

SZENT ISTVÁN EGYETEM
Gépészmérnöki Kar
Gépipari Technológiai Intézet



TDK-dolgozat
TÁJÉKOZÓDÁSI STRATÉGIA ÉS IRÁNYÍTÓESZKÖZ FEJLESZTÉSE
LÁTÁSSÉRÜLTEK SZÁMÁRA

Készítette:

Erdélyi Viktor Ferenc
mechatronikai mérnöki szakirány, MSc. 1. évfolyam

Témavezető: Dr. Jánosi László egyetemi tanár, tanszékvezető

Gödöllő
2013

Tartalomjegyzék

1. Bevezetés	1
1.1. Célkitűzés.....	1
1.2. Témaválasztás	1
2. „Echolocation”	2
2.1. Működési elv	2
2.2. FM-sweep.....	3
2.3. CF.....	4
2.4. Idegi mechanizmusok az agyban	4
3. Ultrahangos távolságmérő rendszerek áttekintése.....	5
3.1. Ultrahangos távolságmérő berendezések napjainkban	5
3.2. Ultrahangjeladók és -vevők bemutatása.....	6
3.3. Mikrokontroller, mint a beágyazott rendszer alapja.....	8
4. Készülék tervezése	10
4.1. Előzetes megfontolások	10
4.2. Tervezési irányvonal.....	11
4.3. Alkalmas eszközök kiválasztása	12
4.4. Tervezési szakasz	12
5. A kísérleti eszköz megépítése és vizsgálata	14
5.1. A tesztelő - megjelenítő eszköz tervezése.....	14
5.2. A távolságmérő áramkör megépítése	16
5.3. A működtető programok logikai felépítése.....	17
5.4. A kísérleti eszköz vizsgálata	21
5.5. Eredmények.....	25
6. Továbbfejlesztési lehetőségek.....	27
7. Összefoglalás.....	28
8. Summary.....	29
9. Irodalomjegyzék	30
10. Köszönetnyilvánítás	31
11. Mellékletek	32
1. melléklet.....	32
2. melléklet.....	34
3. melléklet.....	36
4. melléklet.....	43
5. melléklet.....	48

1. Bevezetés

1.1. Célkitűzés

Olyan módszert és eszközt kifejleszteni, mely hanghullámok visszaverődésén alapul, és virtuális kép előállítását teszi lehetővé.

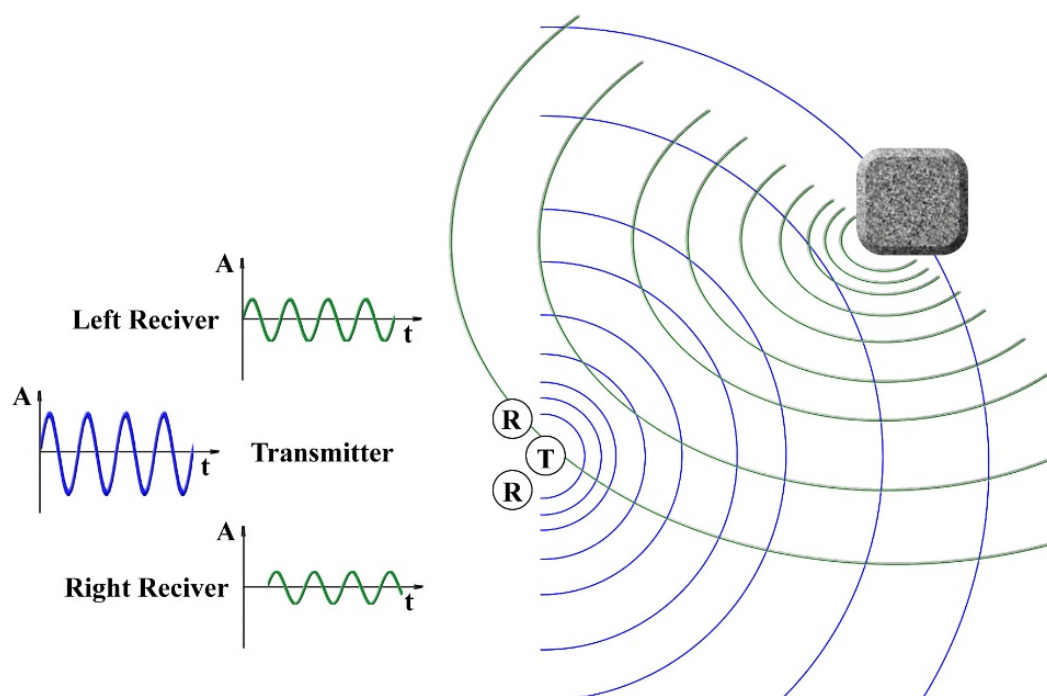
1.2. Témaválasztás

Az állatvilágban az emlősök egy része ultrahangos módszerrel tájékozódik, illetve léteznek olyan ősi „éjjellátó” technikák, melyeket jóval az elektronika megjelenése előtt is használtak bizonyos népek és azok harcosai. Ezek a technikák sok esetben a visszaverődött hangra épülnek. A denevér köztudott módon ultrahangot használ a zsákmányállatok felderítésére, illetve a tájékozódásra. L. Dijkgraaf (1949) és Matsuda et al. (2013). Ezt a jelenséget „echolocation”-nak nevezik, mely szabad fordításban annyit tesz, visszhangos helyzetmeghatározás. A denevérek ilyen módon való tájékozódását Donald Griffin és Robert Galambos állapította meg 1938-ban. L. Griffin (1958). Kézenfekvőnek tűnt tehát, ha az állatok (l. Jones. 2005), sőt az emberek is tudták használni ezt a technikát a sötétben való tájékozódásra, használjuk ugyanezt a módszert –az elektronika segítségével némiképp kiterjesztve–, virtuális képalkotásra. A készüléket kiegészítve egy hangszintetizátorral egy vakok és gyengén látók tájékozódását segítő eszközt kaphatunk. L. Mihajlik et al. (2004).

2. „Echolocation”

2.1. Működési elv

Ha a műszaki gyakorlathoz hasonlítanánk az echolocation technikáját, a legkézenfekvőbb analógia az aktív szonár lehetne. A SONAR angol mozaikszó, jelentése: SOund Navigation And Ranging. Működése a következő: Jones (2005) szerint az állat vagy –jelen esetben– a távolságmérő berendezés hangimpulzust bocsát ki, mely visszaverődik a felületekről, így visszajut a kibocsátóhoz, mely a jel intenzitásának csökkenéséből, illetve a kibocsátás és visszaérkezés közt eltelt időből számítja a távolságot, valamint a két fülbe vagy érzékelőbe jutó hangerősségből és időbeni eltérésből a vízszintes irányszöget. Az állatoktól elcsúszott tájékozódási módszert először a Titanic katasztrófája (1912) után egy angol meteorológus, Lewis Richardson szabadalmaztatta (1913). Hill (1962)



1. ábra: Az "Echolocation" működése

Suga et al. (1975) szerint a kibocsátott hangot különféle szempontokból vizsgálhatjuk.

- **Frekvencia:**

- o **Frekvenciamoduláció (FM):** A frekvencia modulált keresőhang sweep, azaz söprő módon működik, vagyis FM-pingelésnél széles tartományban a magasabb frekvenciától lefelé söpri végig a frekvenciatartományt.
- o **Állandó frekvencia (CF):** A keresőhang frekvenciája állandó. A CF-keresésnél a frekvenciasáv igen keskeny, így a keresőjel frekvenciája konstansnak tekinthető.

A denevérek jellemzően a két módszert vegyesen alkalmazzák. L. Stuart et al. (2000).

- **Intenzitás:** Az állatvilágban a keresőhang intenzitása jellemzően 60-140 decibel között mozog. A keresőhang amplitúdóját kis távolságok esetén csökkentik, mivel a visszaverődött hang hangnyomása károsíthatná a denevér fülét.
- **Impulzus időtartama:** A denevérek a keresőhang intervallumát 0,2-100 ms-ig terjedő skálán adják le, attól függően, hogy mekkora távolságot mérnek vele, illetve vadászat közben sűrítik a pingelést, így jobb felbontású képet képesek kapni a környezetükről. Rövidebb impulzushosszt kisebb távolságok esetén alkalmaznak.

2.2. FM-sweep

A széles frekvenciatartományon végigsöprő keresőhang alkalmazásának legnagyobb előnye, hogy igen nagy precizitással lehet vele távolságot mérni. J. A. Simmons (1975) kísérletekkel bizonyította, hogy a denevérek, ha ezt a módszert alkalmazták, képesek voltak fél milliméteres távolságkülönbséget is érzékelni két felület közt. Ez a képességük abból fakad, hogy a leadott hang egy frekvenciatartományon söprő végig, ami elősegíti a precízebb távolságmérést, mivel jobb felbontásban lehet érzékelni a leadott és visszaverődött hang közti időbeni eltérést (ezáltal javítva a keresztkorrelációt a kettő közt). A pontosságot tovább javíthatja, ha a keresőhang harmonikusokat is tartalmaz. A hátránya az FM-metódusnak Suga et al. (1975), hogy csak kis távolságokban működik hatékonyan.

2.3. CF

A CF legnagyobb előnyének számít a nagyobb hatótávolság, illetve a Doppler effektusból származó sebességérzékelési képesség is. Mivel a denevér mozgó zsákmányállatokat keres, a Doppler effektusból származó vörös-, illetve kékeltolódás segítségével képes érzékelni annak

$$f = \left(\frac{c + v_r}{c + v_s} \right) f_0 \text{ mozgási irányát. Suga et al. (1975)}$$

Ahol:

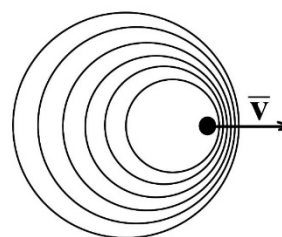
f : a visszaverődött hang frekvenciája

c : a hang sebessége

v_r : a denevér sebessége a közeghez képest

v_s : a visszaverő sebessége a közeghez képest

f_0 : a kibocsátott hang frekvenciája



2. ábra: A Doppler eltolódás

2.4. Idegi mechanizmusok az agyban

Mivel a denevérek a hallásuk segítségével tájékozódnak, így a fülük és az agyuk hallásért felelős része is speciálisan módosult. Az ember alapvetően látás alapján tájékozódik. A tapasztalat azt bizonyítja, hogy a megvakult emberek hallásért felelős agyszekciója, a halántéklebeny jelentősen jobb vérellátottságú a látó emberekénél, sőt idővel a látásért felelős nyakszirti lebeny nem elsorvad, hanem segít a hallott információk feldolgozásában. Burton (2003)

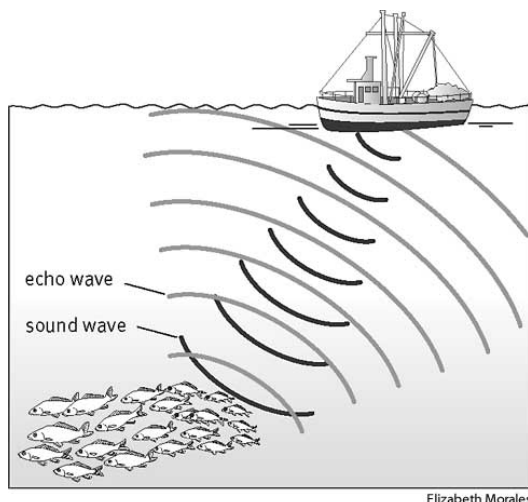
3. Ultrahangos távolságmérő rendszerek áttekintése

3.1. Ultrahangos távolságmérő berendezések napjainkban

A műszaki gyakorlatban előszeretettel alkalmazunk ultrahangos távolságmérést, mivel olcsó, és sok esetben elégséges pontossággal működik. Leggyakrabban az autóiparban és a robotikában használják ezt a típusú távolságmérést, de a halászatban, a harcászatban is előszeretettel alkalmazzák víz alatti tárgyak, élőlények felderítésére (SONAR). Mindennapi életünk során sokszor



3. ábra: Tolatóradar (Forrás: Nissan)



Elizabeth Morales

4. ábra: Sonar működése (Forrás: YourDictionary)

találkozhatunk a tolatóradarokkal, nagyobb gépjárművek esetén parkolóradarokkal, melyek szintén ultrahangos távolságmérés alapú megoldások. Sok esetben bizonyos biztonságtechnikai berendezések, például riasztók is alkalmazzák ezt a típusú mérést nyílászárók nyitásának érzékelésére, üvegfelületek épségének vizsgálatára, és szintén az autóiparból kiragadott példa a gépjárművekben a mozgásérzékelő rendszer, mely az utastérben történő mozgást figyeli. Ilyen megoldással lehet például egy lakásban a mozgást olyan módon regisztrálni, hogy nem csak a mozgás meglétét, de a mozgást végző magasságát, valamint helyzetét is figyelemmel lehessen kísérni. L. Siddiquie (2009). Ultrahangos távolságmérést alkalmaznak továbbá a gyógyászatban is, tumorok, elváltozások felderítésére, de magzatvizsgálatra is. L. Arany et al. 2008. Nem csak ilyen területeken, de a gyártásban is használják az ultrahangot repedésvizsgálatra.

3.2. Ultrahangjeladók és -vevők bemutatása

Az ultrahangjeladók és -érzékelők igen hasonló felépítésűek. Ennek köszönhetően, sok esetben nem két külön eszközt alkalmaznak távolságmérésre, hanem ugyanazt a készüléket előbb hangimpulzus kibocsátására, majd a visszaverődött hang vételére használják (pl. tolatóradarok). Ezzel egy érzékelőt megspórolnak, a pontosság és a legkisebb mérhető távolság rovására. Ha viszonylag kis távolságokat mérünk, szükséges két eszközt: egy adót és egy vevőt használni. Egyik ezek közül csak hangforrásként üzemel, míg a másik csak vevőként működik. A hang sebessége ismeretében az adás és vétel között eltelt időből tudjuk számítani a hangot visszaverő objektum távolságát.

$$d = \frac{c}{2} \cdot t$$

ahol:

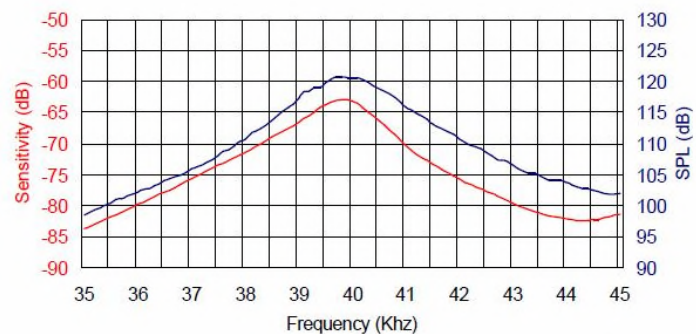
- d: a mért távolság [m]
- c: a közegben jellemző hangsebesség (levegő esetén $\approx 340 \frac{m}{s}$)
- t: a hangimpulzus adása és vétele közt eltelt idő [s]

A kísérleti eszközben két darab intelligens szenzor került felhasználásra, melyek a trigger jel hatására 8 darab négyszögjelet küldenek 40 kHz-s frekvencián a transmitternek, minek hatására az 40 kHz-es 200 μ s hosszúságú ultrahang jelet bocsát ki megközelítőleg 120 dB hangnyomással. Ez a jel (Ultrasonic burst) az útjában lévő objektumról visszaverődik, így a vevőegységbe jut, ahol elektromos jellé alakul vissza. A küldés és visszaérkezés időbeni eltérését egy impulzushossz reprezentálja a készülék Echo kimenetén. A prototípus eszköz az Echo időtartamát 100 μ s pontossággal méri, amely elégséges, hisz ez távolságban 17 mm-t jelent. Mind a transmitter, mind a receiver piezoelektromos készülék. Az alkatrészek karakterisztikái a 6. ábrán találhatóak.



5. ábra: Felhasznált intelligens távolságmérő szenzor

Látható, hogy mindkét készülék egy bizonyos (éppen 40 kHz környéki) tartományban a leghatékonyabb, illetve a legérzékenyebb. Ez egyéb hangkeltőknél probléma, de jelen esetben igen hasznos, hisz így jó eséllyel más hangforrás nem zavarja a mérést. Ezen készülékkel elégséges pontossággal és megbízhatósággal 17



6. ábra: A felhasznált szenzor transmitterének és recieverének átviteli görbéi (Forrás: 400ST/R120 adatlap)

– 4000 mm-ig lehet mérni, ennél nagyobb távolságoknál nagyobb adóteljesítmény szükséges, és figyelembe kell venni, hogy a mért távolság növekedésével csökken a felbontás is, mivel az eszköz 15°-os kúpszögben mér.

A tesztorozat eredményeként megállapítottam, hogy az ultrahangos szenzor csak kisebb távolságon belül, egyméteres tartományban képes viszonylag stabil és zajmentes adatszolgáltatásra. Ennek megfelelően módosítottam a mikrokontroller programján, mely így csak az egy méteren belül érkezett jeleket veszi figyelembe, viszont azt már 0,5 cm pontossággal. A további pontosságjavítás és hatótáv-kiterjesztés érdekében szükségessé válik egy szofisztikáltabb szenzor beszerzése.

3.3. Mikrokontroller, mint a beágyazott rendszer alapja

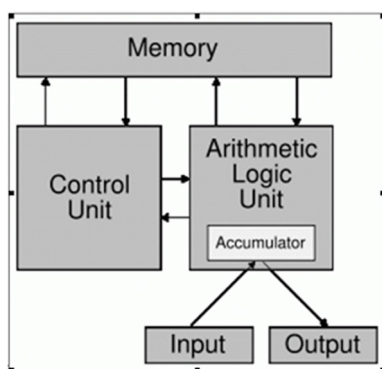
A mikrokontroller egyetlen lapkára integrált, általában vezérlési feladatokra optimalizált számítógép. L. Augarten et al. (1983). A szilíciumlapkán megtalálható a mikrokontroller processzora, a perifériák és a mikrovezérlő memóriája is. Régebbi típusoknál előfordul, hogy a memóriát nem integrálták a tokba, de a modern eszközökben ez szinte mindig megtalálható. A mikrokontrollerek beépített perifériái igen sokrétűek lehetnek, kezdve a hagyományos TTL vagy CMOS logikai szintű ki- és bemenetektől, az analóg-digitális, digitális-analóg konvertereken (ADC, DAC) és komparátorokon keresztül, egészen az analóg LCD kimenetekig és kommunikációs portokig (I2C, RS232, CAN, USB, Ethernet stb.). Allen et al. (2010)

Általában a mikrokontrollereket eltérő perifériákkal szerelik, így fontos a vásárlás előtt tájékozódni a kiszemelt mikrokontroller perifériáiról, hogy megfelelnek-e az adott irányítási feladat megoldásához (ezt leggyakrabban az adatlapok segítségével tehetjük meg). Fontos tulajdonsága még a mikrokontrollereknek, hogy hány bites felbontással képesek dolgozni, az ALU-juk tudja-e kezelni a lebegőpontos számításokat, valamint mekkora utasításkészlet áll rendelkezésre az adott mikrovezérlőn. Allen et al. (2002)

Harvard-architektúra, a mikrokontrollerek szokásos architektúrája

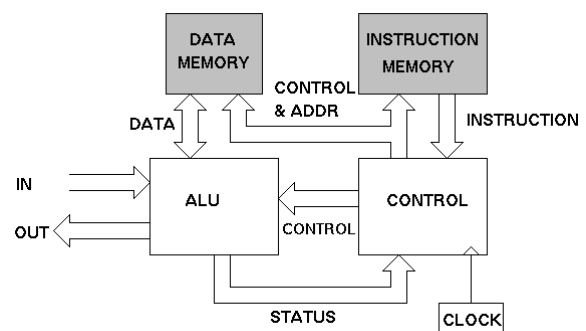
A modern számítógépek és mikrovezérlők architektúrája alapvetően különbözik egymástól, hisz míg az asztali számítógépünk Neumann-architektúrát használ, addig a mikrokontrollerek a Harvard-architektúrán alapszanak. A kettő közti lényeges különbséget először is a memóriánál találjuk meg, hisz míg a Neumann-féle megoldásnál nincs külön adat- és programmemória, addig ezt a Harvard-architektúrájú gépeknél külön találjuk.

A másik feltűnő különbség, hogy míg a Neumann-architektúrájú szerkezetekben egyetlen busz található (melyet ütemezetten vagy adattranszferre, vagy utasításátvitelre használhatunk), addig a Harvard-architektúrájú készülékekben az utasítás- és adatbuszok elkülönülnek, így egyidejűleg, azaz párhuzamosan folyhat a kommunikáció. Ez azt jelenti, hogy, bár a mikrokontrollerek felépítése bonyolultabb, mint egy általános célú processzoré, mégis gyorsabb műveletvégzést tesz lehetővé - köszönhetően a párhuzamos adat- és utasításkommunikációnak. Allen et al. (2002)



7. ábra: Neumann-architektúra

(Forrás: Embedded World)



8. ábra: Harvard-architektúra

(Forrás: Wikipedia)

4. Készülék tervezése

4.1. Előzetes megfontolások

A készülék tervezésének előkészítési fázisában figyelembe vettem az emberi agy adottságait, azt, hogy különbözik a speciálisan ilyen tájékozódásra szakosodott fajokétól. Tanulmányoztam az ősi, jól működő „éjjellátó” módszert, mely az ember által keltett hang visszaverődésén alapszik.

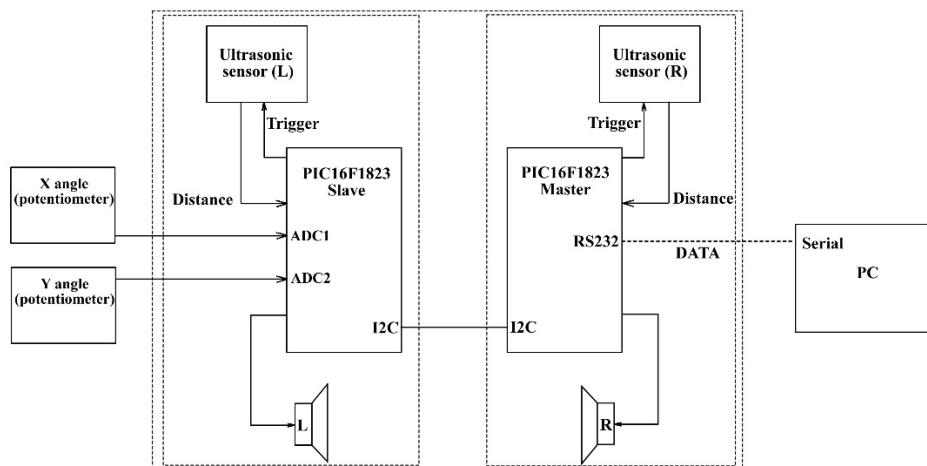
A koncepció felállításánál figyelembevettem szempontok:

- hordozhatóság: Kis méret kis súly.
- akkumulátor-élettartam: Kisméretű áramforrás mellett hosszú működési idő.
- térhallás kihasználása: Az ember alapvető adottságainak felhasználása.

Ezen szempontok alapján egy állandó hullámhosszú távolságmérő eljárást választottam (CF).

4.2. Tervezési irányvonal

Két, egymástól 200 mm-re elhelyezkedő intelligens, ultrahangos távolságjeladó szenzort egy-egy mikrokontrollerrel a távolságmérésre utasítok (trigger), minek hatására a távolságmérő szenzor kimenetén egy –a távolsággal arányos hosszúságú– impulzus jelenik meg. Ennek hossza és a szenzor tulajdonságai ismeretében meghatározható a hangforrás és az objektum közti távolság.



9. ábra: A rendszer felépítésének vázlata

A két fülbe jutó hang erőssége alapján határozhatja meg az agy az előtte lévő akadály térbeni helyzetét, valamint annak távolságát, gyakorlatilag úgy, mintha a hangot maga az akadály adná ki.

A két mikrokontroller egyszerre adja ki a mérésre való utasítást, így a fülbe jutó hangnak nem csak az amplitúdója, de a frekvenciaeltolódása is értékelhető lesz.

Az intelligens távolságmérő szenzorok kimenetén megjelenő távolságértékeket mikrokontrollerrel feldolgozva hallható hangot ad a fülhallgatóba, mely alapján akár az emberi agy is felépítheti a háromdimenziós tér virtuális „képét”.

4.3. Alkalmas eszközök kiválasztása

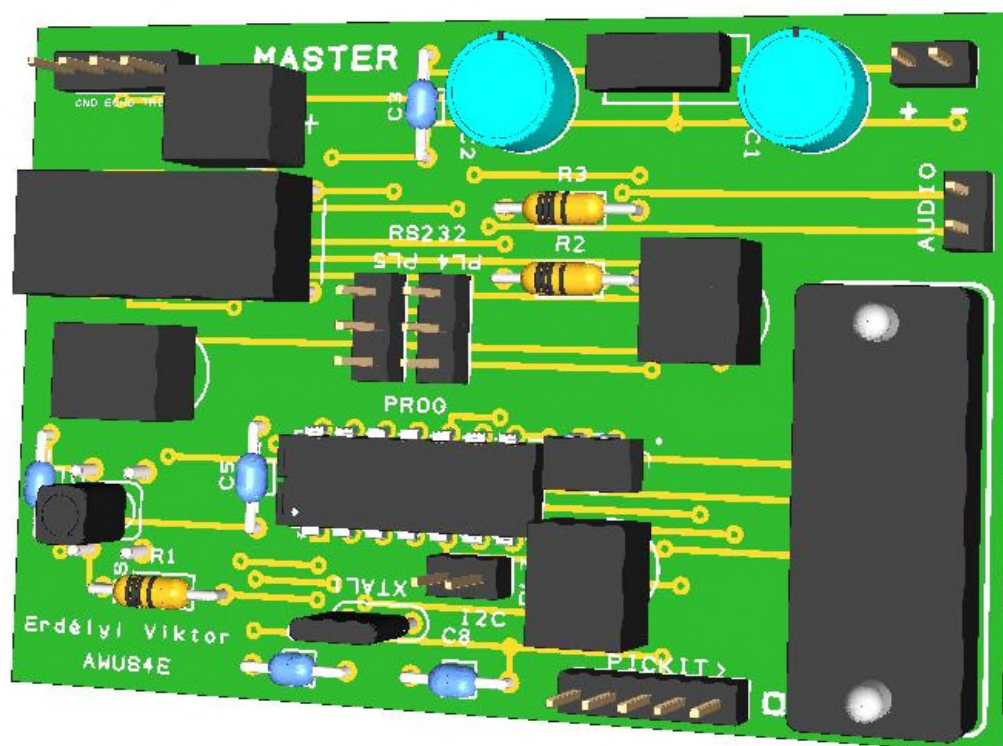
Távolságmérésre a kereskedelemben kapható intelligens távolságmérő szenzorokat választottam, melyek a mérést elvégezve a kimenetükön a távolsággal arányos hosszúságú jelet adnak. Ennek feldolgozására két darab PIC16F1823 típusú mikrokontrollert választottam, mely rendelkezik PWM modullal (a hangerő beállításához), I2C, SPI, kommunikációval (az egymással történő kommunikáció érdekében), USART porttal, mellyel képes számítógéppel kommunikálni esetleges mérési adatgyűjtés céljából, illetve elég gyors a μ s-nagyságrendű jelek feldolgozásához (max. 8 MIPS @ 32 MHz). A hallható tartományba eső hangot egy kereskedelemben kapható sztereo fejhallgató szolgáltatja, melyet egy normál tranzisztoros végfokkal hajtok meg. A hangot szintén a mikrokontroller generálja a beérkezett jellel azonos időben, így a két hang frekvenciája bizonyos eltolásba kerül egymással.

4.4. Tervezési szakasz

Az áramkörök megtervezése:

Az áramkört a DesignSpark nevű programmal terveztem. A Prototípust furatszerelt (DIP) alkatrészekből készülök összeállítani, ami nagyobb helyet foglal, viszont könnyebb a szerelhetősége. Ha a készülék beváltja a hozzá fűzött reményeimet, elkészítem ugyanezen kapcsolás SMD kivitelét is, így a készülék várhatóan kisebb helyen elfér majd. Ez fontos szempont a hordozhatóságot tekintve.

A mérések láthatóvá tételét elősegítendő, az áramkör kapott egy kommunikációs csatornát is, mely segítségével az általa mért adatokat egy számítógépen MATLAB segítségével meg lehet jeleníteni. Ez a kommunikáció egy RS232 soros adatátviteli megoldás segítségével valósul meg. A szükséges programokat C, illetve C# nyelveken írtam meg. (Ezek részletesen a mellékletekben találhatók.)

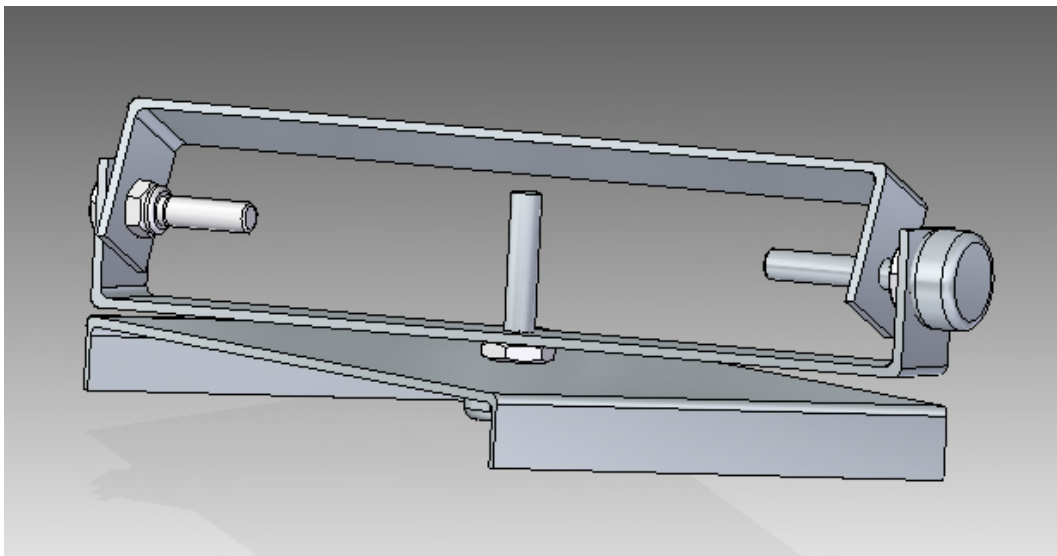


10. ábra: A prototípus-nyák 3D-s modellje

5. A kísérleti eszköz megépítése és vizsgálata

5.1. A tesztelő - megjelenítő eszköz tervezése

Ahhoz, hogy a rendszer működését a látó emberek számára is láthatóvá tegyük, illetve az értékelést elősegítsük, egy megjelenítő eszköz tervezése is szükségessé vált. Ennek segítségével a teret virtuálisan le lehet képezni. A rendszer továbbfejlesztéséhez ez hasznos információkkal szolgál.



11. ábra: Két szabadságfokú szkennertartó állvány szögmérővel

A háromirányú információkat megjelenítő eszköz működése:

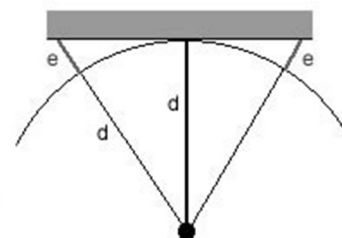
- függőleges és vízszintes irányszög mérése
- kétoldali távolságadat mérése
- adatok küldése PC-re (soros porton keresztül)
- adatok feldolgozása, megjelenítése MATLAB szoftver segítségével.

Az adatgyűjtést nem külön eszköz, hanem maga a fejlesztett berendezés végzi. A vízszintes és függőleges irányszögeket egy-egy potenciométer segítségével méri a slave mikrokontroller. Ezt a két szögállást a potenciométerekből kialakított feszültségosztó kimenetén megjelenő feszültség mérésével végzi a rendszer. A feszültségmérés 10 bites analóg-digitális konverter segítségével valósul meg, melynek a 8 szignifikáns bitje egy-egy változóban kerül tárolásra.

A slave mikrokontroller ezen kívül méri és tárolja is a távolságot, mely szintén egy 8 bites változóban tárolódik. A master mikrokontroller kérésére a slave vezérlő I²C segítségével adatot szolgáltat, melyet a master mikrokontroller a benne tárolt egy távolságadattal együtt a számítógép felé továbbít. A számítógép fogadja a 4 bájt adatot, illetve tárolja is azokat. Így az eszközt kézzel mozgatva (szkennelve) az eltárolt adathalmazból MATLAB segítségével képet készítünk, még hozzá úgy, hogy a rögzített adatokból egy háromdimenziós felületet hozunk létre.

Gömbi torzulás kompenzálása szoftveres úton

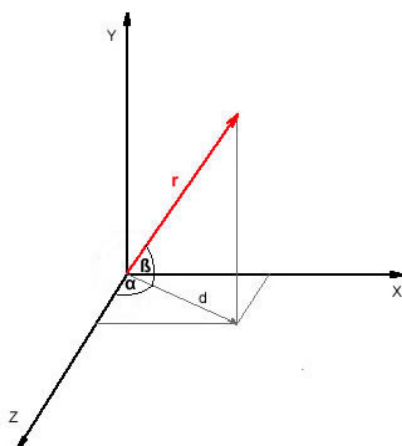
A szkennertartó állvány adottságai miatt a szkennelésnél szükséges figyelembe venni a gömbi torzulás jelenségét, mivel a mérőegységet tengelyek körül forgatjuk, így azonos távolságra lévő pontokról (síkfelületről) változó távolságadatokat kapnánk. Ha korrekció nélkül dolgoznánk fel a beérkezett adatokat egy tökéletes síkfelület, egy gömbsüveg képét adná vissza. Mivel a szkennerek gömbkoordináta-rendszerben működik, a torzulás elkerülése érdekében koordináta-



12. ábra: Gömbi torzulás oka

transzformációt kell végrehajtani.

Az így kapott adatok már elhelyezhetőek Descartes-féle koordinátarendszerben.



13. ábra: Koordináta-transzformáció

$$d = \cos\beta \cdot r$$

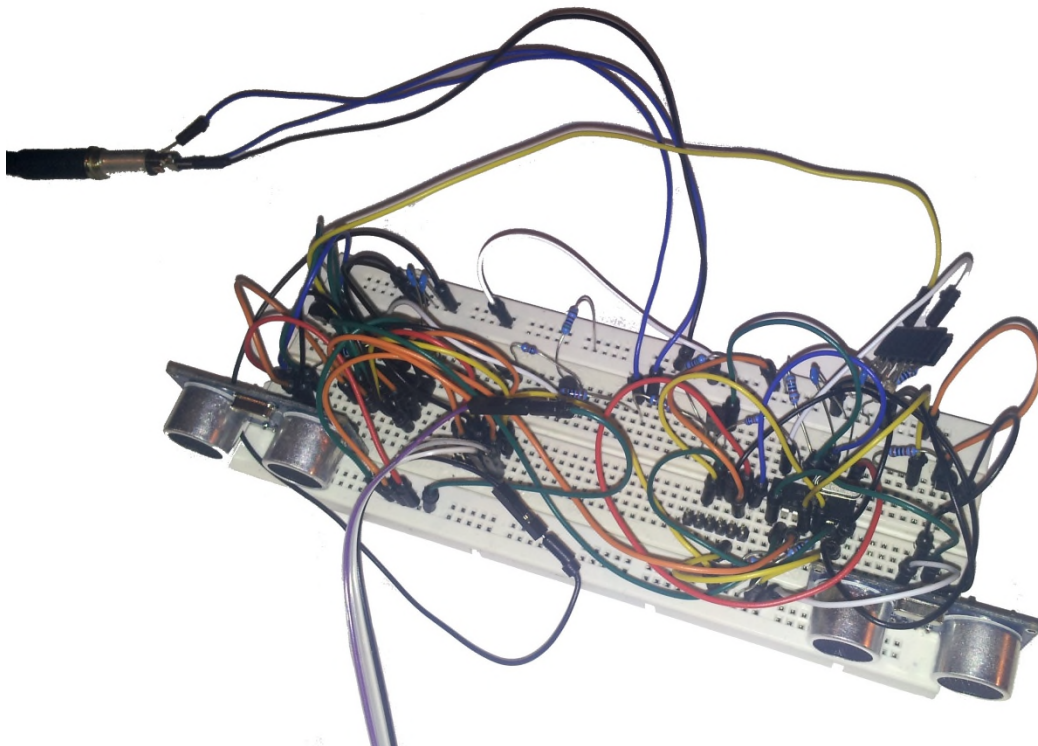
$$x = \sin\alpha \cdot d$$

$$y = \sin\beta \cdot r$$

$$z = \cos\alpha \cdot d$$

5.2. A távolságmérő áramkör megépítése

A prototípus készüléket forrasztásmentes próbapanelen állítottam össze furatszerelt alkatrészekből. Az építés során az áramköri terveket néhol módosítottam. A tesztkészülék két nagyobb egységre bontható. Ezek a jobb és bal oldali szenzorok jeleinek feldolgozását, illetve a hanggá alakítást végzik.



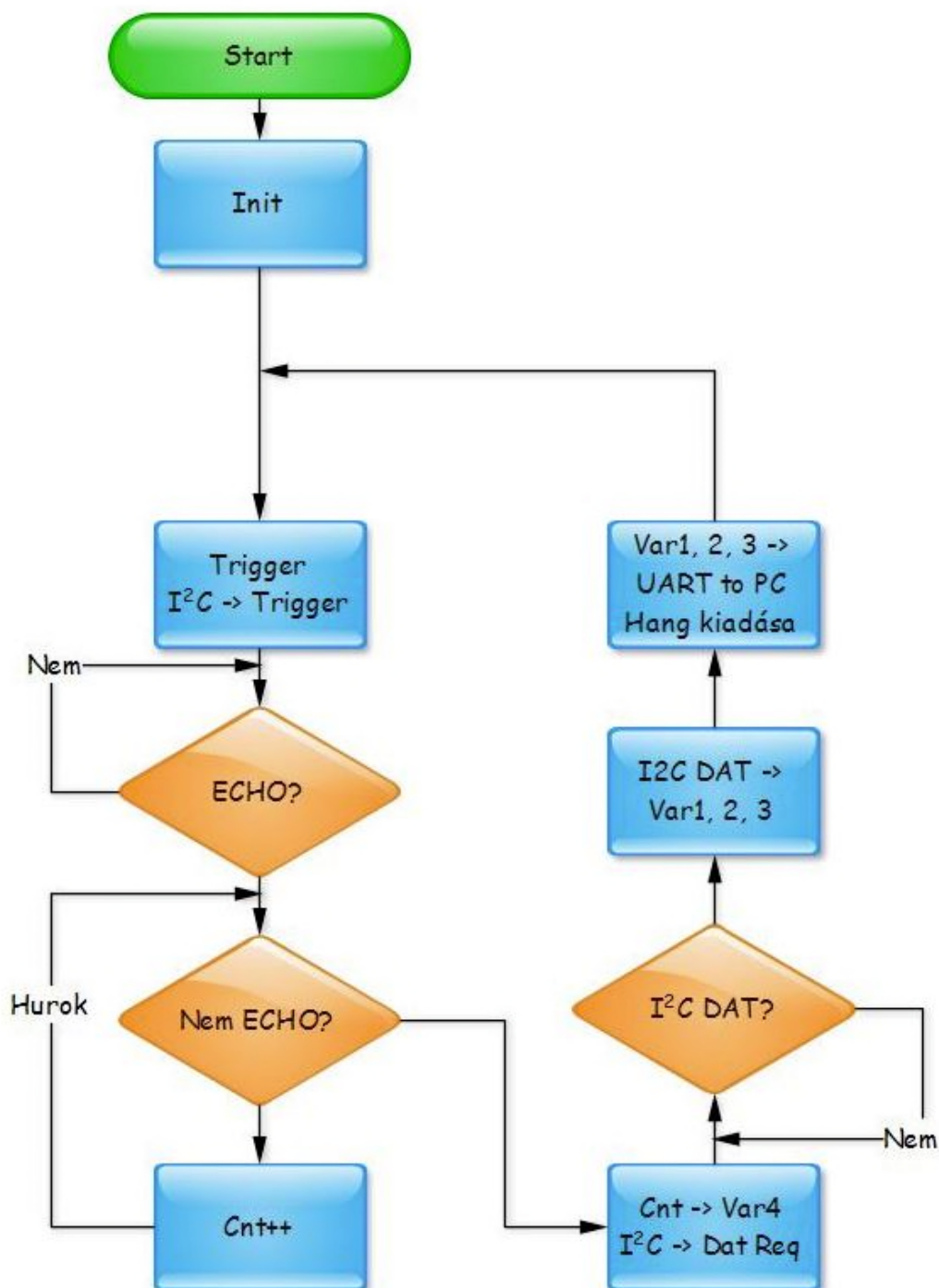
14. ábra: Prototípus

A megépített eszköz lehetőséget nyújt arra, hogy az elképzelést (miszerint az ember is képes lehet hangok alapján tájékozódásra) vizsgáljuk.

5.3. A működtető programok logikai felépítése

A vezérlőprogramok az alábbi ábrán látható módon működnek.

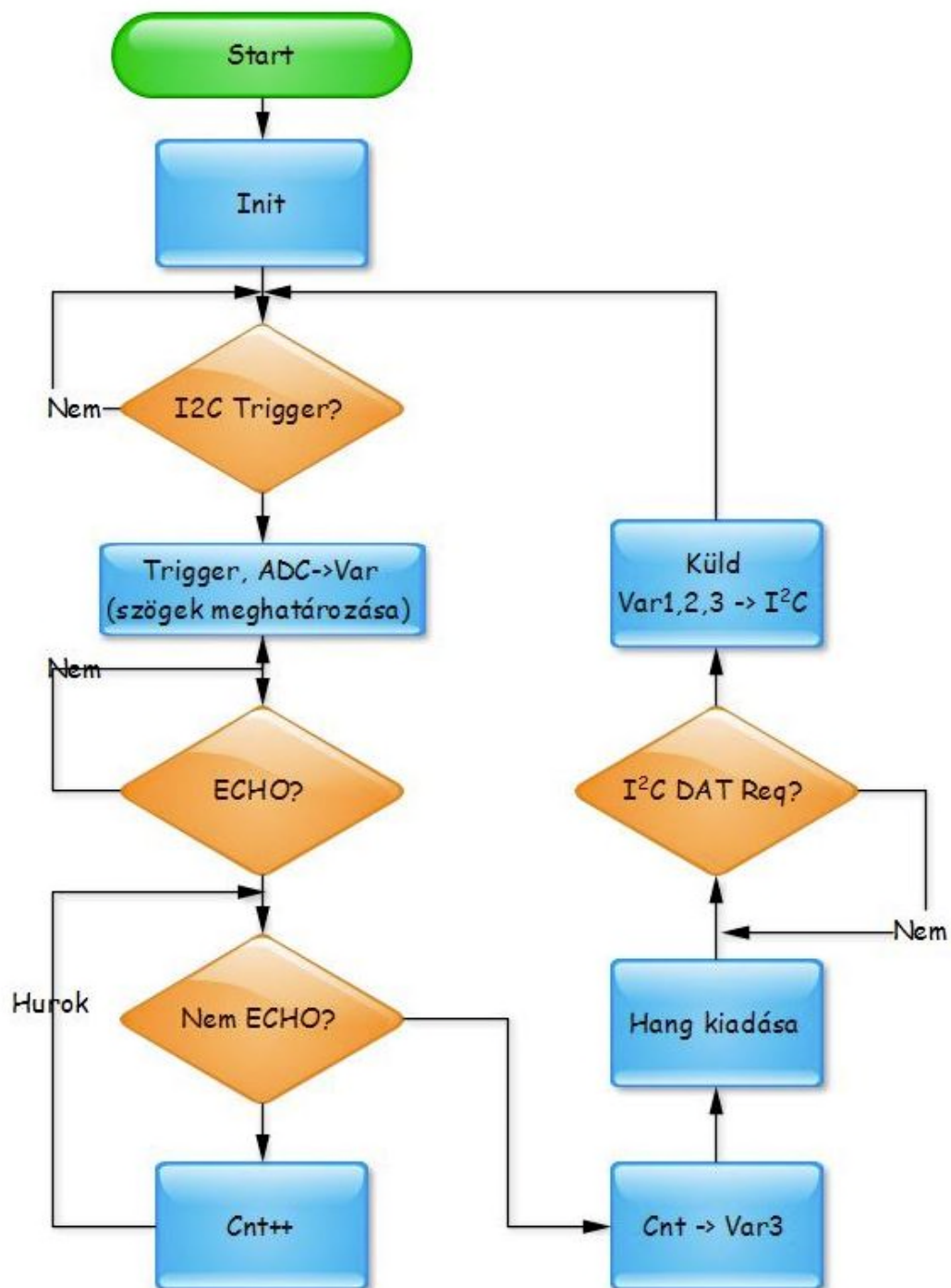
Program Master



15. ábra: Master vezérlő programjának logikai vázlata

A program elindulása után, egy hardver inicializáló ciklus fut le, mely beállítja a mikrokontroller portjait, konfigurálja a GPIO-kat, engedélyezi a megszakításokat illetve a megfelelő értékekkel feltölti a mikrokontroller option regiszterét. Ezután a mikrokontroller végtelen ciklusba kapcsol, így újra és újra elvégzi a szükséges feladatokat. A hurok első lépése, hogy 10 μ s hosszúságú trigger jelet ad az intelligens szenzor bemenetére, ezután I²C-n keresztül ugyanerre utasítja a slave mikrokontrollert is. A trigger jel elküldése után azonnal az echo jel megérkezését kezdi figyelni a mikrokontroller, egy alhurkot használva. Amint az echo jel megjelenik, a program egy számlálóciklusba ugrik, ahol 30 μ s –onként növeli a cnt számláló értékét, addig, amíg az echo jel fennáll. Ha az echo jel megszűnik a távolsággal arányos cnt értéként egy másik változóba (Var4) írja a program. Ezzel egyidőben a slave mikrokontroller felé I²C porton egy adatkérő üzenetet küld. Amint a szolga mikrokontroller adatot közöl, az adatot további három változóba írja, majd az ezekkel együtt 4 változót soros porton a számítógép felé továbbítja. Az információtovábbítás után a hangszintetizáló szubrutin fut le, melynek hatására megjelenik a távolsággal arányos erősségű jel a fejhallgatóban.

Program Slave



16. ábra: Slave vezérlő programjának logikai vázlata

A program elindulása után, egy hardver inicializáló ciklus fut le, mely beállítja a mikrokontroller portjait, konfigurálja a GPIO-kat, engedélyezi a megszakításokat illetve a megfelelő értékekkel feltölti a mikrokontroller option regiszterét. Ezután a mikrokontroller végtelen ciklusba kapcsol, így újra és újra elvégzi a szükséges feladatokat. A hurok első lépése az I²C-n érkező trigger utasítás figyelése. Ha ez megjelenik, 10 μ s hosszúságú trigger jelet ad az intelligens szenzor bemenetére, illetve megméri a két szögadó potenciométer kimenetén lévő, a szögállással arányos feszültséget. Ezt a két szögadatot egy-egy változóban tárolja. Ezek után az echo jel megérkezését kezdi figyelni a mikrokontroller, egy alhurkot használva. Amint az echo jel megjelenik, a program egy számlálóciklusba ugrik, ahol 30 μ s –onként növeli a cnt számláló értékét, addig, amíg az echo jel fennáll. Ha az echo jel megszűnik a távolsággal arányos cnt értéként egy másik változóba (Var3) írja a program. Ezután a hangszintetizáló szubrutin fut le, melynek hatására megjelenik a távolsággal arányos erősségű jel a fejhallgatóban. A következő lépésben a mikrokontroller figyeli, hogy érkezett-e adatkérés a master oldalról. Ha igen, a tárolt három változót (szögadatok, távolság) I²C-n elküldi a mester mikrokontrollernek.

5.4. A kísérleti eszköz vizsgálata

A kísérleti eszközt többféleképp vizsgáltam.

1. Hallás alapján tájékozódás

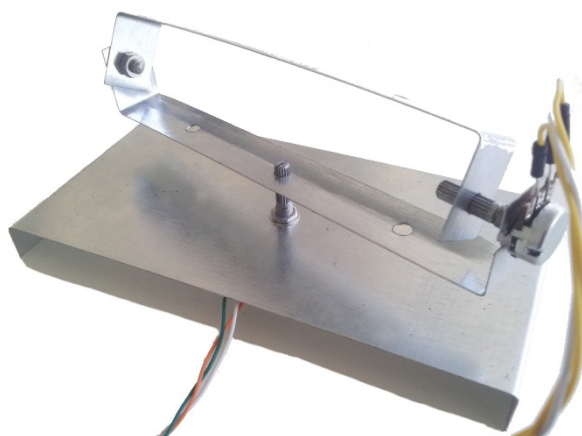
A készüléket egy fejhallgatóhoz erősítve először nyitott, majd csukott szemmel teszteltem. Idővel a látás nélkülözésével is kielégítő képet kaptam a körülöttem – különösen a közelemben– lévő tárgyak helyzetéről, illetve távolságáról. Ebből a tényből arra következtettem, hogy hosszabb tanulási procedúra nélkül is lehet az eszközt tájékozódásra használni.



17. ábra: Tájékozódás az eszköz segítségével

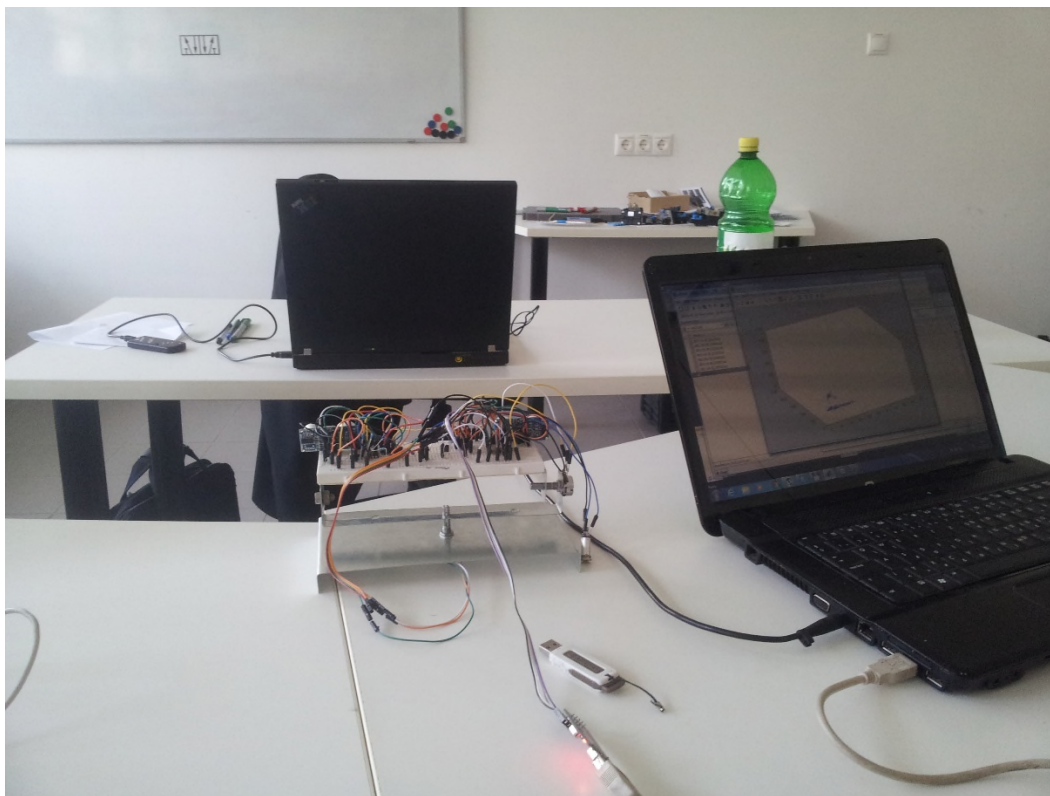
2. Tárgyak szkennelése, megjelenítése PC-n

Az eszközre kiegészítőleg felszereltem két potenciométert, melyekkel a vízszintes és függőleges irányszögeket méri a rendszer. Ezeket a szögeket, illetve a mért távolsági adatokat soros kommunikáció segítségével egy számítógépre küldi a készülék, ahol az adatok feldolgozásra kerülnek.



18. ábra: Két szabadságfokú mérőállvány szögadókkal

A feldolgozás eredményeként a MATLAB program egy képet jelenít meg, ahol az X és Y tengelyeken a koordinátatranszformáció következtében [mm] dimenziójú adatok jelennek meg, a Z tengely a mérőberendezés és az előtte lévő tárgy között lévő korrigált távolságadatokat reprezentálja. Ezáltal egy háromdimenziós ponthalmaz jelenik meg, így láthatóvá téve az elképzelés működését.



19. ábra: Kísérlet: Sík és hengeres felületek szkennelése

A további kutatások eredményeként várhatóan kifejlesztésre kerül egy OpenGL alapú környezet, mely képes lesz valós időben az adatok megjelenítésére, illetve a ponthalmazra való felületillesztésre is.

3. Teljesítményfelvétel

Mivel a rendszer tervezésénél kiemelkedően fontos szempont volt a hordozhatóság, vizsgálnunk kell az eszköz teljesítményfelvételét, hiszen ez nagyban befolyásolja az akkumulátoros üzem időtartamát. Távolságmérés közben végeztem az adatgyűjtést - fejhallgató és szögmérőrendszer csatlakoztatása mellett. A méréseket egy Maxwell MX-25 304 típusú multiméterrel hajtottam végre, a fejhallgató típusa: Panasonic RP-HT265 volt. Az eredmények az 1. táblázatban láthatóak.

1. táblázat: Teljesítményfelvétel távolság függvényében

#	Mért távolság [mm]	Áramfelvétel [mA]	Teljesítményfelvétel [mW]	Várható üzemidő 2000 mAh esetén [h]
1	50	27,5	137,5	72,73
2	100	27,6	138	72,46
3	150	27,5	137,5	72,73
4	200	27,5	137,5	72,73
5	250	27,4	137	72,99
6	300	27,5	137,5	72,73
7	350	26,4	132	75,76
8	400	27,3	136,5	73,26
9	450	27,4	137	72,99
10	500	27,4	137	72,99
11	550	27,3	136,5	73,26
12	600	27,3	136,5	73,26
13	650	27,3	136,5	73,26
14	700	27,2	136	73,53
15	750	27,1	135,5	73,80
16	800	27,2	136	73,53
17	850	27,1	135,5	73,80
18	900	27,1	135,5	73,80
19	950	27	135	74,07
20	1000	27	135	74,07
Átlag	525	27,255	136,275	73,39

A teljesítményfelvétel csökkenése a távolság függvényében azért következik be, mert a távolság növelésével a hangerőt csökkenti a rendszer, minek következtében a fejhallgató által felemésztett teljesítmény csökken.

4. A távolságmérés pontossága

A virtuális képalkotásnál elengedhetetlen fontosságú a mérés pontossága. Ezt megállapítandó a készüléket ismert távolságokra helyeztem el egy sík felülettől, majd számítógép segítségével a soros porton érkező jelet feldolgozva megállapítottam a készülék által mért távolságot. A mérési eredményeket a 2. táblázatban foglaltam össze.

2. táblázat: A távolságmérés pontossága, hibája

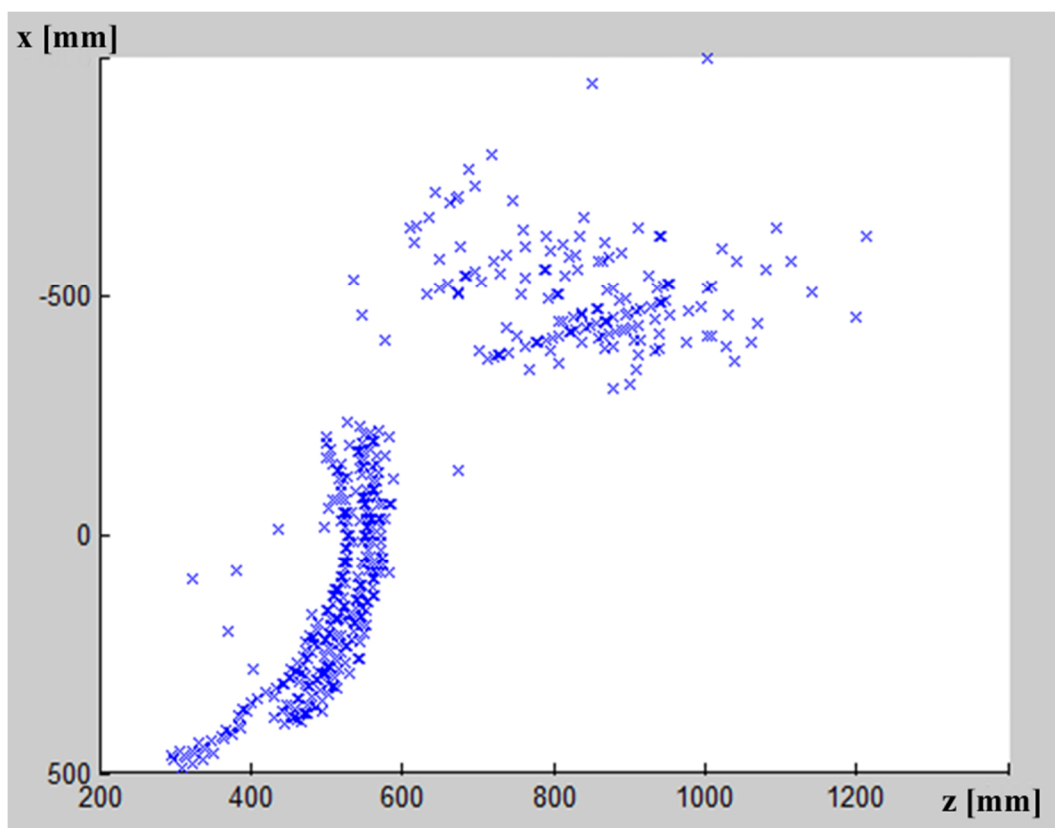
#	Ismert távolság [mm]	Mért távolság [mm]	Hiba [%]
1	50	55	10
2	100	100	0
3	150	155	3,333333
4	200	200	0
5	250	245	2
6	300	305	1,666667
7	350	345	1,428571
8	400	400	0
9	450	445	1,111111
10	500	490	2
11	550	545	0,909091
12	600	595	0,833333
13	650	640	1,538462
14	700	695	0,714286
15	750	740	1,333333
16	800	790	1,25
17	850	840	1,176471
18	900	890	1,111111
19	950	940	1,052632
20	1000	985	1,5
Átlag			1,64792

Az eredményekből látható, hogy az intelligens szenzor hibája a kisebb távolságoknál nagyobb, míg a mérési tartomány végén ez az érték kisebb, habár a mérési hiba nagyobb.

5.5. Eredmények

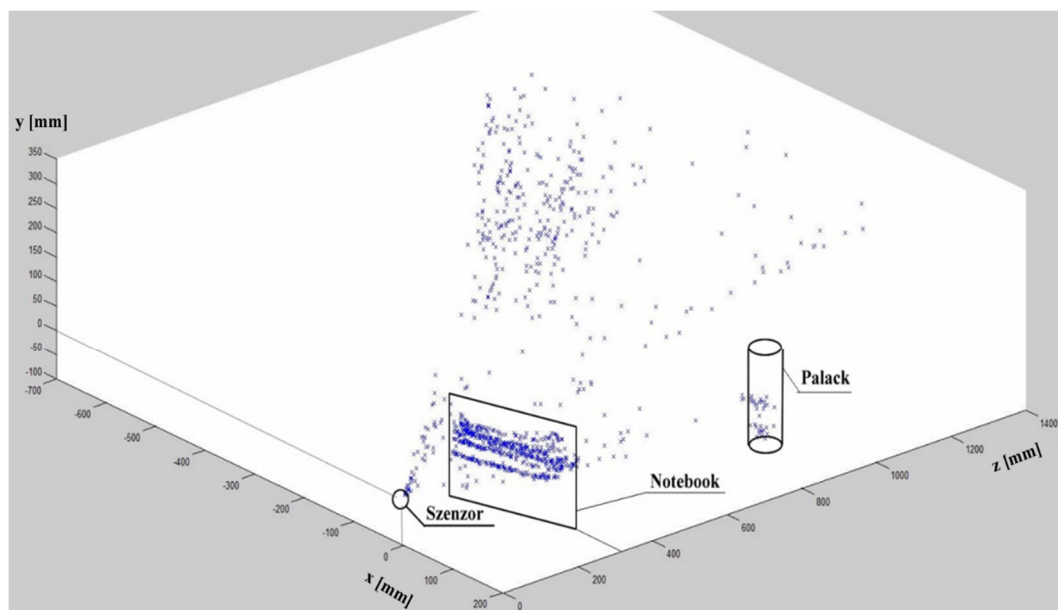
A kísérleti készülék tesztelése során arra a megállapításra jutottam, hogy az eszköz megfelel a vele szemben támasztott követelményeknek. Segítségével lehetőség nyílik virtuális képalkotásra, még hozzá költséghatékonyabban a jelenleg használatos rendszereknél (igaz kisebb pontossággal), illetve felhasználható a vakok és gyengén látók tájékozódásának segítésére.

A következőkben néhány, a készülékkel szkennelt kép látható:



19. ábra: A gömbi torzulás jelensége (sík mért felület, felülnézet)

A hibát megfelelően kompenzálva a notebook sík fedlapja és a hengeres palack képe:



20. ábra: Kézi szkennelés eredménye

6. Továbbfejlesztési lehetőségek

A készülék használhatónak bizonyult a tesztek során, de továbbfejleszthető. Elsődlegesen továbbfejlesztendő tulajdonsága a hangszintézis, mivel jelenleg a hang előállítása négyszögjel alkalmazásával történik, és ez hosszú távon kellemetlen hangot eredményez. Ezen aspektusából az irányvonal egy szinuszos hanggenerátor használata, melynél felharmonikusokat is alkalmazva egy kellemesebb hangot érhetünk el. Felmerülhet az igény egy szofisztikáltabb végfok alkalmazására is.

További fejlesztésre adhat okot az ultrahangos távolságmérő szenzorok korlátozott hatótávolsága, valamint a nagy kúpszögben történő mérés, mely nagyobb távolságok esetén jelentős felbontáscsökkenést eredményez. Ennek kiküszöbölésére érdemes lenne az ultrahangos távolságmérést lézeres távolságmérésre cserélni, és ezzel tesztelni a rendszer működését.

Végezetül lehetséges, hogy érdemes lenne a hangszintézist egy kifinomultabb hardverre bízni, például egy okostelefonra, hisz ennél a megoldásnál könnyűszerrel lehetőség nyílna személyre szabni a tájékoztatót segítő hangot, illetve a telefon hangrendszerén keresztül lehetőség nyílna a telefon kamerájával lefotózott képekről szövegfelolvasásra is. Ebben az esetben a távolságmérő rendszer Bluetooth segítségével kapcsolódhatna a mobiltelefonhoz - így könnyítve a rászoruló emberek tájékozódását.

7. Összefoglalás

A dolgozatom kitűzött célja olyan tájékozódási stratégia, és egy ezt megvalósító eszköz kifejlesztése volt, mellyel virtuális képalkotásra nyílik lehetőség, illetve melynek segítségével vak és gyengén látó emberek tájékozódását lehet megkönnyíteni.

A módszer felépítéséhez áttekintettem a szakirodalmat, mely alapján egy ultrahangos távolságmérő rendszerre esett a választásom. Mivel a természetben ez működő módszer, megvizsgáltam az emlősök ultrahanggal történő tájékozódási technikájának irodalmát, melyet szakzsargonban „echolocation”-nek neveznek. Ez alapján kezdtem a műszaki megoldás kialakítását, melyhez mikroelektronikai és informatikai, illetve gépészeti eszközöket vettem igénybe. A feltételezésem bizonyítására (miszerint az ember is képes lehet a visszaverődött hangok alapján tájékozódni) egy tesztkészüléket építettem, melyet különféle módokon próbáltam ki, értékeltem.

A tesztsorozat eredményei megfelelőek voltak, maradéktalanul kielégítették a célkitűzésben foglaltakat. A készülékkel megvalósítható a virtuális képalkotás, segítséget nyújthat világtalan embereknek a térbeni tájékozódásban, de tovább is fejleszthető. Utóbbi főként kényelmi szempontokat hivatott magasabb színvonalra emelni, de a funkcionalitás határait is kiterjesztheti.

8. Summary

The aim of my thesis was to develop a strategy for orientation, a device which give ability to create virtual images, a device which grants blind and visually impaired people to find their way.

To create the method, i reviewed the literature, and I chose the ultrasonic distancemeter system. This method works well in the nature, so I examined the literature of mammals' ultrasonic orientation technics (in the terminology: Echolocation). With this knowledge, I began to build a technical solution, using microelectronical, informatical and mechanical tools. To prove my expectation (that a person is able to navigate himself based on reflected sounds), I made a prototype and I tested it different ways.

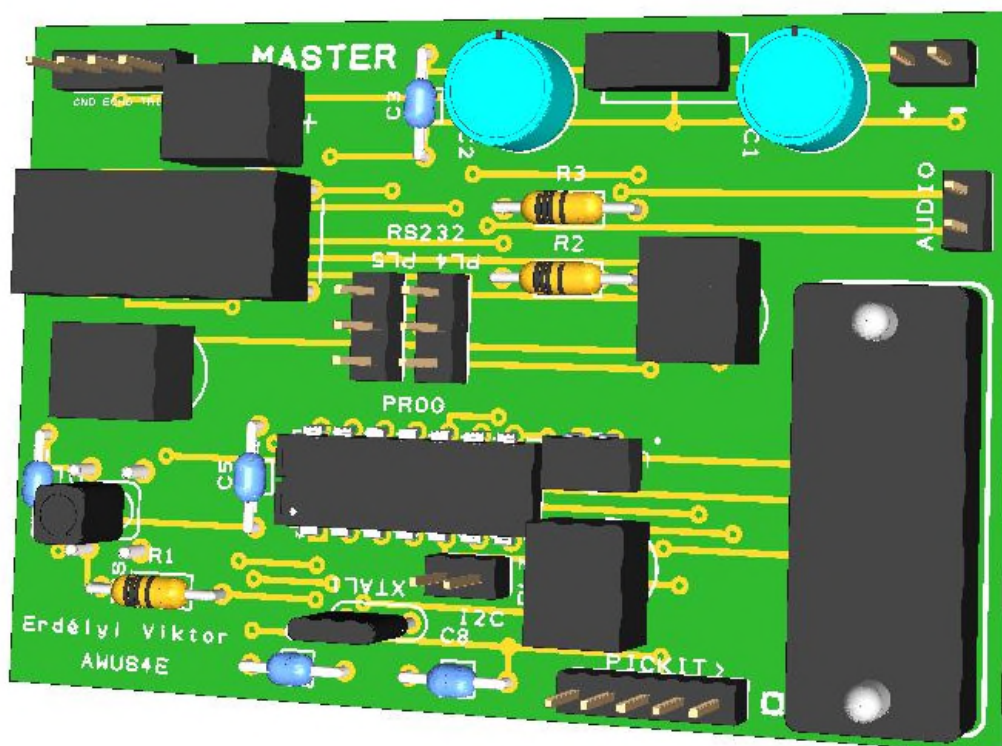
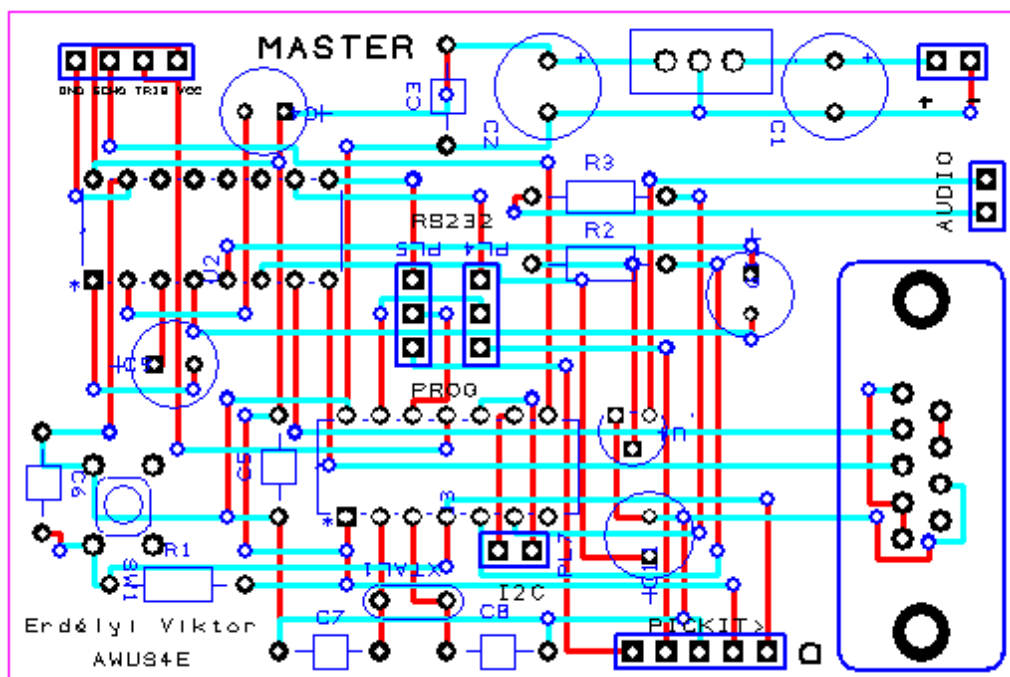
The result of the testseries was appropriate and satisfying. With this device we can achieve to create virtual images, we may help blind people navigate in space, further development also planned. The further development is mainly intended to raise the confort to a higher level, but also can extend the edges of functionality.

9. Irodalomjegyzék

1. Allan R. Hambley (2010) *Electrical Engineering Principles and Applications* fifth edition p. 408-415., 2011. ISBN-13:978-0-13-213006-6
2. Allan R. Hambley Byte Craft Limited – *First Steps with Embedded Systems* p. 5., 2002
3. Augarten, Stan (1983). "The Most Widely Used Computer on a Chip: The TMS 1000". *State of the Art: A Photographic History of the Integrated Circuit* (New Haven and New York: Ticknor & Fields). ISBN0-89919-195-9. Retrieved 2009-12-23. p. 38-40.
4. D. R. Griffin (1958). *Listening in the dark*. Yale Univ. Press, New York.
5. Dr. Arany Antal, Dr. Vízer Miklós (2008): A három- és négydimenziós ultrahang-diagnosztika a szülészetben. IME 3. p. 36-39.
6. H. Burton (2003) *The Journal of Neuroscience*, 23(10): p. 4005-4011.
7. Hill, M. N. (1962). *Physical Oceanography*. Allan R. Robinson. Harvard University Press. p. 498.
8. Jones G. (2005). "Echolocation" *Current Biology* 15(13): p. 484-488.
9. Mansoor-Ul-Hassan Siddique (2009) *Ultrasonic radar and its applications* Recent Advances in Applied Informatics and Communications ISBN: 978-960-474-107-6. p. 225-231.
10. Mihajlik Péter, Tatai Péter, Balog András (2004): Vakok tájékozódását segítő eszköz integrált sztereo ultrahangos és rádiós rendszerrel. IME III. évfolyam 2. szám p. 43-46.
11. Naohiro Matsuta, Shizuko Hiryu, Emyo Fujioka, Yasufumi Yamada, Hiroshi Riquimaroux and Yoshiaki Watanabe (2013) Adaptive beam-width control of echolocation sounds by CF–FM bats, *Rhinolophus ferrumequinum nippon*, during prey-capture flight *The Journal of Experimental Biology* 216. p. 1210-1218.
12. S. Dijkgraaf (1949). Spallanzani und die Fledermäuse. *Experientia* 5: p. 90-92.
13. Stuart Parsons and Gareth Jones (2000) Acoustic identification of twelve species of echolocating bat by discriminant function analysis and artificial neural networks *The Journal of Experimental Biology* 203, p. 2641–2656.
14. Suga N, Simmons J.A. & Jen P.H. (1975) Peripheral specialization for fine analysis of doppler-shifted echoes in the auditory system of the "CF-FM" bat *Pteronotus parnellii*. *Journal of Experimental Biology*, 63. p.161-192.

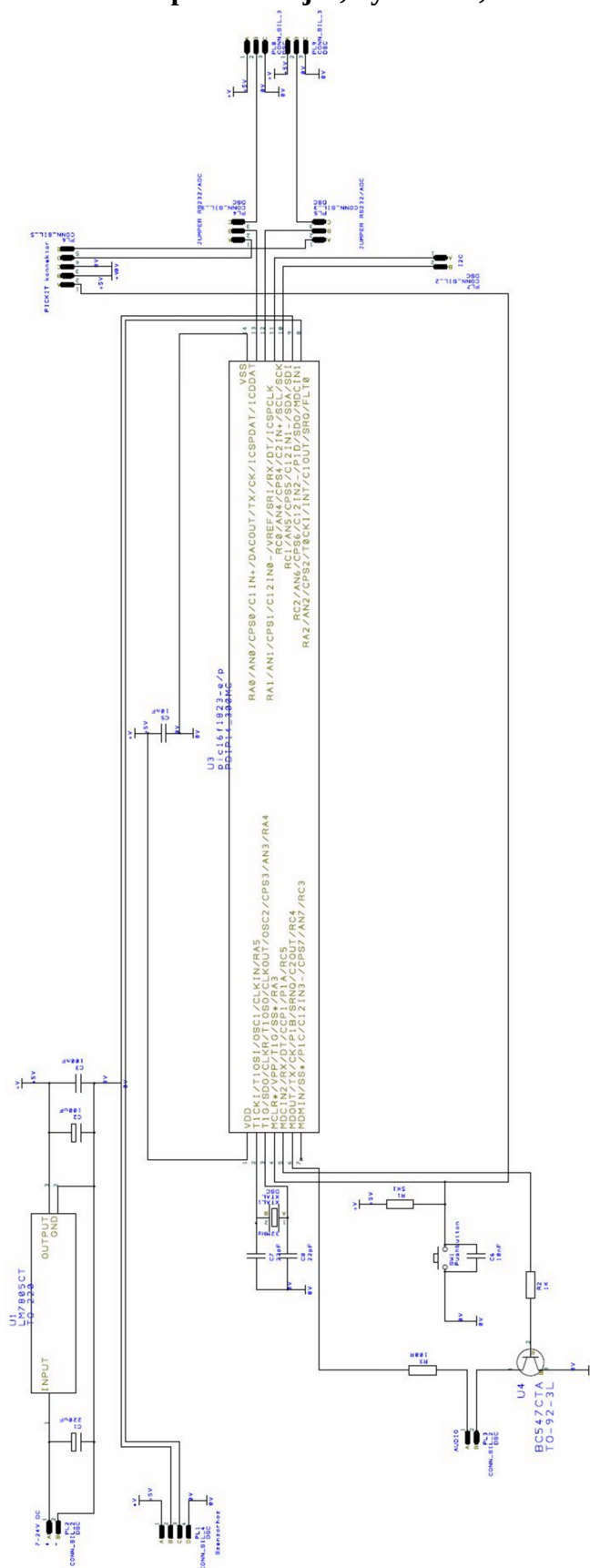
10. Köszönetnyilvánítás

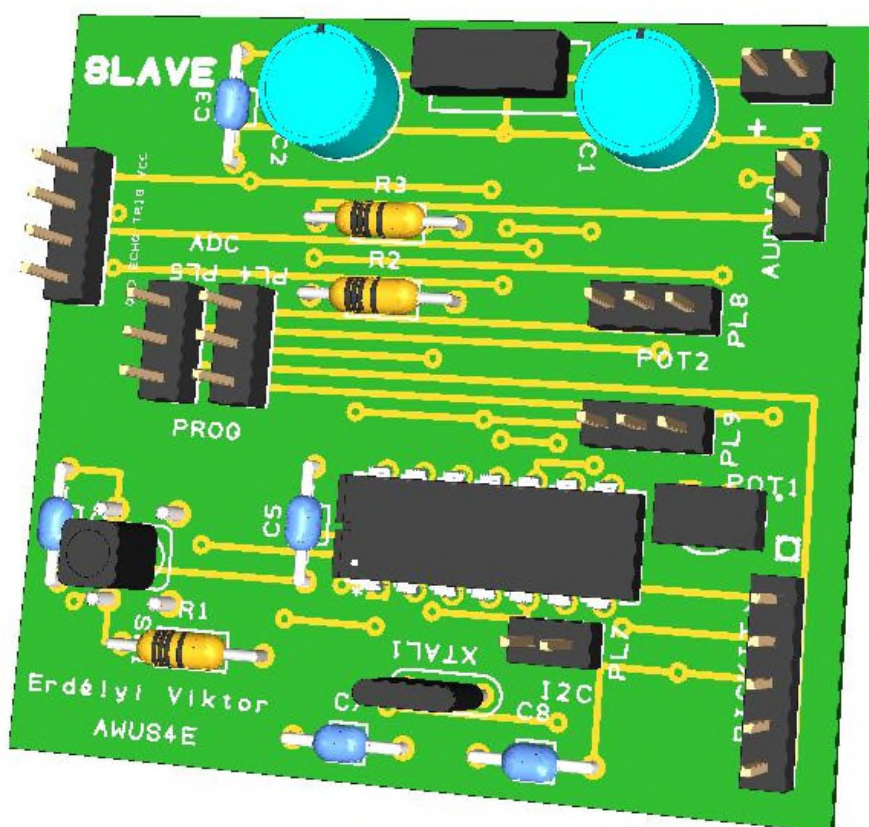
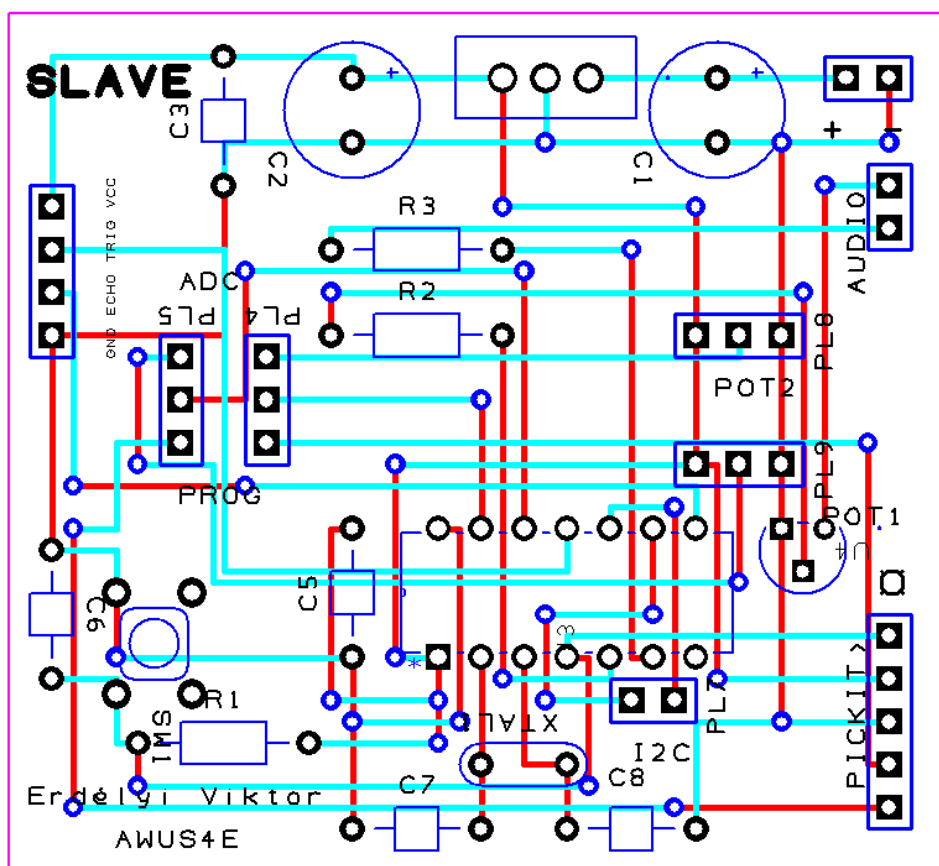
Ezúton is szeretném megköszönni, és tiszteletemet kifejezni **Dr. Jánosi Lászlónak**, aki munkám során végig támogatott, hasznos tanácsokkal látott el, **Gergely Zoltánnak**, akire számíthattam, ha elektronikai, illetve hardver közeli programírási problémákba botlottam, **Tóth János** kollégámnak, akihez a felmerülő számítástechnikai gondjaimmal fordulhattam, és **Dr. Keresztes Róbertnek**, aki lehetővé tette, hogy a tanműhelyben elkészítsem a két szabadságfokú mérőállványt, mely fontos része a kísérleti készüléknek.



2. melléklet

Slave áramkör kapcsolási rajza, nyákterve, 3D modellje:





3. melléklet

A Master mikrokontroller programkódja

```
/*
 * "BatEye"
 *TÁJÉKOZÓDÁSI STRATÉGIA*
 * ÉS IRÁNYÍTÓ ESZKÖZ
 * FEJLESZTÉSE
 *LÁTÁSSÉRÜLTEK SZÁMÁRA
 * PIC16f1823 @ 32MHz
 * 2013 Erdélyi Viktor
 * MASTER
 */
//I2C variables
#define TX_ELMNTS 1
#define RX_ELMNTS 3
// array for master to write data from slave
volatile unsigned char I2C_Array_Tx[TX_ELMNTS] = {0xFF};

// array for master to write to
volatile unsigned char I2C_Array_Rx[RX_ELMNTS] = {0x01, 0x00, 0xFD};

unsigned char I2C_slave_write_add = 0x30; // slave address R/nW = 0
(write)
unsigned char I2C_slave_read_add = 0x31; // slave address R/nW = 1
(read)
unsigned char BRG_val = 0xB3; // 0xB3 sets the SCL freq
to 100kHz

unsigned int index_i2c = 0; // index pointer
unsigned char junk = 0; // place to put unnecessary
data
unsigned char clear = 0x00; // used to clear array
location
unsigned char init_start = 1; // used to get start bit
set
unsigned char stop = 0; // used to know when to
stop
unsigned char write_to_slave = 1; // used to set up master
transmission
unsigned char trans_wrt_add = 1; // used to set up address
for write
unsigned char init_trans_data = 0; // used to set up data
transmission
unsigned char init_mstr_rec = 0; // used to set up master
reception
unsigned char rec_byte = 0; // used to know when byte
received
unsigned char set_RCEN = 0; // used to set RCEN bit
unsigned char trans_data = 0; // used to get transmission
going
unsigned char done = 0; // used to know when
finished
unsigned char read_from_slave = 0; // used to set up reception
routine
unsigned char trans_rd_add = 1; // used to set up address
for read
```

```

unsigned char read_rec_byte = 0;           // used to read data into
array
unsigned char set_ACKEN = 0;              // used to set ACKEN bit

// 8 bit variables
typedef unsigned char uint8;

// flags
bit oldstate;
bit echoold;

// Variables
uint8 Dist1;                             //Distance and azimuth
variables
uint8 Pot1;
uint8 Pot2;
uint8 Dist2;
uint8 Pitch;

// denominate I/O bits

sbit Echo at RC2_bit;
sbit Echo_d at TRISC2_bit;
sbit Trig at RA2_bit;
sbit Trig_d at TRISA2_bit;

void Tone1() {
    Sound_Play(500, 20);                  // Frequency = 659Hz, duration
= 1000ms
    //Sound_Play(Pitch, 20);              // Uncomment to modulate pitch
}

void hardinit(void) {
    CPSCON0.CPSON = 0;                    // Disable CPS module
    OPTION_REG.b7 = 0;                    // Disable weak pull up
    WPUC = 0;
    ANSELA = 0;                           // Configure AN pins as digital
    ANSELB = 0;
    C10N_bit = 0;                          // Disable comparators
    C20N_bit = 0;
    APFCON = 0b10000100;
    //OSCCON = 0b01111010;                  // Internal OSC @ 16MHZ
    OPTION_REG = 0b11010111;
    //WDTCN = 0b00010111;                  // prescaler 1:65536
// period is 2 sec (RESET

value)
    PORTC = 0x00;                          // Clear PORTC
    LATC = 0x00;                            // Clear PORTC latches
    TRISC = 0b00000011;                     // Set RC0, RC1 as inputs for
I2C
    //I2C MASTER MODULE SET UP
    SSP1STAT = 0b10000000;                  // Slew rate control disabled
for
    // standard speed mode (100
kHz, 1 MHz)
    SSP1CON1 = 0b00101000;                  // Enable serial port, I2C
master mode,
    // clock=Fosc/(4*(SSPAD+1))

```

```

        SSP1CON3.SDAHT = 1;           // Minimum of 300 ns hold time
on SDA                               // after the falling edge of
SCL
        SSP1ADD = BRG_val;           // This sets Baud rate
        PIR1.SSP1IF = 0;             // Clear the serial port
interrupt flag
        PIR2.BCL1IF = 0;             // Clear the bus coll interrupt
flag
        PIE2.BCL1IE = 1;             // Enable bus collision
interrupts
        PIE1.SSP1IE = 1;             // Enable serial port
interrupts
        INTCON.PEIE = 1;             // Enable peripheral interrupts
        INTCON.GIE = 1;             // Enable global interrupts
        Echo_d = 1;                  // Init I/O ports
        Trig_d = 0;
        Trig = 0;
        UART1_Init(9600);            // Initialize UART module at
9600 bps
        Delay_ms(100);               // Wait for UART module to
stabilize
        Sound_Init(&PORTC, 4);       // Init sound
        PWM1_Init(25000);            // Initialize PWM1 module at
25KHz
        PWM1_Start();                // Start PWM1
        oldstate = 0;                // Init flags
        echoold = 0;
        Dist1 = 0;                   // Initialize variables
        Dist2 = 250;
        Pot1 = 0;
        Pot2 = 0;
        Pitch = 400;
}

```

```

void distM(void) {                   // Subroutine for distance
measuring
        Dist2 = 255;
        echoold = 0;
        oldstate = 0;
        Trig = 1;
        oldstate = 1;
        delay_us(11);
        Trig = 0;
        if (oldstate == 1){
            while (echoold == 0){
                if (Echo == 1){
                    echoold = 1;
                    while (Echo == 1) {
                        delay_us(100); // 17 mm precision
                        Dist2 --;
                        //Pitch = Dist2*5+400; // +17 mm => +5 Hz
(uncomment to modulate Pitch)
                    }
                }
            }
        }
}

```

```

        }
    }
}

void trans(void){
    if(init_start)                // get ready to start
    {
        SSP1CON2.SEN = 1;        // Initiate START condition
        init_start = 0;
    }
    if(init_trans_data)          // get ready to transmit data
    {
        trans_data = 1;
        init_trans_data = 0;
    }
    if(done)                     // when all data is
transferred                      // send stop bit
    {
        stop = 1;
        done = 0;
        SSP1CON2.PEN = 1;        // stop bit set
    }

} // end transmission

void read(void){
    if(!write_to_slave) && (!read_from_slave)) // get ready to now read
    {
        read_from_slave = 1;      // get read routines going
        done = 0;
        index_i2c = 0;
        init_start = 1;           // get ready to start
    }

    if(read_from_slave)           // read from slave
    {
        if(init_start)           // set up start condition
        {
            SSP1CON2.SEN = 1;     // Initiate START condition
            init_start = 0;
        }
        if(init_mstr_rec)        // get ready to receive data
        {
            set_RCEN = 1;         // set master into receive
mode
            init_mstr_rec = 0;
        }
        if(rec_byte)             // check to see if data
arrived
        {
            read_rec_byte = 1;
        }
        if(set_ACKEN)            // set up ACKEN bit
        {
            set_ACKEN = 0;
            init_mstr_rec = 1;
            SSP1CON2.ACKEN = 1;

```

```

        }
        if(done)                // if array is full time to
stop
    {
        stop = 1;
        SSP1CON2.PEN = 1;      // set stop bit
    }
}
} //end read routine

void main() {
    hardinit();
    while(1) {

        trans();
        distM();                // Measuring distance
        PWM1_Set_Duty(dist2);   // Set volume
        Tone1();
        read();
        UART1_Write(I2C_Array_Rx[0]); // Data transmission to PC
        UART1_Write(I2C_Array_Rx[1]);
        UART1_Write(I2C_Array_Rx[2]);
        UART1_Write(Dist2);
        UART1_Write('\n');

    }
}

void interrupt()
{
    if(PIR1.SSP1IF)
// is it an SSP interrupt?
    {
        if(write_to_slave)
// write to slave
        {
            if(!done)
// if not done, continue
            {
                if(trans_wrt_add)
// get ready to transmit
                {
// address with write bit enabled
                SSP1BUF = I2C_slave_write_add;
// load buffer with write address
                trans_wrt_add = 0;
                init_trans_data = 1;
            }
            if(trans_data)
// ready to transmit
            {
                RC3_bit = 1;
                if(index_i2c < TX_ELMNTS)
// make sure valid data
                {
                    SSP1BUF = I2C_Array_Tx[index_i2c];
// load buffer with data
                    I2C_Array_Tx[index_i2c++] = clear;
// clear that location

```

```
    }
// and increment index
    else
    {
        trans_data = 0;
// done transmitting
        done = 1;
    }
}
}
if(stop)
// done transmitting
{
    write_to_slave = 0;
// clear so no more data transmitted
    stop = 0;
}
}
if(read_from_slave)
// read from slave
{
    if(!done)
// if not done continue
    {
        if(trans_rd_add)
// set up read address
        {
            SSP1BUF = I2C_slave_read_add;
// address and read bit
            trans_rd_add = 0;
            init_mstr_rec = 1;
        }
        if(set_RCEN)
// set up module for read
        {
            set_RCEN = 0;
            SSP1CON2.RCEN = 1;
// set RCEN to enable read
            rec_byte = 1;
        }
        if(read_rec_byte)
// reading data
        {
            if(index_i2c < RX_ELMNTS)
// make sure valid location
            {
                read_rec_byte = 0;
                rec_byte = 0;
                I2C_Array_Rx[index_i2c] = SSP1BUF;
// load array with data
                index_i2c++;
// increment counter
                set_ACKEN = 1;
// get ready to ACK
            }
        }
    }
    else
```

```
    {
// received all data
        init_mstr_rec = 0;
        rec_byte = 0;
        read_rec_byte = 0;
        done = 1;
    }
}
if(stop)
{
    read_from_slave = 0;
// clear so no more data is received
}
}

if(PIR2.BCL1IF)
// Did a bus collision occur?
{
    junk = SSP1BUF;
// dummy read to clear BF bit
    PIR2.BCL1IF = 0;
// clear BCLIF flag bit
}
PIR1.SSP1IF = 0;
// clear SSPIF flag bit
}
```

4. melléklet

A Slave mikrokontroller programkódja

```
/******  
*      "BatEye"      *  
*TÁJÉKOZÓDÁSI STRATÉGIA*  
* ÉS IRÁNYÍTÓ ESZKÖZ *  
*      FEJLESZTÉSE   *  
*LÁTÁSSÉRÜLTEK SZÁMÁRA *  
* PIC16f1823 @ 32MHz *  
* 2013 Erdélyi Viktor *  
*      SLAVE         *  
******/  
#define RX_ELMNTS      1  
#define TX_ELMNTS      3  
  
// 8 bit variables  
typedef unsigned char uint8;  
typedef unsigned int uint16;  
  
uint8 I2C_slave_address = 0x30;           // slave  
address  
volatile unsigned char I2C_Array[RX_ELMNTS] = {0}; // array for  
master to write to  
  
volatile unsigned char I2C_Array_TX[TX_ELMNTS] = // array for  
master to read from  
{0xFF, 0xEE, 0xDD};  
  
uint16 index_i2c = 0;           // used as an index pointer in array  
uint8 junk = 0;                 // used to place unnecessary data  
uint8 first = 1;                // used to determine whether data  
address                          // location or actual data  
  
// flags  
bit oldstate;  
bit echoold;  
  
// Variables  
uint8 Dist1;                    //Distance and azimuth  
variables  
uint8 Pot1;  
uint8 Pot2;  
uint8 Pitch;  
uint16 ADC0;  
uint16 ADC1;  
// denominate I/O bits  
  
sbit Echo at RC2_bit;  
sbit Echo_d at TRISC2_bit;  
sbit Trig at RA2_bit;  
sbit Trig_d at TRISA2_bit;  
sbit LED at RC3_bit;  
sbit LED_d at TRISC3_bit;  
  
void Tone1() {
```

```

    Sound_Play(500, 20);                // Frequency = 500Hz, duration
= 20ms                                // Uncomment to modulate pitch
    //Sound_Play(Pitch, 20);
}

void hardinit(void) {
    OPTION_REG = 0b11010111;           // WPU disabled, INT on rising
edge, FOSC/4                           // Prescaler assigned to TMR0,

rate @ 1:256
    CPSCON0.CPSON = 0;                 // Disable CPS module
    WPUC = 0;                          // Disable weak pull up
    ANSELA = 0b00000011;               // Configure AN pins as
digital, expect AN0 and AN1
    ANSEL0 = 0;
    C1ON_bit = 0;                      // Disable comparators
    C2ON_bit = 0;
    PORTC = 0x00;                      // Clear PORTC
    LATC = 0x00;                      // Clear PORTC latches
    TRISC = 0b0000011;                 // Set RC0, RC1 as inputs for
I2C
    //I2C SLAVE MODULE SET UP
    SSP1STAT = 0b10000000;             // Slew rate control disabled
for standard                           // speed mode (100 kHz and 1
MHz)
    SSP1CON1 = 0b00110110;             // Enable serial port, I2C
slave mode,                            // 7-bit address
    SSP1CON2.SEN = 1;                  // Clock stretching is enabled
    SSP1CON3.BOEN = 1;                 // SSPBUF is updated and NACK
is generated                           // for a received address/data
byte,                                  // ignoring the state of the
SSPOV bit                             // only if the BF bit = 0
    SSP1CON3.SDAHT = 1;                // Minimum of 300 ns hold time
on SDA after                           // the falling edge of SCL
    SSP1CON3.SBCDE = 1;                // Enable slave bus collision
detect interrupts                      // Load the slave address
    SSP1ADD = I2C_slave_address;       // Clear the serial port
    PIR1.SSP1IF = 0;
interrupt flag                         // Clear the bus collision
    PIR2.BCL1IF = 0;
interrupt flag                         // Enable bus collision
    PIE2.BCL1IE = 1;
interrupts                             // Enable serial port
    PIE1.SSP1IE = 1;
interrupts                             // Enable peripheral interrupts
    INTCON.PEIE = 1;                  // Enable global interrupts
    INTCON.GIE = 1;

    APFCON = 0b10000100;
    Echo_d = 1;                       // Init I/O ports
    Trig_d = 0;
    Trig = 0;

```

```

    Sound_Init(&PORTC, 4);           // Init sound
    PWM1_Init(25000);               // Initialize PWM1 module at
25KHz
    PWM1_Start();                   // Start PWM1
    ADC_Init();                     // ADC init
    oldstate = 0;                   // Init flags
    echoold = 0;
    Dist1 = 0;                       // Initialize variables
    Pot1 = 0;
    Pot2 = 0;
    Pitch = 400;
    LED_d = 0;
    LED = 0;
    I2C1_init;

}

void adcread(void){                 // Reading ADC values,
    converting into 8 bit format
    ADC0 = ADC_Read(0);             // reading 10-bit value from ADC
    ADC1= ADC_Read(1);

    Pot1 = ADC0 >> 2;               // writeing significant 8 bit to
variable
    Pot2 = ADC1 >> 2;

    I2C_Array_TX[0] = Pot1;
    I2C_Array_TX[1] = Pot2;
}

void distM(void) {                  // Subroitune for distance
    measuring
    Dist1 = 255;
    echoold = 0;
    oldstate = 0;
    Trig = 1;
    oldstate = 1;
    delay_us(11);
    Trig = 0;
    if (oldstate == 1){
        while (echoold == 0){
            if (Echo == 1){
                echoold = 1;
                while (Echo == 1) {
                    delay_us(100);    // 17 mm precision
                    Dist1 --;
                    //Pitch = Dist2*5+400;    // +17 mm => +5 Hz
                }
            }
        }
        (uncomment to modulate Pitch)
        I2C_Array_TX[3] = Dist1;
    }
}

void main() {
    hardinit();

```

```

        while(1) {

                if (I2C_Array[index_i2c++] == 0xFF) {
                        adcread(); // Reading ADC
channels        distM(); // Measuring
distance        PWM1_Set_Duty(Dist1); // Set volume
                Tone1(); // Play Tone
                I2C_Array[index_i2c++] = 0x00;
                }
        }

void interrupt () {
        if (PIR1.SSP1IF) // check to see if
SSP interrupt
        {
                RC3_bit = 1;
                if (SSP1STAT.b2) // Master read (R_nW = 1)
                {
                        //RC3_bit = 1;
                        if (!SSP1STAT.b5) // Last byte was an address (D_nA
= 0)
                        {
                                SSP1BUF = I2C_Array[index_i2c++]; // load with value
from array
                                SSP1CON1.CKP = 1; // Release CLK
                        }
                        if (SSP1STAT.b5) // Last byte was data (D_nA
= 1)
                        {
                                SSP1BUF = I2C_Array[index_i2c++]; // load with value
from array
                                SSP1CON1.CKP = 1; // Release CLK
                        }
                }
                if (!SSP1STAT.b2) // Master write (R_nW = 0)
                {
                        if (!SSP1STAT.b5) // Last byte was an address (D_nA = 0)
                        {
                                first = 1; //last byte was address, next will be data
location
                                junk = SSP1BUF; // read buffer to
clear BF
                                SSP1CON1.CKP = 1; // Release CLK
                        }
                        if (SSP1STAT.b5) // Last byte was data (D_nA
= 1)
                        {
                                if (first)
                                {
                                        index_i2c = SSP1BUF; // load index with array
location
                                        first = 0; // now clear this since we
have

```

```

    } //location to read
from/write to

    else
    {
        if (index_i2c < RX_ELMNTS) // make sure index
        { //out of range of
            I2C_Array[index_i2c++] = SSP1BUF; //load array
        }
        else
        {
            junk = SSP1BUF; //array location not valid,
discard data
        }
    }
    if (SSP1CON1.WCOL) // Did a write collision
occur?
    {
        SSP1CON1.WCOL = 0; // clear WCOL
        junk = SSP1BUF; // dummy read to clear
BF bit
    }
    SSP1CON1.CKP = 1; // Release CLK
}
}
}
if (PIR2.BCL1IF) // Did a bus
collision occur?
{
    junk = SSP1BUF; // dummy read SSPBUF to
clear BF bit
    PIR2.BCL1IF = 0; // clear bus
collision Int Flag bit
    SSP1CON1.CKP = 1; // Release CLK
}
PIR1.SSP1IF = 0; // clear SSPIF flag
bit
}

```

5. melléklet

A prototípus fényképe

