FFT algoritmus J.W. Cooley és J.W. Tukey munkája által a 60-as évek közepén

vált általánosan ismertté, azonban Danielson és Lánczos már 1942-ben kidolgozták a gyorsabb megoldást. Kimutatták, hogy egy N pontos diszkrét Fourier transzformáció felírható két N/2 pontos Fourier transzformáció összegeként. Az egyik az eredeti N pont páros számú, a másik a páratlan számú pontjaiból tevődik össze:

$$
\begin{aligned}
F_k &= \sum_{j=0}^{N-1} e^{2\pi ijk/N} f_j \\
&= \sum_{j=0}^{N/2-1} e^{2\pi ik(2j)/N} f_{2j} + \sum_{j=0}^{N/2-1} e^{2\pi ik(2j+1)/N} f_{2j+1} \\
&= \sum_{j=0}^{N/2-1} e^{2\pi ikj/(N/2)} f_{2j} + W_k \sum_{j=0}^{N/2-1} e^{2\pi ikj/(N/2)} f_{2j+1} \\
&= F_k^e + W^k F_k^o
\end{aligned}
$$

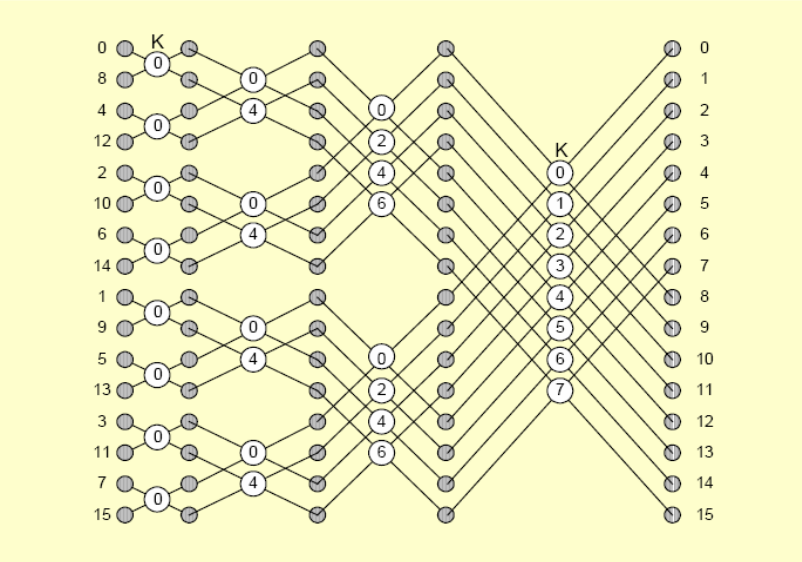
Fek eredeti pontok páros számú, Fok a páratlan számú pontjainak a k. komponense. A fenti módszert dekompozicónak nevezzük.

A Danielson-Lánczos lemma egy nagy előnye, hogy rekurzívan használható. Miután a problémát felosztottuk páros és a páratlan pontok kiszámítására, a páros számú pontok kiszámítását is feloszthatjuk annak N/4 számú páros és páratlan pontjainak kiszámítására.

Abban az esetben a legegyszerűbb a számítás, ha az eredeti N pontok száma a 2 egész számú hatványa, tehát ez az ajánlott méret. Így addig használhatjuk a Danielson-Lánczos lemmát, amíg az adatot egy méretű transzformációkra nem osztottuk. Az egy méretű transzformáció kimenete pedig maga az eredeti adat lesz. Mindegyik kombinálás N számú műveletet igényel, és összesen log N számú kombinációval rendelkezünk, így a teljes algoritmus műveletigénye N log N lesz.

A további egyszerűsítés céljából felcserélhetjük az elemeket úgy, hogy a páros számúak egymás mellé kerüljenek, hasonlóan a páratlan számúakkal, így az elem indexét bináris formában ábrázolhatjuk. Ezt a módszert bit felcserélésnek nevezzük, és a Danielson-Lánczos lemmával együtt használva igen hatékonnyá tehető az FFT. A pontok az egy pontos Fourier transzformációk lesznek. A szomszédos párokat összefűzzük, amivel kételemű transzformációkat kapunk, majd a szomszédos kettő eleműeket négy eleműekké, és így tovább, amíg eredeti adat első és második felét össze nem fésüljük.

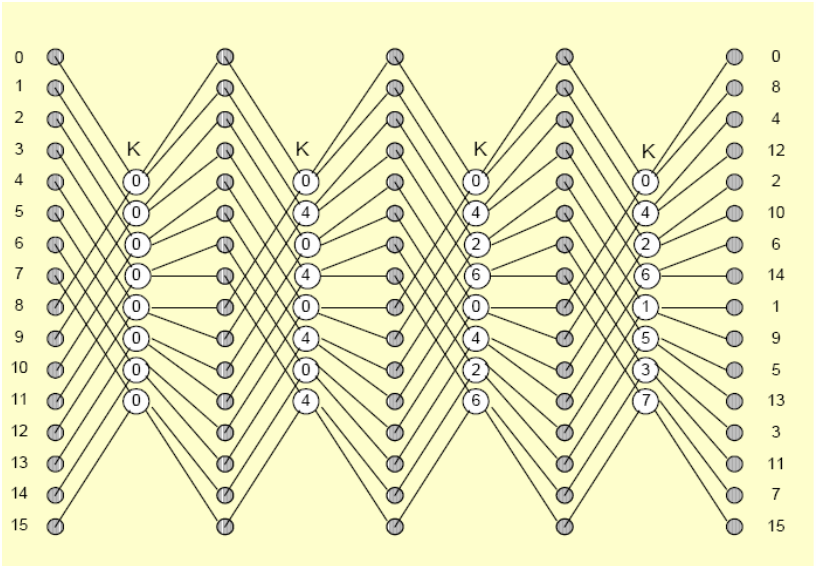
Az algoritmust kétféle elrendezésben szokták használni [12]. A klasszikus, a számított értékeket „helybenhagyó” algoritmust elemezve jól látható a dekompozíció. A tizenhat elem nyolc, majd négy, aztán kétfelé osztottan lesz feldolgozva, négy ütemben, ahogyan az alábbi ábrán látható.



18. ábra A klasszikus, in-place algoritmus

Mindegyik kimeneti oszlop minden eredményét ugyanazzal az algoritmussal, a pillangó művelettel kell számítani. Mindkét elrendezés alapműveletének, sőt alapvetően az FFT speciális műveletének számít a „pillangó” művelet.

A másik változatban, azonos struktúrájúak a fokozatok, az algoritmus ugyanaz marad. Most a kimeneten kell helyretenni a mintákat. A megfelelő sorrendet itt is a bit felcseréléssel lehet elérni.



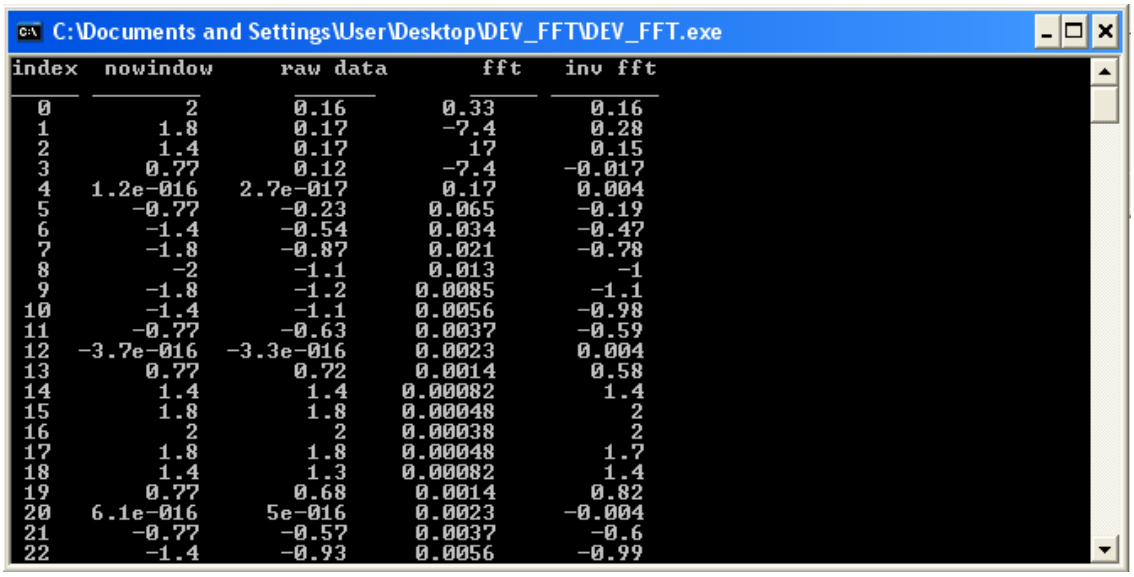
19. ábra Az állandó geometriájú algoritmus

8.2. FFT algoritmus implementálása és tesztelése

Az algoritmust, és az azt kipróbáló tesztprogramot először Visual C++ környezetben írtam meg. Három részre, egy külső és két belső rutinra osztottam az eljárást: Az **FFT()** először megszámolja, hogy a bemeneti adattömb mérete a kettő hányszoros hatványa, ezután meghívja az **FFT2()** és a **REORDER()** belső rutinokat, majd megvizsgálja, hogy inverz transzformációt végeztünk-e, ha igen, akkor az adatokat leosztja N-el (a tömb méretével), és invertálja az Euler formulát. Az FFT2() az algoritmus „magja”, elvégzi a kulcsműveleteket, mint az Euler formula kiszámítása, és a bit reverz művelet. A REORDER() újra az eredeti sorrendbe rendezi a bit felcserélő művelet által megkevert adatelemeket. Az FFT() az egyik paraméterének beállításával az inverz transzformációt hajtja végre. Írtam továbbá egy ablakozó függvényt, amely a négyszög, Parzen, Hanning, Hamming, Welch, és Blackman ablakozást tudja elvégezni.

A tesztprogramomban a próbaadat tömböt a **2*cos(π*i/8)** függvény értékeivel töltöttem fel. Ezután egy Hamming ablakot tettem rá, majd kiszámoltam a valós FFT-t, az inverz FFT-t.. Ezt követően kiírattam őket az ablakozás előtti, ablakozás utáni bemeneti adatokkal együtt.

Annak tesztelésére, hogy a függvény megfelelően működik, elvégeztem ugyanezeket a műveleteket ugyanarra a bemeneti adatra a Matlab **fft()** függvényével is, amely az FFTW könyvtáron alapszik. Ezután összehasonlítottam az eredményeket.



index	nowindow	raw data	fft	inv fft
0	2	0.16	0.33	0.16
1	1.8	0.17	-7.4	0.28
2	1.4	0.17	17	0.15
3	0.77	0.12	-7.4	-0.017
4	1.2e-016	2.7e-017	0.17	0.004
5	-0.77	-0.23	0.065	-0.19
6	-1.4	-0.54	0.034	-0.47
7	-1.8	-0.87	0.021	-0.78
8	-2	-1.1	0.013	-1
9	-1.8	-1.2	0.0085	-1.1
10	-1.4	-1.1	0.0056	-0.98
11	-0.77	-0.63	0.0037	-0.59
12	-3.7e-016	-3.3e-016	0.0023	0.004
13	0.77	0.72	0.0014	0.58
14	1.4	1.4	0.00082	1.4
15	1.8	1.8	0.00048	2
16	2	2	0.00038	2
17	1.8	1.8	0.00048	1.7
18	1.4	1.3	0.00082	1.4
19	0.77	0.68	0.0014	0.82
20	6.1e-016	5e-016	0.0023	-0.004
21	-0.77	-0.57	0.0037	-0.6
22	-1.4	-0.93	0.0056	-0.99

20. ábra A saját FFT implementáció eredményei

```
>> t=(0:31)'  >> c=2*cos(t*pi/8)  >> h=hamming(32).*c  >> f=real(fft(h))  >> i=ifft(f)
```

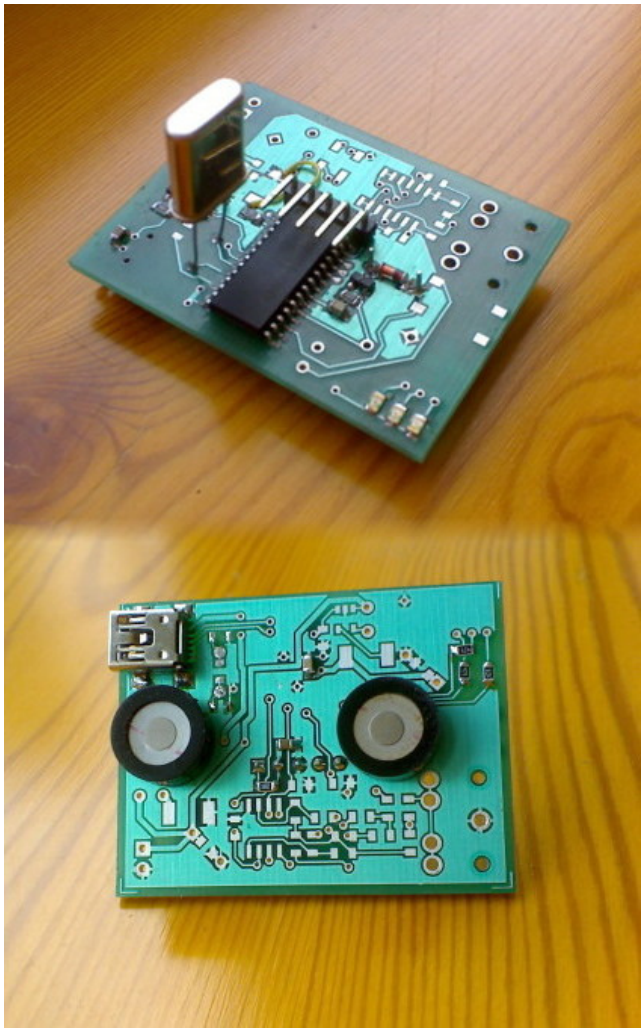
t =	c =	h =	f =	i =
0	2.0000	0.1600	0.3255	0.1600
1	1.8478	0.1652	-7.3706	0.1565
2	1.4142	0.1659	16.8495	0.1462
3	0.7654	0.1243	-7.4116	0.1070
4	0.0000	0.0000	0.1737	-0.0000
5	-0.7654	-0.2271	0.0654	-0.1989
6	-1.4142	-0.5377	0.0343	-0.4787
7	-1.8478	-0.8691	0.0205	-0.7858
8	-2.0000	-1.1266	0.0130	-1.0336
9	-1.8478	-1.2108	0.0085	-1.1258
10	-1.4142	-1.0502	0.0056	-0.9885
11	-0.7654	-0.6288	0.0037	-0.5986
12	-0.0000	-0.0000	0.0023	0.0000
13	0.7654	0.7211	0.0014	0.7008
14	1.4142	1.3844	0.0008	1.3584
15	1.8478	1.8434	0.0005	1.8261
16	2.0000	1.9953	0.0004	1.9953
17	1.8478	1.8088	0.0005	1.8261
18	1.4142	1.3325	0.0008	1.3584
19	0.7654	0.6804	0.0014	0.7008
20	0.0000	0.0000	0.0023	0.0000
21	-0.7654	-0.5683	0.0037	-0.5986
22	-1.4142	-0.9267	0.0056	-0.9885
23	-1.8478	-1.0408	0.0085	-1.1258
24	-2.0000	-0.9407	0.0130	-1.0336
25	-1.8478	-0.7026	0.0205	-0.7858
26	-1.4142	-0.4196	0.0343	-0.4787
27	-0.7654	-0.1707	0.0654	-0.1989
28	-0.0000	-0.0000	0.1737	-0.0000

21. ábra A Matlab FFT impementáció eredményei

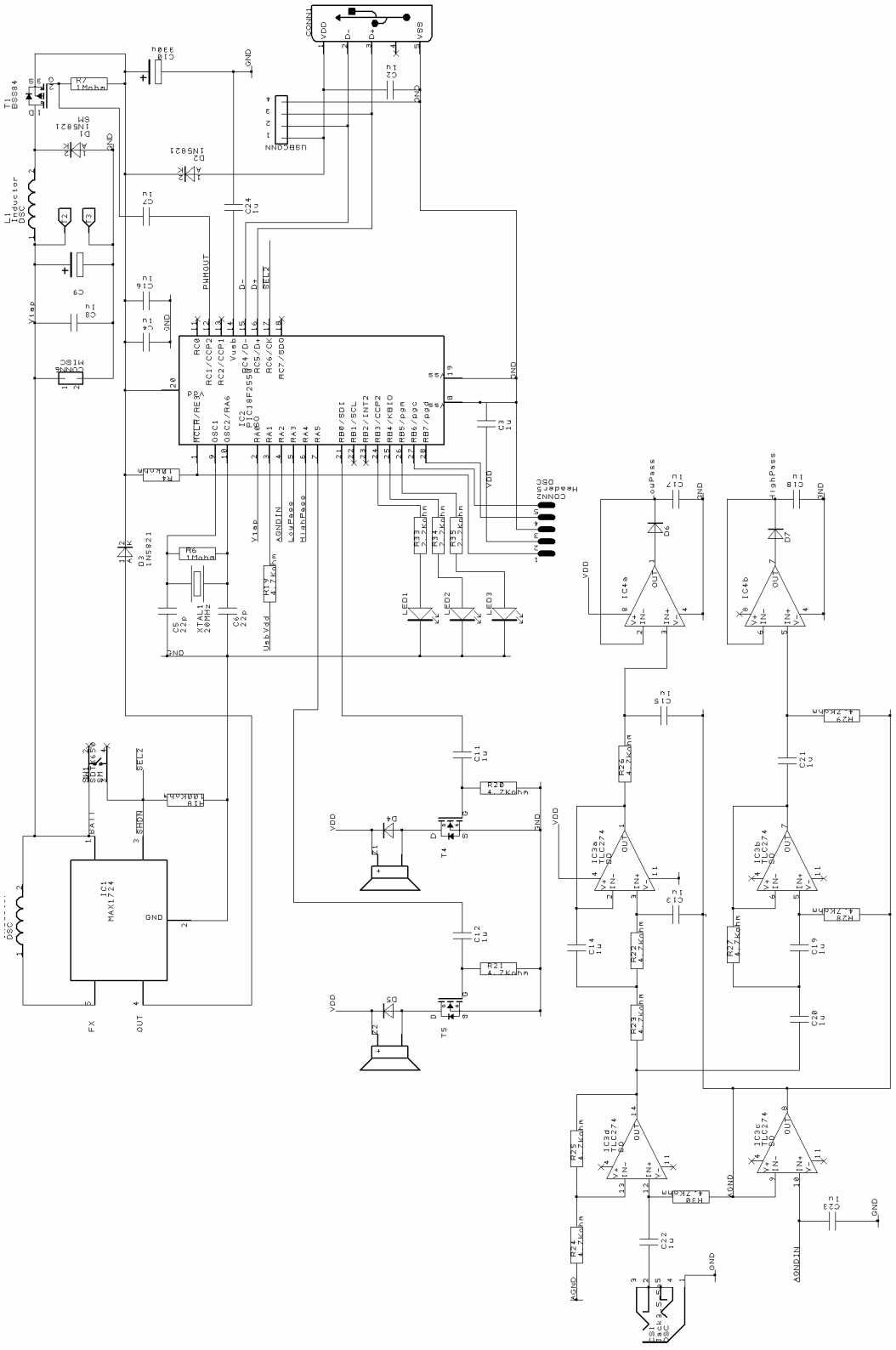
Látható, hogy a kapott eredmények megegyeznek, tehát arra következtethetünk, hogy az általam írt FFT függvény megfelelően működik.

9. A célhardver bemutatása

A projekt előrehaladtával a taktilis kijelző eszközt kicseréltük egy másik, a mobil alkalmazhatóság követelményeinek jobban megfelelő modellre. Ez a műszer a korábbi, telefon membránokból és mágnesekből összeszerelt eszközzel ellentétben egy nyáklap, amelynek főbb alkatrészei két stimulátor, egy kristály oszcillátor, egy USB csatlakozó, és egy vezérlő chip. A leglényegesebb különbség az, hogy a mikrokontroller immár nem egy külön demolapon helyezkedik el, hanem magán a kijelzőn, amely így a kompakt, önálló rendszernek tekinthető. A kijelzőn továbbá még 3 db LED található, a vezérlés tesztelésének céljából.



22. ábra A taktilis kijelző elől- és hátulnézete



23. ábra A taktilis kijelző kapcsolási rajza

A CS1 bemenetre egy mikrofont csatlakoztatunk, melyen keresztül a hangjel előerősítés után kerül a mikrokontroller analóg bemenetére. A jelfeldolgozását, azon belül a mintavételezést a mikrokontroller végzi. Az IC3, IC4 szűrők a zöngés és frikativ hangok megkülönböztetéséért felelősek, amely Földi Balázs munkájához tartozik. A Z1, és Z2 nevű, hangszórónak jelölt egységek szolgáltatják a megfelelő bőrstimulációt. Az áramköri lapon elhelyezett IC az akkumulátor felhasználásával, energiával látja el a kijelzőt, ezen kívül egy akkumulátor töltésére alkalmas áramköri részlet is megtalálható. Az USB csatlakozón keresztül mód nyílik a mikrokontrollerrel való kommunikáció felvételére, annak vezérlésére.

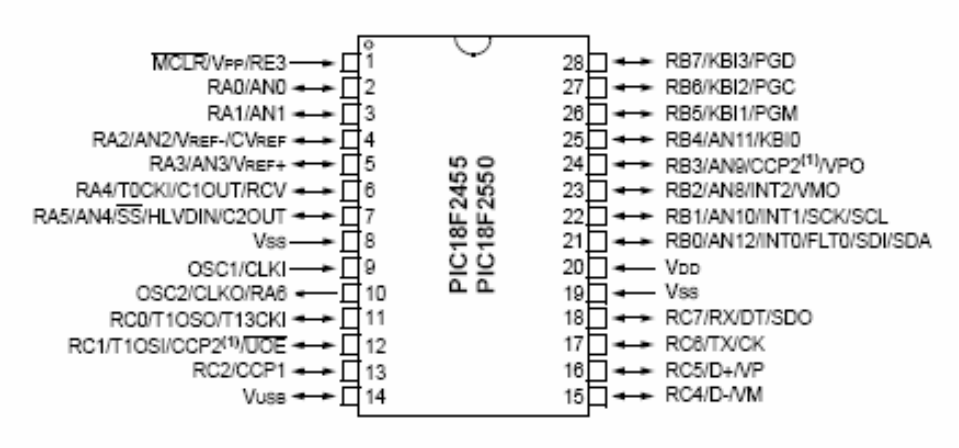
9.1. A mikrokontroller bemutatása

A taktilis kijelzőt vezérlő processzor a Microchip által gyártott, PIC18F2550 típusú mikrokontroller [13]. Felépítésében, és funkcióiban megegyezik a PICDEM FS USB demonstrációs lapot vezérlő PIC18F4550 chippel, a legfontosabb különbségek a következők:

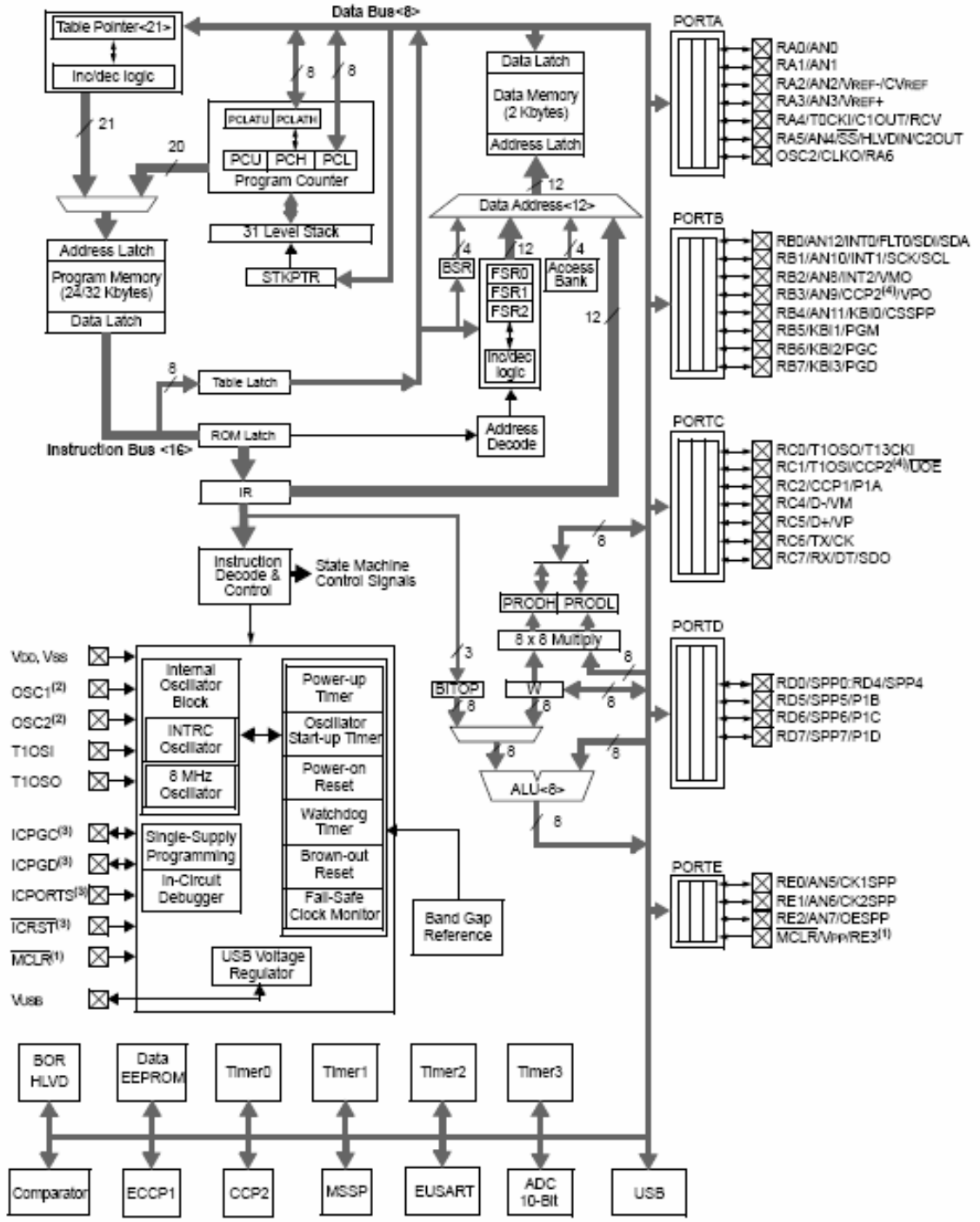
A PIC18F2550 a PIC18F4550-hez képest,

- 44 láb helyett 28 lábbal rendelkezik,
- eggyel kevesebb interrupt forrása van,
- eggyel kevesebb I/O portja van,
- nincs párhuzamos portja,
- a 10 bites A/D átalakítója 13 helyett 10 bemeneti csatornával rendelkezik

A 28 lábas chipre méretoptimalizálási okokból került a választás, a különbségek a működés szempontjából elhanyagolhatóak, a kijelző vezérléséhez szükséges valamennyi funkció a rendelkezésünkre áll.



24. ábra A PIC18F2550 pin kiosztása



25. ábra A PIC18F2550 blokk diagrammja

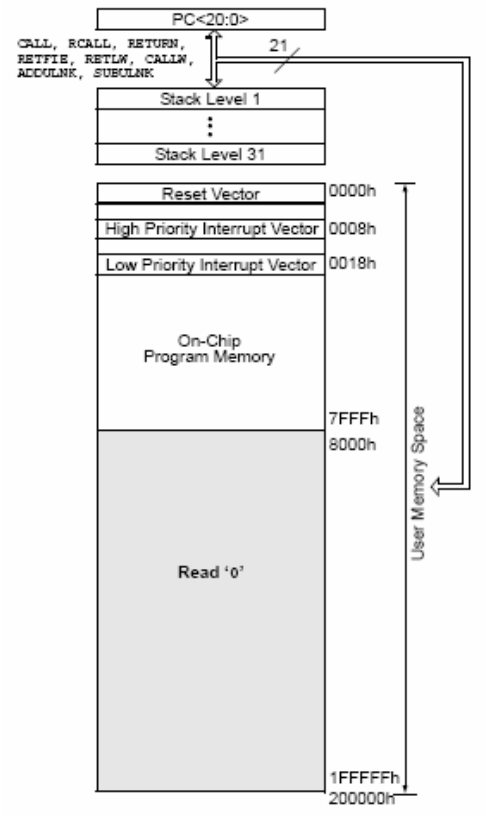
9.1.1. A PIC18F2550 memória felépítése

A chip a Harvard architektúra szerint épül fel, a memóriája három részre lett osztva:

- program memória,
- adat RAM,
- adat EEPROM

Az ilyen architektúráknál megszokottan, a program és az adat memória eltérő buszokat használ, ez a két memória tár egyidejű hozzáférését teszi lehetővé. Az adat EEPROM egy perifériku eszköznek tekinthető, mivel kontroll regiszterek biztosítják a címzést és a hozzáférést. A program memória 16384 byte, az adat memória 2048 byte méretű.

A program memóriához egy 21 bites programszámláló tartozik, amely egy 2 Mbyte-os memóriatárat tud megcímezni. A chip két interrupt vektorral rendelkezik, a Reset vektor a 0000h címen, a high és a low interrupt vektorok a 0008h és a 0018h címeken.



26. ábra A program memória elrendezése

A program memória byteonként hajtja végre a címzést, az utasítások kettő vagy négy byteon vannak tárolva. Egy utasításszó legkisebb fontosságú bájtja mindig páros című memóriahelyen tárolódik. Annak érdekében, hogy az utasítások határainak elrendezései meg legyenek tartva, a programszámláló kettesével inkrementálódik, így a legkisebb fontosságú byte 0 lesz.

Az adat memória egy statikus RAM-ként valósul meg. Mindegyik regiszternek van egy tizenkét bites címe, ami akár 2048 byte méretű adat memóriát tesz lehetővé. A memória tár nyolc szegmensre osztott, mindegyikük 256 byte méretű. Kétféle regiszter típust tartalmaz a memória, az úgynevezett különleges funkciós regisztert (SFR), és az általános célú regisztert (GPR). Az SFR-ek a kontroller vezérlésére és állapotainak megjelölésére, a GPR-ek a felhasználói alkalmazások adatainak tárolására vannak. Bármely kihasználatlan terület 0-ként lesz kiolvasva.

Az utasításkészlet és az architektúra valamennyi szegmensre engedélyezi a műveleteket. A teljes adat memória direkt, indirekt vagy idexelt címzéssel érhető el. Arról, hogy a gyakran használt regiszterek egy utasítás ciklusban legyenek elérhetőek, a hozzáférési szegmens gondoskodik, amely egy 256 byteos memória rész, amely gyors hozzáférést biztosít az SFR és a GPR szegmenseihez.

Az adat EEPROM egy, a program és az adat memóriáktól elkülönülő, nem-illékony memória tömb, amely a programok adatainak hosszabb távú tárolására hivatott. Nincs közvetlen leképezése se a regiszter fájlra, se a program memória tárra, hanem az SFR-eken keresztül közvetve címződik meg. Az EEPROM normális műveletek közben a teljes VDD távolságon belül írható és olvasható. Erre a memória típusra jellemző a hosszú törlési/írási ciklusidő. Egy byte írás automatikusan kitörli régit, és helyére írja az új adatot. Az írási időt egy timer modul irányítja.

A mikrokontroller órajele, attól függetlenül, hogy egy külső vagy belső forrás biztosítja, négy, egymást nem átlapoló órajelre van felosztva (Q1, Q2, Q3, és Q4). A programszámláló minden Q1 alatt inkrementálódik, egy utasítás a Q4 alatt lesz elővéve a program memóriából, és töltődik az utasítás regiszterbe. Valamennyi utasítás Q1 és Q4 között lesz dekódolva és végrehajtva.

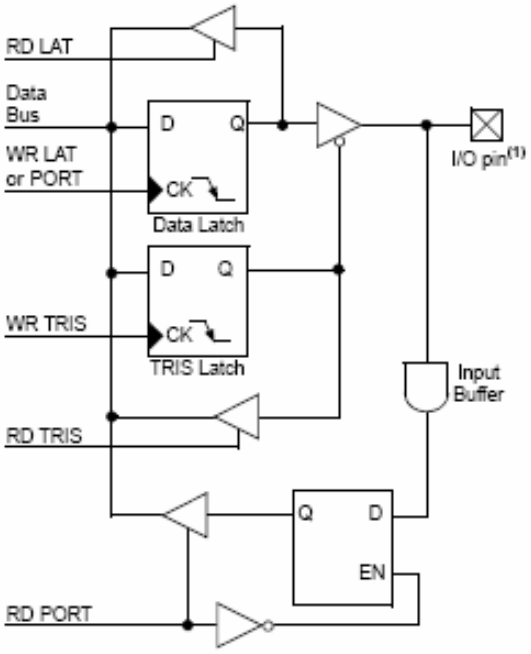
9.1.2. Az I/O portok

A mikrokontroller összesen öt bemeneti/kimeneti porttal rendelkezik. A portokhoz tartozó lábak egy része a chip valamely perifériás funkciójával lettek összevonva. Általánosságban, ha egy periféria van kapcsolva, akkor a hozzá tartozó láb nem használható általános I/O funkciókra.

Valamennyi port három regiszterrel rendelkezik a műveleteinek elvégzésére. Ezek a következők:

- TRIS regiszter (adatirány regiszter),
- PORT regiszter (az eszköz lábait olvassa le),
- LAT regiszter (kimeneti latch)

Az adat latch regiszter az I/O lábakon haladó információ olvasására, módosítására, írására való.



27. ábra Általános I/O port kapcsolási rajz

9.1.3. A mikrokontroller programozása

A mikrokontroller alapértelmezett programozási felülete assembly nyelvű, amellyel hatékony, de igen bonyolult a fejlesztés. A Microchip ezért egy mikrokontrollerekre optimalizált C nyelvű fordítót [14] szolgáltat alternatív lehetőségként, amely lényegesen megkönnyítheti és felgyorsíthatja a fejlesztést.

A fordító többé-kevésbé követi az ANSI C könyvtár szabványait, azonban eltérések tapasztalhatóak a memóriakezelés terén, mivel a Microchip mikrokontrollerei Harvard alapúak, így az adatokat az alapján is megkülönbözteti, hogy a program memóriában, vagy az adat memóriában helyezkednek el. Az algoritmus fejlesztése során ezt is figyelembe kell venni. A számábrázolás terén, valamennyi egész számtípusra létezik megvalósítás, viszont lebegőpontos számoknál csak az egyszeres pontosságú ábrázolás (float) támogatott, a bonyolultabb és nagy pontosságot igénylő részműveletek megírásánál erre is figyelni kell.

Több részművelet gyakran használt alapszámítása a sinus és cosinus értékek meghatározása. A szabványos C könyvtár **sin()** és **cos()** függvényei egy sorba fejtős algoritmus segítségével határozzák meg a radiánban megadott szögekhez tartozó értékeket. A mikrokontroller architektúrája nem kedvez az ilyen számítások elvégéséhez, ezért egy másfajta megközelítést kellett alkalmazni. A legkézenfekvőbb megoldásnak egy olyan módszer bizonyult, amely egy táblázatból keresi ki az értékeket a bemenő paraméternek megfelelően, jelentősen leegyszerűsítve az egyébként számításigényes műveletet. A táblázatot egy tömbben lehet megvalósítani, amelyet az újraírt sin() és cos() függvények használnak. A program mérete valamennyivel nagyobb lesz, viszont a számítások drasztikusan csökkennek, így az egész művelet jelentősen gyorsabb lesz.

A kétfajta számítási módot teszteltem egy, a Microchip MPLAB fejlesztőkörnyezetében írt programmal, a mérési eredmények mind az utasítás ciklusok száma, mind a futási idő tekintetében igazolták a módszer célszerűségét.

28. ábra Az eredeti és a módosított sin függvények utasítás ciklusainak száma és futási ideje

29. ábra Az eredeti és a módosított cos függvények utasítás ciklusainak száma és futási ideje

Amellett, hogy a trigonometrikus részműveleteket a mikrokontroller környezethez igazítottam, az általam írt FFT rutin és az ablakozó függvény egyszerűsítésére is szükség volt. A pillangó művelet kódján egyszerűsítettem, és az ablakozó függvényben csak egyfajta ablak típust hagytam meg, az FFT művelet számára legideálisabb Hamming ablakot. A kódban felhasznált tömböket alacsonyabb elemszámúra kellett változtatnom, hogy megelőzzem a memória túlcsordulást. Ennek következtében legfeljebb 32 elemű tömbökre lehetett elvégezni a számításokat. További egyszerűsítésre, méretcsökkentésre már nem jutott idő.

10. Összefoglalás

Elmondható, hogy a taktilis kijelző vezérléséhez, a vele való kommunikációhoz, és a rajta történő fejlesztésekhez sikerült a környezeti feltételeket biztosítani, amelyek a mikrokontrollerre írt tesztprogramokkal lemérhetővé váltak. Az USB protokoll további lehetőségeket biztosított, hogy a kijelző eszköz egy szabványos csatornán léphessen kapcsolatba a külvilággal, és többféle elektronikus eszközhöz legyen csatlakoztatható.

Annak céljából, hogy a kijelző felhasználható legyen zajos környezetben, szükség volt egy gyors, megbízható és mikrokontroller achitektúrájába illeszthető beszéddetektor algoritmusra. Fontos volt, hogy ne csak a beszéd meglétét ismerje fel, hanem képes legyen a beszéd zajtól vagy zenétől való megkülönböztetésére. Az irodalom tanulmányozása és számos megoldás megismerése után kiválasztottam, lemodelleztem és leteszteltem szimulációs környezetben a célnak legmegfelelőbb algoritmust, amely a mel-cepstrum modulációs energiát használja megkülönböztető jellemzőként. Ez a módszer teljesítette a hozzá fűzött elvárásokat, nagy mennyiségű és sokrétű tesztadatoknál állta meg a helyét. Zajos környezetben, például metrón, forgalmas csomópontokon készült felvételekből vagy az utólag zajjal kevert jelekre, a programnak sikerült kinyernie az emberi beszédet tartalmazó szakaszokat, a paraméterek megfelelő beállításával. Ezen kívül alacsony komplexitásúnak, gyorsnak, és a kijelző felhasználásának koncepciójába illeszthetőnek bizonyult.

A kijelző vezérlőegységébe történő megvalósítás első fázisában az MCME algoritmus alapműveletét, az FFT függvényt írtam meg és teszteltem PC környezetben. A függvény az elvárásoknak megfelelően működött, az eredmények helyesek voltak.

Mivel az algoritmus alapművelete már megírásra került, a további feladatok közé tartozik az algoritmus befejezése, és a Földi Balázs által megvalósított vezérlő programban való integrálás. Ezek után a valós felhasználási körülmények között történő tesztelés következik.

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani **Tihanyi Attila** konzulens és laborvezető úrnak támogatásáért, hasznos tanácsaiért, melyekkel időt és fáradságot nem kímélve, nagyban hozzájárultak a diplomatervem zökkenőmentes előrehaladásához, munkám megkönnyítéséhez és eredményessé tételéhez.

Köszönetet mondok szüleimnek, akik minden erejükkel, szeretetükkel segítettek tanulmányaim elvégzésében, és biztosították számomra az ehhez szükséges körülményeket.

Irodalomjegyzék

[1] Eszter András, Taktilis kijelző alkalmazások, http://digitus.itk.ppke.hu/~tihanyia/taktilis/EA2006_mernoki.pdf, Pázmány Péter Katolikus Egyetem Információs Technológiai Kar, 2005.

[2] Microchip, PICDEM™ FS USB Demonstration Board User's Guide, Microchip Technology Inc., 2004.

[3] The MPUSBAPI Library, http://www.garcia-cuervo.net/picmania.garcia-cuervo.net/USB_MPUSBAPI_DLL.php, PicMania by Redraven,

[4] Singh, D., Boland, F., Voice Activity Detection, Crossroads - The ACM Student Magazine

[5] Kurtosis, http://en.wikipedia.org/wiki/Kurtosis, Wikipedia, 2008.

[6] Takács György, Beszédjelek lineáris predikciója, http://digitus.itk.ppke.hu/~takacsgy/beszedea4_08.ppt, Pázmány Péter Katolikus Egyetem Információs Technológiai Kar, 2008..

[7] Takács György, PARCOR, csőmodell, http://digitus.itk.ppke.hu/~takacsgy/beszedea5_08.ppt, Pázmány Péter Katolikus Egyetem Információs Technológiai Kar, 2008..

[8] Tatarinov, J., Pollák, P., Hidden Markov Models in voice activity detection, http://noel.feld.cvut.cz/speechlab/publications/033_robust04.pdf: Czech Technical University, Faculty of Electrical Engineering, Prága

[9] Sukittanon, S., Surendran, A. C., Platt, J. C., Burges, J. C., Convolutional networks for speech detection, http://research.microsoft.com/~acsuren/ConvNetV2.pdf, University of Washington, Seattle,

[10] Kim, B. W., Choi, D. L., Lee, Y. J., Speech/Music Discrimination Using Mel-Cepstrum Modulation Energy, Text, Speech and Dialogue, 10th International Conference, TSD 2007 Pilsen ed. , Springer, 2007.

[11] Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P., Numerical Recipes in C, Second ed. , Cambridge University Press, 1992.

[12] Fellegi József, Digitális jelfeldolgozás I. - Mintavételezés, digitális szűrés. diszkrét Fourier transzformációk, Budapesti Műszaki Főiskola Kandó Kálmány Villamosmérnöki Főiskolai Kar, Budapest, 2004.

[13] Microchip, PIC18F2455/2550/4455/4550 Data Sheet, Microchip Technology Inc., 2007.

[14] Microchip, MPLAB® C18 C Compiler User's Guide, Microchip Technology Inc., 2005.