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

Horváth András

EMBERI AKTIVITÁS AKUSZTIKUS MONITOROZÁSA SZENZORHÁLÓZAT SEGÍTSÉGÉVEL

MSc diplomaterv

KONZULENS

Orosz György

BUDAPEST, 2013

MŰEGYETEM 1782

DIPLOMATERV-FELADAT

Horváth András (LHFLSW)
szigorló villamosmérnök hallgató részére

Emberi aktivitás akusztikus monitorozása szenzorhálózat segítségével

A szenzorhálózatok manapság egyre több területen elterjedő, intelligens egységekből felépülő mérőrendszerek. A szenzorhálózatok legfőbb előnye, hogy a vezeték nélküli kommunikációnak köszönhetően telepítésük egyszerű, nem igényli hálózati infrastruktúra kiépítését, a hálózatot alkotó eszközök elrendezése pedig flexibilis.

A szenzorhálózatokat gyakran alkalmazzák emberi aktivitás megfigyelésére, például egy épületen belül személyek helyzetének meghatározása, a megfigyelt személyek életvitelének nyomonkövetése, stb. A hallgató feladata egy olyan kísérleti tesztrendszer kiépítése, amely segítségével akusztikus jelek alapján képes bizonyos információkat kinyerni a helyiségben megfigyelhető jelekből (például a hangforrás emberi eredetű-e, mi a hangforrás pozíciója, stb.). Ebben a kísérleti rendszerben vezetéknélküli szenzorhálózat végzi az adatgyűjtést.

A hallgató feladatának a következőkre kell kiterjednie:

- Ismerje meg a rendelkezésre álló hardver és szoftver környezetet (Berkeley micaz mote-ok és TinyOS operációs rendszer).
- Valósítson meg egy egyszerű adatgyűjtő rendszert szenzorhálózat segítségével!
- Végezzen előzetes méréseket, vizsgálja az adatgyűjtő rendszer által gyűjtött jelek minőségét!
- Tekintsen át néhány olyan módszert, amely alkalmas akusztikus jelek alapján a hangforrás pozíciójának becslésére! Válasszon egy tetszőleges eljárást, és implementálja az elkészített adatgyűjtő rendszerben!

Tanszéki konzulens: Orosz György, adjunktus

Budapest, 2013. március 14.

Dr. Jobbágy Ákos
tanszékvezető

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

1117 Budapest, Magyar Tudósok krt. 2. I. ép. I.E.444.
Telefon: 463-2057, Fax: 463-4112
http://www.mit.bme.hu • e-mail: mitadm@mit.bme.hu

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Horváth András**, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2013. 12. 18.

...…….
Horváth András

Kivonat

A vezeték nélküli szenzorhálózatok egy rohamosan fejlődő ágazat, melynek igen széles felhasználási lehetőségei az ipari felügyelettől a medikain át az otthoni alkalmazásig terjednek. A szenzorhálózatok egyes elemei (mote-ok) felszereltségétől függően sokféle környezeti tényező állapotát képesek érzékelni.

Diplomamunkám során a vezeték nélküli szenzorhálózatok segítségével az akusztikai alapon történő emberi aktivitás monitorozását kívántam megvalósítani. Ehhez az eszközök megismerése és egy saját rendszer kiépítése után a Time Difference of Arrival elv használata szükséges, mellyel a hangérzékelés időkülönbségéből meg lehet becsülni a hangforrás helyét. Ehhez szükséges a mote-ok szinkronizációja, egymáshoz viszonyított helyzetük és a hangterjedés tulajdonságainak ismerete. A meghatározáshoz jelentős számítási kapacitás szükséges, mely a kisfogyasztású hálózattal nehezen egyeztethető össze, így ezt egy számítógépen futó Matlab program végzi majd. A rendszer és a technológia tesztelését és az esetleges javítási lehetőségek vizsgálatát is el kívánom végezni.

A feladat részeként megismerkedtem a Berkeley MICAz mote-okkal, operációs rendszerükkel, programozási nyelvükkel, a különböző hálózati topológiákkal, emellett tanulmányoztam a szinkronizációs módszereket, a környezettől függő hangfelismerési lehetőségeket, és az akusztikus helymeghatározás elméletét, majd a megvalósítást. A rendszer elkészülése után részletes teszteket végzek valós környezetben tapasztalható korlátainak feltérképezésére, értékelem ezeket, és vázolom a későbbi továbbfejlesztési lehetőségeket.

Abstract

The wireless sensor networks (WSN) are very fast evolving technology, whose possible applications are in industry, medical and home environment. The motes of the sensor networks can sense many environmental factors.

In this thesis I would like to realize the acoustic monitoring of human activity with WSN. The monitoring system requires to build a data acquisition network from the sensor nodes, which measures the ambient noises. The aim of the monitoring system is to give an estimation of the possible position of noise source.

The position of the voice source is estimated using Time Difference of Arrival principle. To perform an accurate estimation, we need the synchronization of the nodes, and we should also know their positions. The estimation needs large computing capacity, so the localization algorithm is implemented on a PC with Matlab. I have investigated the proper operation and limitations.

As a part of the work I have got to know the Berkeley MICAz motes, their operation system, programming language and network topologies. Besides I meet and study synchronization methods, voice detection and recognition possibilities, the theory of the acoustic positioning, then the implementation. After completing the design of the network, I have tested the capabilities of the real system by performing measurements under different conditions. Measurement results and possible ways of further developments are also summarized in my thesis.

1 Bevezetés

A vezeték nélküli szenzorhálózatok manapság - széleskörű felhasználási lehetőségeik miatt - igen nagy elterjedtségnek örvendenek. További előnyeik a kis fogyasztás, a vezeték nélküli technológia alkalmazásából adódó szabadság és a felhasználható szenzorok sokrétűsége.

A későbbiekben bemutatásra kerülnek a vezeték nélküli szenzorhálózatok (WSN, azaz Wireless Sensor Network), amelyekkel lehetővé válik a távoli felügyelet, adatgyűjtés, amely hasznos lehet a forgalomirányításnál, egészségügyi alkalmazásoknál, katasztrófa esetén, ezen kívül számos más alkalmazásban is. A szenzorhálózat lényege, hogy a csatlakoztatott node (hálózati csomópont) csoportot egy bizonyos feladatra dedikáljuk (pl. lokalizációra). Ezek a node-ok együttműködnek és hálózatot alkotnak.

A lokalizációs technikák alkalmazása elengedhetetlen a vezeték nélküli szenzorhálózatoknál, mivel a vezeték nélküli kommunikáció miatt az egyes hálózati csomópontoknak, vagy alkalmazástól függően a megfigyelt objektumnak tetszőleges pozíciója lehet. A helymeghatározáshoz szükségünk van bizonyos információra a mote-ok helyzetére vonatkozóan és a megfigyelt eseményre vonatkozóan. Esetemben a mote-ok pozíciója ismert volt, és a lokalizáció egy bizonyos hangforrásra történik.

A következőkben bemutatom az egyik legelterjedtebb – általam is használt – MICAz mote-ot, mely egyszerűen felhasználható különböző vezeték nélküli hálózatok kiépítésére. Az eszköz rendelkezik mikrokontrollerrel, memóriával, saját tápellátással és rádiós kommunikációra képes adóvevővel. A mote-ot moduláris felépítésűre tervezték, amely így lehetőséget biztosít különböző bővítőkártyák csatlakoztatására, amelyeken szenzorok kaptak helyet. Bemutatásra kerül továbbá operációs rendszerük és programozási nyelvük. A mote-ok segítségével összeállítok egy olyan hálózatot, amely lehetőséget biztosít a kitűzött cél, a lokalizáció megvalósítására. A hálózat egy bázisállomáson csatlakozik a PC-hez, amely az adatgyűjtést és a számításigényes feladatokat Matlab környezet segítségével végzi.

Az általam kiválasztott lokalizációs technika elsősorban az emberi aktivitás akusztikai alapon történő monitorozására szolgál. Felhasználási területei lehetnek intelligens otthonok megvalósításai, egészségügyi monitorozás, továbbá rendszerfelügyelet és vagyonvédelem.

A 2. fejezetben ismertetem a vezeték nélküli szenzorhálózatok tulajdonságait, általános alkalmazásait, és kiemelek néhány konkrét felhasználási lehetőséget.

A 3. fejezet a tervezett rendszer különböző tulajdonságairól szól, megemlítem a felhasznált eszközöket, és ismertetem a fejlesztés főbb fázisait.

A 4. fejezetben az alkalmazott hardver és szoftvereszközök kerülnek bemutatásra. A felhasznált hardver a MICAz mote, amely TinyOS operációs rendszert futtat, melyet NesC nyelven programozhatunk.

Az 5. fejezetben a kommunikáció tulajdonságait mutatom be, és az alkalmazott hálózati technológiákat, továbbá a rendszer alapjául szolgáló adatgyűjtő hálózat ismertetése is megtörténik, a hálózatban részvevő eszközök feladatának bemutatásával.

A 6. fejezetben a későbbiekben elengedhetetlen szinkronizációról, a lehetséges technikákról és a kiválasztott technika megvalósításáról olvashatunk.

A 7. fejezet bemutatja a különböző lokalizációs metódusokat, és a kiválasztottat, azaz a TDOA-t (Time Difference of Arrival), amely az egyes mote-ok által érzékelt esemény beérkezési időkülönbségeiből számítja a hangforrás pozícióját.

A 8. fejezet a megvalósításról szól, amelyben a hálózat eleminek bemutatása, és működésük ismertetése történik.

A 9. fejezetben a mérési eredmények összefoglalása, és a különböző tesztekhez használt környezetek bemutatása kerül ismertetésre.

A 10. fejezetben az esetleges további lépések és a rendszer fejlesztési lehetőségei olvashatók.

A 11. fejezetben összefoglalom a munka során gyűjtött tapasztalatokat.

2 Szenzorhálózatok

2.1 Általános bemutatás

A szenzorhálózatok autonóm node-okból álló kooperatív rendszerek, amelyek elemei a központi feladat elvégzését elosztott módon valósítják meg. A későbbiekben elsősorban a vezeték nélküli szenzorhálózatokkal (Wireless Sensor Network, WSN) foglalkozom. [23]

Ilyen típusú hálózat alkalmazása a feladat és a mérendő rendszer tulajdonságai miatt szükséges, mivel ezek nem teszik lehetővé egy eszköz használatát. A hálózati kommunikáció emiatt elengedhetetlen, amely gyakran kis sávszélességű rádiós csatornát takar. Tipikus elrendezésben az érzékelők különböző környezeti paramétereket mérnek (fényerő, hőmérséklet, légnyomás, stb.), és a mért vagy számított értékeket a bázisállomás felé továbbítják. A fejlesztési irány leginkább a méret és az ár csökkentése, amelyhez elengedhetetlen a MEMS (Micro Electromechanical System) technológia szerves beépülése a hálózati elemekbe.

Az egyik vízió az ún. smart dust, röviden intelligens por, amely lényege, hogy a jövőben porszem méretű node-okból álló vezeték nélküli szenzorhálózatokat hozhassunk létre, amelyhez a MEMS (Micro Electromechanical System) megjelenésével nagy előrelépés történt ebben az irányban.

A chip ára függ az analóg és digitális áramköri elemek számának arányától. A technológiai fejlődéssel együtt a digitális alkatrészek mérete csökken, az analóg összetevőké viszont tipikusan nem változik – pl. azon passzív alkatrészeknél, melyek paraméterei a fizikai méreteik függvénye.

A rádiós áramkörök energiafogyasztása a mérettel arányos, így ezen a téren a fogyasztási követelmények egybeesnek a méretbeliekkel. Az legnagyobb alkotórész az RF adóvevőknél a vevő oldali csatornaszűrő, melyhez analóg esetben nagyméretű kapacitások, digitális esetben pedig anti-alias szűrő beépítése szükséges.

További szempont a darabszámból adódó egységár, mivel annál olcsóbb egy darab eszköz, minél többet gyártanak belőle. Emiatt lényeges, hogy a fizikai réteg minél több ország szabályaival legyen összhangban. A rádiós frekvenciáknál ez többnyire az ISM sáv

(Instrumentation, Scientific, and Medical band, azaz Ipari, Tudományos és Orvosi frekvenciasáv) használatát jelenti. A kiválasztott ISM sáv kompromisszumok eredménye az ár, fogyasztás és a hatékonyság között. A 2.4 GHz-es – általam is használt – ISM sáv számos technológia működési frekvenciája. Az Egyesült Államokban rendelkezésre áll az ultra-szélessáv szabvány, amely a 3.1 és 10.6 GHz közti tartományt jelenti. [7]

2.2 Jellegzetességek

A hálózatot alkotó autonóm elemek a legtöbb esetben rendelkeznek saját mikrokontrollerrel, energiaforrással, érzékelőkkel és kommunikációs interfészekkel – rádió, soros interfész –, néhány esetben beavatkozó eszközökkel is. A gazdasági szempontok miatt az ilyen eszközök többnyire olcsók, kisméretűek, korlátozott számítási kapacitással, tápellátással, memóriával és kommunikációs képességekkel rendelkeznek. További probléma, hogy a szenzoron futó programot és a rendszert mint elosztott hálózatot más szempontok alapján kell működtetni. Az architektúrális megoldások meglehetősen eltérőek lehetnek, így az operációs rendszerek támogatása általában nem széleskörű. További problémák még a kommunikációs csatornák esetleges túlterheltsége, a szabályozás szükségessége, melyekhez a vezetékeshez képest nagyobb hibaarány társul. Az egyes érzékelők elromolhatnak, lemerülhetnek, esetleg más környezeti tényező miatt kieshetnek a hálózatból, így pl. egy továbbított csomag útvonala gyakran ad-hoc módon jön létre.

2.3 Követelmények

A szenzorhálózatoknál lényegesebb követelmények a skálázhatóság, önszerveződés, a megbízhatóság, pontosság és az ár együttese. A hálózat skálázható, ha a hálózat méretváltozása következtében fellépő minőség- és teljesítménybeli változás minimális. A hálózatnak tudni kell kezelnie a változásokat, lehetőleg emberi beavatkozás nélkül, azaz legyen önszerveződő. Ez azért is lényeges, mivel kis fogyasztásuk miatt a szenzorhálózatok akár évekig is működhetnek emberi beavatkozás nélkül. A hálózat kialakításánál fontos szempontok a megbízhatóság, pontosság és ár, amelyet leginkább az alkalmazás határoz meg. Nincs általánosan alkalmazható ideális megoldás, mindenképp valamilyen kompromisszumot kell kötnünk az említett szempontoknál.

2.4 Alkalmazások

A vezeték nélküli szenzorhálózatok az emberiség hétköznapjainak számos területén megjelennek. Az 1970-es években a hadiiparban kezdődött a fejlesztése, ám a civil felhasználók körében is gyorsan elterjedtek. A technikai fejlődéssel együtt az árak csökkentek, a képességek pedig bővültek, manapság a beltér és kültér megfigyelésére, irányítására számos lehetőség kínálkozik. Az alkalmazások főbb csoportjai a következők:

- **Egészségügy**: kórházi menedzsment, katasztrófa-elhárítás (pozíció- és állapotszolgáltatás, vészriasztás), otthoni betegellátás (tevékenység-monitorozás, riasztás)
- **Gyártás, raktározás**: gyártósor monitorozás, készletnyilvántartás (intelligens raktárak, vasúti szerelvények, tartálykocsik, tankhajók)
- **Környezetvédelem**: élőhely-monitorozás (állatok viselkedése, növény-társulások mikroklímája), katasztrófa-előrejelzés (árvíz, vulkánkitörés)
- **Mezőgazdaság**: „precíziós mezőgazdaság” (szőlőtermesztés: öntözés, növényvédelem, szüret igény szerint)
- **Mérnöki alkalmazások**: épületek, műtárgyak monitorozása (hidak statikai ellenőrzése), közlekedésfelügyelet, intelligens épületek
- **Védelmi alkalmazások**: megfigyelés, követés, detektálás (elektronikus kerítés), lokalizálás (akusztikus lövés-lokalizáció)
- **Űrkutatás**: Mars-szondák [14]

A témához kapcsolódó irodalom áttekintése során több mintarendszerrel is találkozhatunk, melyek közül az alábbiakban ismertetek néhányat.

Életminőség javítása - AAL

A dolgozathoz leginkább kötődő terület az AAL (Ambient Assisted Living), jelentése elsősorban a környezet ismeretében történő életminőség javítását foglalja magában. Egy értekezés ezzel kapcsolatban az aktivitás monitorozás intelligens otthonokban aktív tanulási metódussal egymást átfedő tevékenységek jelenlétében. Az idős, egyedül élő emberek számának növekedésével ez egyre növekvő felhasználási területeket jelent. A rendszer

lehetőségeket ad az egyes tevékenységek felismerésének megtanulására, az esetleges hasonló aktivitások elkülönítésére, mindezt annak érdekében, hogy érzékeljük az orvosi vészhelyzeteket, vagy a viselkedésbeli változásokat. [9]

További felhasználási területei hasonló alkalmazásokban az alvásmonitorozás, beszédfelismerés, különböző aktivitások nyomon követése.

Lőfegyver helyének meghatározása

A projekt célja Berkeley mote-ok segítségével történő helymeghatározás, mindez a DARPA támogatásával. A hangforrás egy lőfegyver vagy támadó, amely helyét minél pontosabban kell meghatározni. Az érzékelőket egy védett területen nem fix topológiában, egyenletesen szétszórják. A projekthez egy FPGA-val (Field-Programmable Gate Array, azaz felhasználó által konfigurálható áramkör) és 3 mikrofonnal kiegészített akusztikus bővítőkártyát használtak. A hangfelvételek adatait párhuzamosan feldolgozva a beérkezési időpontok különbségéből meghatározható egy irány. Az egyes mote-októl származó információt összegezve a lövés helye kiszámítható. Az FPGA-nak köszönhetően helyben történő adatfeldolgozásra nyílt lehetőség, így a rendszer válaszideje 2 s körül van. A pontosság átlagosan 1 méteres. [15]

1. ábra: Lőfegyver helyének meghatározása [15]

Környezeti paraméterek megfigyelése

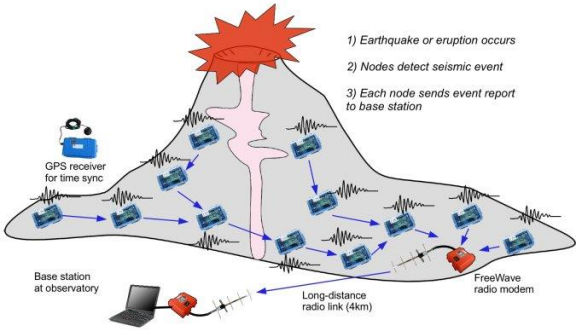
A Berkeley Egyetem egyik projektje a Great Duck Island sziget mikroklímájának megfigyelése volt, amely ökológiailag törékeny terület, így a lehető legkevesebb emberi beavatkozással kellett kiépíteni a monitorozó rendszert. A mote-ok az általam is használt Berkeley mote család tagjai, melyeket később részletesen is ismertetek. Az erőforrás kezelés

2. ábra: Környezetmonitorozáshoz használt Berkeley mote [16]

lényeges szempont volt a projektben, mivel többhónapos mérési eredmények szükségesek. A megfigyelt paraméterek a hőmérséklet, fényerő, páratartalom illetve a szenzor közelében elhelyezkedő fészek aktivitása. Az adatokat egy hierarchikus topológiájú hálózaton továbbították a bázisállomáshoz, amelynek segítségével az adatok az interneten is elérhetővé váltak. [16]

Vulkán működésének megfigyelése

A Harvard Egyetem projektje aktív vulkánok működésének megfigyelését tűzte ki célul. Az esetleges kitörések és földmozgások tanulmányozásával jobban megérthetők a veszélyesen működő vulkánok. A felhasznált szenzorhálózat olcsó, kisméretű és alacsony fogyasztású, amely az eddigi megfigyelések eszközeihez képest hatalmas előny. A megfigyelések az equadori Tungurahua és Reventador vulkánon zajlottak, elsősorban szeizmikus adatokat gyűjtve, melyeket a bázisállomáshoz továbbítottak. A rendszer segítségével meghatározható a földrengés vagy kitörés, a jövőben pedig az adatok akár előrejelzési célokat is segíthetnek. [17]



3. ábra: Vulkáni környezetbe telepített szenzorhálózat [17]

3 Rendszerterv és feladatok összefoglalása

A rendszerterv az általam használt rendszerről, azok hardver és szoftverelemeinek rövid bemutatásáról szól. A működést, az alkalmazott módszereket és a későbbi fejezetek szükségességét is megalapozza a rész.

Cél

A cél az akusztikai alapon történő helymeghatározás, amely a rendelkezésre álló Berkeley MICAz mote-ok segítségével valósult meg.

Ismeretek

A rendszer létrehozása előreláthatóan mikrokontrolleres, hálózati, jelfeldolgozási és Matlabon belüli ismereteket követel meg, és ezek magasabb szintű ismeretéhez segít hozzá.

Hardver

A hardver a Berkeley Egyetemen tervezett MICAz mote, amely 8 bites mikrokontrollerrel, rádiós adóvevő egységgel és különböző bővítőkártyákkal számos alkalmazáshoz felhasználható. A mote-ot elsősorban nagy elemszámú vezeték nélküli szenzorhálózat létrehozásához tervezték.

A hálózat mellett a rendszer részét képezi egy PC, amely a nagyobb számításigényű adatfeldolgozást végzi.

Szoftver

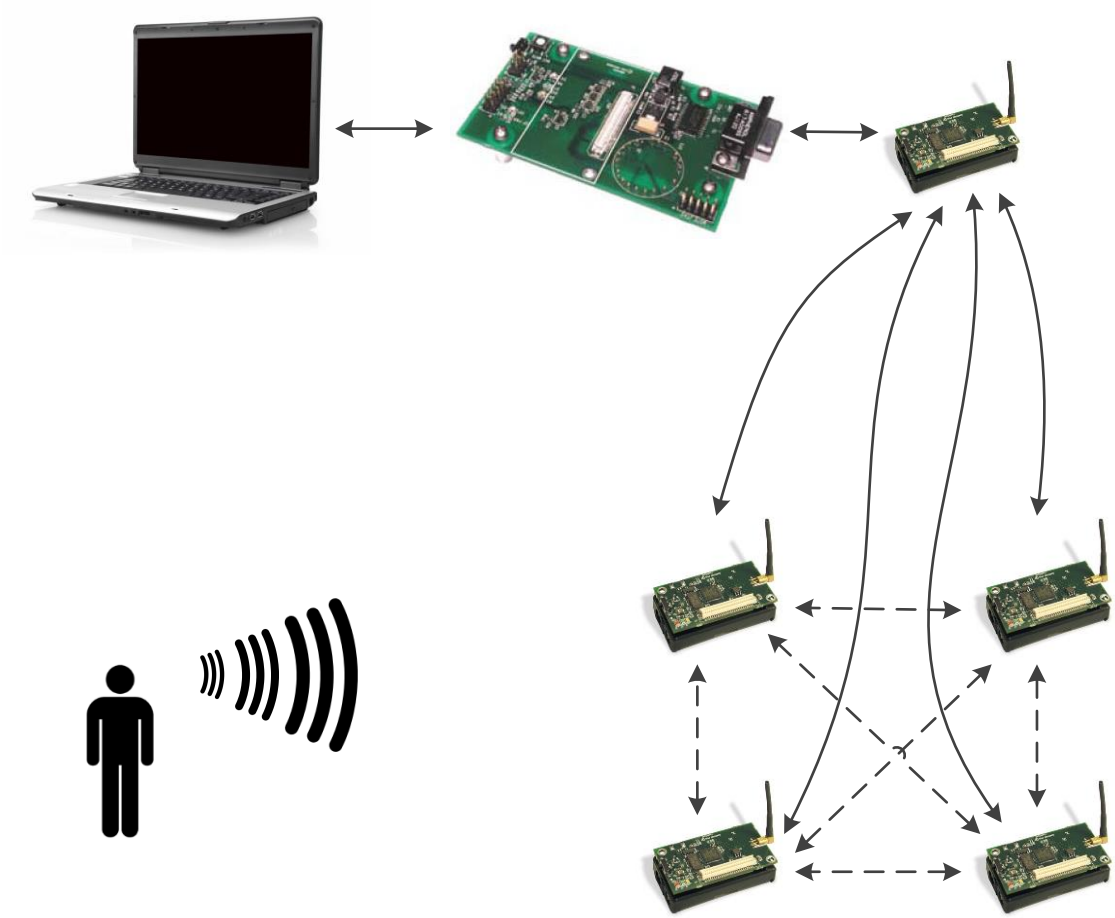
A mote-okon egy TinyOS nevű beágyazott operációs rendszer fut, amelyet NesC nyelven programozhatunk. A TinyOS speciális üzenetkezelést biztosít és az egyes modulok összehuzalozásával konstruálhatunk programot. A PC-n Matlab környezetben kerül feldolgozásra a begyűjtött adat.

Hálózat

A hálózat többelemű, csillag topológiájú, amelyben egy központi node irányítja a kommunikációt, bizonyos esetekben viszont megengedett bármely node-ról érkező broadcast

kommunikáció is. A résztvevők szenzormote-ok és egy kitüntetett szerepű bázis, amely kapcsolatban van a PC-vel.

Felépítés



4. ábra: A hálózat felépítése

A hálózat a fenti ábra szerinti összeállításban működik. Az adatgyűjtés egy triggerfeltétel bekövetkezése után indul, amellyel egy időben kezdődik a mikrofon jelének elmentése. A bázis a mintavételezés után begyűjti az adatokat a szenzoroktól, majd továbbítja a számítógépre, ahol megtörténhet a további feldolgozás.

Szinkronizáció

A feladat megoldása során szükséges a térben elosztottan érzékelt jelek időbeli összerendelésére. Ahhoz, hogy ezt megfelelő pontossággal tegyük, a szenzorok működését és időzítési paramétereit össze kell hangolni. Erre szolgál a szinkronizáció.

Helymeghatározás

A feladat célja, hogy egy kiválasztott helymeghatározó algoritmust implementáljak, és megvizsgáljam annak tulajdonságait, továbbfejlesztési lehetőségeit. A választott elv az ún. TDOA, azaz a hangok beérkezési idejének különbségéből történő pozícióbecslés. Ez egy passzív lokalizációs eljárás, amelynek lényege, hogy egy megfigyelni kívánt eszköz által keltett zajokat érzékeli. Alkalmazásomban a zajok leginkább tranziens jellegűek, amely lehetővé teszi az ún. triggerüzenetek (eseményindító üzenet) használatát. A triggerüzenet a szenzorok által érzékelt hangok alapján kerül kiküldésre egy broadcast üzenet formájában. A szenzorokon a mintavételezés folyamatos, csupán az adatmentés függ a triggerjeltől. Ha bármelyik szenzor meghatározott ideig átlépi a megadott hangerősség szintet, indítja az adatmentést és triggerjelet küld.

A beérkezett hangminták eltolódása a keresztkorrelációs függvényük tulajdonságaiból határozható meg. A kapott adatokból ezután a legvalószínűbb hangforrás pozíció az egyes mote párok által meghatározott helyek alapján becsülhető.

4 A felhasznált hardver- és szoftvereszközök ismertetése

A fejezet bemutatja a felhasznált hardver- és szoftvereszközöket, a Berkeley mote családot, a TinyOS operációs rendszert és a különböző fejlesztőeszközöket, melyeket igénybe vettem a fejlesztés során.

4.1 A Berkeley mote-ok

A Berkeley mote családot a Berkeley Egyetemen fejlesztették vezeték nélküli szenzorhálózatok alkotóelemeinek. A platform moduláris, nyílt és rengeteg felhasználóval büszkélkedhet.

Négy fontosabb részre bonthatjuk: tápellátás, szenzorok, számítás és kommunikáció. A mote-ok legnagyobb előnyei a kis méret, alacsony ár, a szerény energiafelhasználás és a kivitelezés, illetve a felhasználás sokszínűsége. Elterjedtebb típusok a Mica2, Mica2dot és a Micaz [5. ábra.]. Hardveres felépítésben nincs számottevő különbség, csupán a MICAz rádiós modulja különbözik a többitől, így nagyobb adatátviteli sebességet kínál, rövidebb hatótávval. További különbség a Mica2dot fizikai felépítésében (pénzérme méret és alak) és az ebből adódó tápellátásban keresendő. Az mote-ok többféle szenzorkártyával bővíthetők, amelyek a felhasználásnak megfelelően választhatók ki. [18]

5. ábra: A MICA mote család 3 tagja (balról jobbra: MICAz, MICA2 és MICA2dot) [18]

Közös operációs rendszert futtatnak, amely a vezeték nélküli szenzorhálózatok tagjain történő felhasználásra készült. Az operációs rendszer a TinyOS, mely kis erőforrásigényű, eseményvezérelt módon működik és moduláris felépítésű. Az operációs rendszerre ún. NesC nyelven fejleszthetünk, amely C alapú, és a rendelkezésre álló hardverekhez szükséges könyvtárak különböző modulként kerültek implementálásra.

4.2 MICAz mote főbb jellemzői

A Crossbow cég gyártja a MICA mote családot (így a MICAz mote-ot is), melynek technológiai alapja a Berkeley egyetem került kifejlesztésre.

6. ábra: A Berkeley MICAz mote [1]

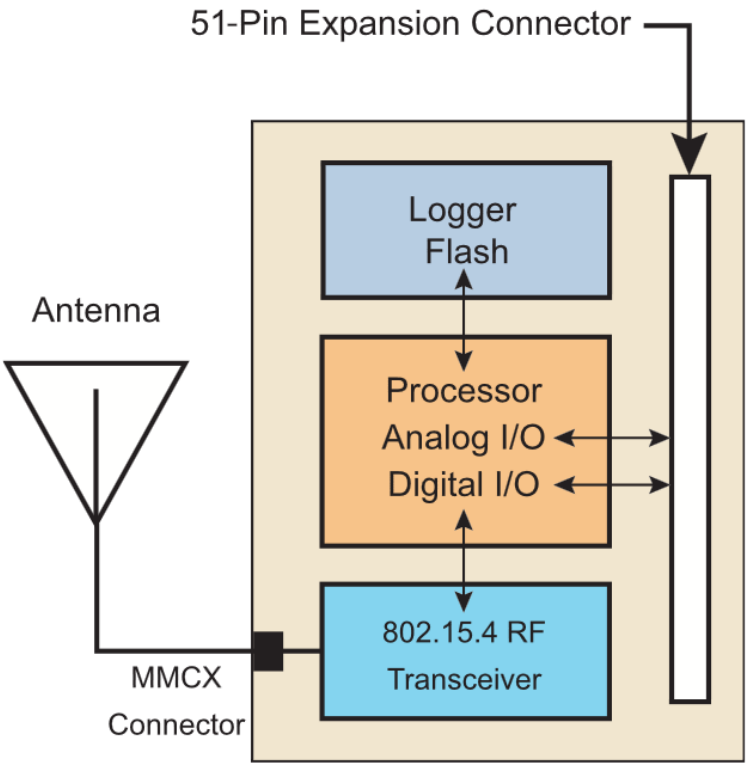
A MICAz mote főbb komponensei a nyolc bites mikrokontroller és a Zigbee szabvány szerint működő kis sebességű rádiós adóvevő. A mote beágyazott szenzorhálózatokhoz készült, és vezeték nélküli kommunikációban minden node képes routerként működni. Moduláris felépítésű, bővíthető fény, hőmérséklet, légnyomás, gyorsulás, hang, mágneses és egyéb szenzorokat tartalmazó board-okkal. Alkalmazásai pl. a beltéri épületmonitorozás és biztonság, akusztikus, vibrációs és más szenzoradatok gyűjtése és továbbítása, emellett alkalmas óriás hálózatok kiépítésére, melyek ezer fölötti elemszámúak.

A MICAz mote részletes jellemzői

A mote főbb tulajdonságai összegezve:

- MICAz mote (MPR2400CA) - 6. ábra
    - mikrokontroller: ATmega128L (Low power, azaz kis fogyasztású)
        - 8 bites ALU
        - 4 kB RAM
        - 128 kB programmemória
        - 7,3728 MHz órajel
        - 8 csatornás, 10 bites ADC
    - rádiós adóvevő: CC2420, 2.4-2.48 GHz
    - bővítő csatlakozó (51 pines)
    - LED-ek: 3 db
    - elemtartó: 2 db AA-s elem számára
- opcionális szenzorkártya (MTS300/MTS310) – 9. és 10. ábra
    - fényérzékelő
    - hőmérséklet
    - mikrofon
    - zümmer
    - hangérzékelő áramkör
    - 2 szabadságfokú gyorsulásérzékelő (MTS310)
    - 2 szabadságfokú magnetométer (MTS310)

A MICAz mote blokkvázlata látható a lenti ábrán. A felépítés legfontosabb elemei jól látszanak. Középen az AVR, ehhez csatlakozik a soros flash, a rádiós adóvevő és a bővítőkártya csatlakozója.



7. ábra: A MICAz mote blokkvázlata [3]

A mote pontos típusa MPR2400CA, melyben egy ATmega128L kis fogyasztású mikrokontroller kapott helyet, amely analóg és I/O bementeket, I2C, SPI és UART interfészeket támogat. A bővítőkártyák csatlakoztatása az 51-pines bővítő csatlakozó segítségével történik, így perifériák széles választéka csatlakoztatható. A processzor tulajdonságai a 128 kbyte programmemória, 4 kbyte konfigurációs EEPROM, UART (0 – 3 V), soros kommunikációs interfész, 10 bites ADC, emellett I/O, I2C, SPI kompatibilitás. Fogyasztása 8 mA aktív és < 15 µA alvó állapotban.

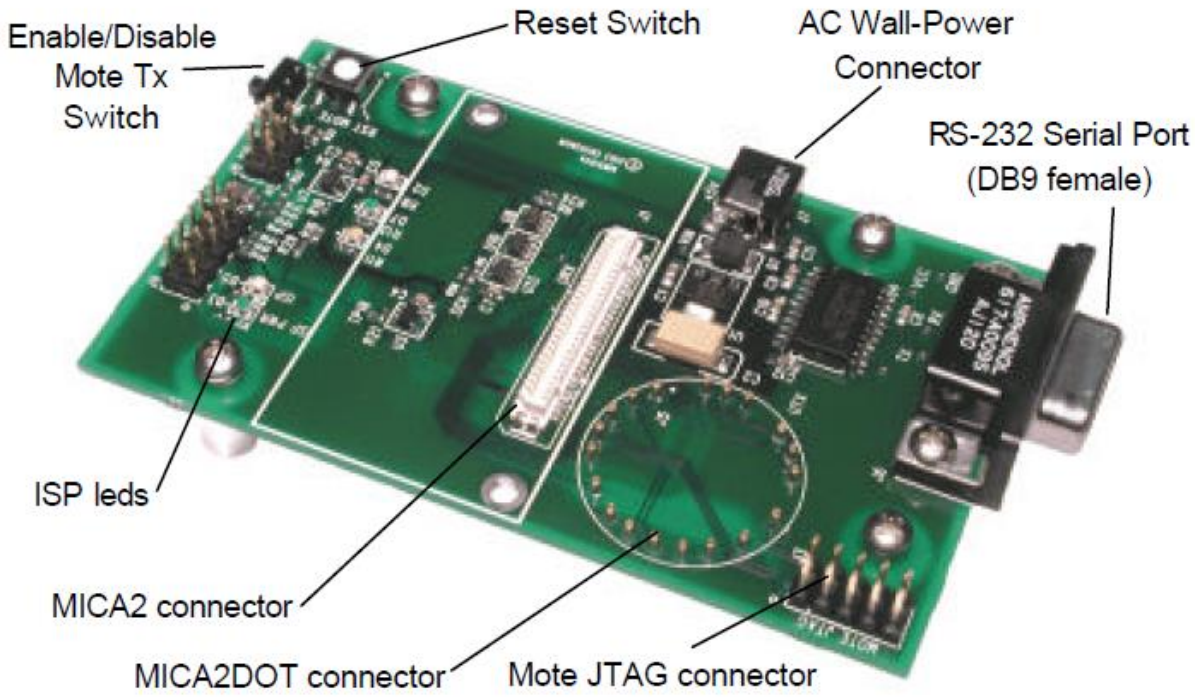
A vezeték nélküli összeköttetésért az IEEE 802.15.4 kompatibilis RF (rádiófrekvenciás) adóvevő felel, amely a 2,4-2,48 GHz-es ISM sávban képes működni, RF zavarokkal szemben ellenálló és támogatja a hardveres titkosítást (AES-128). A rádiós adóvevő frekvenciasávja programozható 2400 – 2483,5 MHz között 1 MHz-es lépésekben. Az adatátviteli sebesség maximum 250 kbps, teljesítménye -24 dBm és 0 dBm között állítható. A vevő érzékenység minimálisan -90 dBm lehet, a szomszédos csatorna elnyomás 47/38 dB (+5/-5 MHz channel spacing). A kültéri hatótávolság 75 – 100 m, a beltéri 20 – 30 m. Az egység áramfelvétele vétel esetén 19.7 mA, adáskor 11 mA (-10 dBm-es beállításnál),

14 mA (5 dBm-es beállításnál), 17.4 mA (0 dBm-es beállításnál), 20 µA (üresjárásban, bekapcsolt feszültségszabályzóval) és 1 µA alvó állapotban.

Egyéb technikai paraméterek pedig a 2 db AA elemes működtetés, külső táp csatlakoztatási lehetőség, 2,7 és 3,3 V közti feszültséggel (Molex csatlakozóval), továbbá három LED. Mérete mm-ben 58 x 32 x 7, súlya 18 gramm.

4.3 Bázisok – MIB510

A gateway board egyrészt lehetővé teszi a PC-vel vagy más platformmal történő kommunikációt, másrészt ennek segítségével programozhatjuk az eszközt. A gateway board-ra csatlakoztatva egy mote-ot bázisként működhet. Az általam használt MIB510 soros interfésszel rendelkezik. Alkalmazható továbbá a MIB520 USB-s és a MIB600 TCP/IP alapú hálózatokhoz történő csatlakoztatásra.



8. ábra: A MIB510 programozó kártya [1]

Az általam használt programozó a MIB510, amely a MICA mote család tagjainak csatlakoztatására ad lehetőséget. A board-ot egy DB9 típusú soros csatlakozón keresztül köthetjük össze a számítógéppel. A kártya lehetőséget biztosít továbbá JTAG

csatlakoztatására, saját tápcsatlakozóval szerelték fel, rendelkezik reset gombbal, ISP LED-ekkel és a soros kommunikációs igény esetén letiltó kapcsolóval. Az ISP (működés közbeni programozó) képes egy bizonyos bájtkombináció érkezése esetén átvenni a soros port irányítását, ezzel egyidejűleg letiltva a mote soros Tx és Rx vonalát. Ez reseteléskor és felprogramozáskor történhet meg – helyes működés esetén.

A gateway board lehetőséget biztosít a kiegészítő kártyák csatlakoztatására is, melyet a mote felhelyezését követően a kártya alsó részén tehetünk meg. A reset gomb nemcsak a mote-ot, hanem az ISP-t is újraindítja, ISP – mote sorrendben. A JTAG csatlakozó debuggolásra használható. A csatlakoztatott eszköz tápellátása biztosított a programozó kártya által. A tápcsatlakozó regulátorral ellátott, és 5-7 V-ig táplálható, amelyet 3 V-ra szabályoz. A programozóhoz mellékelve található egy 230 V-ról üzemelő 5 V-os táp. További védelemként az ISP feszültségét figyelő áramkör is helyet kapott a board-on, amely a feszültség 2,9 V alá csökkenése esetén letiltja a programozást, és ezt LED-del is jelzi. Az RS-232 interfész DB9-es csatlakozón keresztül érhető el, amely csupán a küldés és fogadás vonalakat használja.

4.4 A szenzorkártyák

A mote-okhoz csatlakoztatható többféle szenzortípust tartalmazó bővítő kártya, melyek például a légnyomásmérő, páratartalom mérő, a gyorsulásérzékelő, a GPS modul és mágnes térerősség mérésére alkalmas szenzor is. Ennek fényében elmondhatjuk, hogy a Berkeley MICA mote-ok családja a szenzor kártyákkal együtt rendkívül sokrétű felhasználásra ad lehetőséget.

9. ábra: Az MTS300 bővítőkártya [1]

A mote-hoz készült bővítő modulok közül az MTS300 típusú állt rendelkezésemre. A bővítő modulok elsődleges szerepe a mote funkcióinak bővítése, mely a modularitás révén a kisfogyasztású eszközöknél ésszerű megoldás. Ezzel az energiafelhasználást mérsékelhetjük, emellett a kívánt szenzorok is elérhetővé válnak. Az energiahatékonyságot tovább növeli működésük is, melynek lényege, hogy a szenzorok mindaddig inaktívak, amíg a felhasználó be nem kapcsolja azokat. Az általam használt bővítő modulnak több verziója is elérhető, melyek csupán a felszereltségben térnek el. A továbbiakban a fent említett modult mutatom be részletesebben, emellett említést teszek egyéb elérhető kártyákról is.

10. ábra: Az MTS310 bővítőkártya [1]

Mikrofon

A mikrofon fő felhasználási területei az akusztikai terjedésvizsgálat (pl. tengerfenék feltérképezése) és a hangrögzítés –, illetve vizsgálat. A mikrofonhoz csatlakozik egy előerősítő és egy digitálisan szabályozható erősítő második fokozatként.

A mikrofon jelét lehetőségünk van közvetlenül az AVR egyik lábára kötve használni, ha a mikrofon kimenetválasztóját ennek megfelelően állítjuk be. Ez az általam használt megoldás is, mivel ez akusztikus adatgyűjtéshez használatos.

A második fokozat kimenete aktív zajszűrőn keresztül csatlakozik a hangdetektorhoz. A hangdetektor IC a mikrofon jelét egy egybites digitális jellé alakítja, mely a 4 kHz-es hang érzékelése alapján jelez. A 4 kHz-es hangot a modulon elhelyezkedő rezonátor képes biztosítani.

Zümmer

A rezgő (buzzer, sounder) egy fix 4 kHz-en működő piezoelektromos rezonátor. A vezérlés, és a frekvenciaszabályzás beépített, csupán a periféria be- és kikapcsolására van lehetőségünk.

Fény- és hőmérséklet

A fényszenzor CdSe fotocellás, maximum érzékenységen 690 nm hullámhosszú fényt képes érzékelni. Nagy fényerősségnél magas, sötétben nulla közeli értéket vesz fel.
A hőmérséklet szenzor egy feszültségosztó részeként kapott helyet a board-on, típusa Panasonic ERT-J1VR103J. A mérési tartomány feléhez van beállítva a 25 °C, működési tartománya -40 és 70 °C közt alakul.

A szenzorok kezelésére egyszerűen használható vezérlőjeleket alkalmaztak, amelyeknek a szenzorok időszakos használatánál, így az energiatakarékosságnál van fontos szerepe. [1]

4.5 TinyOS és NesC– a szenzorhálózat szoftverelemei

A mote-okon az ún. TinyOS (TOS) fut, amely egy szenzorhálózatokhoz fejlesztett C nyelv alapú eseményvezérelt operációs rendszer. Többek közt MICA mote-okon fut és kezeli a főbb funkciókat, a perifériákat, a fogyasztást és a rádióhálózatot. Az operációs rendszer az alkalmazásunk része, nem létezik az alkalmazás alatt futó mag, amely a legtöbb általános célú PC-n is fut. Támogatott a többtaszkos ütemezés, nem preemptív ütemezés mellett, azaz a taszkok nem szakíthatják meg egymás futását. Nincs dinamikus memóriafoglalás, azonban a kevés memória miatt ennek megléte nem is indokolt. TinyOS futtatható a következő platformokon: MICA, MICA2, MICA2dot, MICA128, MICAz (a Crossbow termékek közül), emellett támogatja még az avrmote-ot és PC-t is.

Az rendszer mögöttes célja, hogy minimalizálja a fogyasztást. A működést a "hurry up and sleep" (sietve végezzük a feladatot, és alvás), azaz a gyors feladatvégzést követő energiatakarékos módba kapcsolással lehetne jellemezni. Ennek keretein belül akkor történik parancsvégrehajtás, ha azt valamilyen esemény indikálja. Az imént ismertetett működésre azért van szükség, hogy az akkumulátoros üzemidő kibővüljön, a fogyasztás csökkentésének eredményeképp.

A programozás nesC nyelven történik. Az nyelv a C nyelv egy változata, amelyben implementálásra került a “wiring”, azaz huzalozás metódus. Jellemzői, hogy egyszerre hordozza a modularitás és a hatékonyság előnyeit illetve nyílt forráskódú. A rendszer előre megírt szoftverkomponensek összessége (hálózata), melyeket tetszőlegesen összeköthetünk az interfészeiken keresztül.

Az alkalmazások komponensekből állnak, amelyek a nesC nyelv építőkockáinak tekinthetők, egyfajta esemény (event) és parancs (command) listának - ezek a komponens interfészein keresztül hívhatók. Az interfészek az egyetlen eszközei a komponens adattovábbításának. Egy komponens tartalmazhat például LED, timer vagy egy ADC interfészt. A programok különböző komponensek interfészeinek összehuzalozása révén alakulnak ki.

A nesC programozási nyelvet modulárisnak tervezték, ezzel lehetőséget adva a felhasználónak, hogy a kódrészleteket egyszerűen összekapcsolja. Az interfészek igazából parancsokat linkelnek, és eseményeket szolgáltatnak egyik komponenstől egy másiknak, amely használni kívánja azokat. Egy komponens interfésze modulban definiált. A parancsok és az események, amelyeket átadnak az interfészek, modulok részeként kerülnek definiálásra. A komponensek konfiguráción keresztül huzalozottak.

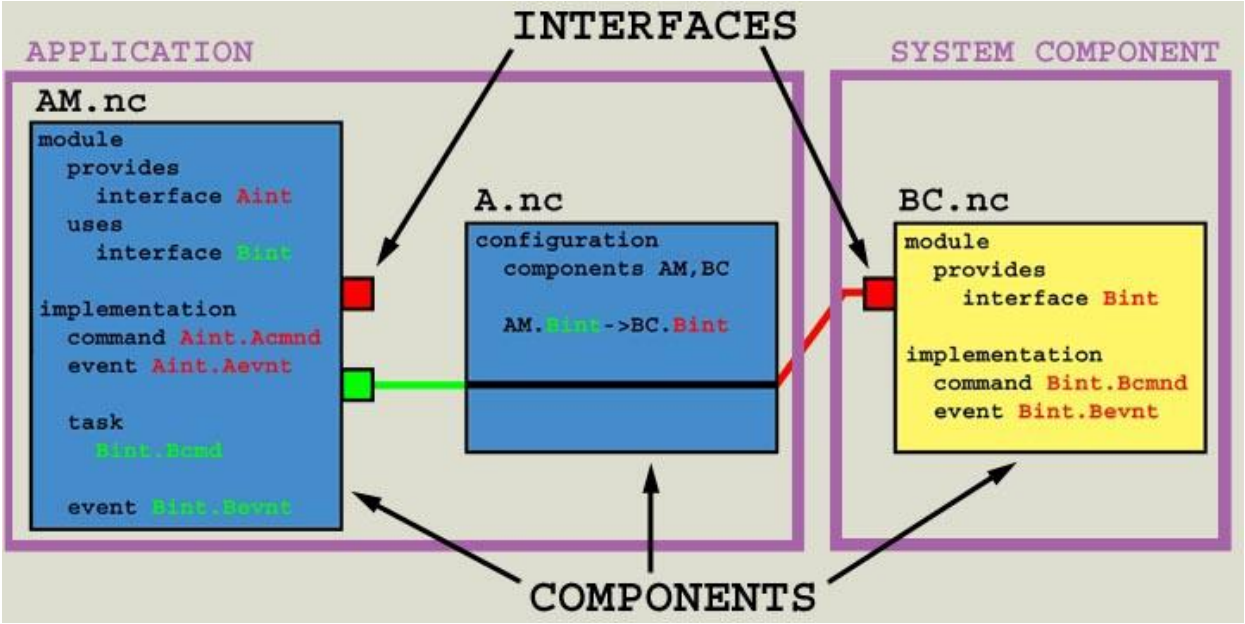
A komponensek hierarchiába szervezhetők, ezáltal a komponensek tartalmazhatnak más komponenseket, melyek a fejlesztés során rejtve maradnak. Ennek előnye, hogy a fejlesztés átláthatóbb, és könnyebben kezelhető maradhat. Hátránya az alsóbb szintekben lévő hiba esetén mutatkozhat meg, mivel ekkor az implementált rétegek sokaságában kell hibát keresnünk. Ez egy-egy programnál akár többtíz szintet is jelenthet.

Az operációs rendszer alapszabálya, hogy a különböző események kezelésénél a processzort nem foglalhatjuk le sokáig. Ha erre mégis szükségünk lenne, egy taszkot (task) kell ütemeznünk. A taszk a TinyOS-en belül lényegében egy függvény. A taszk hívásakor (post) a függvényre mutató pointerek egy FIFO-ba kerülnek, ahonnan a feladatütemező indítja a meghívott függvényeket (posztolt taszkokat). A taszkok így nem szakíthatók meg, de külső esemény által felfüggeszthetők.

A fordítás eredményeképp létrejön az AVR-re tölthető gépi kód, melyből a fordító előzetesen egy C nyelvű kódot is generál, amit megtalálunk egy adott mappában. Ezáltal az

egyes komponenseink felhasznált része C nyelven is rendelkezésre áll, az esetleges átültetéshez.

A jobb megértés érdekében nézzünk egy példát.



11. ábra: A nesC nyelv ismertetése [10]

A fenti ábrán jól látható módon a BC modul a Bint interfészt szolgáltatja, melynek implementációjában definiálva van a Bint.Bcmnd parancs és a Bint.Bevnt esemény. Ha az AM modulban használni szeretnénk ezt, a uses paranccsal kell megtennünk. Ehhez szükséges egy konfigurációs fájl is, melyben össze kell huzaloznunk az AM Bint interfészét a BC Bint interfészével. Ezek után az AM modul hozzáfér a BC modul parancsaihoz. Az AM modulban definiálhatunk tetszőleges parancsokat, eseményeket és taszkokat, mely a BC modul parancsainak használatára alkalmasak. Az AM modulba kötött BC modul eseményeit szükséges lekezelnünk az AM modulban.

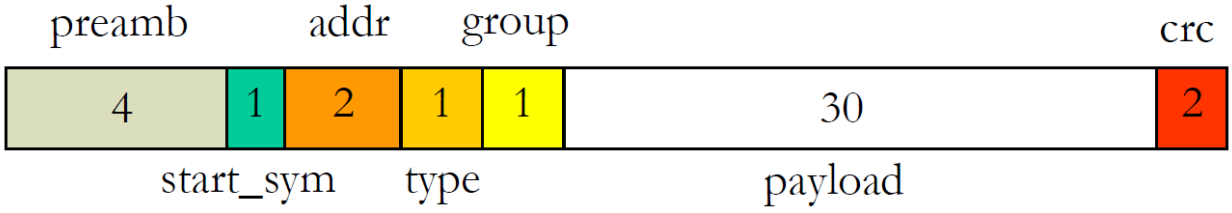
Egy átlagos vezeték nélküli hálózatban az egyes csomópontok működése három csoportba sorolható, melyek az érzékelés, feldolgozás és kommunikáció. Vegyünk egy átlagos hálózati csomópontot, és vizsgáljuk meg a működését. Konkrét példaként vegyünk egy tűzjelző rendszert, melyben az egyes füstérzékelők megfelelő kiegészítéssel vezeték nélküli szenzorhálózatot alkotnak. Első lépés az érzékelés, mely itt a füstöt detektálja, egy

füstérzékelő segítségével. A következő lépésben a feldolgozás következik egy AD átalakítás formájában, és a küldeni kívánt adatot becsomagolja egy üzenetbe. Harmadik lépésként az előkészített adatot küldi, fogadja a beérkezőket, és státusz információt szolgáltat. Az általam felépített hálózat is hasonló működésű lesz, kiegészülve egy számítógéppel. [10]

4.5.1 A TinyOS hálózati kommunikációja

Az operációs rendszer egyik legfontosabb és legbonyolultabb része a rádiós kommunikációt reprezentáló könyvtár. A rádión keresztüli adattovábbítás miatt meg kell felelni az elküldendő adatsorozatnak bizonyos feltételeknek, azaz a nulla és egy bitek szerepeljenek rövid távon is egyenlő számban. Ez azért szükséges, mivel a vevő a vett jel amplitúdóját egy mozgó átlaghoz viszonyítja, amely alapján eldönti, hogy egy vagy nulla értéket kapott-e (on-off keying). A kódolások közül több teljesíti ezt a feltételt, ilyen pl. a Manchester-kódolás. További probléma az ütközések kezelése, mivel nem észlelhetőek és megelőzésük sem oldható meg tökéletesen. A rendszerben emiatt CSMA alapú hozzáférés-vezérlés került implementálásra, amelynek lényege a következő: az adó véletlen hosszú ideig vár az üzenetküldés előtt, majd belehallgat a csatornába. Adás esetén ismét véletlen hosszú ideig várakozik, ellenkező esetben elkezdi az üzenetküldést. Bizonyos esetekben ez sem segít, képzeljük el, hogy két, egymást nem halló csomópont kíván adni egyszerre egy köztük lévő vevőnek. Emiatt az ilyen rendszerekben a minél rövidebb csomagok használatára törekednek, így növekedhet a sikeres átvitelek aránya. Hibás ellenőrzőösszeg (CRC) esetén ugyanis a csomag eldobásra kerül. A CRC ellenőrzés lényege, hogy a beérkezett csomagból ismert metódussal újraszámítjuk a CRC-t, majd összehasonlítjuk a fogadott üzenet megfelelő bitjeivel.

A TinyOS-ben az alábbi üzenetfelépítés használatos.



12. ábra: A rádiós üzenet felépítése [11]

A preamble (preamb) szerepe az üzenet elején a kódolás miatt szükséges említett középérték beállítása. A start symbol speciális minta, amely az üzenet kezdetét hivatott jelezni. Ezek után a fejléc következik, amely a cím (address és group), csomagtípus (type) bitjeit tartalmazza. A következő mező a hasznos adat (payload), az üzenet végén pedig a CRC található.

A szenzorhálózatokban a hibakeresés a működési sajátosságok miatt bonyolult és nehézkes feladat. Ezt tovább nehezíti a kimeneti perifériák hiánya, továbbá az eszköz belső állapotához nehéz hozzáférni. Létezik ugyan megfelelő megoldás, mint pl. a JTAG, de megfelelő hardver nélkül ez sem használható. Az egyik legfontosabb hibakeresési lehetőség a szimuláció, amely nem helyettesítheti a hardveren történő viselkedés vizsgálatát, de az egyes működés közbeni állapotok vizsgálhatók, és az egyes hibák determinisztikusan előhozhatók. A TinyOS része egy szimulációs eszköz (TOSSIM), amely segítségével esemény alapú szimulációt végezhetünk, nagyságrendileg 1000 csomóponttal. A hálózat közvetlenül a TinyOS forráskódjából épül fel, majd a környezet segítségével a kommunikációt is vizsgálhatjuk, amelynek érdekében üzeneteket juttathatunk be, illetve ki. [11]

A TinyOS-ben az ütemezés az alábbi szinteken megvalósuló működés eredményeképp alakul ki:

- **hadveres megszakítás:** a legmagasabb szintű megszakítás, mely bármilyen alacsonyabb prioritású kódrészletet megszakíthat, ha engedélyezve van a megszakításkezelés
- **aszinkron command, aszinkron event:** olyan parancsok és események, amelyek megszakításrutinokban generálódnak
- **taszk:** a olyan művelet, melyeket bármely programrészletből bármikor elindíthatunk. A korábban leírtakat összefoglalva: a taszkok híváskor egy FIFO-ba kerülnek, majd az ütemező által hajtódnak végre
- **szinkron command, szinkron event:** olyan parancsok és események, melyek a taszkokban generálódnak.

A kölcsönös kizárás megoldható, ha csak taszkokon keresztül végezzük az erőforrás kezelést, vagy a beépített atomic parancs használatával, mivel ekkor tiltva vannak a megszakítások.

4.6 PC-s környezet

A környezet installáláshoz a következő kritériumoknak kell eleget tennünk:

- Linux, vagy emulált Linux fusson a számítógépen
- ne legyen installált TinyOS
- a Java Developer Kit ne legyen installálva

Ezek nélkül is meg lehet próbálni, de a bőséges error üzenetekkel számolni kell.

A kódokat Windows-on Cygwinben fordíthatjuk le, mely egy linux emulátor. A jegyzettömbben fejlesztett kódokhoz navigálva Cygwinnel fordíthatunk, majd tölthetünk programot a mote-okra. Szerkesztőprogramnak a JCreator alkalmazást használtam, amely lehetőséget biztosít a különböző nyelvekben használt parancsok definiálására, ezáltal megkönnyítve a fejlesztést. A TinyOS környezet segítségével alkalmunk nyílik a pc-re történő fordításra, emellett az egyes elkészült programokhoz tartozó dokumentum és blokkvázlat generálására. Ezek igen hasznos funkciók.

A felprogramozáshoz a MIB510 soros programozó board-ot használtam. A csatlakoztatott mote-ból ajánlott kivenni az elemeket, mivel a programozó saját tápellátásával együtt használva ez gondot okozhat. Természetesen a mote is képes energiát szolgáltatni a programozónak, ám a korábban említett okok miatt érdemes a programozó tápjával együtt használni.

Matlab

A PC-n történő feldolgozáshoz több szempont miatt a Matlab környezetet választottam. Mellette szólt, hogy rendelkeztem korábbi tapasztalattal a programról, széleskörű és kiválóan támogatott adatfeldolgozási képességei vannak, továbbá a TinyOS-szel is biztosított a kompatibilitása.

A nyelv vektor alapú, segítségével könnyen írhatunk matematikai és adatfeldolgozó alkalmazásokat. Támogat továbbá toolboxokat, amelyek által különböző adatok megjelenítése, szűrők tervezése, gépi tanulás, statisztikai analízis, stb. válik lehetőség a programban. Szkriptek írására is lehetőséget biztosít, így nagy programok

esetén akár jobban használható, mint egy objektum orientált nyelv. Közvetlen parancsok segítségével is beavatkozhatunk például egy szenzorhálózat működésébe.

A Matlabban futó programról a rendszer működésének ismertetésénél részletesebben is beszámolok.

5 Az adatgyűjtő hálózat

A fejezetben bemutatom a kommunikációs közegeket és tulajdonságaikat, majd az adatgyűjtő hálózat működését, amely a későbbi helymeghatározó rendszer alapjául szolgált.

5.1 Kommunikációs közegek

A rendelkezésre álló hardver tartalmaz egy rádiós kommunikációra képes chipet, ám adódnak más lehetőségek is.

Rádiós átvitel tulajdonságai

A rádiós átviteli frekvenciák néhány kHz-től egészen a többszáz GHz-es tartományig terjednek. Bizonyos frekvencián könnyen áthatol rengeteg közegen, emiatt pl. tengeralattjáróknál is használatosak.

Infravörös tartomány hullámhossza a látható fény hullámhossza fölé esik. Az erre épülő IrDA (Infrared Data Association) hátránya, hogy az eszközöknek közvetlen rálátással szükséges rendelkezniük a kommunikációhoz, emiatt közel kell őket helyezni, illetve a sebességbeli fejlődésre is szükség lenne. Ilyen technológiával szerelték fel pl. az Olivetti ActiveBadge eszközé.

Az ultrahang terjedési sebessége ~340 m/s, amely nagyságrendekkel lassabb a fény és néhány elektromágneses hullám terjedésénél. A frekvencia 40 kHz körül van, további tulajdonságai, hogy egyszerre egy adó adhat, több szenzorra van szükségünk, a hatótávolság viszonylag kicsi, és elfogadható a pontosság. A falakon nem képes áthatolni. Alkalmazták pl. az Olivetti ActiveBat termékben.

Zigbee technológia

A vezeték nélküli technológiák manapság fejlődés szempontjából sebességorientáltak. Az energiafelhasználás csökkentése nem az elsődleges cél, ennek ellenére kifejezetten fontos szerepet tölt be főleg a mobilis eszközök – mint például a vezeték nélküli szenzorhálózatok – területén. Az egyik napjainkban is használt rádiós vezeték nélküli technológia a Zigbee, melynek talán legnagyobb előnye az energiatakarékosságában rejlik.

A Zigbee a szenzorhálózatokat, házi automatizálást és egyéb alacsony intelligenciájú csomópontokból álló hálózatok kialakítását támogatja. Zigbee kis adatcsomagok átvitelét támogatja nagyméretű, statikus felépítésű, rendszertelenül működő eszközökkel teli hálózatban.

A technológiát egy Zigbee Alliance nevű szervezet fejleszti. A társulásnak több neves ipari partner is tagja, mint például a Honeywell, Motorola, Mitshubishi, Philips, összesen 35 cég. A fejlesztés célja kis adatfogalmú és fogyasztású, olcsó eszközök létrehozása. A hálózati és alkalmazási réteg feladatit szem előtt tartva folyik a fejlesztés, a fizikai és közegelérési rétegét az IEEE dolgozta ki, és a 802.15.4 szabványban rögzítette.

Felhasználási területei ipari és kereskedelmi alkalmazások (pl. monitorozás, érzékelés, automatizálás, vezérlés), egészségügyi alkalmazások (pl. érzékelés, diagnosztika), szórakoztató elektronika (pl. tévék, videók, távirányítók, játékok), számítógép perifériák, vagy a házi automatizálás (pl. biztonság, világítás, hőszabályozás).

Rádiós interfész

Az IEEE 802.15.4 frekvenciatartományai az alábbi táblázatban láthatók.

	Sáv	Lefedettség	Adatátviteli sebesség (kbps)	Csatornák száma	Moduláci ós eljárás	Chip sebesség (kchip/s)	Szimbólum sebesség (ksymbol/s)
2.4 GHz	ISM	Világ	250	16	Q-QPSK	2000	62.5
868 MHz		Európa	20	1	BPSK	300	20
915 MHz	ISM	Amerika	40	10	BPSK	600	40

1. táblázat: Az IEEE 802.15.4 frekvenciatartományai [2]

A fizikai szinten elérhető sebesség 20 és 250 kbps közt mozog. Az interferenciák ellen direkt szekvenciális spektrumszórást (*Direct Sequenced Spread Spectrum,* DSSS) alkalmaz, mely 250 kbps-os sebességnél 62500 szimbólumváltást eredményez másodpercenként. A technológia 2,4 GHz-en 11 darab átlapolódó 83 MHz-es csatornát használ, mindegyiket 3 darab 22 MHz-es nem egymásra lapolódó csatornaként. Így DSSS nagyobb adatátviteli sebességet képes produkálni frekvenciaugrásos megoldásoknál.

Alkalmazott metódus továbbá a CSMA/CA, azaz minden adás előtt vivő érzékeléses, többszörös hozzáférést kezelő, ütközést elkerülő algoritmus, melyet az esetleges ütközések kezelése érdekében használnak.

A maximális hatótávolság 10-75 m, amely erősen függ a környezettől.
A protokoll képes lefoglalt időrésekkel garantálni az időkritikus alkalmazások igényeit, emellett kézfogásos (handshake) üzenettovábbításra is lehetőséget ad.

Összehasonlításképpen a folyamatos működés egy 802.11 típusú rádiós interfészhez képest (667 mW) kevesebb, mint huszadannyi energiát (30 mW) igényel, amelyhez még hozzá kell számolni a kis időbeli kihasználtságot.

Egy hálózatban részvevő node-ok az általam használt környezetben a 16 bites cím miatt maximum 65536-an lehetnek, amely az esetleges foglalt címek miatt csökkenhet – pl. a TinyOS az 0xFF címet alkalmazza az UART-hoz.

A hálózati forgalom szempontjából három fajta eszközt különböztetünk meg. Periodikus adatforgalmat bonyolító eszközök, melyek rendszeresen küldenek adatot (pl. szenzorok), rendszertelen adatforgalmú eszközök, melyek valamilyen külső hatás alapján dolgoznak (pl. villanykapcsoló), a harmadik típusba az alacsony késleltetést igénylő eszközök sorolhatóak, melyek kihasználhatják a rendszer által biztosított garantált időrés opciót (pl. egerek). A Zigbee protokollveremnek több változata létezik, ezek közül a nagyobb funkcionalitású verzió önmagában 32 kbyte, az kisebb pedig 8 kbyte memóriát igényel.

Az imént említett hálózati topológiának köszönhetően egy entitás akár 30 ms alatt beépülhet egy hálózatba, mivel alvó állapotból 15 ms-ra van szükség az aktív állapotba kerüléshez, míg egy aktív egységnek átlagosan 15 ms-ra van szüksége ahhoz, hogy kommunikációs csatornához jusson.

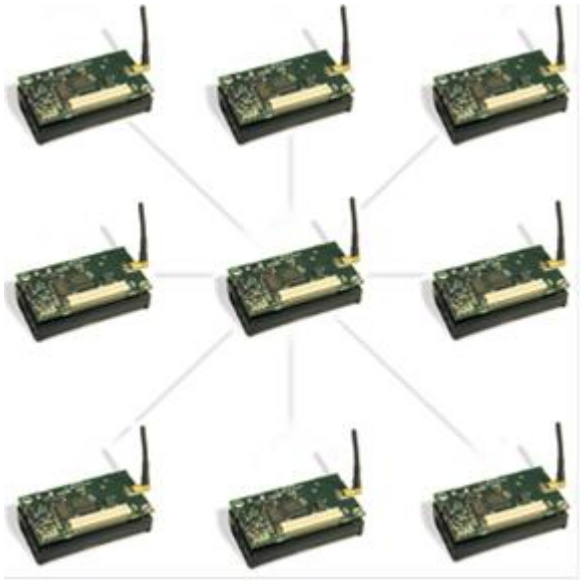
Biztonsági kérdések

A parancsok, beacon üzenetek és visszaigazolások titkosítására a Zigbee MAC szintű titkosítást kínál, egy ugrásnyi (hop) távolságnál nagyobb esetben azonban a felsőbb rétegek biztonságára támaszkodik. A MAC szint a továbbfejlesztett titkosítási szabvány (Advanced Encryption Standard, AES) nevű kriptográfiai algoritmust használja, és sok különböző biztonsági csomagot definiál, melyek az AES algoritmusra épülnek. Ezek a biztonsági csomagok támogatják a MAC keretek bizalmasságát, integritását és hitelességét. Habár a biztonsági feldolgozást a MAC szint végzi, a felsőbb rétegek állítják elő a biztonsági kulcsokat és határozzák meg az adott esetben használandó biztonsági szinteket. Amikor a MAC szint továbbít (fogad) egy titkosított csomagot, megnézi a keret célcímét (forráscímét), leellenőrzi a címhez hozzárendelt kulcsot, majd eme kulcsot használja a keret

feldolgozásához, a kulcshoz rendelt biztonsági csomag alapján. A keretekben egy bit jelzi a titkosítás használatát. [2]

Hálózattípusok

Az elkészült lokalizációs rendszer típusa hálózati szempontból nem határozható meg egyértelműen. Adatgyűjtés szempontjából csillag, viszont bizonyos üzemmódban – triggerelés esetén – ún. mesh topológiáról beszélhetünk.



13. ábra: A csillag topológia

A csillag topológia egy hálózattípus, amelyben létezik egy kitüntetett, központi résztvevő, amely a többivel kapcsolatban áll. Esetemben ez a bázis, amely a lekérdezési szakaszban irányítja a működést, azaz a bázis kérdezi le az összes szenzort, akik csupán a kérdésre válaszolnak.

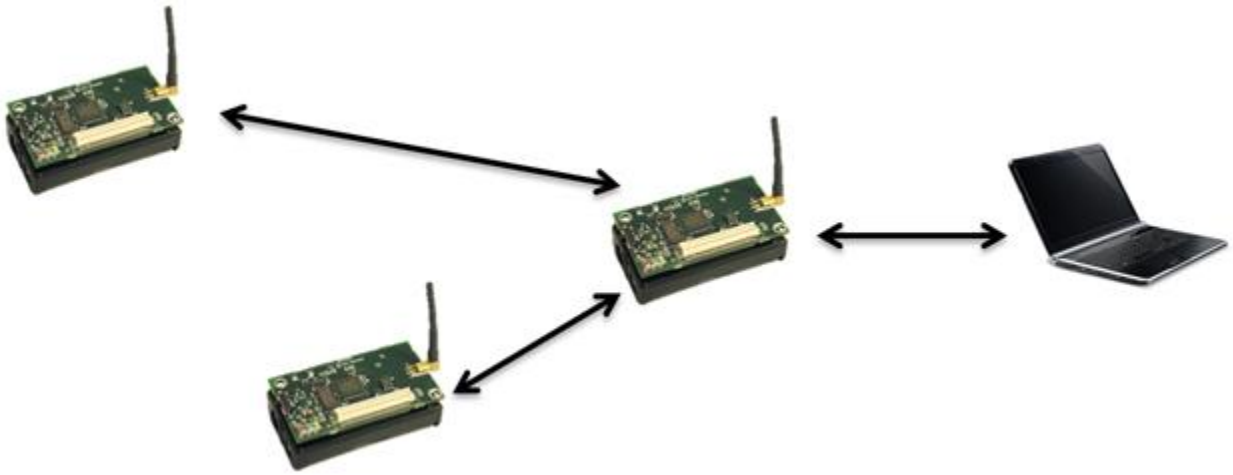
A mesh egy olyan hálózattípus, ahol minden egyes node különálló routerként viselkedik, általában nincs egy központi node, amely irányítja a hálózatot. Kitüntetett feladatot elláthat egy-egy résztvevő, de a lényegi különbség a node-ok közti kapcsolatoknál keresendő. Az ilyen típusú hálózatban bármely node kapcsolódhat egy másik hálózatban szereplő node-hoz. Az olyan mesh hálózat, melyben minden node kapcsolódik egymáshoz, ún. teljesen csatlakoztatott hálózat, mely megvalósulhat több ugráson keresztüli kapcsolatok formájában is. A mesh hálózatok legtöbbször vezeték nélküli hálózatoknál használatosak, éppúgy alkalmazhatók vezetékes megvalósításnál is. A hálózat egy résztvevő kiesése vagy csatlakozási problémák esetén is

14. ábra: A mesh topológia

működőképes marad. A node-ok közt legtöbbször nem csak egy út létezik, így a hálózat igencsak megbízhatónak számít. A mesh hálózatok eredetileg katonai célból jöttek létre.

5.2 Az adatgyűjtő hálózat felépítése

Az alábbi alfejezetben ismertetem az adatgyűjtő hálózat felépítését és működését, amely a későbbi helymeghatározó rendszer alapjául szolgált. A hálózat a bázison keresztül (a gateway board segítségével) csatlakozott a PC-hez.



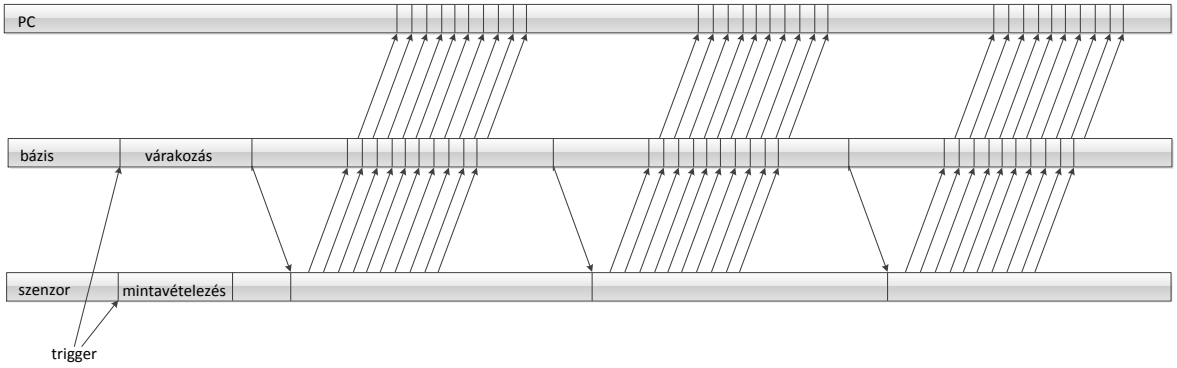
15. ábra: Az adatgyűjtő hálózat felépítése

A hálózat felépítése a fenti ábrán látható, ahol a PC-hez kapcsolódó mote a bázis, a többi pedig a szenzormote.

5.2.1 Működés

A munka célja a későbbiekben az akusztikus módon történő helymeghatározás megvalósítása egy hálózatban. Ehhez először egy adatgyűjtő hálózatot építettem ki, mely működése alapján egy PC-hez kapcsolódó csillag topológiát reprezentáló összeállítás. A résztvevők a PC, a bázis és a szenzor mote-ok. A szenzor mote-ok szerepe az adatgyűjtés, mely ebben az esetben bármilyen szenzor adat lehet. A bázis, amely a PC és a hálózat között teremt kapcsolatot, irányítja a kommunikációt, lekérdezéses módon szerzi be az adatokat, a számítógép pedig ezen adatokat jeleníti meg egy programban. A teszthálózat első lépésben kisszámú mote-ra valósult meg. Ez megkönnyítette a tesztelést, ezen kívül a fejlesztés alatt szem előtt tartottam a későbbi hálózatbővítést. Az alábbi ábrán a lokalizációs hálózatban

megvalósult idődiagram látható, amelynek adatgyűjtő része az itt létrejött metódus alapján működik. A hálózatban a bázis kérdését követően tetszőleges számú válaszüzenetet programozhatunk.



16. ábra: A szenzorok lekérdezésének menete egy szenzorra vonatkozóan

A bázis

A fentebb említettek alapján a bázisnak a hálózatban irányítania kell. A rajta futó program más, mint a szenzor mote-okon. A bázis jelenlegi esetben egy időzített lekérdezéssel gyűjti az adatokat. A program indulásakor elindul egy időzítő, amely lejárta indikálja a lekérdezés megkezdését. A kiküldött csomag, amelyre a szenzor mote-ok válaszolnak majd, a hálózatban részt vevő mote-ok azonosítója alapján címez. A válaszban megkapott csomagokat leellenőrzi, majd soros porton továbbítja a PC felé. A soros porton küldendő üzenet ugyanolyan felépítésű, mint egy rádióüzenet, csupán a címben különböznek.

A bázis szoftverének elkészítésekor szem előtt tartottam, hogy a bázisnak az első verzióban nem kell szenzorokat kezelnie, viszont a kommunikáció irányítása és a pc-re történő továbbítás miatt szinte folyamatosan működnie kell, így ésszerűen a programozó board tápjáról működik. Ezáltal nincs szükség az energiatakarékosság magas prioritással történő kezelésére.

A szenzorok

A szenzorok működése eseményvezérelt, ez esetben a beérkezett rádióüzenetekre válaszolva küldik a szenzoradatokat. A mote-okon szintén működik egy időzítő, mely

másodpercenként lekérdezi a kívánt adatokat, majd ezekkel frissíti a kiküldendő üzenetet. A beérkezett utasítás hatására így megindulhat a küldés, amelyben a legfrissebb adatok találhatók. Az adatok védelme érdekében bizonyos műveleteket atomic típusúként használok, így ezeknél nincs megszakításkezelés.

A számítógép

A PC-n egy adatok megjelenítésére alkalmas programban nyomon követhetjük a szenzor mote-ok adatainak alakulását, emellett ellenőrizhetjük azokat. A későbbiekben a jelfeldolgozási igényeknek megfelelően áttérek a Matlab környezetre.

Tesztelés

A hálózat felépítése után a tesztelés következett. A rendelkezésre álló hardverek sajátosságai miatt a teszteléshez a következő segédeszközöket használtam: a mote-okon található LED-eket és a számítógépet, mely a beérkezett üzeneteket jeleníti meg. Ezek alapján jól látható, hogy a programok tesztelése lassú és nehézkes, mivel például egy program működésének debugolása a LED-ekkel történhet (ha valamilyen oknál fogva nem működik a soros porton történő üzenetküldés), ezt pedig többszöri újraprogramozás kíséri az egyes apróbb módosítások után.

Az adatgyűjtő hálózat megvalósítása után továbbléphetek a végső feladatra, a lokalizációs rendszer felépítésére. Ehhez szükséges a szinkronizációs és a lokalizációs lehetőségek tanulmányozása, majd megvalósítása, továbbá a Matlab program és a jócskán kibővített bázis- és szenzorprogram elkészítése.

6 Szinkronizáció

A lehető legpontosabb helymeghatározáshoz a különböző mote-ok óráinak szinkronizálására van szükség. Ehhez a rendelkezésre álló kommunikációs csatornán kell megtalálni az egyensúlyt a pontos szinkron és a szükséges üzenetek mennyisége közt. A későbbiekben a többi mérlegelendő szempontot is vázolom, amelyek a működést befolyásolhatják.

6.1 Szinkronizáció alapjai

A szinkronizáció alapja az időmérés, mellyel feladatokat hangolhatunk össze, vagy mérhetünk jelenségeket. Az időméréshez szükséges időalapokat szolgáltató eszközök az egyes rendszerekben különbözőek lehetnek, felhasználási területet tekintve létezik globális és lokális.

A szenzorhálózatok szinkronizációját meghatározza a felépítés, az erőforrás korlátja és az eltérő követelmények. A felépítés ad-hoc jellegű, melynek lényege, hogy a hálózat kiépítéséhez nem szükséges infrastruktúra, szinte tetszőleges helyen létrehozható és a hálózat elemei többnyire vezeték nélküli technológiával kapcsolódnak egymáshoz. Az ilyen hálózatokban a pontosság tág korlátok közt mozoghat, ami jó esetben μsec-sec tartományt jelent. A működés közben legtöbbször nincs lehetőség és igény sem az emberi felügyeletre. Az erőforrás korlátos az energia, számítási kapacitás, memória és kommunikáció területén is. Egy központi probléma az energiafelhasználás, annak optimalizálása és minimalizálása.

Ezzel szemben a hagyományos rendszerek szinkronizációja fix felépítésű, nem jellemző az erőforrásprobléma, így az egyes csomópontok folyamatosan működhetnek, tudnak kéréseket fogadni és a hálózat folyamatosan használható. További jellemző, hogy általános szolgáltatásról beszélhetünk széleskörű felhasználói rétegnek. A pontosság ~100 μs körül alakul, emellett emberi felügyelet is biztosítható. [21]

Időalapok

Általánosan elterjedtek a globális időalapok, amelyek pl. a számítógép, vagy a mobiltelefonok idejének alapját is szolgáltatják. Több globálisan elterjedt rendszernek az időméréshez szükséges mennyisége 1 másodperc (amely a 133-as tömegszámú Cézium izotóp ún. hiperfinom átmeneti rezgési periódusidejének 9 192 613 770-szerese). Ilyenek pl. a Föld keringési idejéhez kapcsolódó Universal Time (UT) – a GMT modern megfelelője –, és a Coordinated Universal Time (UTC). Létezik továbbá az International Atomic Time (TAI – a francia megfelelőből). Az időalapok által hordozott időinformáció lehetséges forrásai pl. az UTC-hez szükséges információt biztosító GPS (Global Positioning System) és a DCF77 (D: Deutschland, C: long wave signal, F: Frankfurt, 77=77.5 kHz) rádiós órareferencia is, mely Frankfurt mellől sugározza az atomidő szinkronjelét.

Az általános időalapot a beágyazott rendszerekben sokszor egy kvarcoszcillátor biztosítja. Az ilyen rendszerekben alternatívaként megjelenő RC oszcillátorokkal összehasonlítva nagyságrendekkel jobb, használatával maximum kb. 100 ppm-es abszolút hiba érhető el. Áruk és méretük tekintetében az RC oszcillátorok kedvezőbbek, ám a kvarcoszcillátorok jobb ár/pontosság arányuk miatt a legtöbb felhasználásban elsőbbséget élveznek.

A kvarcok előnyei viszonylagos pontosság, az ár és a méret. Pontosságukat tekintve beszélhetünk rövid és hosszútávú stabilitásról. A rövid távú stabilitást befolyásolhatja a hőmérséklet, a rezgés, a mágneses hatás és a tápfeszültség, a hosszú távú függ a gyártástól és az öregedéstől. Magasabb árú termék általában nagyobb pontosságot nyújt, amelyet a technológia és a fogyasztás is javíthat (pl. fűtött, vagy termosztátos kvarc). Általánosságban elmondható, hogy a pontosabb kvarc esetén ritkább szinkronizáció szükséges, ám semmi esetre sem hanyagolható.

Mozgó objektumok esetén bonyolultabb szinkronizációra lenne szükség, mivel ebben az esetben az elmozdulás okozta időtorzítást is figyelembe kellene venni. Az általam alkalmazott hálózatban ez nem fordulhat elő.

Óramodell

Az alkalmazott óramodell néhány fontosabb jellemzője a következő. Az időmérés diszkrét felbontású, számlálós megvalósítású, az idő léptethető vagy felülírható. A modellben

szerepel a referencia idő – amelyhez viszonyítunk – és az egyes node-ok lokális ideje. Ezen adatokkal megvalósítható egy, a felhasználáshoz szükséges szinkronizáció.

A szinkronizáció célja, hogy a hálózatban található összes óra értéke minél kisebb intervallumon belül legyen a referenciaidőhöz képest.

További cél, hogy konzisztens becslést adjon az események sorrendiségére, azaz legyen az egyes események sorrendje egyértelműen meghatározható. Amennyiben ez nem lehetséges, arról is legyen információnk. A kommunikáció ideje lehetőleg legyen minimális és jól kezelhető.

A fenti célok a hálózatomban a következőképpen teljesülnek. A szinkronizáció egy referenciaidőhöz képest kell, hogy megvalósuljon. Ez a hálózatban szereplő bázis idejét jelenti, amely biztosítja a referenciaidőt. A sorrendiség biztosított, amelyet egy későbbi fejezetben részletezek. A kommunikáció ideje jól kezelhető, és nagyságrendi különbségről beszélhetünk a mérendő esemény idejéhez képest, amely enyhíti a követelményeket.

A szinkronizációt egységesre tervezve a szenzornode-ok azonos program futtatása esetén minimális különbségekkel működhetnek. Üzenet beérkezése esetén, ha nem lép fel hiba, akkor ezek a különbségek a processzor órajelének nagyságrendjébe esnek.

A pontosságot befolyásolhatják a különböző hibák, és a programok működéséből adódó késleltetés. Hiba többek közt a szinkronizációs üzenet kiesése, vagy feldolgozásának menete közben történő késleltetés. Az üzenet elveszhet, ha a rádiós jelterjedés közege nem megfelelő. A későbbi rendszer beltéri felhasználása előtt emiatt szükséges méréseket végezni az adott környezetben. A teszt elsősorban a rádiós kommunikáció különböző adóteljesítménnyel történő csomagkiesések vizsgálata, amellyel könnyen megállapíthatunk egy biztonságos működési szintet. További lehetőség a többszörös szinkronizációs üzenetek használata, amely esetén szintén kiemelten figyelni kell az esetleges kiesett csomagokra.

A feldolgozásból adódó késleltetésre a szoftverben kell felkészülnünk. Esetemben szinkronizációs üzenetek beérkezése után történő szinkronizáció az adott esemény első eleme, amely védve van a különböző megszakítások ellen, továbbá biztosított a biztonságos kommunikációs távolság.

Szinkronizáció

A szinkronizáció típusainak több szempont szerinti csoportosítása létezik. Adattovábbítástól való függetlenség szempontjából beszélhetünk implicit és explicit

szinkronizációról. Az implicit szinkronizációnál az adattovábbítással történik a szinkronizáció. Előnye, hogy nem jelent nagy terhet, viszont ritka üzenetváltásnál a pontosság romlik. Az explicit módszer lényege, hogy szinkronizációs üzeneteket alkalmazunk, amelyek az adatáramlástól függetlenül kezelhetők. Előnye, hogy nem igényel folyamatos adatáramlást, ellenben a szinkronizáció extra terhet jelent a hálózat számára, csatornafoglaltság és erőforrás szempontjából. A hálózatomban implicit szinkronizációról beszélhetünk.

A szinkronizáció lehet globális és lokális. Globális szinkronizációnál egy globális időreferenciához szinkronizálunk (pl. UTC-hez, GPS segítségével), jellemzően a hálózat élettartamán túl is rendelkezésre álló adatokról beszélhetünk – pl. egy adatbázisban rögzített eseményhalmazról. A lokális vagy belső szinkronizációnál az egységek egymáshoz képest szinkronizáltak, és nincs információnk egy globális időreferenciához képesti helyzetről.

A tesztelt hálózatban belső szinkronizáció került implementálásra, mivel a működés nem követelte meg a globális időinformációt.

6.2 Megvalósított szinkronizáció, tesztek

Az elsődleges szempont a szinkronizáció minősége volt, amelyre vonatkozó tesztek kimutatták, hogy az implementálni kívánt algoritmus megfelelő minőséget képes biztosítani. A tesztek az egyes oszcillátorok eltérésének mérésére irányultak, amely alapján becsülhető, hogy milyen sűrű szinkronizáció szükséges a lokalizáció pontosságának megfelelő szinten tartásához.

Órák eltérésének mérése

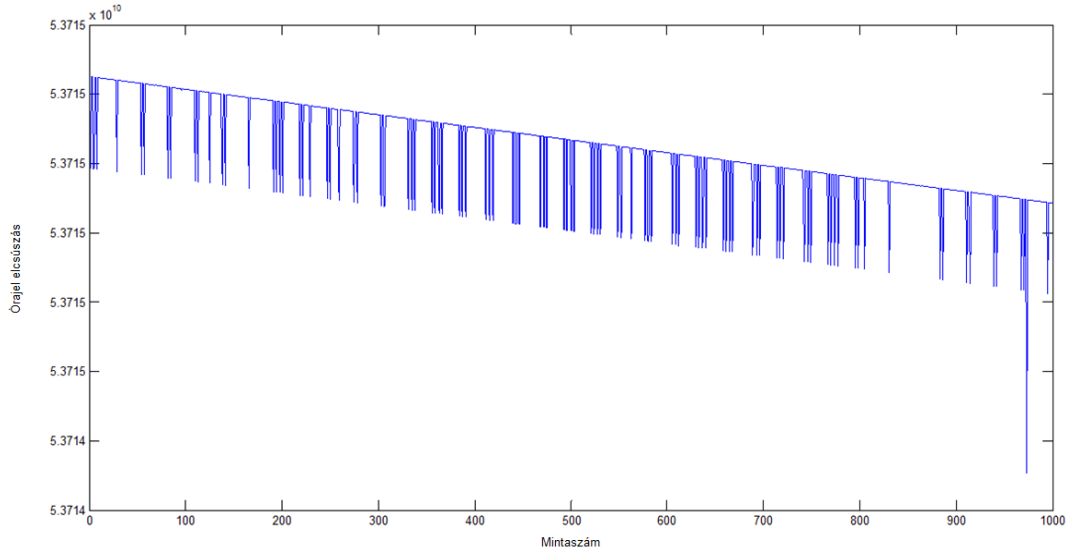
A tesztek alapján a referenciaidőhöz képesti eltérést kívántam mérni. A teszt menete nagy vonalakban: a referenciaidőt mérő node bizonyos fix időközönkénti eseményhez egy lekérdezést fűzök az egyik vizsgált node irányába, amely hatására a lekérdezett node az érkezett órainformáció mellé fűzi a saját óraállását, így ezt eljuttatva a számítógépre kiértékelhetővé válik az adat. A mérés állandó hőmérsékleten (szobahőmérsékleten) zajlott.

A teszthez a Zigbee szabványból adódó működésbe is be kellett avatkozni. A korábbiakban leírtam az ún. CSMA algoritmust, amely lényege a rádiós kommunikáció ütközéseinek elkerülése. A CSMA a rádiós üzenet kiküldése előtt véletlen hosszú ideig vár, majd belehallgat a csatornába, amely ha szabad, kiküldi az üzenetet. Az ilyen működés

elrontja a szinkronizációt. A pontosításhoz a véletlen hosszú ideig történő várakozásért felelő ún. backoff timer-t kellett kikapcsolni, mely alsóbb szinten történő beavatkozás során valósult meg. Az így létrejött kommunikáció alkalmas volt szinkronizációra, ám ebben az esetben kiemelten figyelni kell az esetleges ütközésekre.

A rendelkezésre álló oszcillátorok 10 ppm hibával működnek, amely azt jelenti, hogy 100 s alatt 1 ms lehet az elcsúszás. A szobahőmérsékleten vett hangterjedési sebességgel számolva: 1 ms * 340 m/s = 34 cm, ha az időzítéseket nem vesszük figyelembe.

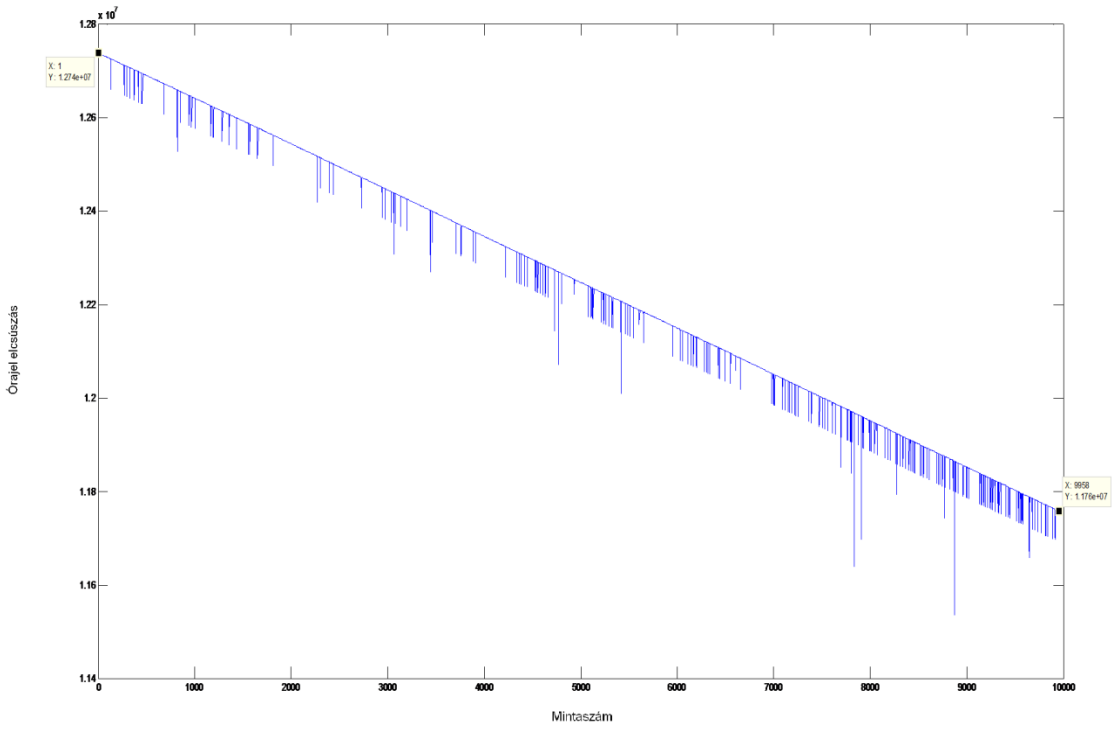
A tesztek megmutatták, hogy az időalapokat tekintve egymáshoz képesti egyirányú elcsúszásról beszélhetünk, azaz az ingadozás kismértékű. Emiatt az ilyenfajta elcsúszásokat a későbbiekben az órák lassításával, vagy gyorsításával korrigálhatjuk. Erre az egyes órák állandó hőmérsékleten vizsgált, egymáshoz képesti elcsúszásnak ismeretében elméletben akár a számítógépen is lehetőségünk van, és időbélyeg konverziónak nevezzük.



17. ábra: Az oszcillátorok elcsúszása nem szinkronizált esetben
(y tengely: Órajel elcsúszás – órajelben, x tengely: Mintaszám – db-ban)

A fenti ábrán az oszcillátorok elcsúszására vonatkozó teszt látható, amely 1000 mintaalapján látható csúszást mutat – a minták kb. 1 sec gyakorisággal érkeztek. A mérésből látható továbbá egy állandó hiba, amely az ábrán tüskék formájában mutatkozik meg. Az oszcillátorok ilyen jellegű hibájának a program tanulmányozása, majd oszcilloszkópos

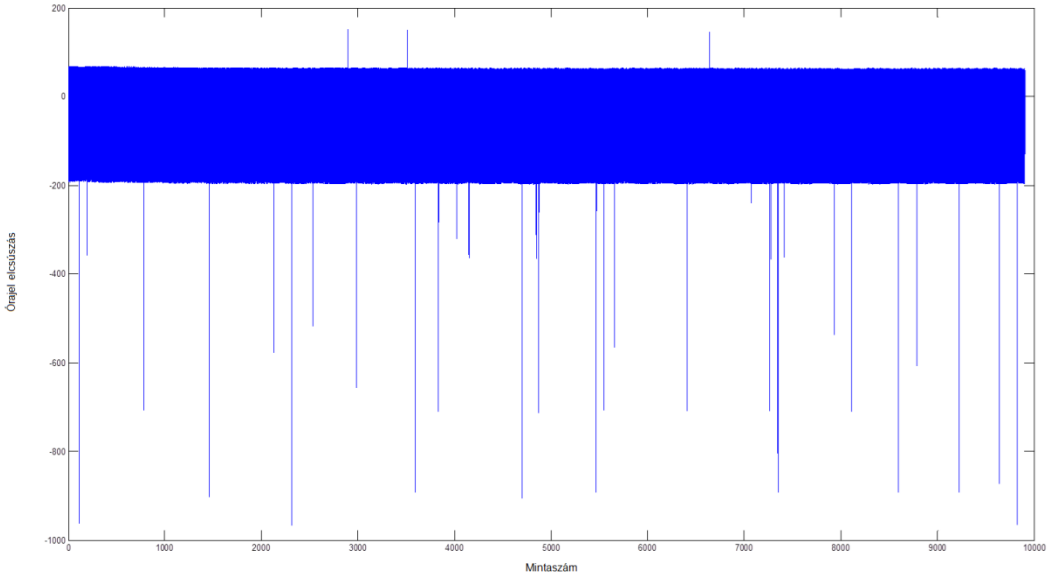
mérések után sem derült fény az okára. Az ábráról nem látszik egyértelműen, de ez az ingadozás százalékosan kismértékű, ám állandó jelleggel megjelenő torzító tényező. Az elcsúszásról elmondható, hogy állandó hőmérsékleten az egyes oszcillátorok közt nagyrészt egyirányú elcsúszásról beszélhetünk, amely folyamatos szinkronizáció és ehhez mérten nem túl hosszú hangminták felvétele esetén nem okoz nagy mérési hibát. A mérési eredmények közt ezt a rendszer működése által is láthatjuk.



18. ábra: Az oszcillátorok elcsúszása nem szinkronizált esetben
(y tengely: Órajel elcsúszás – órajelben, x tengely: Mintaszám – db-ban)

A fenti ábrán egy közel 10000 mintából álló mérés eredménye látható, amelyből a szélső minták eltérése ~ 1000000, amely órajelben értendő. Az időt ebből megkaphatjuk, ha az oszcillátor 7,3728 MHz-es órajelét elosztjuk ezzel. Az eredmény kb. 140 ms, amely a mérés több, mint 20 perces időtartama alatt történt. Az akusztikus lokalizációnál ez az idő akár 40 méteres hibát is okozhatna, ebből is látszik, hogy mennyire fontos a szinkronizáció.

A szinkronizált esetben az eredmény a következő.



19. ábra: Az oszcillátorok elcsúszása szinkronizált esetben
(y tengely: Órajel elcsúszás – órajelben, x tengely: Mintaszám – db-ban)

A fenti ábrán látható, hogy az átlagos órajel ingadozás nagyságrendileg 300 órajel periódus, amely időben kb. 40 µs, tehát hangterjedésben hozzávetőlegesen 14 mm-t jelent.

Megvalósított szinkronizáció

A kommunikációs struktúrám egyszintű volt, így lehetőség adódott a központi szinkronizációra, melynek itteni jelentése a következő: a bázis mote minden más, a hálózatban szereplő mote-tal közvetlen kapcsolatba léphet. Ennek a feltétele, hogy az adott környezetben elérhető rádiós hatótávon belüli node-ok alkossák a hálózatot. Esetemben ez az alkalmazásnál biztosítva lesz, mivel a beltér (szoba, lakás) mérete ennek megfelelő.

A szinkronizáció a tesztelt hálózatban egy folyamatos, bázis által irányított belső szinkronizációként került implementálásra. A módszer lényege, hogy bizonyos időközönként a bázis szinkronizációs üzenetet küld a szenzoroknak, amelyet a megfelelő változók beállítása és a hardveres óra újraindítása követ. A működéstől függően lehetőségünk van energiatakarékosság miatt a szinkronizációt csupán a mintavétel elejére korlátozni. Ebben az esetben a trigger üzenettel egyidőben történik szinkronizáció.

A szinkronizációhoz szükség volt a hardveres óra használatára, amely az adott környezetben a lehető legpontosabb időalapot képes biztosítani.

Esetemben fejlesztési fázistól függően létezik implicit és egy speciális, implicithez hasonló szinkronizáció. Előbbivel a hálózatbeli bázis fix időközönként szinkronizációs üzeneteket küld a szenzoroknak. Ebben az esetben a triggerjelet az egyik szenzortól várjuk, amely a mikrofonjel beállított értékének átlépése esetén érkezik, és bizonyos számú, triggerjel előtti hangmintát is feldolgozunk. Utóbbi eset a működés biztonsága és a fogyasztás szempontjából is előnyösebb. A módszer lényege, hogy a szinkron akkor történik, ha a hálózatban esemény indul, így az ún. triggereseményt használom szinkronizációs üzenetként. Feldolgozáskor csak az ettől a pillanattól kezdődően rögzített hangok állnak rendelkezésre. A biztonság abban mutatkozik meg, hogy a szinkronizáció miatt nem használatos a rádiós ütközéselkerülő algoritmushoz tartozó véletlen időtartamú időzítő, ezért erre fokozottan figyelni kell. Ebben az esetben a bázis csak a triggeresemény után kérdezheti le a node-okat, így biztosított az ütközéselkerülés. A rendszerben az utóbbi metódus használatos.

7 Lokalizáció

A fejezetben bemutatom a főbb lokalizációs technikákat, azok tulajdonságait, a kiválasztott technikát és előnyeit. A célom egy akusztikai alapon működő helymeghatározó rendszer létrehozása. Ennek fényében választom ki és implementálom a megfelelő metódust.

7.1 Lokalizáció szenzorhálózatokban

A szenzorhálózatokban történő lokalizáció több felhasználáshoz elengedhetetlen, vegyük például az intelligens otthonok működését, a rendszerfelügyeletet, vagy a környezetmonitorozást. Sokszor a fizikai helyre vagyunk kíváncsiak, de előfordulhatnak olyan esetek is, melyeknél a szimbolikus helyről szükséges információ. Ez kevésbé pontos, lényege, hogy valamilyen információt társítunk egy eszköz helyzetéhez. Ilyen például egy jármű haladási iránya, a Bécsbe tartó vonat, vagy a szenzor a ház mely szobájában található. [13]

A helytudatos működéssel megtakarítható energia is, mivel így nincs szükség útvonal felderítésre és többszintű hálózatokban az útvonalválasztás is egyszerűbb lehet.

A helymeghatározáshoz kapcsolódóan beszélhetünk abszolút és relatív pozícióról. Abszolút esetben ismert az egyes eszközök földrajzi helyzete, míg relatív helyzetben az egymáshoz viszonyított távolságokról van információnk. Ha rendelkezünk a megfelelő információkkal a hálózatról, az abszolút és a relatív pozíció átszámítható egymásba.

A relatív pozíció a viszonyítási alap node-onként különböző megválasztására ad lehetőséget. A relatív pozícióra épülő módszert használhatunk például akusztikai helymeghatározáshoz, melyről beszámolok lentebb. Ehhez tudnunk kell a node-ok elhelyezkedését, amely esetemben ismert. Beltéri alkalmazásoknál túlnyomó többségben a relatív pozícióra van szükségünk.

Az abszolút pozícióra sok esetben nincs szükség, ám ha mégis, akkor lehetőségünk van a GPS (Global Positioning System) használatára. A rendszer működéséért műholdak felelnek, amelyeken egymással szinkronizált atomórák, stabilizációs és navigációs egységek, valamint a földi és műholdközi kommunikációért felelős berendezések vannak. Előnyei az elterjedtség, pontosság, ellenben ára magas - leginkább a szenzorhálózat node-jainak más

alkotóelemeihez képest. A GPS akár méteres pontossággal képes meghatározni helyünket, de ehhez több műholdra kell rálátással rendelkeznünk, amely beltérben sokszor nem biztosítható. [12]

Az abszolút pozíciót fix telepítésű szenzorhálózat esetén egyszeri kalibrációval is megállapíthatjuk, így az egyes node-ok helyzete ismert, és a hálózat működése alatt mindvégig pontos maradhat. Az ilyen jellegű mérések ideiglenesen felszerelt bővítőkártyákkal is végezhetők a költség- és fogyasztáscsökkentés jegyében. Nem fix felépítésű szenzorhálózat esetében bizonyos események hatására történhet a frissítés, amelyet indikálhat lokális elmozdulás észlelése, bizonyos idő eltelte, stb..

A pontosság alkalmazásfüggő, szemléltetésként vegyük a következő kérdéseket. Melyik földrészen telelnek a magyarországi költöző madarak? Melyik teremben lesz a diplomavédésem? Hol van a kulcsom (köbcentiméter pontossággal)? Általában elmondható, hogy a pontosság csökkentésével nagyobb megbízhatóság érhető el.

A lokalizáció referenciapontok, és távolságmérés segítségével oldható meg. Az általános problémák, a precíz méréshez speciális hardver szükségesessége és a hálózati topológiától való függés. A helymeghatározás lehet centralizált – központi helyen történő pozíciószámítás – vagy elosztott – azaz az egyes node-ok saját helyzetüket számítják. Az elosztott rendszereknél alkalmazott módszerek hatókör alapú és hatókör nélküli kategóriákra oszthatók. A hatókör alapú megoldások érkezési idők, a vett jel erőssége, két különböző jel érkezési ideje és irányszög mérés alapján működhetnek. A hatókör nélküliek hop számon alapuló vagy referenciapontok segítségével történő számítás segítségével valósíthatók meg.

Alapvető távolságmérési technikák

- Akusztikus: Egy hangfrekvenciás és egy rádióhullámú jelet egyidejűleg elküldve, a vevőhöz érkezés időkülönbségéből tudjuk becsülni az adó és a vevő távolságát.
- Rádióhullám interferenciás: Két adóra és két vevőre van szükségünk, majd a két adó vivőfrekvenciájának változtatásával a vevőknél fellépő jel relatív fázistolásból következtethetünk a távolságokra.
- Rádiós jelerősség alapú: A becslés a vett jel erősségéből történik.

- Centroid módszer: A hallható referenciapontok középpontjába pozícionálással határozzuk meg az egyes node-ok helyzetét, amelyhez a referenciapontoknak egyenletesen sűrűn kell elhelyezkedniük.
- DV-hop: Minden node nyilvántartja az út hosszát (hop-szám) minden általa ismert referenciaponthoz. Szükséges az úthosszak hirdetése a hálózaton belül, és nem szükséges sűrűn elhelyezett referenciapontok megléte.

7.1.1 A helymeghatározási technikák

A helymeghatározási technikák nagyobb csoportjai:

- irányszögelés (Angle Of Arrival, AOA)
- késleltetés érzékelés (Time (Difference) Of Arrival, TOA, TDOA)
- jelerősség érzékelés (Radio Signal Strength Indication, RSSI vagy RSS)
- bithibaarány mérés (Bit Error Ratio, BER)

Irányszögelés - AOA

A metódus lényege, hogy a forrás irányának ismeretéből számítunk pozíciót. Az irányszögelés legnagyobb hátránya, hogy speciális hardver (irányított antenna) szükséges hozzá. A pontos működéshez közvetlen rálátás szükséges, amelyet szóródás és a többutas terjedés ronthat. Beltéri pozícionáláshoz nem a legjobb megoldás.

Jelerősség érzékelés - RSS

Az RSS a rádiós jelerősség alapján történő pozícionálást jelenti, azaz a beérkezett jel erősségéből képesek vagyunk következtetni az adótól való távolságra. Előnyei, hogy egyszerűen mérhető és kiegészítő hardver nem szükséges. Hátránya a pontosság és romló minőség a távolság növekedésével. A pozícionáláshoz legalább három adó jelerősségadatainak ismerete szükséges. A módszer jól használható akkor, ha a zavarok nem torzítják az adott környezetben történő mérést. A metódus minőségét javítani lehet, ha a területet feltérképezzük, azaz az egyes pontokban RSS ujjlenyomatokat készítünk.

Késleltetés mérése – TOA/TDOA

TOA esetében az érkezési idők alapján köröket rajzolunk, és ezek metszete adja a pozíciót.

TDOA használatával hiperbolákat szerkesztünk, amelyek metszete adja a pozíciót. A hiperbolák szerkesztése az egyes node párokon érzékelt időkülönbség alapján történik. Általánosságban elmondható, hogy a hiba csökken a távolság növelésével (ésszerű keretek közt - például a jel/zaj arány kezelhető szintjét feltételezve).

Az előnyök, hogy pontosabb becslés érhető el vele, az RSS-hez hasonlítva és rengeteg vezeték nélküli technológia alkalmazása esetén használható. Hátránya a nagy költség, mivel a node-oknak rendkívül jól szinkronizáltnak kell lenniük.

A metódusban használt hiperbolák a mote párokhoz tartozó azonos időkülönbségű helyek összességét jelzik. Ezek metszéspontjában határozhatók meg az esetleges hangforrás pozíciók.

Bithiba arány mérése – BER

A BER alapú pozícionálás elsősorban Bluetooth esetén használható, alapja, hogy a távolság függvényében változik a z érkezett hibás bitek aránya.

Mobilitás szenzorhálózatokban

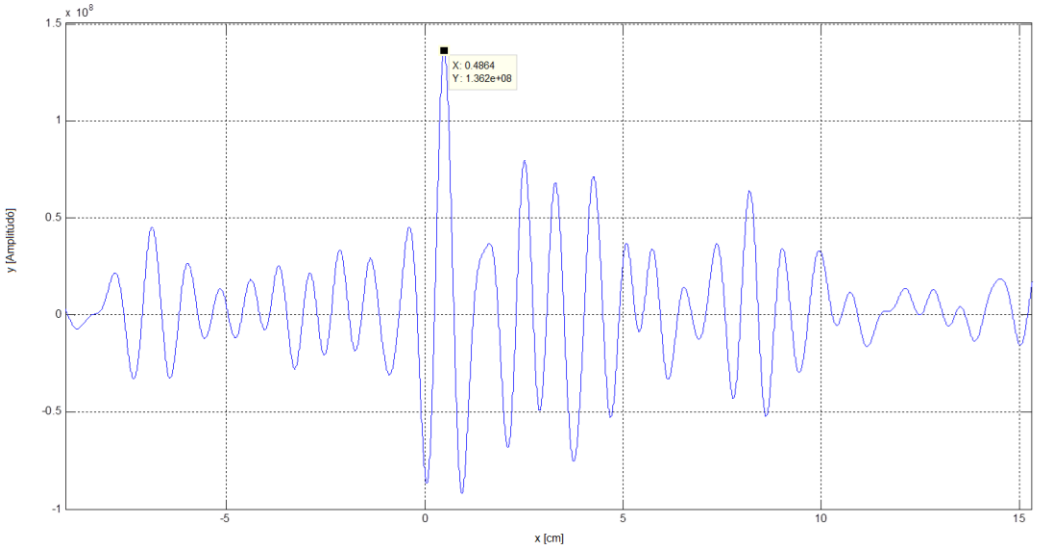
A hálózatban mozoghatnak a szenzorok, a bázisállomás és a követendő objektumok, amely megnehezíti a lokalizáció kérdését. A mobilitás célja lehet energiahatékonyság, a lefedettség biztosítása, a topológia ellenőrzése vagy a megfigyelés minőségének javítása. A mozgó bázisállomásnak további haszna lehet a fogyasztás csökkentése, amely esetén a bázisállomás ténylegesen körbejár a szenzor node-ok közt, és lekérdezi a gyűjtött mintákat. A bázisállomás visszatérve a kezdőpozícióba újra töltődhet, így ezen a téren nem kritikus tényező a fogyasztás. Az esetemben a hálózat elemei működés közben nem változtatnak pozíciót.

További beltéri technikák az RFID-s rendszerek és a wifi jelerősség alapján történő pozícióbecslés. Az RFID-s rendszerek fix telepítésű olvasók alapján képesek meghatározni a személy helyzetét, a hozzárendelt kártya használata alapján, a wifis pozícióbecslést pedig a korábban említett RSS alapú technológia. [13]

7.1.2 A választott metódus

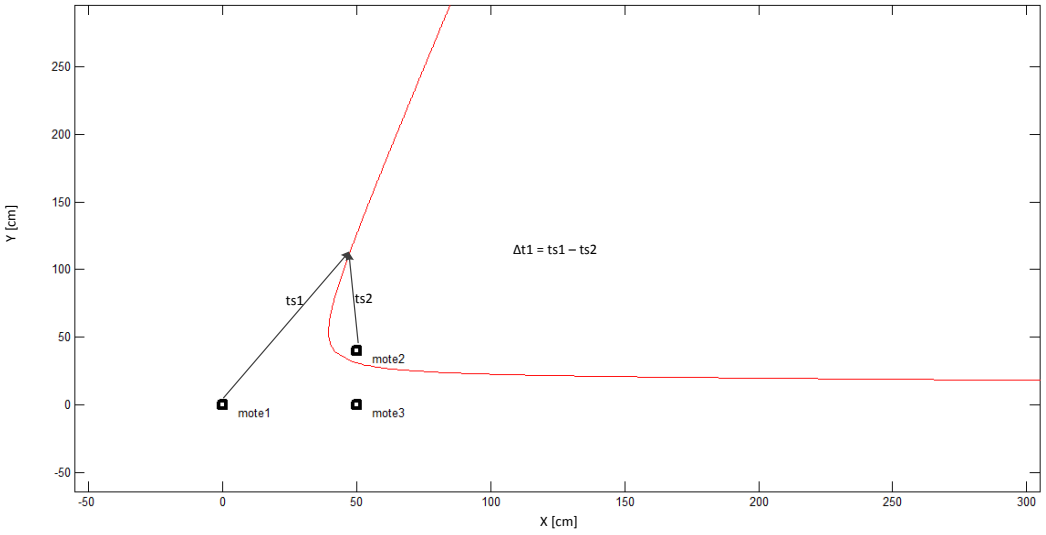
Az általam választott technológia a TDOA, vagyis az egyes node-ok által érzékelt jelek időkülönbségeiből becsülöm a hangforrás pozícióját. A korábbiakban leírtak szerint akusztikai alapon történik a helymeghatározás. A működés nagyobb lépései a következők. Az esemény hatására a node-ok elkezdik a mintavételezést, amelyeket ezután a bázis lekérdez, és a mintákat továbbítja a PC-re. A számítógép Matlabban dolgozza fel az adatokat. A technika lényege, hogy a mote párokat vizsgálva megállapítjuk a hangminták közti eltolódásból az időt, amely alapján az egyes párok közt hiperbolát rajzolhatunk.

Az időeltolódást az egyes mote párok keresztkorrelációs függvényének maximumából határozhatjuk meg, mivel a keresztkorrelációs függvény megadja két függvény hasonlóságát, olyan módon, hogy a mintasorokat eltolva egymáson minden pontban kiszámítja az értéket, így a maximumhely nullához képesti különbségéből megkapjuk a mintakülönbséget, amiből távolságot számíthatunk (20. ábra).

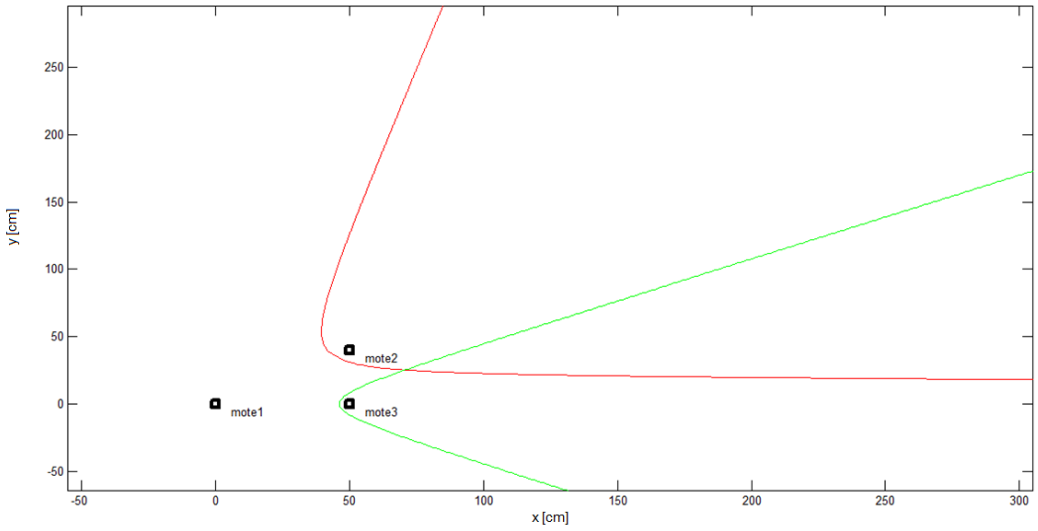


20. ábra: A keresztkorrelációs függvény maximuma (x: cm-ben, y: amplitúdó)

A hiperbolák megadják azokat a helyeket, amelyek az egyes párokhoz tartozó állandó időkülönbségű helyeket jelentik (21. ábra), így ezek mutatják a lehetséges hangforrás pozíciókat. A hiperbolák metszetét véve – amely ideális esetben egyetlen pont – megkapjuk a pontos pozíciót (22. ábra).



21. ábra: Hiperbola - állandó időkülönbségű helyek (a tengelyek cm-ben értendők)



22. ábra: Hiperbolák metszéspontja (a tengelyek cm-ben értendők)

A metódus eredménye ugyanakkor egyfajta irányszögelésnek is felfogható, mivel a mérési eredmények bebizonyították, hogy a rendszer az irányt kisebb hibaaránnyal képes

megállapítani, mint a pontos pozíciót. Ez elsősorban a különböző hibák fellépése és a hangminták sajátosságai miatt történt így.

Összefoglalva a TDOA (érkezési időeltolódásból számított távolság) elv használatával megvalósított AOA (irányszögelés) eredményét sikerült nagyobb pontossággal megvalósítani. Az AOA hagyományosan speciális hadvert igénylő technika, így a megvalósítás ennek alapján nem nevezhető irányszögelésnek.

8 Megvalósítás

A fejezet az ismertetett eszközökkel megvalósított rendszer részletes működéséről szól. Az alábbiakban szó esik a PC-n és a hálózat elemein futó programokról.

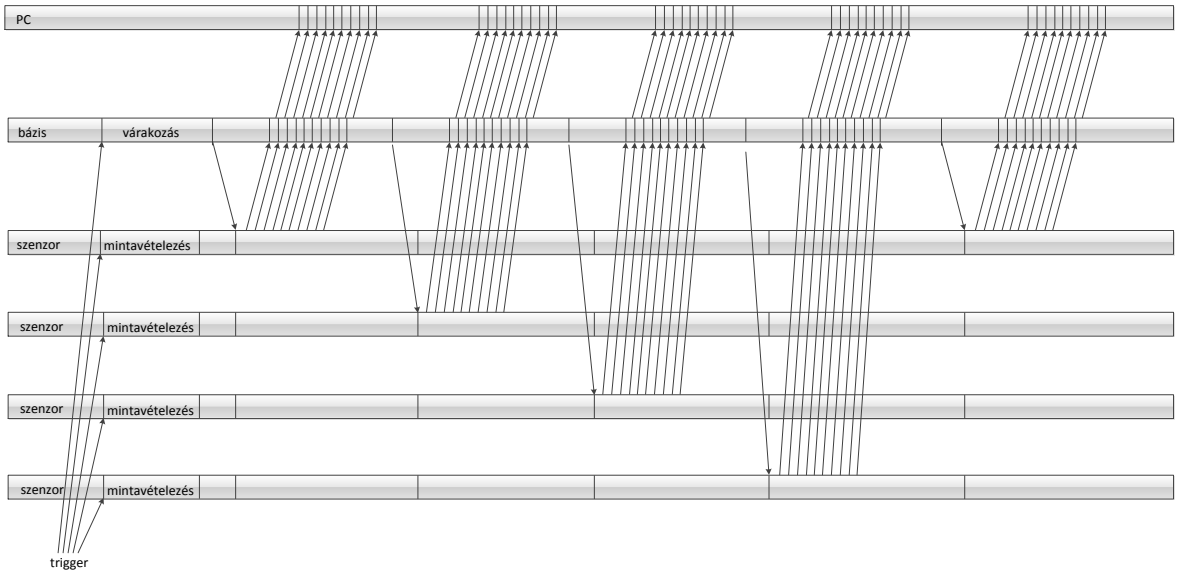
8.1 A szenzorhálózat működése

A rendszer felépítése a 4. ábrán látható. A következő pontokban a helymeghatározó rendszer felépítését és működését ismertetem. Röviden a hálózat egy PC-hez csatlakoztatott bázisállomásból és a szenzorokból áll. Az adatgyűjtés egy maghatározott feltételre indul a szenzorokon, melyet a bázis begyűjt, majd továbbít a PC-re. A számítógépen futó program az adatok kiértékelését és a pozíció becslését végzi, grafikus szemléltetéssel.

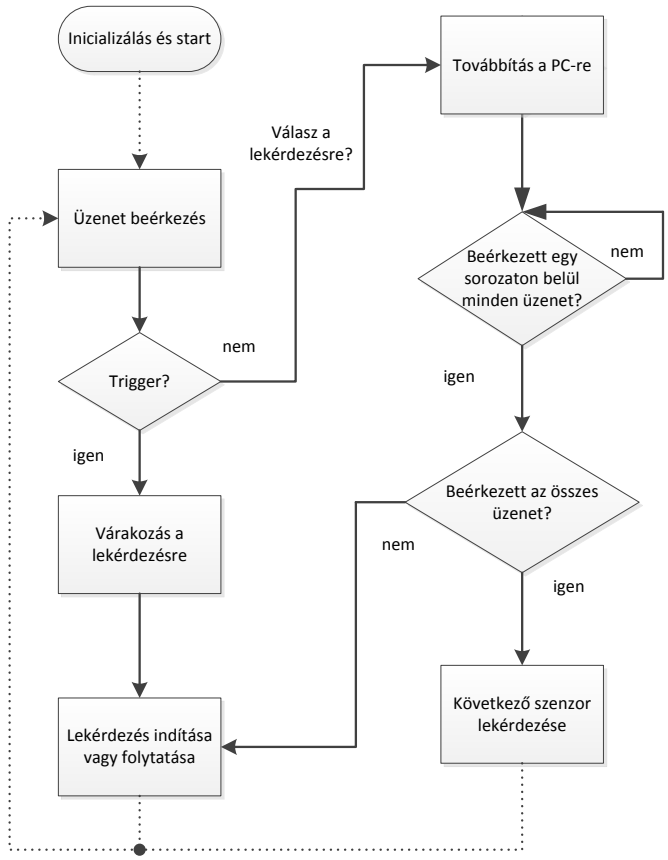
8.1.1 Bázisállomás

A bázisállomás végzi a rendszer irányítását. Az idődiagramot a 25. ábrán láthatjuk, a folyamatábrát a 24. ábrán, a modulábrát pedig a Függelék V oldalán.

A pozíció becslése a következőképp zajlik. A triggerfeltétel teljesülése után megkezdődik a mintavételezés, mindegyik szenzoron egyszerre. Ez egy olyan rádiós üzenettel biztosított, amely a szinkronizációt is elvégzi. Az ily módon szinkronizált node-ok a begyűjtött mintákat eltárolják a memóriájukban, majd várnak a bázis által irányított lekérdezésre. A bázis egy előre kiszámított ideig vár, amely alatt biztosan végbemegy a mintavételezés. Ha ez letelt, a lekérdezés megkezdődik, amely egy fix számú üzenetben érkező sorozatot jelent. A beérkezéseket követően a PC-re továbbítódik az üzenet UART-on. A szenzormote-okat sorban lekérdezve begyűjtjük az összes adatot. Az összes adat megérkezése után megkezdődik a kiértékelés, és az állapotgép alapállapotba áll. Az alábbi ábrán az ehhez tartotó idődiagram látható, amely addig ismétlődik, amíg le nem kérdeztük az összes adatot.

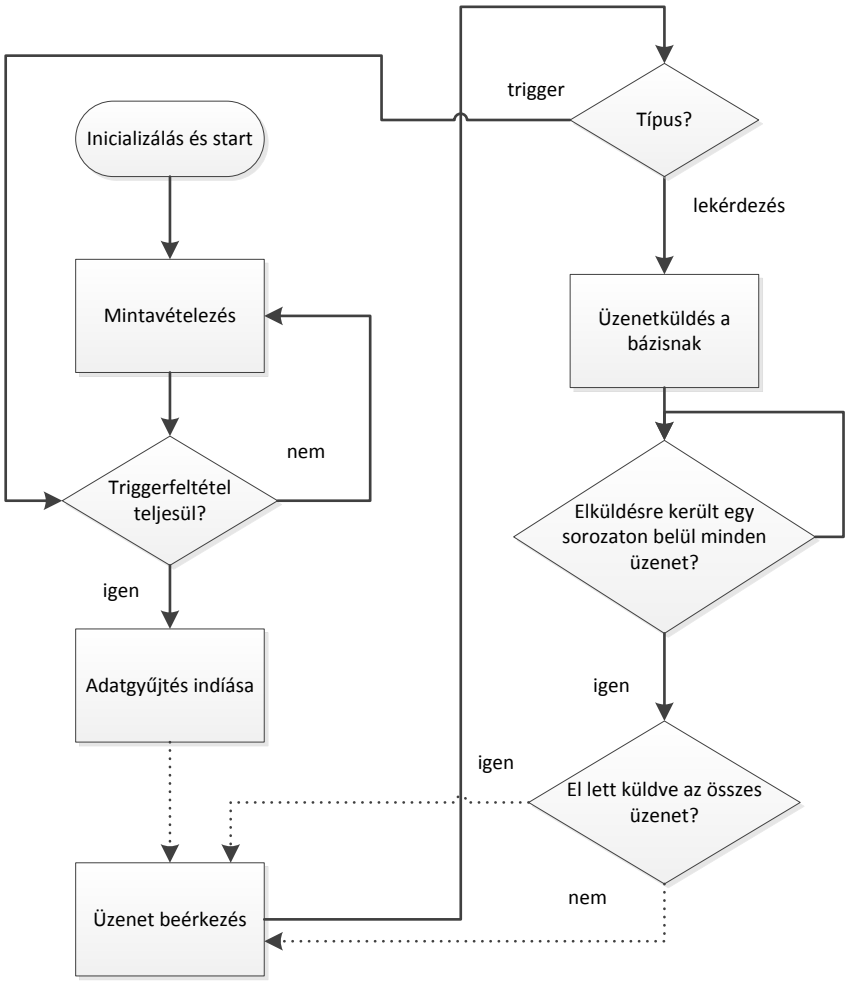


23. ábra: A lekérdezés idődiagramja



24. ábra: A bázis működésének folyamatábrája a triggerelésre és az adatgyűjtésre vonatkozóan

8.1.2 Szenzor mote-ok



25. ábra: A szenzor mote-ok folyamatábrája

A szenzormote-ok folyamatábrája a fenti ábrán látható, a modulábra pedig a Függelék W oldalán. A szenzor mote-ok szerepe a hálózatban az adatgyűjtés, és annak továbbítása a bázis felé. Az adatgyűjtés triggerjelre indul, amely működési módtól függően több helyről érkezhet. Teszt üzemmódban a bázis küld egy meghatározott bájtsorozatot, amely teljesíti a triggerfeltételt, míg automatikus adatgyűjtés módban a mikrofon folyamatos mintavételezésével kapott értékek egy bizonyos szintet átlépő értéke jelenti ezt. A mintavételezést bármelyik szenzor indíthatja, ekkor a hasznos jel érzékelése esetén rádiós

üzenetben értesíti az eseményről a többi hálózati résztvevőt, mely egyben szinkronizációs csomag is. Az említett üzenetet a bázis is megkapja, amely irányítja ezután a lekérdezést. A szenzor node-ok különböző üzenet érkezésekor különböző üzemmódokban képesek működni, melyek a fejlesztés során nagyban megkönnyítették a munkám. Lehetőség van pl. az óraszinkronizáció tesztelésére, melynek lényege, hogy az üzenetek beérkezése után meghatározott időpontban egy választott értékre állítja az órát az összes mote-on. Tesztelhető továbbá az órák elcsúszása, amely az egyik node hardveres órájának járásához igazítva kérdezi le a másik node-ot, így az elcsúszás egy PC-re visszaküldött üzenetbe csomagolva kiértékelhető. Az így gyűjtött információ felhasználható egy későbbi alkalmazás által megkövetelt szinkronizációhoz.

A node-ok működése instabillá vált, ha a RAM kihasználtsága 99 % fölötti volt. Ez nagyságrendileg többtíz bájtot is jelenthet, amely a legtöbb felhasználásnál nem jelent nagy hátrányt, ám nem tűnt indokoltnak.

A szenzoroknál érdemes ügyelni a telep töltöttségére, mivel ez látszólag helyes működés esetén már torzíthatja az ADC eredményeit, emiatt a helymeghatározás pontossága jelentősen romlik.

8.2 A PC oldali program

Mivel a Matlabhoz hivatalos TinyOS támogatottság is rendelkezésre állt, amely lényegében a java környezet Matlabban történő programozását jelenti. Ez a rendelkezésre álló szoftverkörnyezetben nem működik. A problémára az interneten sem találtam megoldást, így egy saját soros port kezelő megírása mellett döntöttem.

A Matlab soros port kezelésének főbb tulajdonságai

A Matlab beépített objektummal képes kezelni a soros portot, így a megvalósítást ezzel tervezem. Az objektum több attribútummal rendelkezik, amelyekben a soros portra vonatkozó beállítások találhatók. A helyes használathoz a portszámot, a sebességet és egyéb tulajdonságokat be kell állítani, mielőtt kinyitnánk a portot. A port nyitott állapotában történhet a kommunikáció. Több soros porton csatlakozó eszköz kezelésére is lehetőségünk van, ebben az esetben a különböző portokhoz más-más objektum létrehozására van szükség. Egy porthoz tartozó objektumokból többet is létrehozhatunk, ám egyszerre csak egy lehet közülük nyitott állapotban.

A buffer (mely lényegében egy FIFO) mérete a soros port objektum egyik fontos tulajdonsága, amelynek telítődése esetén a legrégebbi adat kerül eldobásra. A többi beállítás a soros portra jellemző, amely más soros port kezelő programban is hasonló, úgymint baudrate, com port, stop bit, stb.. A Matlab kínál felvételi funkciót az objektumhoz, amely használata révén az adatáramlást elmenthetjük egy fájlba.

Lényeges működési szempontok a megbízható adatfogadás és feldolgozás. Ennek megvalósításához a TinyOS-en belüli üzenetkezelést kell jobban megvizsgálni.

Keretezés, speciális karakterek

A TinyOS-szel gyűjtött adatok PC-n történő feldolgozáshoz a bitosítani kell a számítógépes kommunikációt. A rendelkezésre álló programozó soros kommunikációt tesz lehetővé. A TinyOS lehetőséget biztosít a soros kommunikáció tesztelésére az Oscilloscope alkalmazás segítségével. A program a fényszenzor értékeit küldi vissza a számítógépre, majd az adatokat egy java-s programon keresztül követhetjük nyomon, emellett a beérkezett csomagok nyers értékei is megjeleníthetők a parancssorban. A későbbi feldolgozás szempontjából a nyers adatok lényegesek, amelyek csomagokban érkeznek. Az egyes csomagsorozatok elejét és végét jelölő egy speciális bájt a 0x7d és 0x7e karakter.

A csomagok a következőképp épülnek fel (amelyek UART-on keresztüli továbbításnál keretbe foglalva érkeznek).

dest addr	handlerID	groupID	msg len	source addr	counter	channel	readings
7e 00	0a	7d	1a	01 00	14 00	01 00	96 03 97 03 97 03 98 03 97 03 96 03 97 03 96 03 96 03 96 03

26. ábra: TinyOS üzenetek felépítése [18]

Az egyes mezők méretei és elnevezései:

- Célcím - Destination address (2 byte)
- Üzenet azonosító - Active Message handler ID (1 byte)
- Csoport azonosító - Group ID (1 byte)
- Üzenethossz - Message length (1 byte)
- Hasznos adatrész - Payload (maximum 29 byte) [18]

A hasznos adatrész felépítése egy header fájlban tárolódik, amelyből egyszerűen definiálhatunk tetszőleges felépítésűt. A header fájlban egy struktúraként adjuk meg az egyes mezők által hordozott információt, amelyekkel az átláthatóbb a fejlesztés. A Matlab később ezen információk alapján dolgozza fel a beékező üzeneteket.

Fontos megemlíteni, hogy az adat a mote-ról little-endian formátumban távozik, azaz például egy 2 byte-os 0x9603 adatot 0x03 0x96 sorrendben küld ki.

A TinyOS 1.1-ben használatos protokoll a „PPP in HDLC-like framing”-en alapul, amely az RFC-1662-ben került leírásra. Ugyanazt a keretezést és 16 bites CRC-t használ. A protokollban meghatározott néhány speciális karakter, amelyek a következők:

- 0x7e kódolva 0x7d, 0x5e. (Flag Sequence)
- 0x7d kódolva 0x7d, 0x5d. (Control Escape)
- 0x03 kódolva 0x7d, 0x23. (ETX) [20]

Ezen karakterek közül a TinyOS 1.1-ben a 0x7e és a 0x7d továbbítódik kódolva.

Definiált üzenet felépítése:

```
struct CounterDiffMsg
{
    uint8_t forrasMoteID;
    uint8_t celMoteID;
    uint16_t szamlaloAllas;
    uint16_t szamlaloSzorzo;
    uint16_t data[BUFFER_SIZE];
};
```

Itt látható, hogy a definiált üzenet tartalma az 1 bájtos forrás mote ID, cél mote ID, majd 2 bájtos számláló állás és számláló szorzó, majd az egyenként 2 bájtos, BUFFER_SIZE mennyiségű adat következik. Az egy bájtos címkezelés helytakarékosságot szolgál, mivel az eredetileg 2 bájtos címeket a hálózatomban nem szükséges fenntartani, mert az 1 bájton kódolható elemek száma 255, amely lényegesen több az általam használt eszközszámnál. A fenti üzenetet használtam a szinkronizációs tesztek végzésére.

Működés

A Matlabban írt soros port kezelő program első megvalósítása folyamatos lekérdezéssel működött. Ennek előnye, hogy a beérkező bájtsorozatokat egyben tudom feldolgozni, így rövid és egyszerű scripttel képes vagyok dekódolni az üzenet részeit. A

Matlab lehetőséget biztosít aszinkron módon történő beolvasásra is, amelynek előnye, hogy a program a beolvasáshoz érkezve megáll, amíg adatot nem kap. Ezáltal a beolvasott adatot egy ciklusban dolgozhatom fel, ahol több korrekció elvégzése után következhet a tömbbe rendezés, és az adatok kiértékelése.

Az implementálás és mérések elvégzése után az aszinkron olvasás minőségével kapcsolatban probléma merült fel, a beolvasás megszakadt, mivel egy korábbi folyamatban volt. Ez elméletileg nem fordulhat elő, mivel a program csak akkor folytatódhat, ha már érkezett adat, amelyet rögtön be is olvasunk. Összességében a módszer használható, de a robosztussága sajnos az elvárásoktól elmarad. A vázolt problémák miatt másik módszert kerestem.

Implementált megoldás

A Matlab programkörnyezet lehetőséget kínál eseményvezérelt működés leírására, ún. function_handle segítségével történő indirekt függvényhívások által. Bizonyos eseményekhez ún. callback függvényeket társíthatunk, amelyeket az esemény hatására automatikusan meghív a Matlab. A callback függvény egy előírt argumentumlistájú, de egyébként tetszőleges, a fejlesztő által megadott funkcionalitással rendelkezhet. A callback függvény ún. function handler segítségével társítható eseményhez. Segítségével lehetőség nyílik függvények átadására paraméterként, adatmentésre és speciális függvényhívásra. [19]

A soros port kezelésnél számomra az indirekt meghívhatóság a lényeges, mivel ily módon adat érkezésekor meghívhatok egy függvényt, amely beolvassa a rendelkezésre álló adatokat. A működéséből adódóan magasabb prioritással rendelkezik más Matlabon belül futó programoknál, ezáltal megszakítva azok működését szinte azonnal lefuthat. A soros porton érkező adatok szempontjából ez lényeges elem. A robosztussága elfogadható.

A megvalósítás jelentősen eltér a lekérdezéses módszertől, mivel az adatok érkezése nem jósolható pontosan. A feldolgozás tömbösített beolvasással történik, majd az összes adat beérkezése után történő feldolgozása következik. A beolvasás megkezdése után indítok egy időzítőt, amely a legutóbbi adat beérkezése óta eltelt időt méri. A számlálót adat érkezésekor újraindítom, így biztosított, hogy a kiértékelés csak akkor indulhat, ha több adat már nem érkezik. A beérkezés után következik a feldolgozás. A beérkezett adatokat egy tömbbe mentem.

A beolvasott adat egy olyan tömb, mely a szenzorhálózat működéséből adódóan a mote-ok adathalmazainak egymásutánja. A beolvasás után bizonyos korrekciók elvégzése szükséges. Korrigálni kell például az esetleges hiányos csomagokat, majd kiszűrni a fejlécet, hogy a hasznos adatrész rendelkezésre álljon.

Az adatmező ezek után rendelkezésre áll, így kezdődhet a speciális karakterek visszafejtése az említett szabvány szerinti módon. Az így kapott adatmezőben még szerepel a TinyOS-ben definiált üzenettípus minden mezője. Ez saját kódolást jelent, amelyet értelmezve kapom meg a kívánt adatokat.

A beérkezett bájtsorozatot a szenzorok azonosítója szerint csoportosítom, így folyamat eredményeképp megkapom az egyes szenzorok által mintavételezett adathalmazt. Az esetleges hibák miatt szükség volt a tömbök egyforma méretűre vágására, és a kimaradt üzenetek helyének feltöltésére.

A Matlab többször nem volt képes megfelelően kezelni a soros portot, ha más programmal is kommunikáltam azon. Ez a rejtett objektumok miatt történt így, amelyek törlése segített a problémán. Az objektum láthatósága az egyik paramétere, melyet lehetőségünk van átállítani. Ha az objektum rejtett, a futási idejű változóknál tovább megmarad, csak rejtettsége miatt a workspace-ben erről nem kapunk információt. A rejtett objektumok kilistáztatására és törlésére van lehetőség.

A tesztelés szempontjából lényeges a folyamatos dokumentálás, mivel enélkül nem lehetne visszakövetni az eredményeket. Ennek érdekében a Matlab kód minden futtatáskor elmenti a workspace-t, amely tartalmazza az összes változót, és annak értékeit a mentés pillanatában. Így a rendelkezésre álló adatok segítségével bármikor újra futtathatjuk a számításokat és a kiértékelést. Ez a metódus tesztelése és javítása közben kifejezetten hasznos.

Jelfeldolgozás

A számítógépen futó Matlab program az adatok kicsomagolásán kívül a helymeghatározó algoritmus futtatásáért is felelős. Az algoritmust a korábbiakban ismertettem (7.1.2 fejezet), összefoglalva az egyes szenzorok adatainak összehasonlításának eredményeképp számolt beérkezési időeltolódásból hiperbolák felrajzolásának metszéspontjai alapján ítélhető meg a hozzávetőleges pozíció.

A program feladata a következő. A triggeresemény érzékelése után következik a beolvasás, amelynek eredményeképp a nyers adatok rendelkezésre állnak. Javítás és csomagolás után a szükséges adatmezők csoportosításával megkapjuk az egyes szenzorokon felvett hangokat. A mintákból ezután távolságadatokat képzünk, amelyeket az előzőleg kiszámított ismert pozíciókból kapott távolságértékekkel összehasonlítva kiszűrjük a hibás mérési eredményeket. Ilyen hiba, ha a mért adatokból adódó távolság nagyobb, mint a mote-ok közti tényleges távolság.

Egy mótpárra megadható egy hiperbola az általuk mért távolságkülönbség alapján (21. ábra), melynek alapja egy óraszinkronizációs mérés [22]. A hangforrás ezen hiperbola valamelyik pontján tartózkodik. Egyetlen mótpár (2db szenzor) esetén csupán egyetlen hiperbola adható meg, a hangforrás hiperbolán belüli pozíciója nem. Amennyiben minden egyes mótpárra meghatározzuk, hogy az általuk mért időkülönbségek alapján hol lehet a forrás pozíciója, úgy egy hiperbolasereget kapunk. A valódi pozíciót ezen hiperbolák metszéspontja határozza meg (22. ábra). Ideális esetben elegendő lenne két hiperbola metszéspontja, amihez 3 mote szükséges, de valós esetben a mérési zajok miatt érdemes minél több eszközt használni. A hiperbolák metszéspontját Matlabban numerikusan számítjuk. Minden hiperbolát megfelelő finomságú pontban kiszámítunk, és az összes pontpár közötti legkisebb távolságot tekintjük metszéspontnak. A valós mérések eredményei összesítve a függelékben találhatók (Függelék A-Q: Pontosságra vonatkozó mérések és 29. ábra).

9 Mérési eredmények

A fejezetben a rendszerrel végzett mérési eredményeket ismertetem.

9.1 Tesztjelek előállítása

A rendszer pontosságának feltérképezésére vonatkozó mérések először ún. chirp jellel gerjesztve készültek. A chirp jel időfüggvénye a következő:

$$x(t) = Asin\left[2\pi\left(f_0 + \frac{\Delta f}{2T}t\right)t + \varphi\right];\ t = [0 \dots T]$$

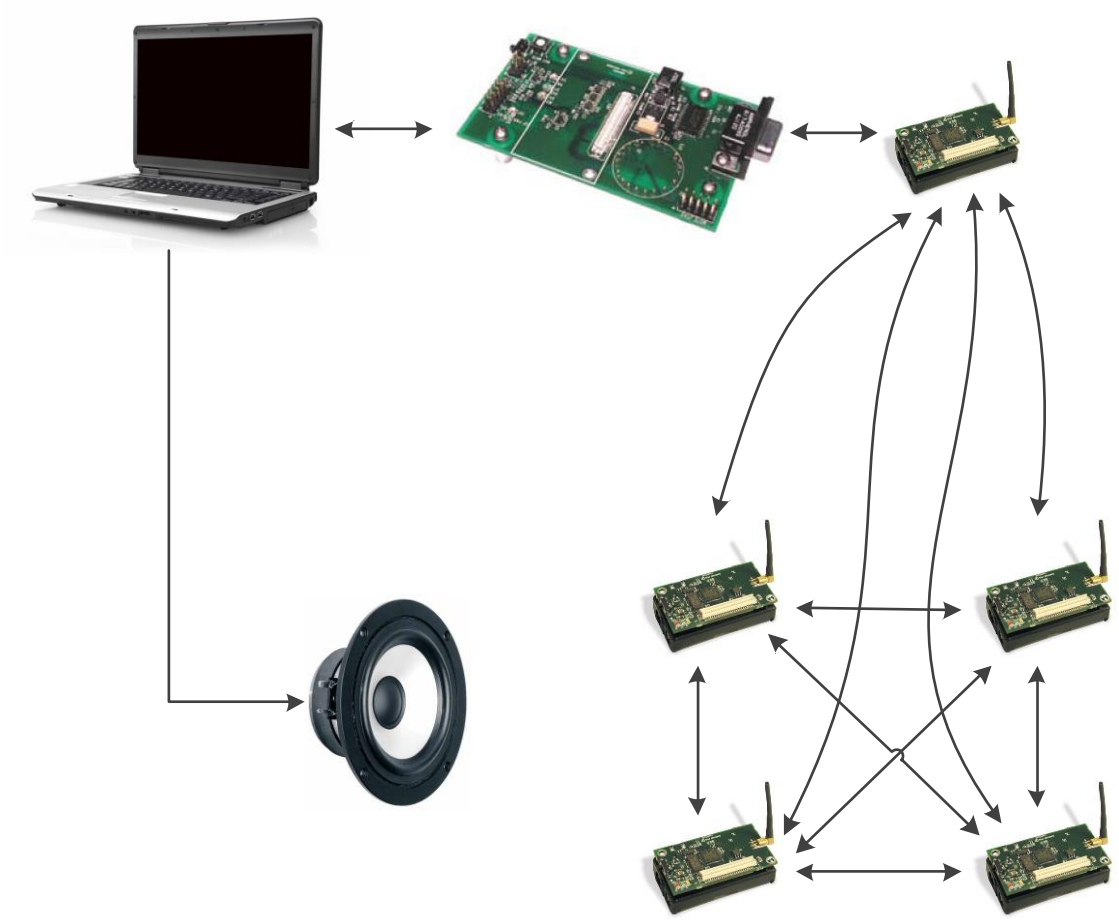
A chirp jel tehát egy T hosszúságú, lineárisan növekvő frekvenciával rendelkező szinuszjel.

A jel előnye, hogy jól kalkulálható sávszélessége miatt a korrelációs függvény számításakor pontosabb eredményt ad (mivel keskenyebb korrelációs függvényt kaphatunk), mint egy tetszőleges hang esetén számított. A jelet a Matlabban a chirp parancs segítségével generálhatjuk, majd a generált jelet a sound vagy a soundsc függvénnyel játszhatjuk le. A jelek elcsúszását az keresztkorrelációs függvények maximumhelyeinek eltolódásából számíthatjuk.

A mintavételi tétel teljesítése érdekében a chirp jel maximális frekvenciáját a mótok adott mintavételi frekvenciájához kell megválasztani. Ez 1800 Hz-es mintavételezés mellett maximum 900 Hz-et jelent. A keskeny autokorrelációs függvény miatt nagy sávszélességre van szükség. [22]

Mérések

A mérések a következőképpen zajlottak. A szükséges hardverek felélesztése után egy választott topológia szerint állítottam fel a hálózatot. Ehhez a megfelelő környezet az alábbi ábrán látható.



27. ábra: A mérési összeállítás

A hálózat indulása után a számítógépen futó Matlabból indítottam a hangminta lejátszását és a hálózati adatgyűjtést. A hangminták felvétele után a Matlab a bázison keresztül begyűjti az adatokat, amelyeket feldolgozva megkapjuk az egyes mote-okon felvett hangmintát. A mintákból a keresztkorrelációs függvény számítása után számoljuk az eltéréseket. A mote párok közti eltérésből rajzolt hiperbolák metszéspontjai megadják a hangforrás helyzetét.

A méréssorozat a hangszóró meghatározott koordinátákra helyezésével került felvételre, amelyet összesítve az alábbi ábrán láthatunk. A függelékben az egyes mérések külön-külön is megtalálhatók.

Méréseket végeztem továbbá a hangerősség vizsgálatára. A lenti ábrán több ilyen jellegű mérés összesítése látható. A hangerősséget nem volt lehetőségem lemérni, a

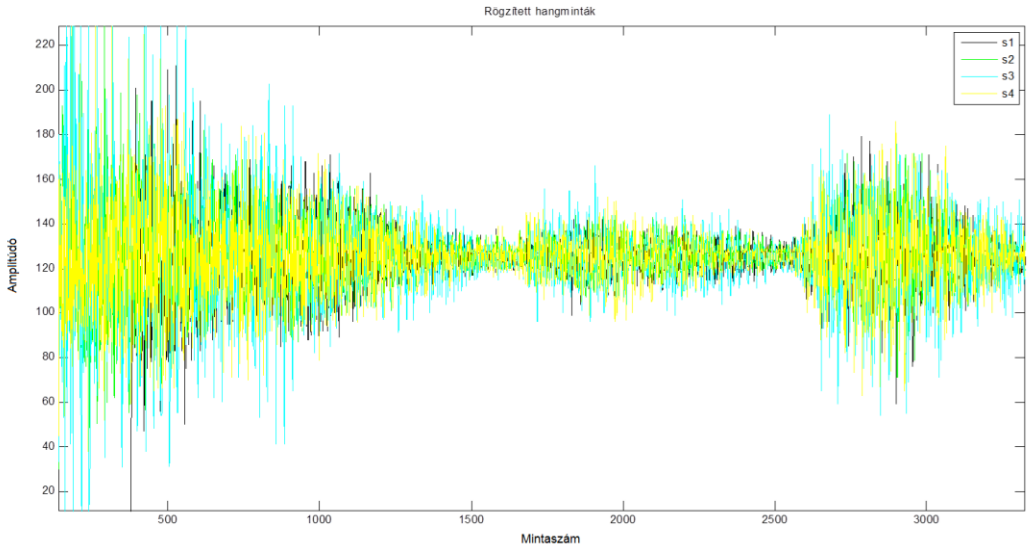
rendelkezésre álló tesztkörnyezet paramétereit ismertetem nagy vonalakban. A teszteket több laborban végeztem, a rendelkezésre álló eszközök minősége közt nagy különbség nem volt. Egy laptopról generáltam a hangot, mely maximális hangerőre volt állítva, ehhez csatlakoztattam egy hifit – melyre 1 db 40 W-os hangfal volt rákötve – 50 %-os hangerő beállítással. A tesztek közben természetesen ez változott.

A konklúzió a hangerősségre vonatkozóan elfogadható eredmény hozott, mivel egy emberi beszéd átlagos hangerejéhez viszonyítva jóval halkabb hangerőn is helyesnek mondható eredmények születtek.

A generált chirp jel után általam felvett beszédhangokkal is teszteltem a hálózatot. A számítógép beépített mikrofonjának segítségével jó minőségű, tiszta mintákat sikerült rögzíteni, amely a tesztekre nézve fontos szempont. Ily módon az eredmények összehasonlíthatók. A felvétel a Matlab segítségével történt, a wavrecord paranccsal, amelynél a mintavételi frekvenciát és a hangminta hosszát szükséges megadni. Emellett élő beszédhanggal is teszteltem a rendszert.

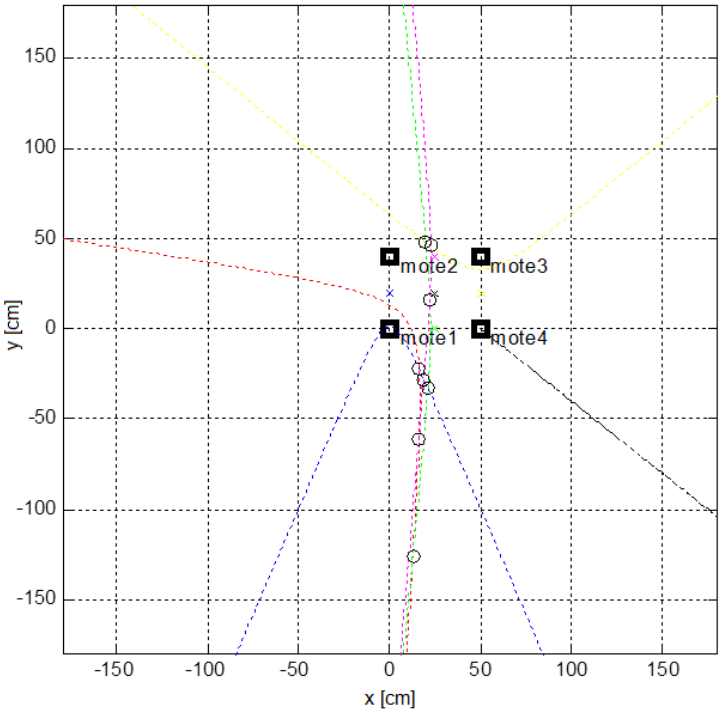
Az alábbiakban látható egy átlagos mérés beszédhangra, amely átlagosnak nevezhető mérés eredményét takarja.

A hangminták a következőképpen néznek ki egymásra rajzolva.



28. ábra: Egy valós mérés hangmintái

A fenti ábrán tehát a 4 szenzor hangfelvételei láthatók, amelyek már a szinkronizált hálózatban kerültek rögzítésre. Az adatok a függelék ábráján külön-külön is megtalálhatóak. Ebből számíthatók a keresztkorrelációs függvények, amelyekből pedig az érzékelési különbségeket, így pedig a távolságok határozhatók meg. A pontosabb számítás érdekében 20-szoros interpolációt alkalmazunk a hangmintákra (a Matlab resample parancsával), majd ezekből számítjuk a keresztkorrelációs függvény maximumát. A korrelációs függvény az xcorr parancs segítségével számítható, a maximum pedig egy egyszerű maximumkeresés. Az így kapott eredmények már távolságértékekben számolódnak.



29. ábra: A számított pozíció (a tengelyek cm-ben értendők)

A fenti ábrán látható, hogy a helymeghatározás nem nevezhető egzaktnak, ám az irány a metszéspontokból jó eredményt mutat. A hangforrás kb. 20, -50 pozícióban helyezkedett el.

A tesztek közt a feldolgozás szempontjából kritikusabb hangokat is kívántam vizsgálni. Ilyenek például a csak azonos magánhangzóból álló hangok, mivel ezeknél a korrelációs függvény számításának eredményeképp pontatlanabb adatokra kell számítanunk.

Az ilyen típusú hangoknál is jól szerepelt a rendszer, ám ez nagyban függött attól, hogy a hang a mintáknak mekkora részét töltötte ki. teljes kitöltés esetén a pontos meghatározás nem lehetséges, viszont egy hozzávetőleges irányt képes kijelölni a rendszer.

A teszteléshez a mote-ok mintavételezési beállításait működés közben a PC-ről is állíthatóvá írtam meg. Ehhez egy új típusú üzenet feldolgozását kellett lekezelnem a szenzorokon, amelyhez a bázist is fel kellett készítenem az ilyenfajta üzenetek kezelésére és továbbítására, továbbá a Matlabbal is új típusú adatot küldtem ki.

A mintavételezés 1800 Hz-en is kielégítő eredményt adott a speciális jelekre, hangfelvételekre viszont többször rontott. Emiatt a 3600 Hz-es frekvenciával történő mintavételezést is kipróbáltam. Az eredmények javultak, de a rendelkezésre álló körülmények közt a felvételi idő csökkent. Az így felvett hangminta hossza kb. egy másodperc, amely elegendő a pozícióbecsléshez. A mintavételi frekvenciát a rendszer tesztelt körülmények közt több, mint 10 kHz-ig képes tartani, viszont ebben az esetben szükség lenne a flash memória használatára és ehhez kapcsolódó tesztelésre. A hálózat működése szempontjából viszont ez nem mérvadó, mivel ennyi adatot a bázis csak lassan lenne képes továbbítani a számítógépre.

10 Kitekintés

A továbbiakban a rendszer hétköznapi alkalmazásban történő felhasználását is elképzelhetőnek tartom. A korábbi fejezetekben bemutatott példák többségéhez felhasználható lenne a rendszer, amely felhasználástól függően javításra szorulhat.

Egy lépés lehet a rendszer közel valósidejűvé alakítása, amely kibővíthetné a felhasználási lehetőségeket – a felhasznált eszközökkel az általam mért minőségben nem lehetséges. Fejlesztési irányok lehetnek a kommunikációs csatorna adatátviteli sebességének kibővítése, a helyben történő adatfeldolgozás, és egyéb kiegészítő hardver tervezése, amellyel biztosíthatók a követelmények. Az adatátviteli sebesség szűk keresztmetszete egyelőre a PC és a bázis közti kommunikáció. A jelenlegi hardverekkel szoftveresen a duplájára növelhető, míg egy esetleges nagyobb adatátviteli sebességre képes soros-USB átalakító és a hozzá tartozó driver esetén a mikrokontroller válhatna a szűk keresztmetszetté. A mote-okon található AVR képes lenne akár a jelenlegi sebesség (57600 kbps) négyszeresének biztosítására is. További lehetőség több bázisállomás használata, amely drágítaná a rendszert. Ilyen esetben növelhető lenne a mintavételi frekvencia, amelyhez szükséges lenne a nagyobb adatátviteli sebesség, és esetlegesen a flash memória használata. A flash memória használatához előzetes teszteket kellene végezni, elsősorban a sebességre vonatkozóan.

A vezeték nélküli szenzorhálózatok felhasználási céljai miatt ésszerű lenne a programkód távvezérelhetővé tétele bizonyos paraméterek szempontjából. Ehhez megfelelő üzenetet/üzeneteket kell definiálni a PC-n, a bázison ennek továbbítását lekezelni, a szenzorokon pedig a feldolgozást. A Matlabban létrehozható ehhez egy grafikus felület, amellyel lehetőség nyílik az üzenetek összeállítására, és az aktuális beállítások lekérdezésére is. Az alapja a hálózatban már működik, mégpedig a mintavételi frekvenciára vonatkozó beállításoknál.

Pontosítás

A hálózat működésében további pontosítás lenne elérhető, ha az alkalmazott hardver- és szoftvereszközöket megvizsgálnám ilyen szempontból, gondolok itt a szinkronizációhoz kötődő kiegészítő hardverre (pl. fűtött kvarcoszcillátor), szoftveresen a szinkronizációs algoritmus javítására (pl. elcsúszás korrekciója), stb.. A beltéri alkalmazásban a hőmérsékletfüggés nem kritikus befolyásoló tényező, így ez szintén a későbbi továbbfejlesztések egyike lesz, mivel a hangterjedés sebessége függ a környezet hőmérsékletétől – ez szintén javíthatna a pontosságon.

További fejlesztési lehetőség a becsült hely folyamatos számítása, és grafikus megjelenítése, amelyet közel real-time alkalmazásban lehetne használni, így a rendszer különböző beavatkozó rendszerek irányítására is szolgálhat (pl. intelligens otthonban világítás szabályzása)

A későbbi tervek közt szerepel a kalibráció beiktatása, amelynek lényege a mote-ok aktuális pozíciójának automatikus meghatározása. Ez történhetne a rendszer indításakor, vagy felhasználói utasításra. Ezáltal a mote-ok elmozdulását, a topológia változást, a hálózati elemek számának változását és az esetleges környezetváltozásokat is érzékelhetjük.

A minőség javítására a mikrofonok típusait és tulajdonságait kívánom majd vizsgálni, amely az eddigiekben megfelelő hardverek hiányában nem valósult meg.

Jelfeldolgozás

A jelfeldolgozás javítása érdekében beépíthető lenne a számítógépen – vagy valamely mote-on – futó zajmérő algoritmus, amely a kis tér miatt a beltérben az állandó zavarokat képes lenne kiszűrni. Alkalmazható aktív zajszűrés, amely megfelelő (kiegészítő, vagy új) hardver esetén valósulhatna meg, pl. FPGA, vagy DSP (Digital Signal Processor).

A Matlab program további tökéletesítését is távlati feladatnak tekintem. A programba extra funkciók beépítése, állítható paraméterek beiktatása és a mote-ok helyzetének grafikus „drag and drop” módszerrel történő megadása is tervezett.

További megkötésként a mote-okról nem feltételeztem a helyváltoztatást, így nem volt szükség folyamatos feltérképezés futtatására. Állandó, de ismeretlen topológia esetén elegendő lenne az induláskor egy broadcast üzenet kiküldésére, és várni a válaszok beérkezésére. A későbbiekben nemcsak a mote-ok esetleges mozgása miatt, hanem új résztvevők működés közbeni felvétele miatt is szükségét látom a periodikus hálózati

lekérdezésnek. Az energiatakarékosság lényegesen javítható lenne, ha a szenzormote-ok időosztással figyelnék az esemény zajlását, majd bekövetkezéskor a többi mote-ot felébresztenék az ún. „wake up” rádióvevővel.

11 Összefoglalás

A diplomatervezés ideje alatt rengeteg ismerettel és tapasztalattal gazdagoztam, mind az alkalmazott hardver- és szoftverkörnyezet, mind a kapcsolódó elméleti ismeretek tekintetében.

A diplomamunka megismertetett (elsősorban a vezeték nélküli technológiát alkalmazó) szenzorhálózatokkal, jellegzetességeikkel és létező alkalmazásaikkal. A fejlesztés közben huzamosabb ideig használtam a Berkeley MICAz mote-okat, és a TinyOS operációs rendszert, amelyre NesC nyelven készültek a kódok. A platform minden előnyével és hátrányával együtt jól használható. Egy esetleges saját szenzorhálózati eszköz tervezése esetén sok tulajdonságban az általam használt mote-okon lévőhöz hasonló megoldásokat választanék, mivel kis mérete, alacsony fogyasztása, moduláris felépítése, és ár/érték aránya kedvezőnek mondható.

A fejlesztés alatt fejlődtem a Matlab ás a mikrokontrollerek programozásában is. Megismerkedtem különböző szinkronizációs és helymeghatározási technikákkal, amelyek közül választottam egy megfelelőt, majd kisebb-nagyobb változtatásokkal implementáltam azt. A szinkronizáció rengeteg alkalmazás elengedhetetlen része, amely bizonyos lokalizációs módszerek alapját képezi, és amelynek minősége meghatározza a helymeghatározás minőségét is. A megvalósított beérkezési időkülönbség mérés alapú lokalizáció hangalapú technikáknál általánosan elterjedt. A mérések megmutatták a technika határait a rendelkezésre álló eszközökkel, és a rendszer tesztelése közben megmutatták a pontatlanságokat és hiányosságokat.

Az elkészült helymeghatározó rendszer összességében jól használható, ám a távolság becslésére csak speciális esetben biztosít pontos értékeket. A pozíció irányának meghatározására ellenben igen jó eredmény mutatott.

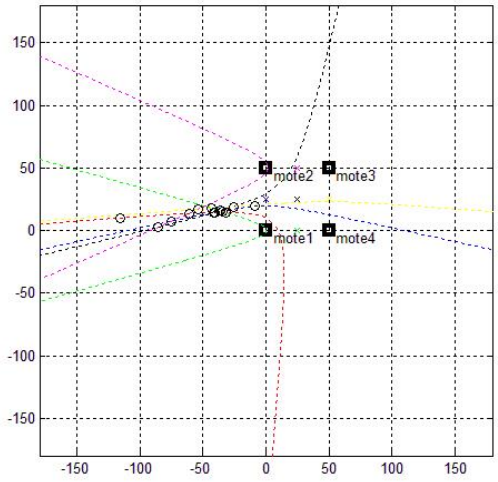
A munka alatt számomra kifejezetten érdekes és hasznos területekkel foglalkoztam, amelyektől remélhetőleg a későbbiekben sem távolodok el.

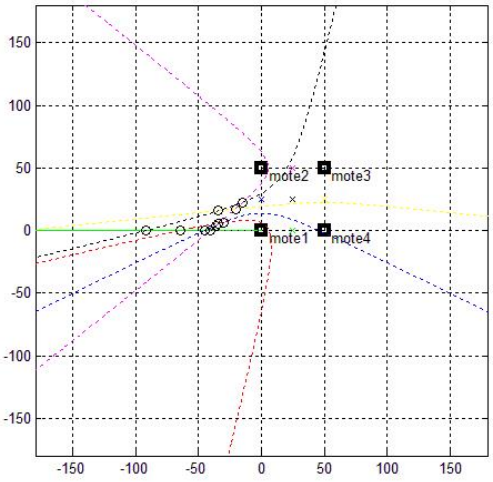
Függelék

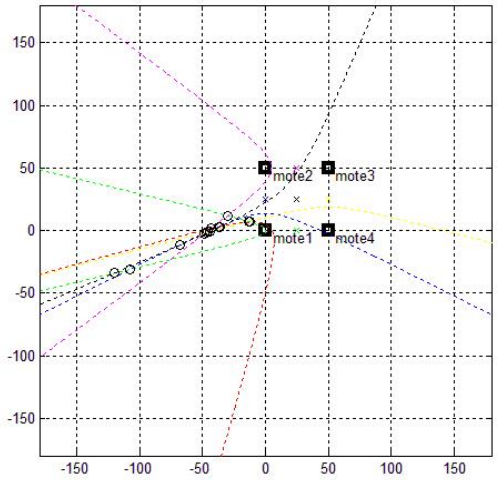
Pontosságra vonatkozó mérések

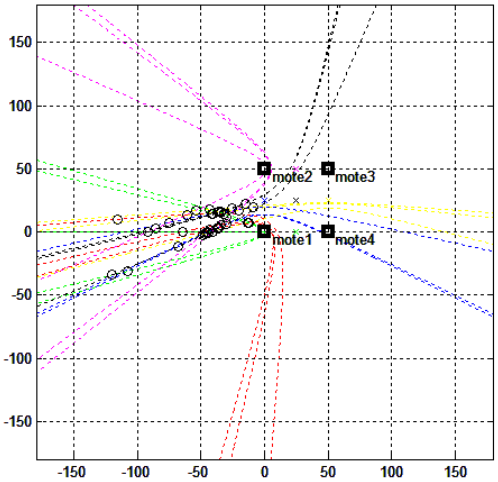
Az alábbi mérési eredményeknél a hangforrás eredeti pozíciója mellett 3 mérési eredmény, majd azok összesítése látható. A tengelyek ezen grafikonoknál cm-ben értendők.

Hangforrás pozíció: -50 cm, 0 cm

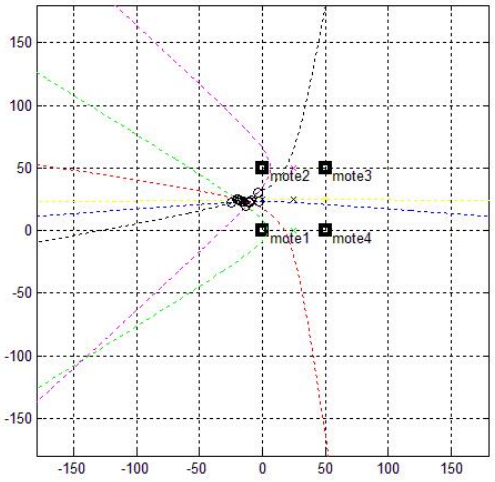


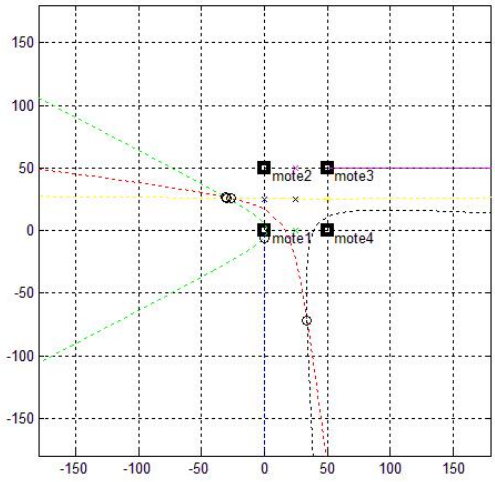


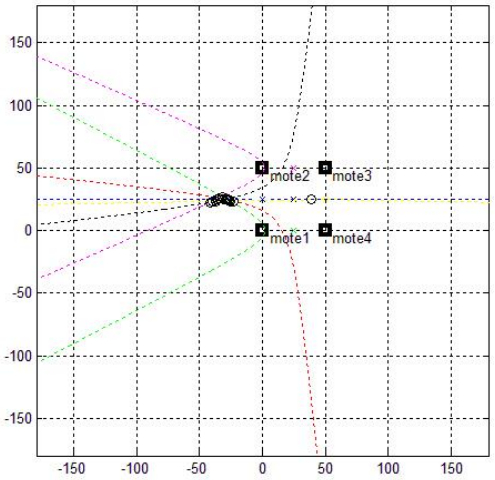


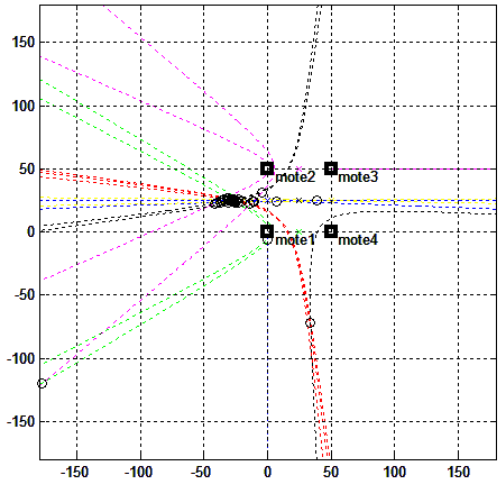


Hangforrás pozíció: -50 cm, 25 cm

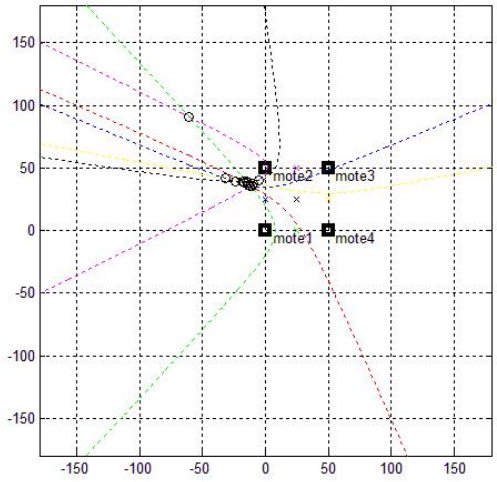


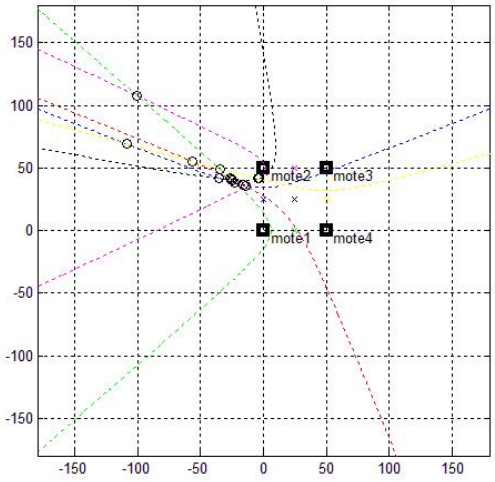


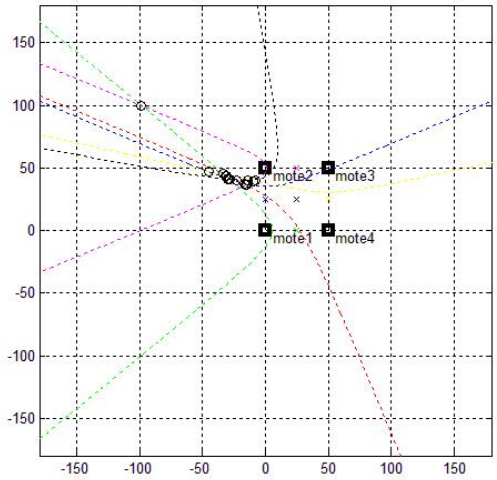


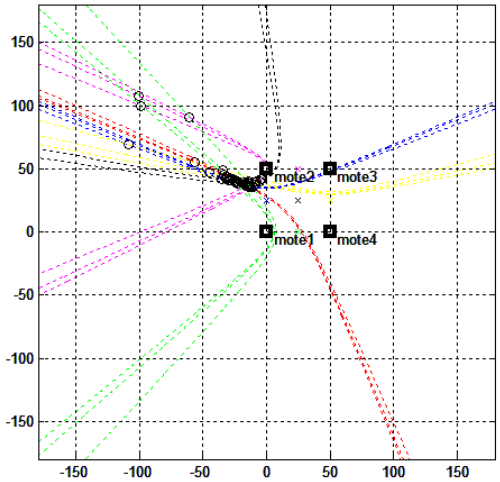


Hangforrás pozíció: -50 cm, 50 cm

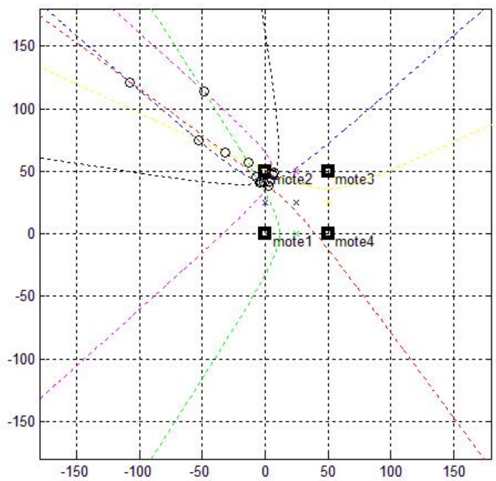


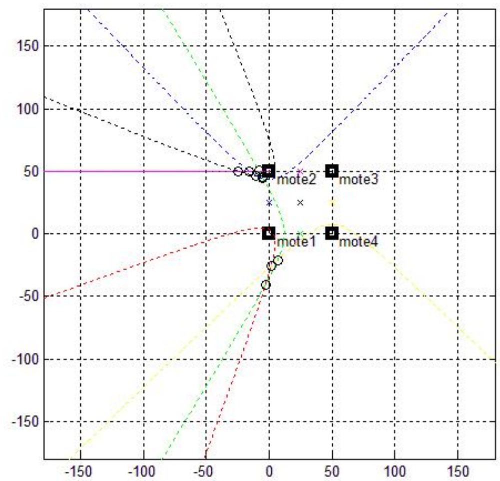


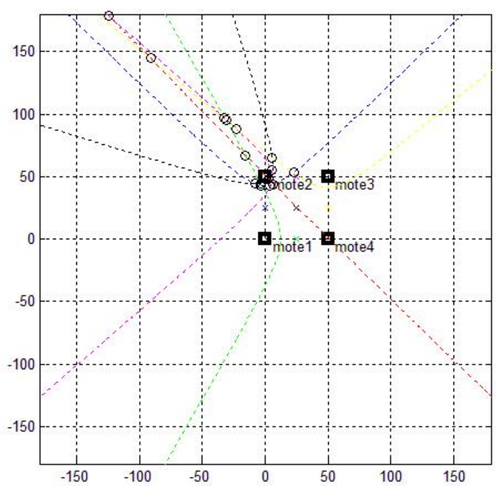


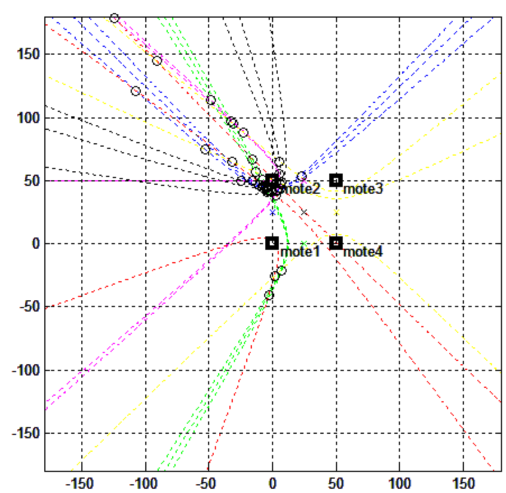


Hangforrás pozíció: -50 cm, 100 cm

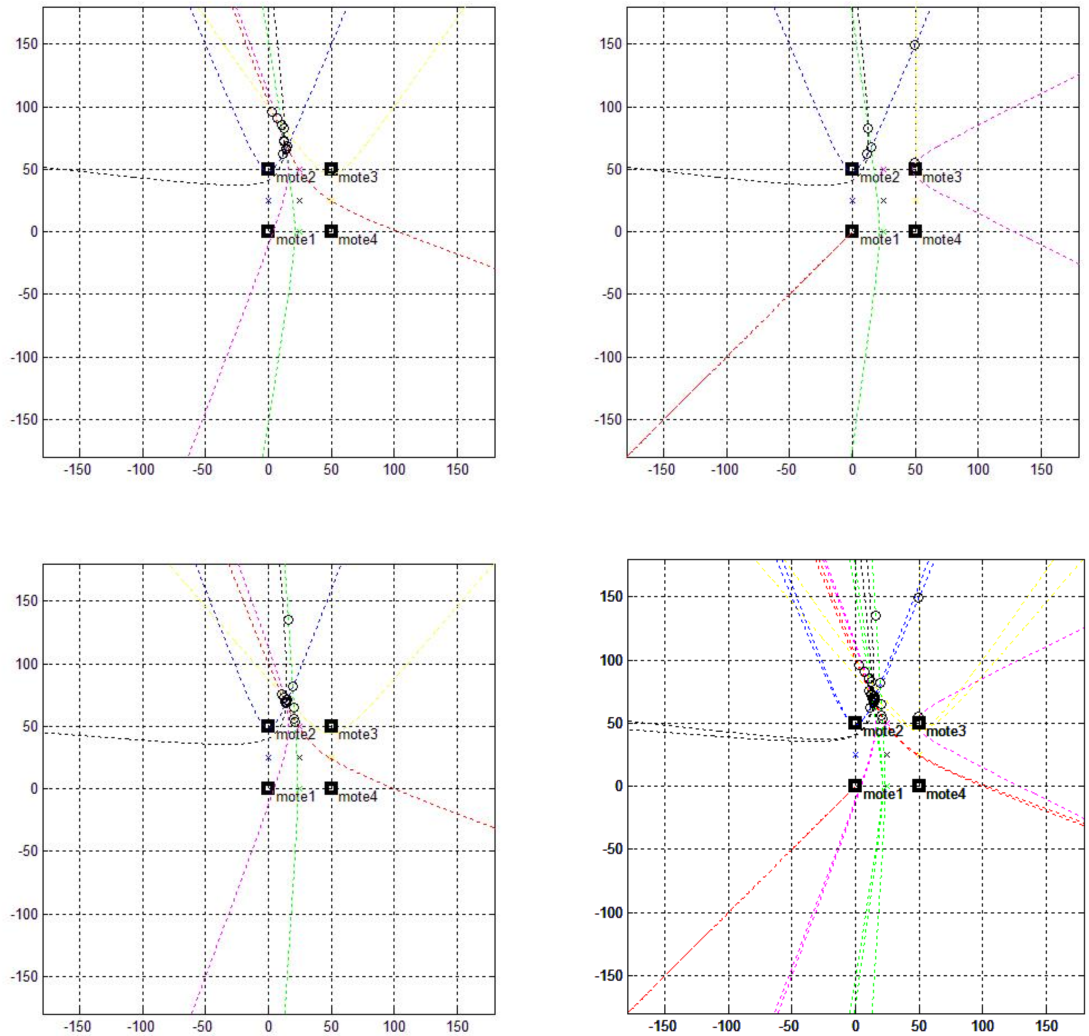




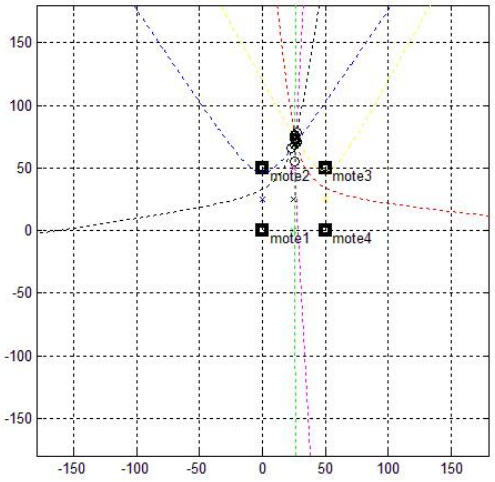


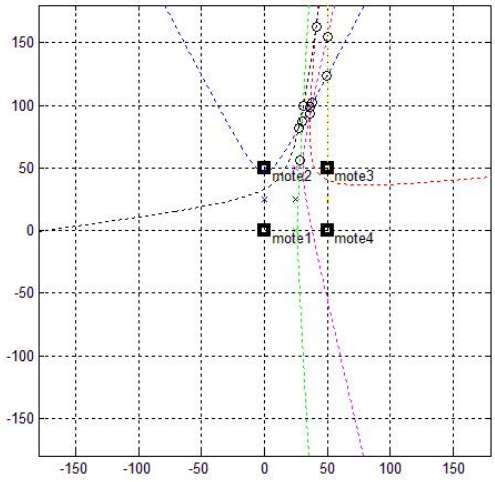


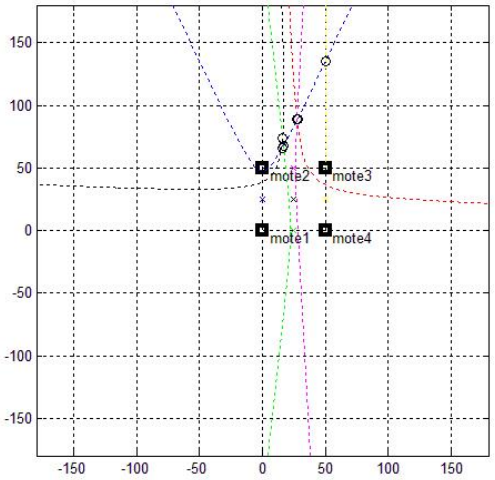
Hangforrás pozíció: 0 cm, 100 cm

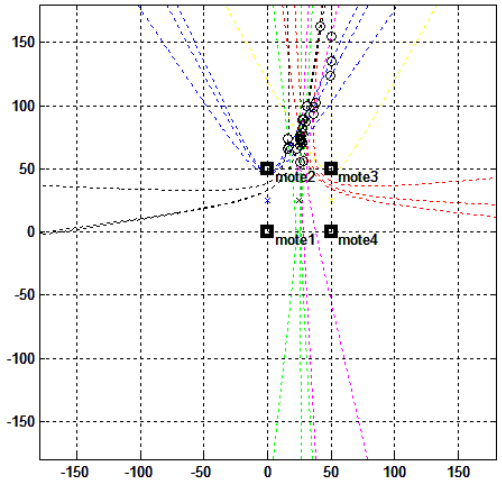


Hangforrás pozíció: 25 cm, 100 cm

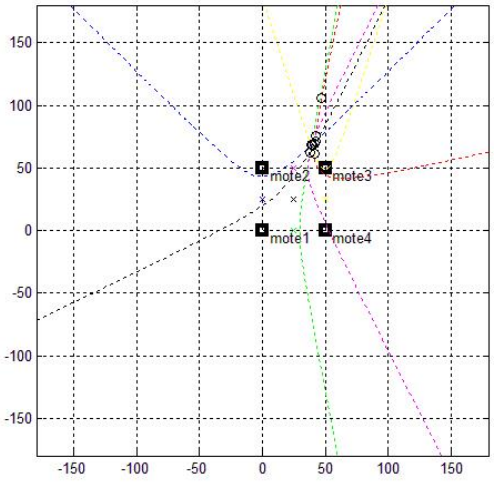


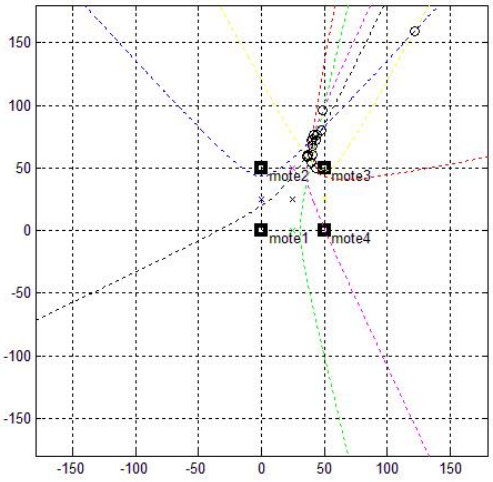


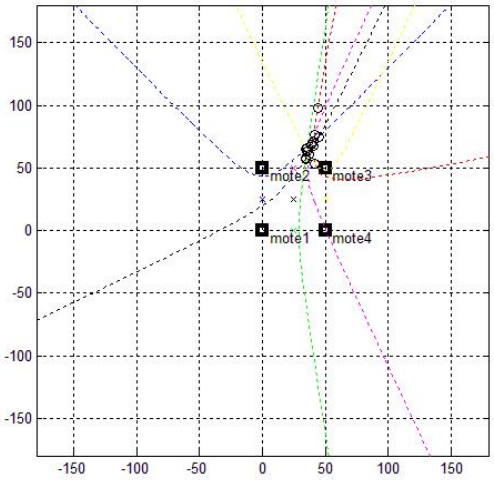


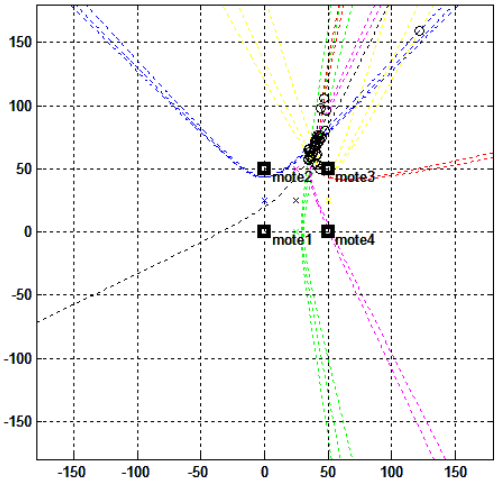


Hangforrás pozíció: 50 cm, 100 cm

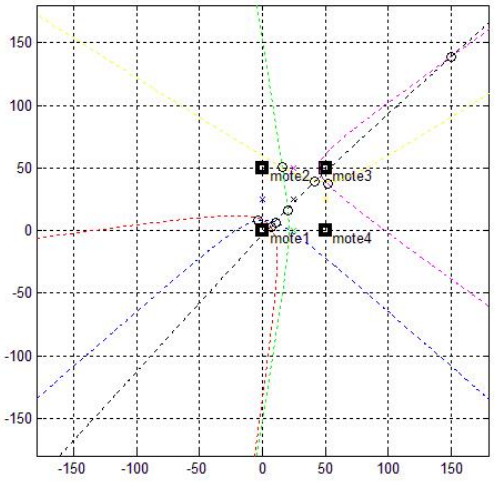


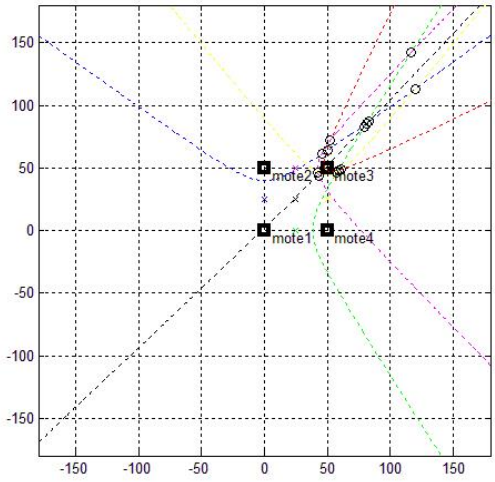


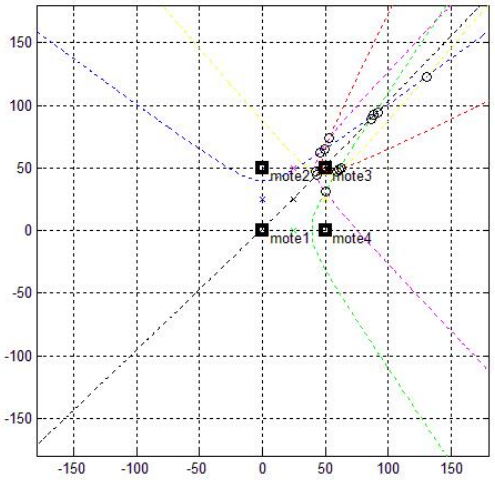


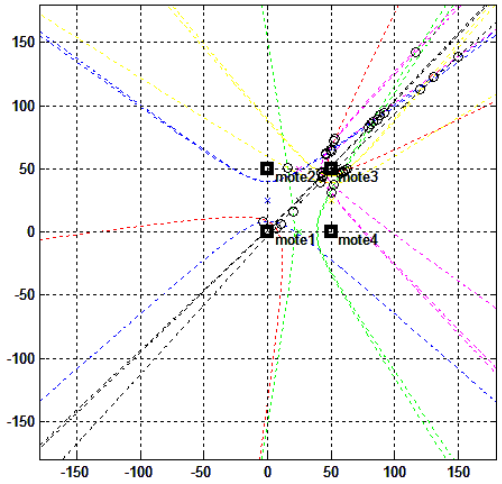


Hangforrás pozíció: 100 cm, 100 cm

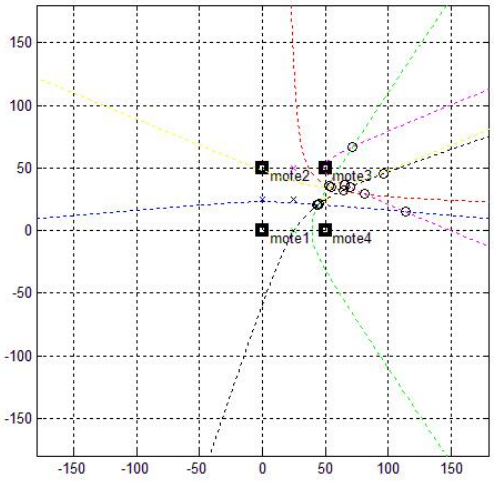


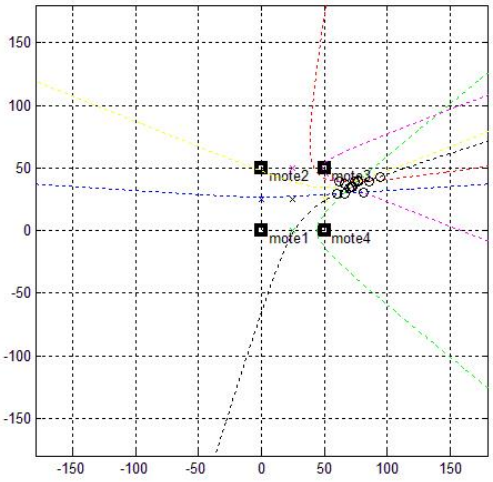


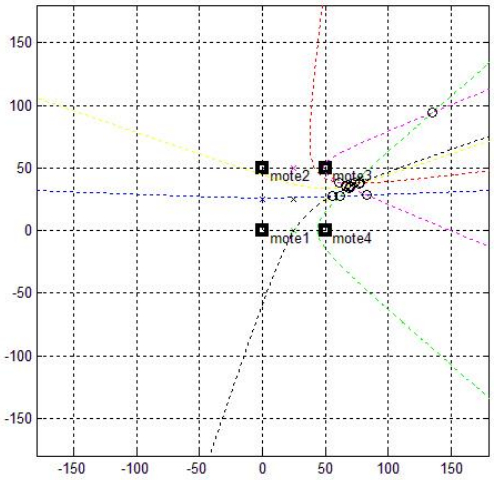


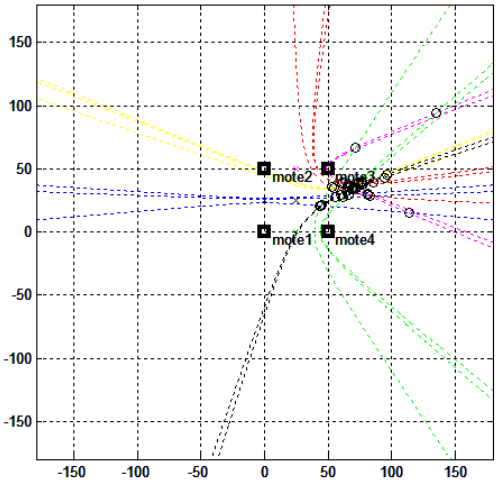


Hangforrás pozíció: 100 cm, 50 cm

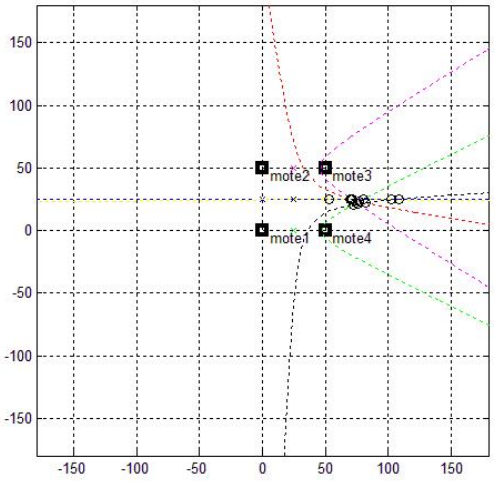


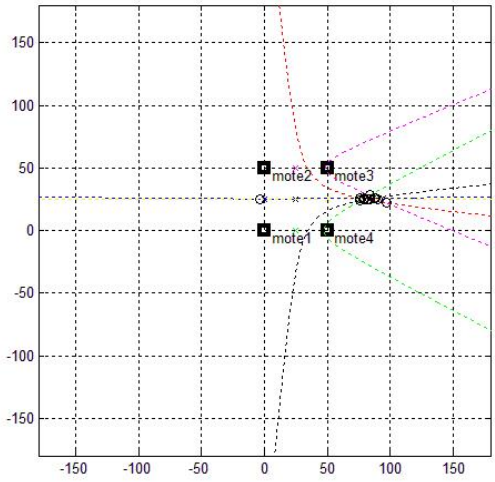


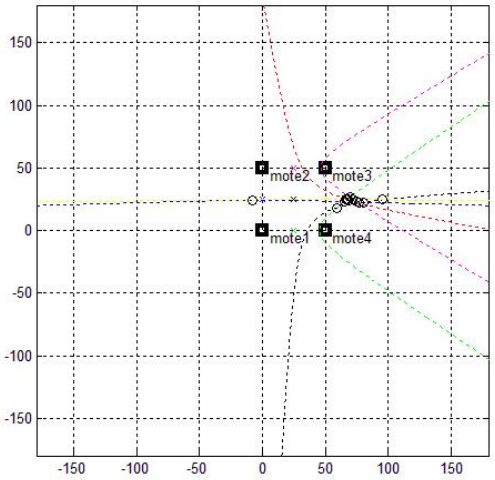


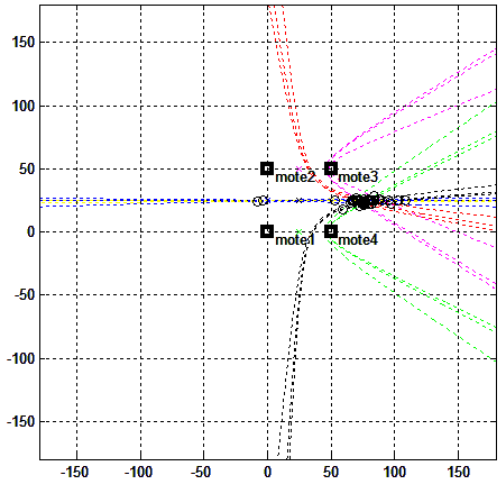


Hangforrás pozíció: 100 cm, 25 cm

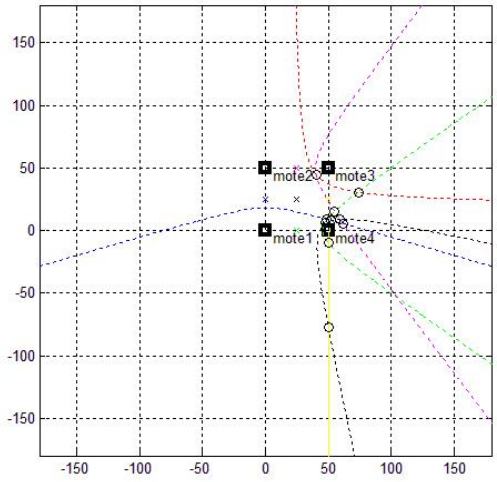


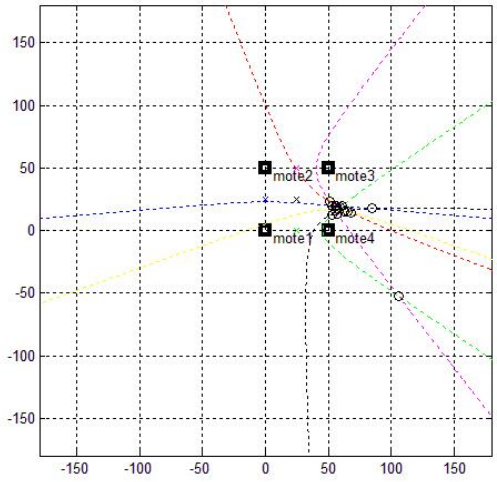


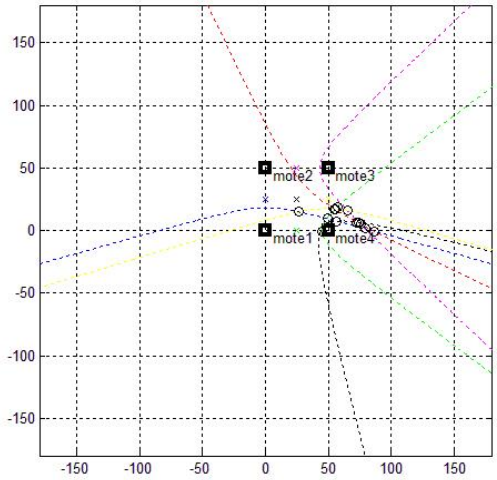


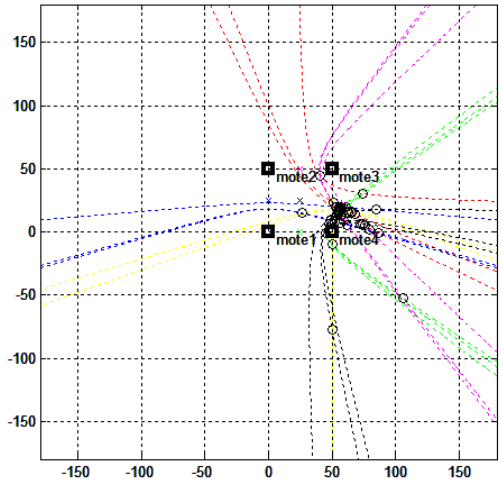


Hangforrás pozíció: 100 cm, 0 cm

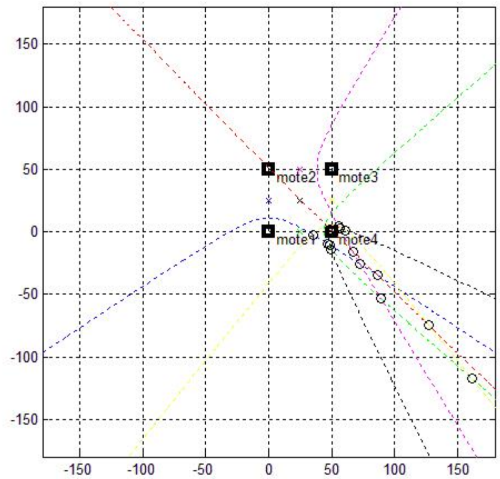


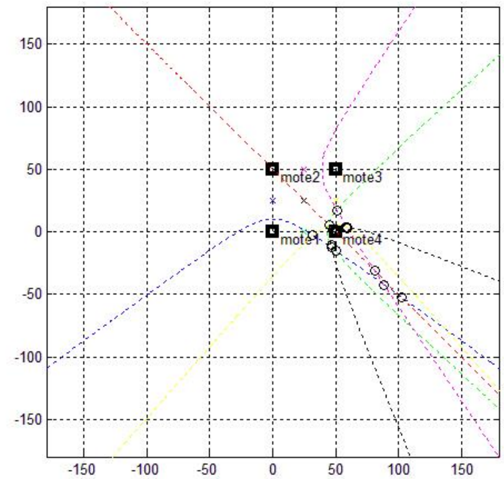


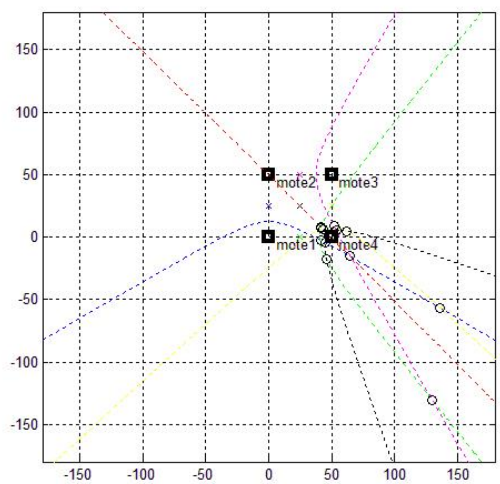


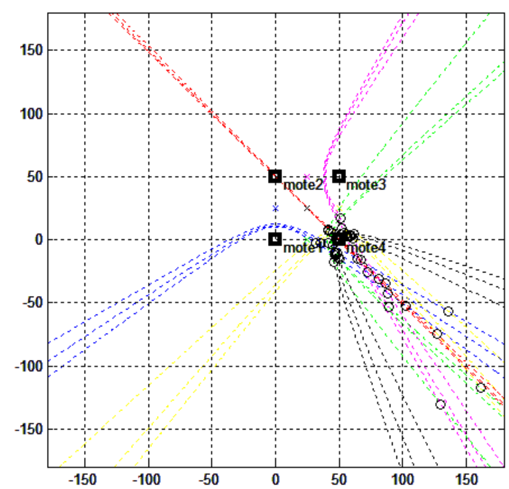


Hangforrás pozíció: 100 cm, -50 cm

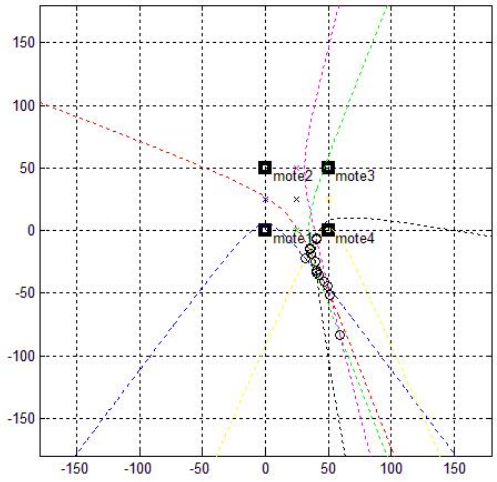


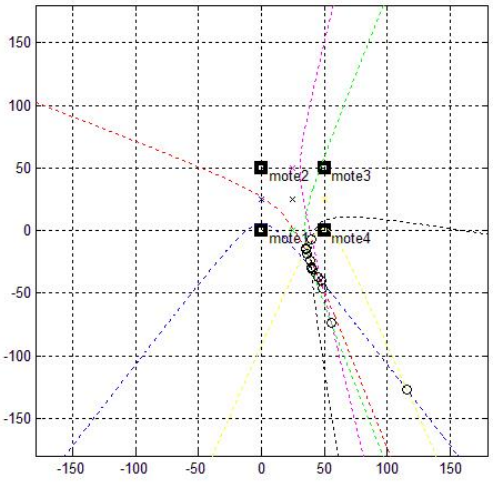


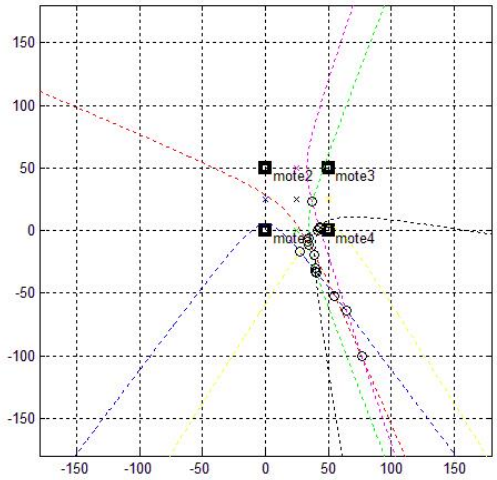


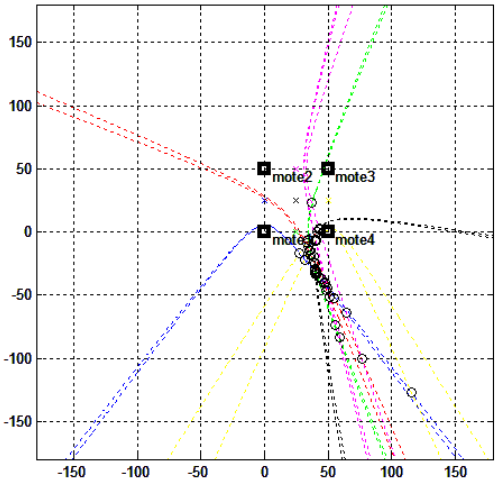


Hangforrás pozíció: 50 cm, -50 cm

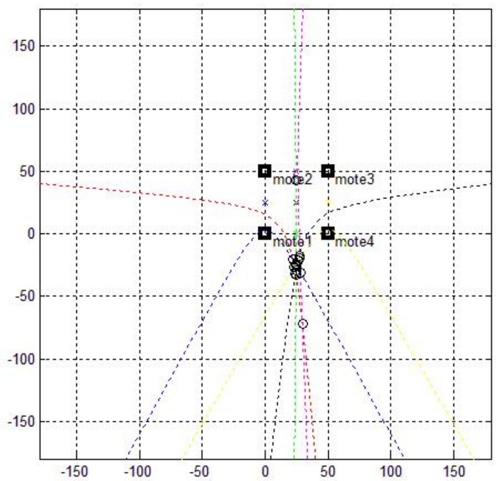


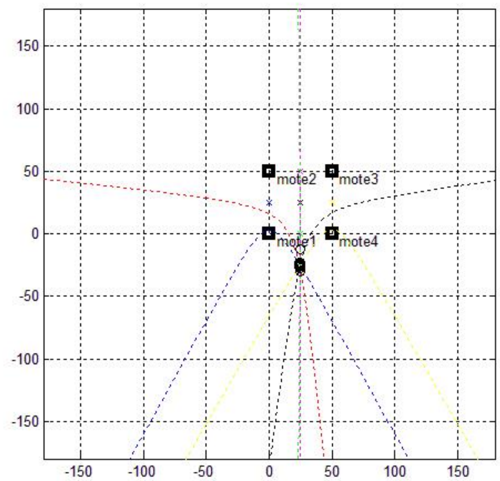


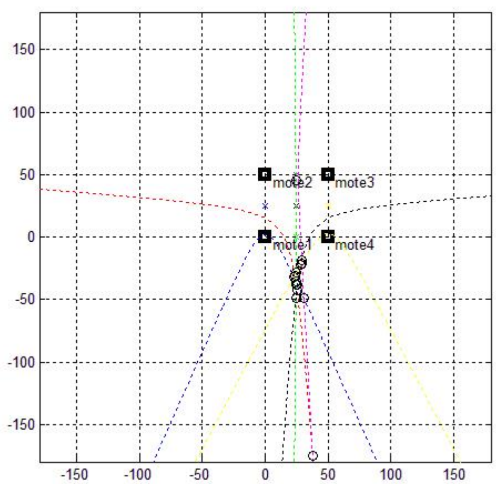


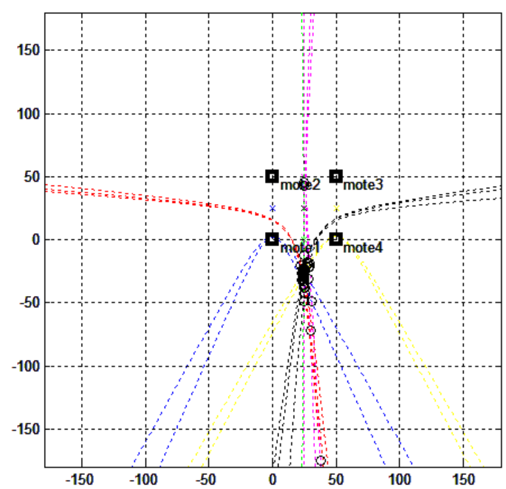


Hangforrás pozíció: 25 cm, -50 cm

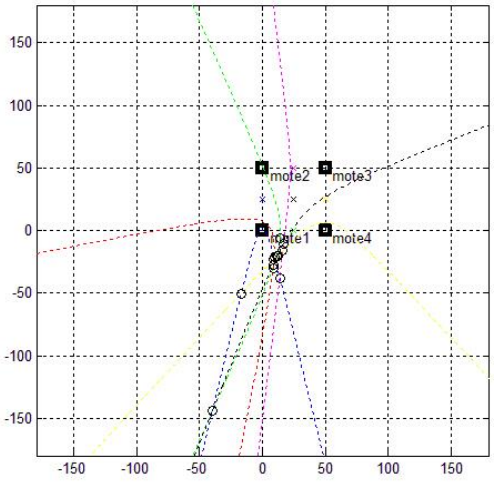


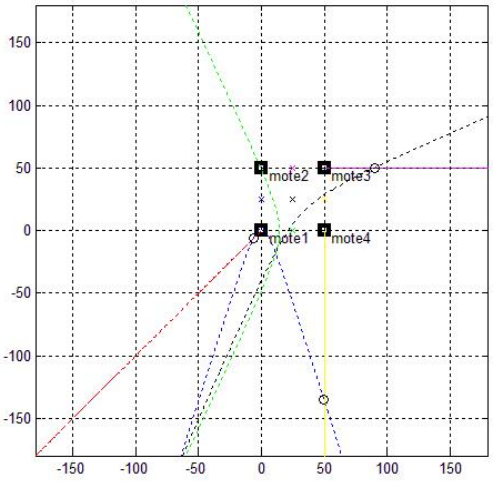


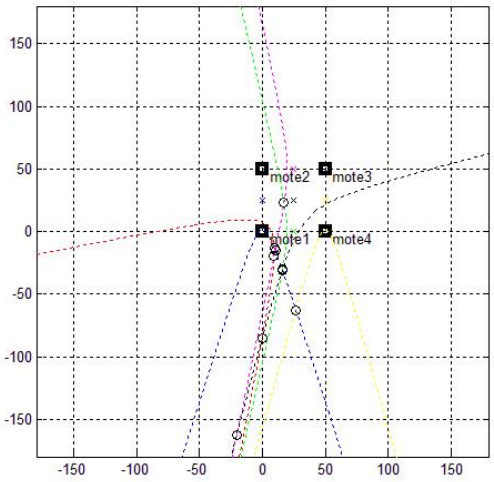


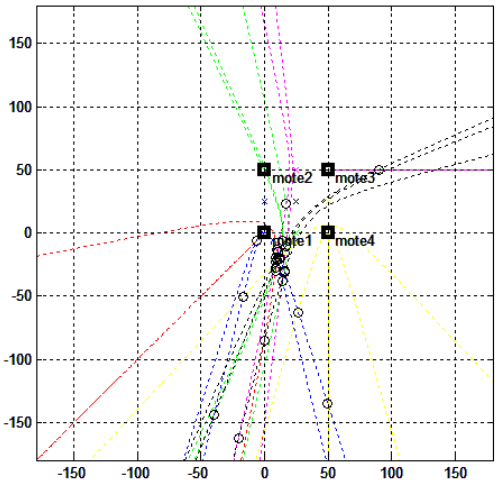


Hangforrás pozíció: 0 cm, -50 cm

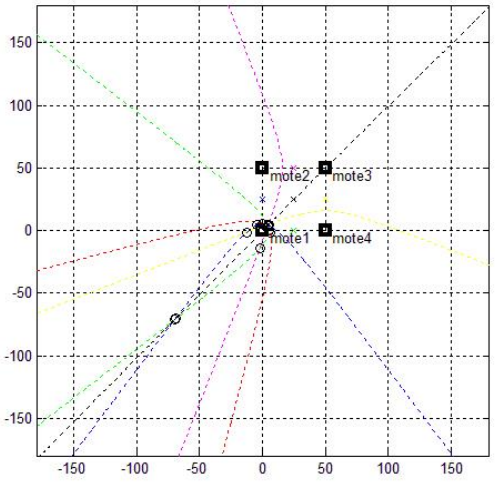


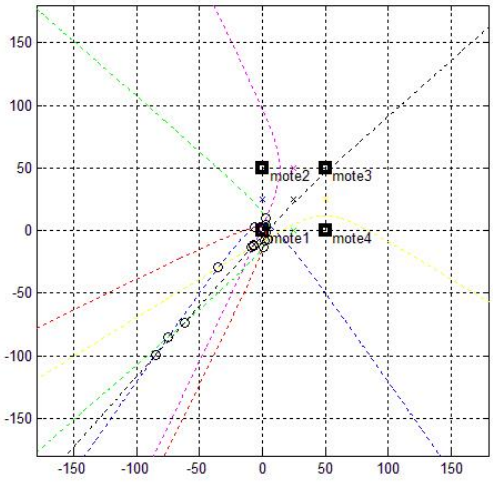


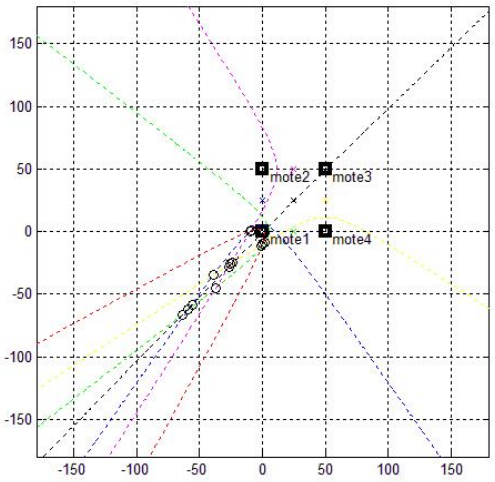


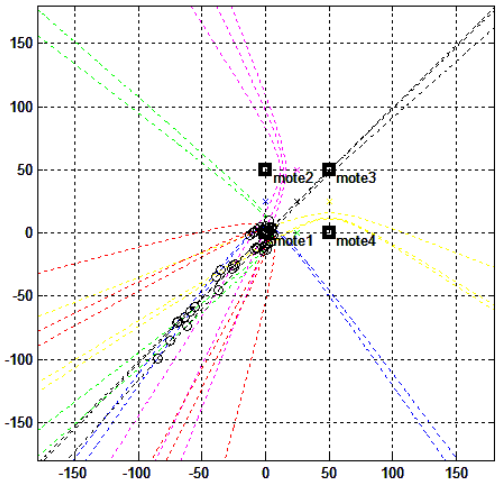


Hangforrás pozíció: -50 cm, -50 cm

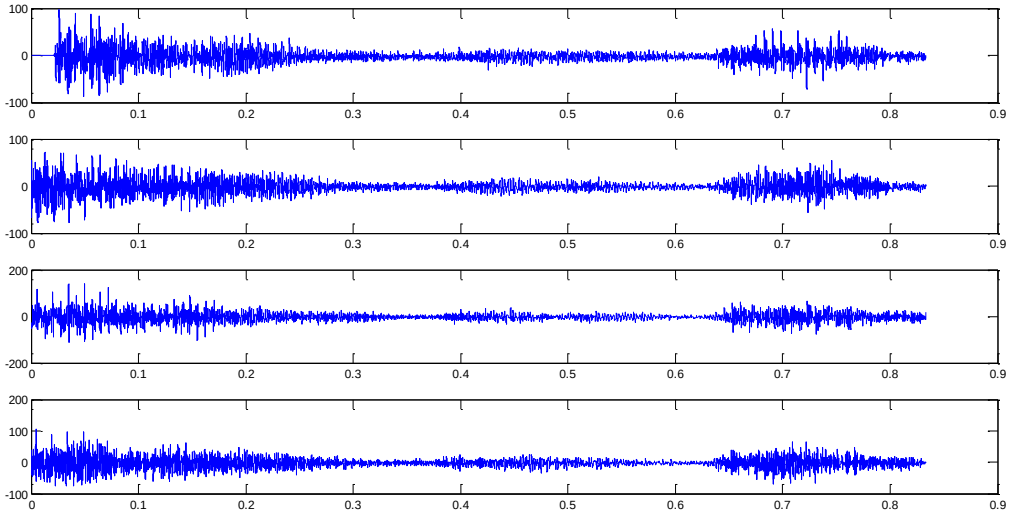




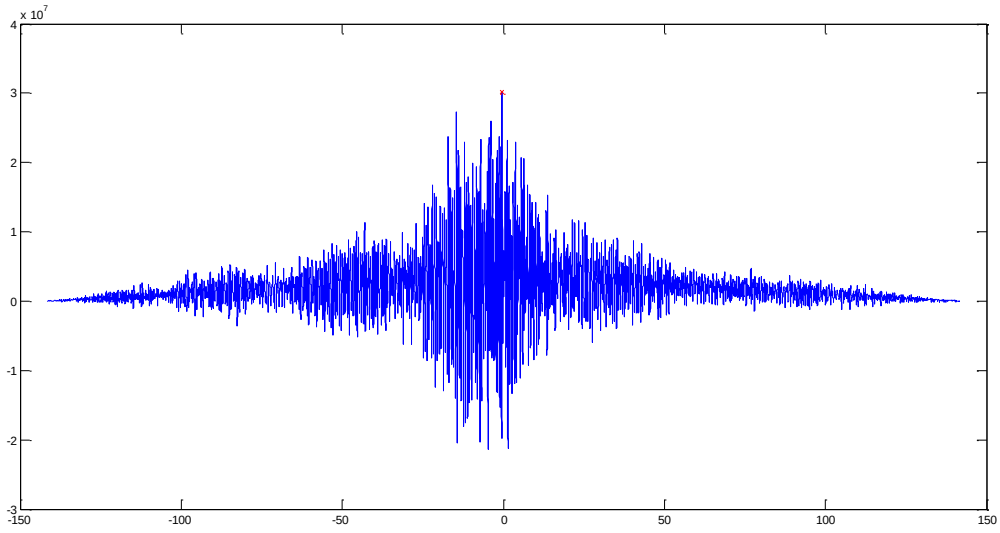




Q

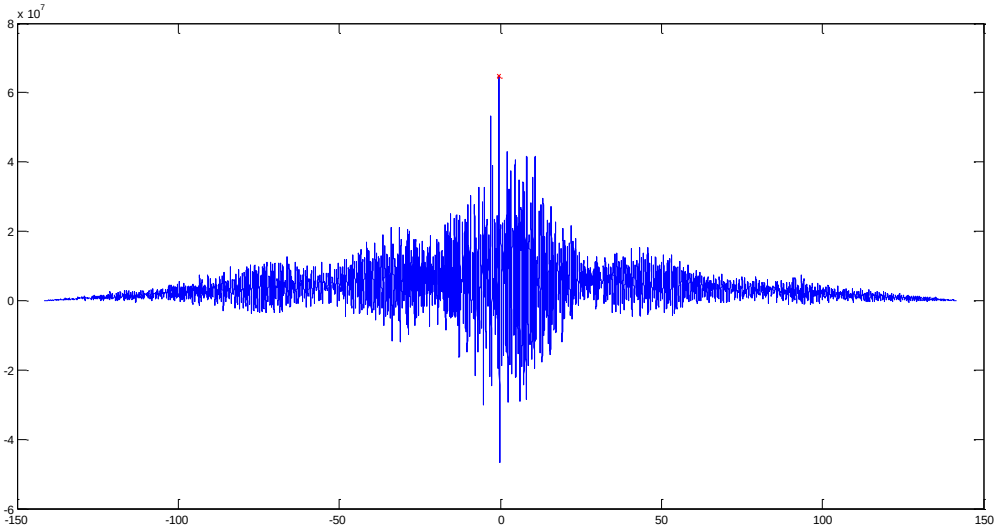


30. ábra: Egy valós mérés hangmintái (x: Mintaszám – db, y: Ampitúdó)



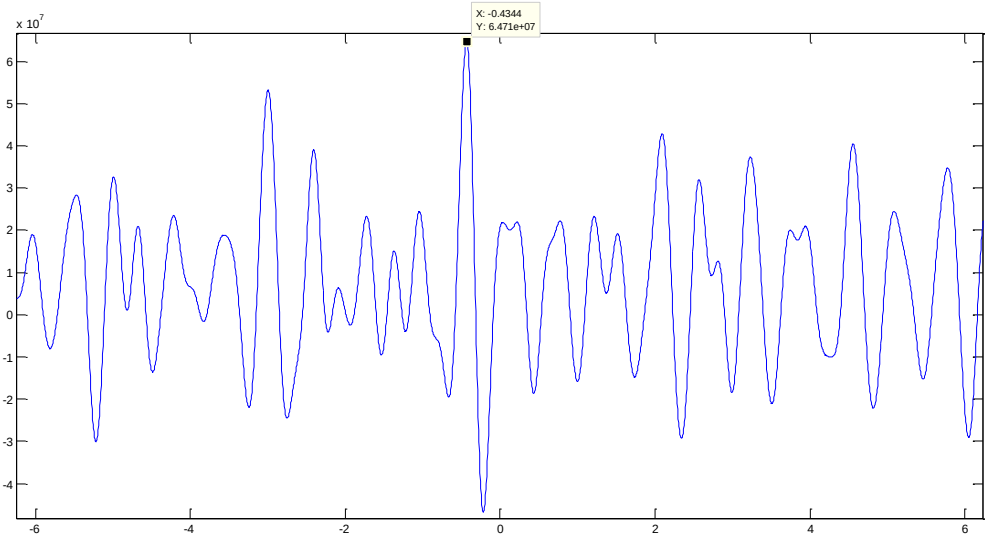
31. ábra: Az 1-2 mote adatainak korrelációs függvénye

(x: keresztkorrelációs függvény pontjai, y: amplitúdó)



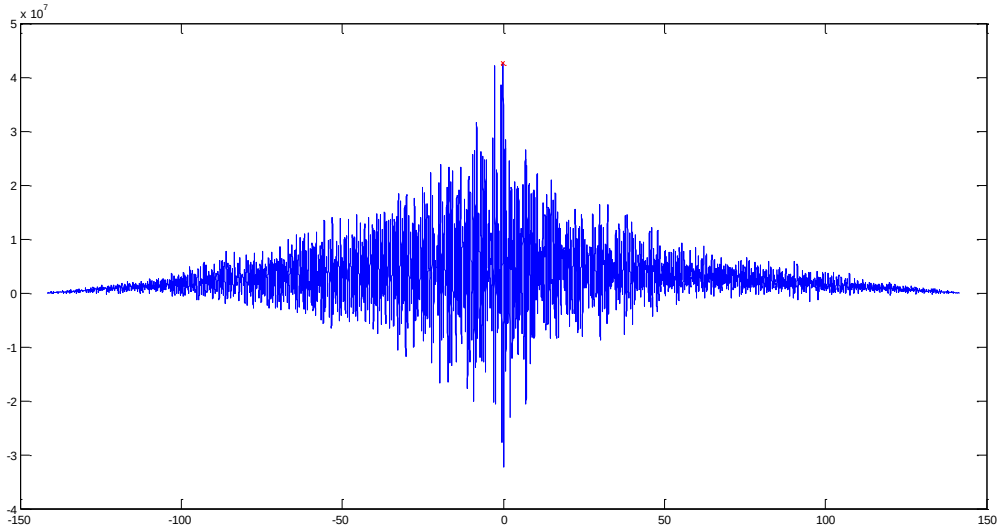
32. ábra: Az 1-3 mote adatainak korrelációs függvénye

(x: keresztkorrelációs függvény pontjai, y: amplitúdó)



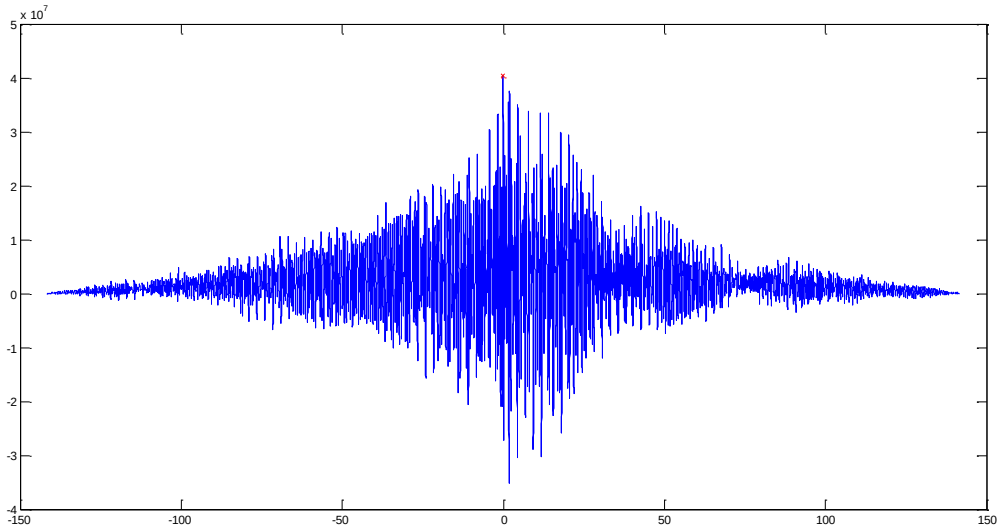
33. ábra: Az 1-3 mote adatainak korrelációs függvényének maximuma

(x: távolság – cm, y: amplitúdó)



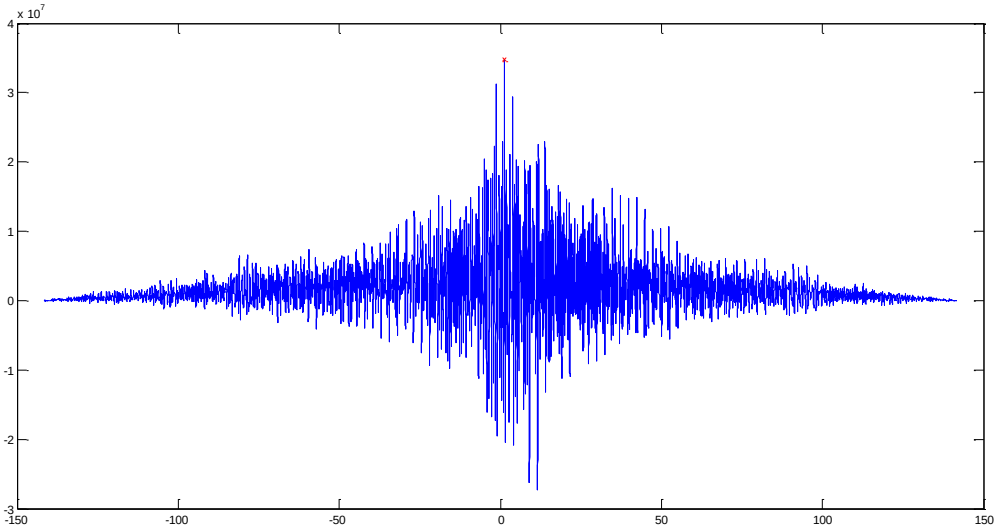
34. ábra: Az 1-4 mote adatainak korrelációs függvénye

(x: keresztkorrelációs függvény pontjai, y: amplitúdó)



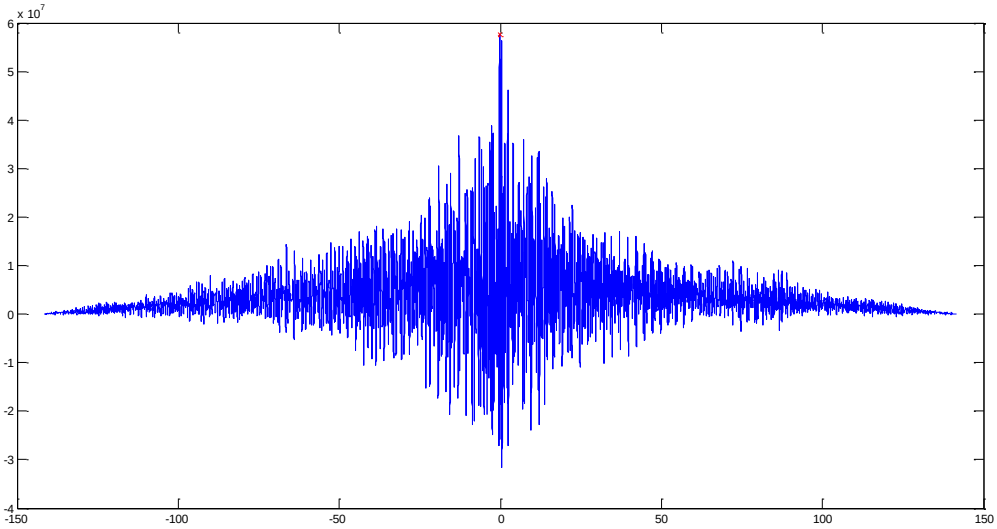
35. ábra: A 2-3 mote adatainak korrelációs függvénye

(x: keresztkorrelációs függvény pontjai, y: amplitúdó)



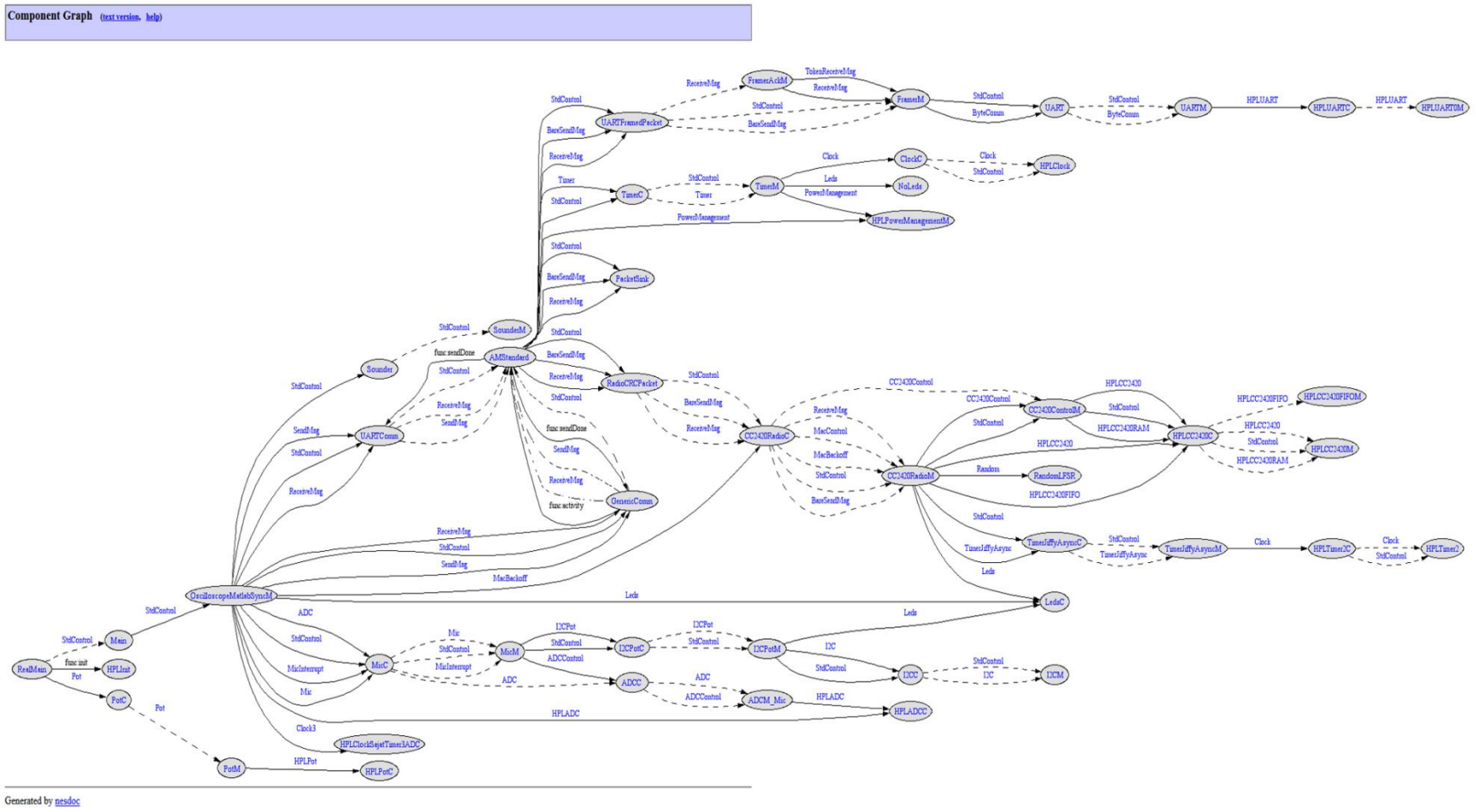
36. ábra: A 2-4 mote adatainak korrelációs függvénye

(x: keresztkorrelációs függvény pontjai, y: amplitúdó)

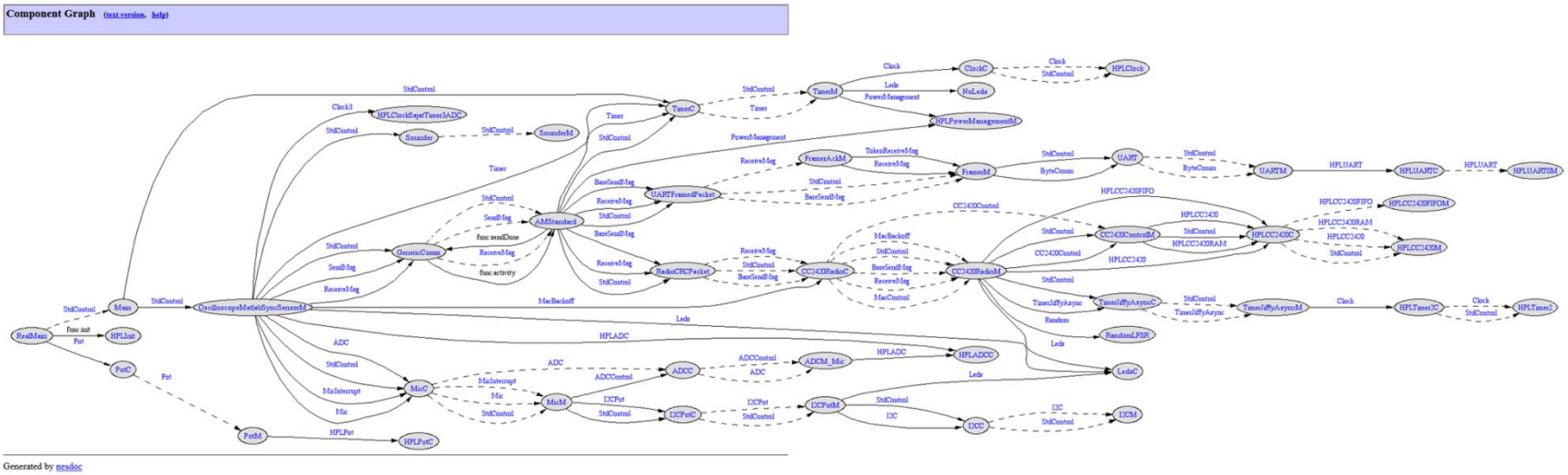


37. ábra: A 3-4 mote adatainak korrelációs függvénye

(x: keresztkorrelációs függvény pontjai, y: amplitúdó)



38. ábra: A bázison futó program



39. ábra: A szenzorokon futó program

Ábrajegyzék

Z

Irodalomjegyzék

[1] *A MICA mote-okhoz tartozó bővítőkártyák felhasználói kézikönyve*
MTS-MDA_Series_Users_Manual.pdf
Crossbow Technology, Inc.
xbow.com

[2] *A Zigbee szabvány részletes bemutatása*
Kovács B., Vida R.
zigbee.pdf

[3] *A MICAz adatlapja*
MICAZ_datasheet.pdf
Crossbow Technology, Inc.
xbow.com

[4] *A crossbow mote-ok bemutatása*
Drexel University
http://www.pages.drexel.edu/~kws23/tutorials/motes/motes.html

[5] *A crossbow cég rövid története*
The Free Library
http://www.thefreelibrary.com/Crossbow%27s+New+MICAz+Mote+Enables+Audio,+Video,+and+Other+High...-a0117927513

[6] *A mesh hálózatok ismertetése*
The Free Library
http://encyclopedia.thefreedictionary.com/Mesh+networking

[7] *Szenzorhálózatok és alkalmazásaik*
Vidács Attila, Vida Rolland
www.tmit.bme.hu/dl396

[8] *A continuous remote emotional health monitoring system for depressive illness*
Robert F. Dickerson, Eugenia I. Gorlin, John A. Stankovic
http://www.cs.virginia.edu/~stankovic/psfiles/robempath.pdf

[9] *AALO: Activity recognition in smart homes using Active Learning in the presence of Overlapped activities*
Enamul Hoque, John Stankovic
http://www.cs.virginia.edu/people/faculty/pdfs/AALO.pdf

[10] *Beginner's guide to crossbow motes*
Keith Sevcik, Ph.D.
http://www.pages.drexel.edu/~kws23/tutorials/motes/motes.html

[11] *Szenzorhálózatok*
Völgyesi Péter
http://www.volgy.com/pubs/BIR_SensorNetworks.pdf

[12] *A GPS működése*
http://gps-teruletmeres.blog.hu/2009/02/25/a_gps_mukodese

[13] *Szenzorhálózatok: Lokalizáció és nyomkövetés, mobilitás*
Vidács Attila
http://www.tmit.bme.hu/dl374%E2%80%8E

[14] *Mik a szenzorhálózatok*
Simon Gyula
http://www.dcs.vein.hu/~simon/SH/

[15] *Shooter localization*
Ákos Lédeczi
http://w3.isis.vanderbilt.edu/projects/nest/applications.html

[16] *An Analysis of a Large Scale Habitat Monitoring Application*
R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson and D. Culler
http://www.ics.uci.edu/~dsm/ics280sensor/readings/apps/sensys04-gdi.pdf

[17] *Vulcano monitoring (more projects)*
Prof. Matt Welsh, Prof. Jeff Johnson, Prof. Jonathan Lees
http://fiji.eecs.harvard.edu/Volcano

[18] *TinyOS 1.x tutorial lesson 6*
David Gay, Philip Lewis
http://www.tinyos.net/tinyos-1.x/doc/tutorial/lesson6.html

[19] *Applications of Function Handles*
The Mathworks, Inc.
http://www.mathworks.com/help/matlab/matlab_prog/applications-of-function-handles.html

[20] *PPP in HDLC-like Framing*
W. Simpson
http://www.ietf.org/rfc/rfc1662.txt

[21] *Óraszinkronizáció szenzorhálózatokban*
Gy. Orosz
http://www.mit.bme.hu/system/files/oktatas/targyak/7613/oraszinkron_alap_09_I.pdf

[22] *Elosztott rendszerek és szenzorhálózatok*
Gy.Orosz
http://www.mit.bme.hu/system/files/oktatas/targyak/8605/meres10_ElosztottRendszEsWSN2_2012.pdf

[23] *„Wireless sensor networks: A survey”*
Akyildiz, I. F., W. Su, Y. Sankarasubramaniam, and E. Cayirci
Comput. Netw., vol. 38, no. 4, pp. 393-422