Diplomaterv

Adaptív jelfeldolgozó eljárások megvalósítása szenzorhálózatban

Orosz György
2006

Nyilatkozat

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

.....

Orosz György

Tartalomjegyzék

Kivonat

A szenzorhálózatok alkalmazása manapság számos területen utat tört magának, azonban ezek a rendszerek főleg mérésadatgyűjtő funkciót látnak el. Az on-line feldolgozást igénylő gyors jelfeldolgozási és szabályozási algoritmusokban használatuk még nem kiforrott. Mivel az aktív zajcsökkentő rendszerek olyan szabályozási körök, melyek főleg adaptív jelfeldolgozó algoritmusokra épülnek, így a szenzorhálózatok aktív zajcsökkentő rendszerekben történő alkalmazása lehetőséget kínál ezen témakör tanulmányozására. Mivel az aktív zajcsökkentő rendszerekben az elnyomandó zaj érzékelésére általában sok mikrofont használnak, ezért a szenzorhálózatok alkalmazása gyakorlati szempontból is megalapozott, hiszen a szenzorok telepítése a jelvezetékek elhagyása miatt egyszerű, és az elrendezés dinamikusabban változtatható.

A szenzorhálózatban egymástól független egységek (mote-ok) végzik a zaj mintavételezését. Annak ellenére, hogy a névleges mintavételi frekvenciák egyenlők, a mintavételi időpontok elcsúsznak egymáshoz képest, mivel a mintavételezést időzítő egységek órajelét szolgáltató oszcillátorok frekvenciái a valóságban nem egyeznek meg pontosan. Az oszcillátorok frekvenciájában jelenlévő kis mértékű eltérés is hosszú távon a mintavételi időpontok jelentős elcsúszásához vezet, mely a rendszer instabilitását okozza. Ez szükségessé teszi a mintavételezés szinkronizálását. Ennek kapcsán a dolgozatban áttekintést adok a szinkronizációval kapcsolatos alapelvekről és a megoldási lehetőségekről. A zajérzékelő hálózatban jelentkező problémák elemzése és a lehetséges megoldások áttekintése után bemutatom a hálózatra kidolgozott szinkronizációs algoritmust.

Amennyiben a zajérzékelő hálózat csupán a zajminták egyszerű továbbítását végzi, akkor a hálózat korlátozott adatátviteli képessége miatt a mote-ok számának növelése nem lehetséges a mintavételi frekvencia csökkentése nélkül a megnövekedő adatmennyiség miatt. Megoldásként kidolgozásra került egy olyan, periodikus jelek elnyomására alkalmas elosztott jelfeldolgozást megvalósító rendszer, melyben a mote-ok a zajminták rekurzív feldolgozásával előállítják a zaj Fourier-együtthatóit. Mivel a Fourier-együtthatók általában lassabban változnak, mint maga a jel, így továbbításuk ritkábban is lehetséges. Az érzékelők száma ezért nem jelent korlátozást a mintavételi frekvenciára, tehát a mintavételi tétel miatt az elnyomandó zaj sávszélességére sem.

A dolgozatban kitérek a rendszerek megvalósításával kapcsolatos részletekre is. A zajcsökkentő algoritmusok implementálásához egy ADSP-21061 EZ-KIT Lite típusú DSP fejlesztői kártyát, a szenzorhálózat építőelemeiként MICAz mote-okat alkalmaztunk, melyek 1.8 kHz-es mintavételi frekvenciával üzemeltek. Az egyszerű mintagyűjtő hálózatban két szenzort, míg az elosztott jelfeldolgozással üzemelő rendszerben négy szenzort alkalmaztunk. Az utóbbi esetben azonban lehetőség kínálkozik akár 10-20 szenzor alkalmazására is. Korlátot a DSP számítási teljesítménye, és a zaj Fourier-együtthatóinak változási sebessége szabhat.

A kész berendezéseken végzett mérések eredményei bizonyítják a szinkronizációval kapcsolatos elméleti megfontolásokat, és a zajcsökkentő rendszer működőképességét. A zajcsökkentés mértéke periodikus jelek első néhány harmonikusára nézve meghaladta a 20-30 dB-t.

A diplomaterv kidolgozása során tehát sikerült egy olyan, adaptív jelfeldolgozási algoritmusokon alapuló szabályozási rendszert létrehozni, melyben a visszacsatoló jelek továbbítása szenzorhálózat segítségével történt.

Abstract

Nowadays wireless sensor networks (WSN) are successfully used in many applications as data acquisition units. In fast control- and digital signal processing systems, however, their application is still in experimental phase. Since the active noise control (ANC) systems are such control loops, which are based on adaptive signal processing algorithms, the utilization of WSN in active noise control systems offers opportunity for investigation on the above mentioned field. The ANC systems usually comprise many microphones for sensing the noise to be suppressed. Because of the wireless data transfer the installation and reconfiguration of the system can be simpler compared to such ANC systems, to which sensors are attached by wires.

In the WSN autonomous sensors, so-called motes, sample the output signal of noise sensing microphones. Although the nominal sampling frequencies are equal on each mote, the time instants of the sampling on the different motes are slipping away from each other because of the timing error of sampling. It is caused by the slight differences in the frequency of quartz oscillators of motes. On long terms it results in the instability of the whole system. This makes necessary the synchronization of sampling.

To solve this problem, we consider the theoretical and practical basics of synchronization. After the analysis of the synchronization problems, and the possible solutions in the noise sensing network, the synchronization algorithm is presented, that was worked out in order to synchronize the time instants of the sampling.

If the WSN performs only the transmission of row data, the expansion in the number of motes is not possible without the reduction of sample rate. It is because of the limited bandwidth of the radio channel, and the increased amount of collected data. As a solution for this problem, also a distributed signal processing system was implemented, in which motes provides the Fourier-coefficients by recursive processing of noise samples. Since these coefficients usually change slower than the noise signal itself, their transmission rate can be reduced. The preprocessing of noise samples made possible to increase the number of motes without decreasing the sampling frequency. According to the sampling theory the bandwidth of the noise to be suppressed is not limited by the number of motes.

In this work the details of the realization of ANC systems are introduced, as well. The ANC algorithms are implemented on ADSP-21061 EZ-KIT Lite evaluation board, and the sensor network is made up of MICAz motes that operate at the sampling frequency of 1.8 kHz. In the simple data collecting network two sensors, while in the distributed signal processing system four motes are used. In the latter case the number of motes could be expanded about up to 10-20 motes. The number of motes is limited only by the computational capacity of the DSP, and the rate of change of Fourier-coefficients.

The measurements, performed on the systems, prove the efficiency of the synchronization algorithm, and the active noise control system. For periodic noise 20-30 dB suppression for several harmonic components of the noise is achieved.

This work introduced successful implementations of adaptive algorithm based control systems, which utilize wireless sensor networks in the feedback path of the control loop.

1. Bevezetés

A technológia fejlődésével manapság már széles körben elérhetővé váltak olyan, kis méretű, rádiós kommunikációt is lehetővé tevő eszközök, melyek segítségével lehetőség nyílik különféle fizikai jelek érzékelésére, feldolgozására és vezeték nélküli továbbítására. Az ilyen eszközökből felépülő szenzorhálózatok felhasználására az ipari, katonai és civil szféra számos területein találhatunk példát, és jövőbeni fejlődésük is ígéretesnek látszik.

A szenzorhálózatok felhasználása számos előnyt rejt magában. Fő indokként jelenik meg általában, hogy elkerülhető az érzékelők telepítésével járó nehézségek és költségek egy része, a kialakított hálózat struktúrája könnyen átkonfigurálható, mivel a hálózat nem igényel meglévő infrastruktúrát.

A szenzorhálózatok használata különösen ott játszik nagy szerepet, ahol több helyen is kell méréseket végezni egy vagy több fizikai jelenségen. Célszerűnek látszik tehát felhasználásuk az aktív zajcsökkentő rendszerekben is, ahol a zajcsökkentés hatékonyságának növeléséhez minél több zajérzékelő mikrofont kell elhelyeznünk.

Az aktív zajcsökkentő rendszerek jelentősége a nagy számítási teljesítményű jelfeldolgozó processzorok (DSP) megjelenésével nőtt meg, hiszen ezek lehetővé tették a bonyolult zajelnyomó algoritmusok implementálását. Ezen rendszerekben a cél az, hogy egy bizonyos térrészben elhelyezett mikrofonok körül úgynevezett csendes zónákat alakítsunk ki. Ennek során a szuperpozíció jelenségét használjuk ki. A rendszerben a DSP a rajta futó zajelnyomó algoritmus alapján olyan jelet állít elő és ad ki a hozzá csatolt hangszórók segítségével, hogy a mikrofonok helyén a zaj és ellenzaj összegeként adódó eredő zajszintet minimalizálja.

A hagyományos zajcsökkentő rendszerekkel ellentétben a mikrofonok jeleinek mintavételezése ebben az esetben nem központilag a DSP-nél, hanem elosztottan, a szenzorhálózat elemeinél, a mote-oknál történik. Ez a megoldás viszont több szempontból is szigorú követelményeket támaszt a hálózattal szemben:

- Ezen hálózatok adatátviteli sebessége általában csak néhány száz kilobit per second (kbps), melyet nagy hatásfokkal kell kihasználnunk.
- Az aktív zajcsökkentő rendszerek visszacsatolt struktúrák, és igen érzékenyek a visszacsatoló ág megváltozására. Mivel a szenzorhálózat a visszacsatoló ág szerves részét képzi, ezért a rádiós adatátvitel miatti kommunikációs bizonytalanságok ellenére stabil adatátvitelt kell biztosítanunk.
- A szenzorhálózat elemei autonóm rendszerek, melyek aszinkron működésűek, tehát működésüket – így a rajtuk található mikrofon jelének mintavételezést is – külön kvarcoszcillátorok biztosítják. Ugyan a kvarcoszcillátorok pontossága igen jó (körülbelül 10-100 ppm), ezen kis eltérés is a mintavételi időpontoknak a névleges frekvenciából adódó referenciapontokhoz viszonyított jelentős elcsúszásához vezet viszonylag rövid időn belül. Ez a visszacsatoló ág késleltetésének, így

átviteli függvényének megváltozását okozza, tehát a rendszer instabilitásához vezethet. Emiatt a hálózatban szinkronizálnunk kell az egyes mote-okat.

Mivel a hálózatban található mote-ok számával adott mintavételi frekvencia mellett nő a DSP felé továbbítandó adat mennyisége, így a hálózat adatátviteli sebessége korlátozza az érzékelők számát. A mote-ok számának növelése így csak a mintavételi frekvencia csökkentésével válhat lehetővé, ami viszont a mintavételi tétel értelmében korlátozza a zajcsökkentő rendszer felhasználási tartományát. Ez tette szükségessé egy olyan rendszer kidolgozását, mely a mintavételi frekvencia csökkentése nélkül képes több mote segítségével érzékelni a zajszintet. Ez a módszer a mintavételezett jelek előfeldolgozásán alapul.

A dolgozatban először megismerkedünk az aktív zajcsökkentő rendszerek alapjaival, felépítésével, és az elterjedt zajelnyomó algoritmusok működésével, illetve azok alkalmazási lehetőségeivel. Mivel az aktív zajcsökkentő rendszerekben fontos szerepet játszik a beavatkozó jel és a zajérzékelő egységek közti átviteli függvény, tehát a visszacsatoló ág ismerete, mely viszont minden rendszerben más és más, ezért fontos szerepet kapnak a különféle identifikációs eljárások, melyek szintén ismertetésre kerülnek.

A szenzorhálózatok általános bemutatása után a már létező szinkronizációs algoritmusok részletes vizsgálata, és ezek tükrében a saját rendszerünkre – a zajcsökkentő alkalmazás által támasztott követelményeknek megfelelően – kidolgozott szinkronizációs eljárás ismertetése történik.

A zajcsökkentő rendszerben felhasznált elemek áttekintését követően megismerkedünk a rendszer felépítésével, az egyes egységek illesztésének és szinkronizálásának megvalósítási problémáival és megoldásaival.

Az egyszerű adatgyűjtő rendszer bemutatása után vázoljuk az elosztott jelfeldolgozást megvalósító zajcsökkentő rendszer működésének alapgondolatait és megvalósítási részleteit, melyek segítségével a zajcsökkentő rendszer sávszélességének csökkentése nélkül növelhetjük a rendszerben található érzékelők számát.

Ezt követően a megvalósított szinkronizációs és zajcsökkentési algoritmusok teszteredményeit közöljük, így lehetőség nyílik azok gyakorlati működésének elemzésére és az algoritmusok összehasonlítására.

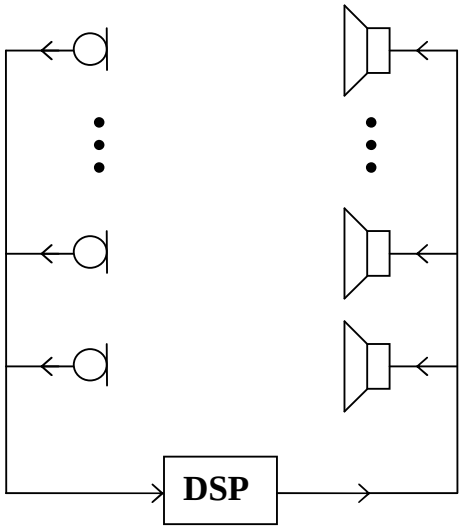
A diplomaterv zárásaként összefoglaljuk a tervezés során szerzett tapasztalatainkat, elemezzük a létrehozott rendszerek teljesítményét, és megvizsgáljuk, hogy milyen továbbfejlesztési lehetőségek kínálkoznak a jövőbeni munkák során.

2. Aktív zajcsökkentés

A zajcsökkentés hosszú ideig kizárólagos módszere az úgynevezett passzív zajcsökkentési technikák használata volt. Ezek alapja, hogy a zajforrást és a zajtól védeni kívánt helyet zajelnyelő, zajvisszaverő anyagokkal választják el. Ezen technikák igen jó hatásfokkal alkalmazhatóak, és mai napig széles körben használják őket, de több hátrányos tulajdonsággal is rendelkeznek, melyek közül a legfontosabbakat említjük meg. A megfelelő zajszigetelést biztosító anyagok beszerzése, felszerelése, a zajtól elszigetelni kívánt helység átalakítása igen költséges lehet. A felhasznált anyagok zajszigetelő képessége ráadásul alacsonyabb frekvenciákon csökken, mivel a zaj hullámhossza jóval meghaladja a zajszigetelő anyagok vastagságát. Ezen hátrányos tulajdonságok adnak létjogosultságot az aktív zajcsökkentés (ANC) használatának, melynek ugyan vannak korlátai, de mindenképpen a passzív zajcsökkentési eljárások jó kiegészítőjének tekinthető.

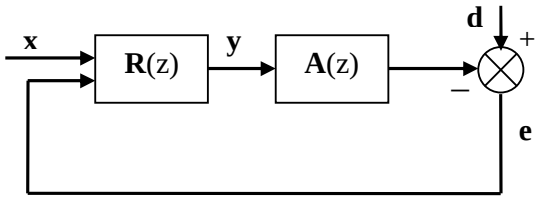
Az aktív zajcsökkentő rendszerek elmélete már régóta ismert, de gyakorlati alkalmazásuk a nagyteljesítményű jelfeldolgozó processzorok (DSP) megjelenésével vált lehetségessé, melyen implementálni lehet a bonyolultabb zajcsökkentő algoritmusokat is, melyek analóg módszerekkel igen nehezen megvalósíthatóak. Ezen módszerek lényege, hogy a zajt egy vele ellentétes előjelű hanggal (ellenzaj) oltjuk ki, a szuperpozíció elvét felhasználva. Mivel az akusztikai rendszerek jó közelítéssel igen nagy dinamikatartományban lineárisnak tekinthetőek, ezért a szuperpozíció alkalmazása megalapozott. Megjegyezzük, hogy ugyan most csupán az akusztikai rendszerekkel foglalkozunk, de ez az elv bármilyen fizikai rendszerben használható rezgések kioltására [1].

Az aktív zajcsökkentő rendszerek általános fizikai kiépítése a 2.1. ábrán látható. A rendszer működése a következő: a DSP hozzá kapcsolódó mikrofonok segítségével érzékeli a zajszintet. Ennek felhasználásával a zajcsökkentő algoritmus alapján kiszámítható az ellenzaj aktuális értéke, melyet a hangszórók segítségével alakítunk fizikai jellé. Sok alkalmazásban a zajcsökkentő algoritmusban felhasználunk a zajjal korrelációban lévő referenciajelet is. A megvalósított rendszerek igen jó zajelnyomásra képesek, ám rendelkeznek néhány hátrányos tulajdonsággal is. Ezek közül a két legfontosabbat említjük meg. A mikrofonok körül kialakuló úgynevezett csendes zónák mérete, tehát az a tér, ahol a zajcsökkentés mértéke még megfelelő, körülbelül a zaj hullámhosszának negyedével egyezik meg. Emiatt az aktív zajcsökkentés felhasználási tartománya az 1-2 kHz-es tartományra korlátozódik, ugyanis ezen határ felett túl kicsi az a tér, amelyben zajcsökkentést érünk el. Másik hátrányos tulajdonság, hogy az aktív zajcsökkentő rendszerek visszacsatolt struktúrák, és igen érzékenyek a visszacsatoló ág, más néven másodlagos út megváltozására. A visszacsatoló ágat esetünkben az AD-, DA átalakítók, az analóg jelkondicionáló áramkörök, a hangszóró, a mikrofon és a közöttük található akusztikus út alkotják. Mind közül ez utóbbi járul általában legnagyobb mértékben hozzá a másodlagos út átviteli függvényének megváltozásához.



2.1. ábra. Aktív zajcsökkentő rendszerek általános felépítése

Az aktív zajcsökkentő rendszerek analitikus tárgyalásakor a 2.2. ábrán vázolt lineáris modell használatos.



2.2. ábra. Aktív zajcsökkentő rendszerek blokkdiagramja

A jelölések a következők:

- **x**: referenciajel, mely az elnyomandó jelről hordoz információkat
- **d**: elnyomandó komponens
- **y**: zajelnyomó algoritmus kimenete
- **e**: hibajel, mely a maradó zajjal egyezik meg
- **R**(z): zajelnyomó struktúra
- **A**(z): másodlagos út átviteli függvénye

Az ábrán a vastag vonalak azt jelölik, hogy többdimenziós jelekről van szó, tehát a 2.2. ábrán egy többcsatornás rendszer blokkvázlata látható.

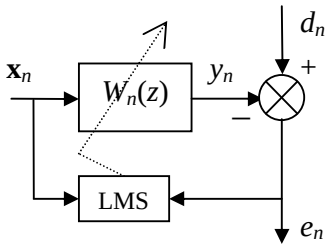
Meg kell jegyeznünk, hogy a megvalósított rendszerben különbségképzés nem valósítható meg, a hibajel az elnyomandó jel és a beavatkozó jel összegeként áll elő. Az, hogy mégis ezt a struktúrát használjuk, az indokolja, hogy a zajcsökkentési algoritmusok ezen elrendezésből indulnak ki. Az ellentmondás feloldásaként gyakorlatban úgy járunk el, hogy a beavatkozó jelet szorozzuk –1 -gyel.

A fent vázolt struktúrából kitűnik, hogy a zajcsökkentési feladat megfeleltethető egy olyan szabályozástechnikai problémának, melyben cél a nulla értékű alapjel követése. Az, hogy mégis főként jelfeldolgozási kérdésként tárgyalják a problémát, annak köszönhető, hogy a szabályozandó akusztikus rendszer átviteli függvénye igen magas fokszámú lehet, ezért nehéz a meglévő analitikai módszerekkel szabályzót tervezni hozzá. Emiatt az aktív zajcsökkentési eljárásokban adaptív algoritmusokat használnak fel. Az algoritmus kiválasztása függ az elnyomandó zaj tulajdonságaitól. A diplomatervben kétfajta algoritmust használok fel. Egyik a sztochasztikus jelek elnyomására tervezett LMS alapú előrecsatolt FxLMS algoritmus, a másik a periodikus jelek elnyomására alkalmas rezonátoros struktúra. A következőkben ezekkel ismerkedhetünk meg részletesebben. Általában az aktív zajcsökkentő rendszerek több hangszórót és mikrofont használó, ún. többcsatornás rendszerek, az algoritmusokat viszont főleg az egy mikrofont és hangszórót tartalmazó egycsatornás rendszerek esetén mutatjuk be nagyobb részletességgel.

2.1. LMS alapú zajcsökkentő rendszerek

2.1.1. LMS algoritmus ismertetése

Az algoritmus megismeréséhez [24] tekintsük a 2.3. ábrát.



2.3. ábra. LMS algoritmus blokkvázlata

jelölések magyarázata:

- $\mathbf{x}_n$: referenciajel utolsó N mintáját tartalmazó vektor
- $W_n(z)$: adaptív szűrő, melynek együtthatóit a $\mathbf{w}_n$, N elemű vektor tartalmazza. Esetünkben **w** transzverzális szűrő
- d_n: adaptálandó jel
- y_n: d_n jel becslője; az adaptív szűrő kimenete
- e_n: a valódi és becsült jel különbsége
- LMS: a **w** szűrőegyütthatókat állító algoritmus
- az alsó indexben található ’*n*’ az adott változó *n*-dik időpillanatbeli értékét jelöli

A rendszert az alábbi egyenletek írják le:

$$y_n = \mathbf{w}_n^T \cdot \mathbf{x}_n \qquad (2.1)$$

$$e_n = (d_n - y_n) = (d_n - \mathbf{w}_n^T \cdot \mathbf{x}_n) \tag{2.2}$$

w-t úgy kell beállítani, hogy a hibanégyzet várható értéke minimális legyen. Ennek érdekében az LMS algoritmus a **w** együtthatókat úgy állítja, hogy a valódi és becsült jel különbségeként adódó hibajel négyzetének pillanatértéke (2.3) csökkenjen.

$$(e_n)^2 = (d_n - y_n)^2 = (d_n - \mathbf{w}_n^T \cdot \mathbf{x}_n)^2 \tag{2.3}$$

Ezen minimum megkeresését iteratív módon végezzük a legmeredekebb lejtő módszerét használva. Ez azt jelenti, hogy a **w** szűrő együtthatóit úgy módosítja az algoritmus, hogy a hibanégyzet az adott pillanathoz tartozó **w** szerinti negatív gradiense irányába mozduljon el. A gradienst a következőképpen határozunk meg:

$$\nabla(e_n)^2 = \frac{\partial(e_n)^2}{\partial \mathbf{w}_n} = \frac{\partial(d_n - \mathbf{w}_n^T \cdot \mathbf{x}_n)^2}{\partial \mathbf{w}_n} = -2 \cdot (d_n - \mathbf{w}_n^T \cdot \mathbf{x}_n) \cdot \mathbf{x}_n = -2 \cdot e_n \cdot \mathbf{x}_n \tag{2.4}$$

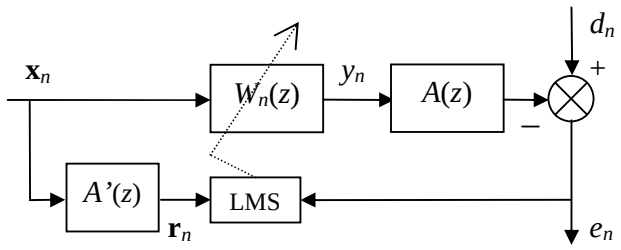
Ezt az értéket egy μ bátorsági tényezővel szorozva megkapjuk a **w** szűrőt módosító vektort, mely alapján a **w** szűrő új együtthatói a következőképpen alakulnak:

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \mu \cdot \nabla(e_n)^2 = \mathbf{w}_n + 2 \cdot \mu \cdot e_n \cdot \mathbf{x}_n \tag{2.5}$$

Ez a megoldás bizonyíthatóan az optimális megoldás felé konvergál megfelelő μ megválasztása esetén.

2.1.2. LMS algoritmus felhasználása aktív zajcsökkentő rendszerekben

Az LMS algoritmus több változatát is használják zajcsökkentő rendszerek megvalósításához. Az alkalmazás alapgondolata az, hogy ha a **w** szűrő *y* kimenetét hangszórón keresztül kiadjuk, akkor az algoritmus a *d* zajjel és az *y* beavatkozójel különbségeként előálló hibajel-, tehát a maradó zajjel négyzetét, ezáltal a zajteljesítményt minimalizálja. Az *e* hibajel az *y* és *d* jel eredőjeként adódik, és a fizikai rendszerben a hibamikrofon kimenőjele. Ehhez természetesen néhány változtatást kell végrehajtanunk a 2.3. ábrán látható blokkdiagramhoz képest. Az első ilyen leszármaztatott struktúra a 2.4. ábrán látható, úgynevezett előrecsatolt, vagy más néven feedforward struktúra [2].



2.4. ábra. Előrecsatolt zajcsökkentő struktúra blokkvázlata

Jól látható módosítás, hogy a $W(z)$ adaptív szűrő kimenete nem közvetlenül jut a hibajelképzőre (mikrofon), hanem az $A(z)$ átviteli függvénnyel rendelkező akusztikus úton keresztül. Emiatt valamelyest módosítanunk kell a 2.2.1. fejezetben bemutatott LMS algoritmust úgy, hogy a $\mathbf{w}$ együtthatók adaptációját nem az eredeti $\mathbf{x}_n$ referenciajel-vektorral végezzük el, hanem annak $A'(z)$-vel szűrt értékeivel, amit $\mathbf{r}_n$-nel jelölünk. $A'(z)$ az $A(z)$ akusztikus átviteli függvény becslője. A leírtak alapján (2.5) a következők szerint módosul, és szűrt xLMS, vagy más néven FxLMS algoritmusként hivatkozunk rá:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + 2 \cdot \mu \cdot e_n \cdot \mathbf{r}_n \tag{2.6}$$

Az algoritmus több csatornára is általánosítható [2]. Látható, hogy az algoritmusba beépül az akusztikus út átviteli függvénye. Ennek meghatározásával később foglalkozunk. Amennyiben a rendszer stabil, az algoritmus továbbra is minimalizálja a hiba négyzetének várhatóértékét. $A(z)$ nem kellően pontos modellezése azonban akár a rendszer instabilitásához is vezethet, de mindenképpen lassítja a rendszer tranziens viselkedését. A stabilitás feltétele, hogy $A'(z)$ az eredeti átviteli függvény fáziskarakterisztikáját adott frekvencián legalább 90°-os pontossággal megközelítse, de 40°-os eltérésen belül már kielégítő a rendszer működése [3].

Előfordulhat olyan eset, amikor nem tudunk a zajról információt hordozó referenciajelet szerezni. Erre az esetre kínál megoldást a visszacsatolt, más néven feedback struktúra, mely a 2.5. ábrán látható [2].

Látható módosítás az előrecsatolt struktúrához képest, hogy a referenciajelet nem kívülről szerezzük, hanem a hibajelből és a beavatkozó jelből számítjuk a következő módon:

$$x_n = e_n + y \cdot A'(z) \tag{2.7}$$

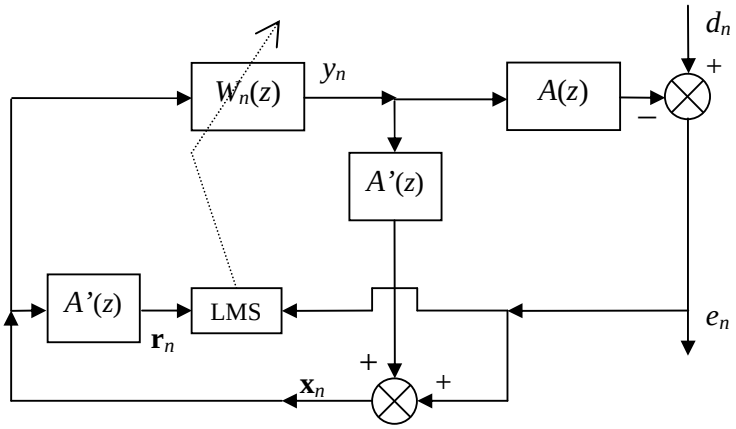
mivel

$$e_n = d_n - y \cdot A(z) \tag{2.8}$$

így

$$x_n = [d_n - y \cdot A(z)] + y \cdot A'(z) = d_n + y \cdot [A'(z) - A(z)] \approx d_n \tag{2.9}$$

feltéve, hogy az akusztikus átviteli függvény becslése megfelelő.



2.5. ábra. Visszacsatolt zajcsökkentő struktúra

Ezek alapján referenciajelként magát a zajt használjuk fel. Mivel a zajmintákat annak régebbi értékei lineárkombinációjaként számoljuk, így ez a struktúra olyan zajok elnyomására nem alkalmas, melyek mintái korrelálatlanok. Ezek alapján ez a struktúra jól használható periodikus zajok, keskenysávú sztochasztikus zajok és egyértelmű referenciával nem jellemezhető periodikus zaj elnyomására (pl. benzinmotor). Használata általában háttérbe szorul, ugyanis a referenciajel sokszor hozzáférhető, ilyen körülmények között viszont az előrecsatolt rendszer jobb paraméterekkel rendelkezik..

Eddigiekben még nem esett szó a μ paraméter megválasztásáról. A μ paraméter a rendszer sebessége szempontjából rendelkezik egy optimális értékkel. A μ-t erről az értékről növelve, a rendszer beállási ideje nő, mígnem instabillá válik. A μ értékét csökkentve a rendszer szintén lassul, $\mu = 0$ esetén pedig leáll az algoritmus. A tranziens viselkedések szempontjából tehát célszerű μ-t ezen optimális értékre választani. Másrészről viszont a zajérzékenység szempontjából célszerű a μ-t kis értékűre választani, ugyanis így a referenciajellel korrelálatlan egyéb jelek (pl. beszélgetés) kevésbé zavarják az állandósult állapotban lévő rendszert. Látható tehát, hogy a μ konvergenciaparaméter megválasztása kompromisszumot igényel a gyorsaság és a zavarérzékenység között, meghatározása gyakorlati tesztelések végrehajtásával lehetséges.

2.2. Rezonátor alapú zajcsökkentő rendszerek

A rezonátor alapú zajcsökkentő rendszereket kifejezetten periodikus zajok elnyomására fejlesztették ki [25]. Bár az előző részben megállapítottuk, hogy az FxLMS algoritmus is alkalmas periodikus zajok elnyomására, de ezzel kapcsolatban számos probléma merülhet fel, melyek megoldására ezen új struktúra kínálhat megoldást. Egyik ilyen például, hogy FxLMS algoritmus használatakor a referenciajelben szerepelnie kell

minden elnyomandó harmonikusnak, ugyanis lineáris rendszerről lévén szó, a $W(z)$ szűrő nem tud új harmonikus komponenseket generálni.

A rezonátoros struktúra kifejezetten periodikus jelek feldolgozására alkalmas, és más alkalmazásokban is sikeresen felhasznált, kiforrott eljárás. Adott feltételek mellett a bemutatott LMS alapú rendszerek bizonyíthatóan maximum elérhetik a rezonátoros rendszerek gyorsaságát, és a rezonátoros struktúra gyorsasága nem függ a referenciajel teljesítményétől.

A rezonátoros algoritmus számításigénye kedvezőbb, hiszen az átviteli függvény számítása csupán az üzemi frekvenciákon szükséges, míg LMS algoritmus esetén a teljes lehetséges működési tartományban megfelelő pontossággal fel kell használni az átviteli függvényt.

A rezonátor alapú algoritmus alapötlete az, hogy a zajjelet harmonikus komponensekre bontjuk, és az egyes harmonikusokat külön-külön nyomjuk el. A következő részekben a rendszer felépítésével ismerkedhetünk meg.

2.2.1. Rezonátoros algoritmus ismertetése

Az algoritmus első ismertetése [4]-ben található. Mivel az algoritmus kifejezetten periodikus jelek elnyomására szolgál, ezért először a periodikus jelek leírásának módját ismertetem.

A periodikus jelek leírása gyakran történik frekvenciatartományban, Fourier-együtthatóik segítségével. Ez azt jelenti, hogy egy x_n periodikus jel felírható komplex exponenciálisok X_k együtthatókkal súlyozott összegeként, ahol X_k az adott komponens fázisát és amplitúdóját jellemzi:

$$x_n = \sum_{k=-N}^{N} X_k c_{k,n} \tag{2.10}$$

és:

$$c_{k,n} = e^{j2\pi f_1 kn}; k = -N \ldots N \tag{2.11}$$

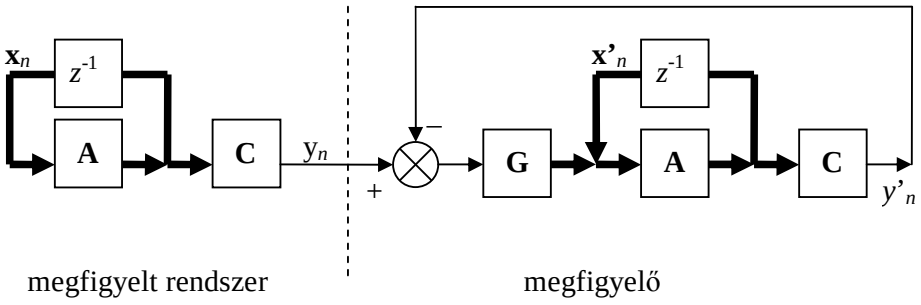
Az egyenletben f_1 az alapharmonikus frekvenciát jelenti a mintavételi frekvenciához (f_s) viszonyítva. Mivel a mintavételi tétel korlátozza a mintavételezett jelek sávszélességét, és a digitális jelfeldolgozó rendszerek mintavételezett rendszerek, így az N paraméter maximális értékének megválasztása a következő egyenletben adott módszerrel történik:

$$N \cdot f_1 < 0.5 < (N+1) \cdot f_1 \tag{2.12}$$

mely – mivel N a felharmonikusok számát jelenti – azt fejezi ki, hogy semmelyik felharmonikus frekvenciája sem haladhatja meg a mintavételi frekvencia felét.

X_k Fourier-együtthatók meghatározására többféle módszer létezik. Esetünkben ezek meghatározása a megfigyelő-elmélet alkalmazásával történik.

Az állapot-megfigyelők olyan rendszerek, melyek a megfigyelendő rendszer struktúrájának és kimenetének ismeretében képes meghatározni a megfigyelt rendszer állapotváltozóinak értékét. Egy ilyen megfigyelt rendszert és megfigyelőt is tartalmazó struktúrát láthatunk a 2.6. ábrán.



2.6. ábra. Állapot-megfigyelőt tartalmazó rendszer felépítése

A megfigyelt rendszert és a megfigyelőt a (2.13) – (2.16) egyenletek írják le:

$$\mathbf{x}_{n+1} = \mathbf{A}\mathbf{x}_n \tag{2.13}$$

$$y_n = \mathbf{C}\mathbf{x}_n \tag{2.14}$$

$$\mathbf{x'}_{n+1} = \mathbf{A}\mathbf{x'}_n + \mathbf{G}(y_n - y'_n) \tag{2.15}$$

$$y'_n = \mathbf{C}\mathbf{x'}_n \tag{2.16}$$

(2.13) és (2.15) különbségét véve, valamint (2.14)-et és (2.16)-ot behelyettesítve a következő egyenletekre jutunk:

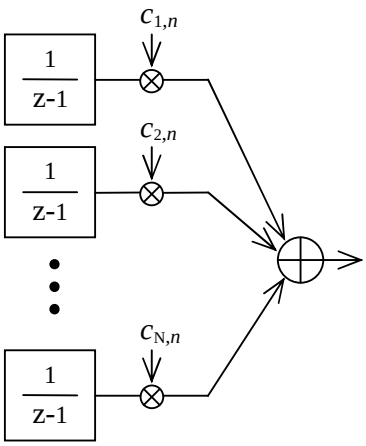
$$\mathbf{x'}_{n+1} - \mathbf{x}_{n+1} = \mathbf{A}(\mathbf{x'}_n - \mathbf{x}_n) + \mathbf{G}(\mathbf{C}\mathbf{x'}_n - \mathbf{C}\mathbf{x}_n) \tag{2.17}$$

$$\mathbf{x'}_{n+1} - \mathbf{x}_{n+1} = (\mathbf{A} - \mathbf{G}\mathbf{C})(\mathbf{x'}_n - \mathbf{x}_n) \tag{2.18}$$

$$\mathbf{x''}_{n+1} = \mathbf{F}\mathbf{x''}_n \tag{2.19}$$

A **G** becsatolómátrixot úgy kell megválasztani, hogy a megfigyelt és a megfigyelő állapotváltozóinak különbségét tartalmazó **x**" állapotvektor minél gyorsabban nullához tartson, tehát hogy a megfigyelő a megfigyelt rendszernél gyorsabb legyen. Ennek elérése általában FIR beállású megfigyelő tervezésével történik.

A kapcsolatot a megfigyelő-elmélet alkalmazása és a Fourier-együtthatók számítása között a koncepcionális jelmodell teremti meg. Ez azt jelenti, hogy a komponenseire bontandó jelet egy rendszer kimeneteként tekintjük. Ezt a rendszert úgy hozzuk létre, hogy állapotváltozói a Fourier-együtthatók legyenek. Ezen rendszer megkonstruálásának több formája is létezik, most csak azt a formát ismertetem, melyet kedvezőbb számítási tulajdonságai miatt a zajcsökkentő rendszerben alkalmazok. A modell felépítése a 2.7. ábrán látható.



2.7. ábra. Periodikus jeleket előállító koncepcionális jelmodell

A nulla értékkel inicializált integrátorok bemeneteire az $n=n_0$ ütemben X_k értéket, aztán nullát adva a megfelelő állapotváltozók beállnak X_k értékekre. Ezután a kimeneten a (2.10) egyenletnek megfelelő x_n jel áll rendelkezésre. Az integrátorok bemeneteire a megfelelő értékeket adva az állapotváltozók minden pillanatban a kívánt értékre beállíthatóak. A Fourier-analizátor megtervezése a 2.7. ábrán látható blokkdiagrammal bemutatott koncepcionális jelmodellhez tartozó állapot-megfigyelő tervezését jelenti. Ehhez először meg kell határoznunk a rendszer **A** állapot-átmeneti és **C** kicsatoló mátrixát. Mivel a rendszer csatornánként egy integrátort tartalmaz, és a csatornák között nincs csatolás, az **A** mátrix egy olyan diagonálmátrix, mely főátlójában csupa egyes található, a többi eleme nulla (2.20). A **C** mátrix a 2.7. ábra alapján a $c_{i,n}$ bázisfüggvényeket tartalmazó vektorként adható meg (2.21).

$$\mathbf{A} = \mathrm{diag}\langle 1\ 1\ \ldots\ 1\rangle \tag{2.20}$$

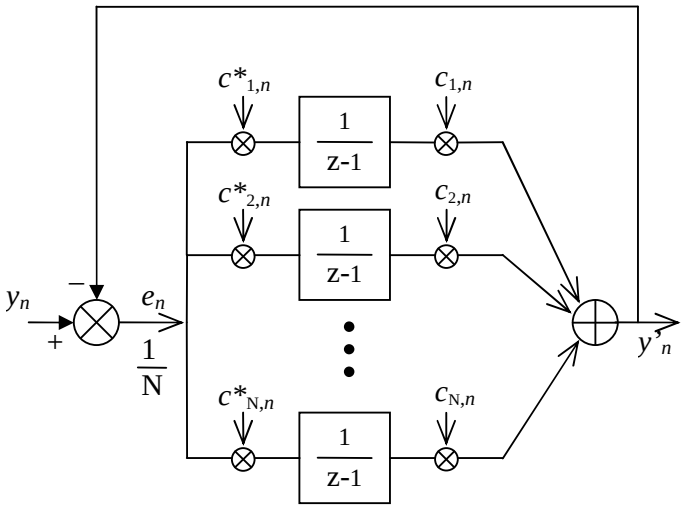
$$\mathbf{C} = [c_{1,n}\ c_{2,n}\ \ldots\ c_{N,n}] \tag{2.21}$$

Ezen struktúrához tervezett **G** becsatolómátrix a (2.22) egyenlet által megadott vektorrá egyszerűsödik.

$$\mathbf{g} = \frac{1}{N}\left[c^*_{1,n} \quad c^*_{2,n} \quad \ldots \quad c^*_{N,n} \right] \tag{2.22}$$

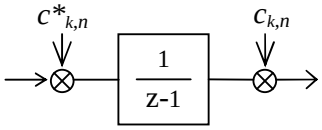
ahol $c^*_{i,n}$ a $c_{i,n}$ konjugáltjait jelölik.

Az így megtervezett rendszer a 2.8. ábrán látható, azzal az egyszerűsítéssel, hogy az $^1/_N$-es becsatoló tényezőt a közös ágba írtuk, ezzel a számítást egyszerűsítve. Ezt a rendszert szokásos ún. rezonátoros megfigyelőnek is nevezni.



2.8. ábra. A koncepcionális jelmodellhez tartozó rezonátoros megfigyelő

A rezonátoros megfigyelő egy csatornája látható a 2.9. ábrán.

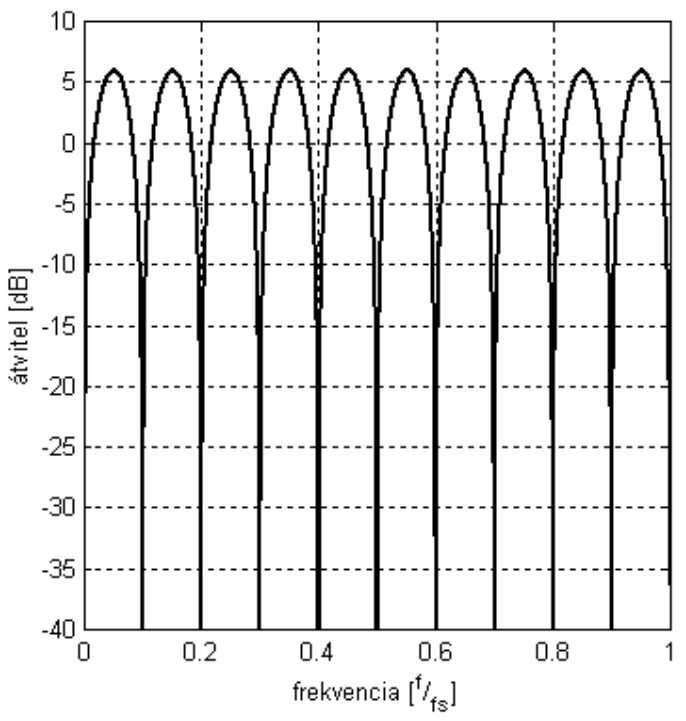


2.9. ábra. Egy rezonátorcsatorna

A rezonátoros megfigyelő működése a következőképpen is értelmezhető: a hibajelben a *k*. rezonátorcsatornához tartozó $k \cdot f_1$ frekvenciájú jelet a *k*. rezonátorcsatorna a bemenetén található $c^*_{k,n}$ segítségével nulla frekvenciára keveri. Mivel az integrátor DC átvitele végtelen, így az integrátor kimeneti értéke, tehát az adott rezonátorcsatorna állapotváltozója addig nő, míg az e_n hibajelben megtalálható a $k \cdot f_1$ frekvenciájú komponens. A rezonátor állapotváltozót $c_{k,n}$-val szorozva kapjuk meg a *k*. rezonátorpozícióban lévő jelet, tehát az állapotváltozó a (2.10) alapján a *k*. Fourier-együtthatónak felel meg.

A következőkben a rezonátoros megfigyelőre jellemző néhány jellemző átviteli függvényt ismerhetünk meg.

Egy rezonátorcsatorna átviteli függvénye rezonátorfrekvenciákon az integrátor miatt végtelen. Az y_n bemenő jel e_n-re vonatkoztatott átviteli függvénye 10 rezonátoresetén a 2.10. ábrán látható.

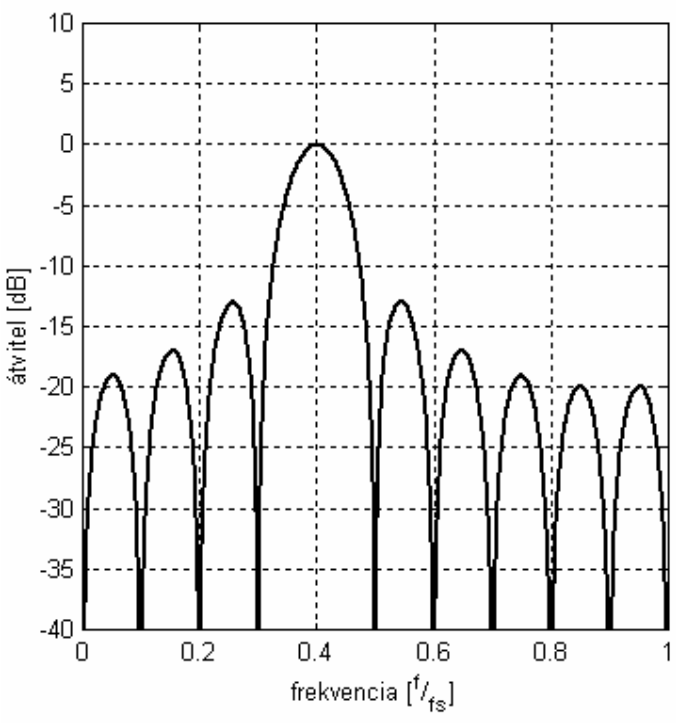


2.10. ábra. Hibajelre vonatkoztatott átvitel N=10 rezonátor esetén

Az átviteli függvényről általánosságban elmondható, hogy rezonátor-frekvenciákon az átvitel nulla, tehát ezekben a pozíciókban a rendszernek zérusai vannak. Ebből kifolyólag a rendszer a rezonátorfrekvenciákon lévő jeleket hiba nélkül képes előállítani.

A 2.11. ábrán egy rezonátorcsatorna zárt hurkú átvitele látható. A rendszernek ebben az esetben az adott rezonátorfrekvenciát kivéve minden rezonátor frekvencián zérusa van. Ez azt jelenti, hogy az adott rezonátorcsatorna az annak megfelelő frekvenciájú jelet hiba nélkül képes előállítani.

Ezen átviteli függvények egyenletes rezonátor-elhelyezkedésre vonatkoznak, tehát amikor $f_1 \cdot N = f_s$. A függvények jellege, és a levont következtetések azonban nem egyenletes rezonátor-elhelyezkedés esetében is helyénvalóak.



2.11. ábra. 4. rezonátor zárthurkú átvitele N=10 rezonátor esetén

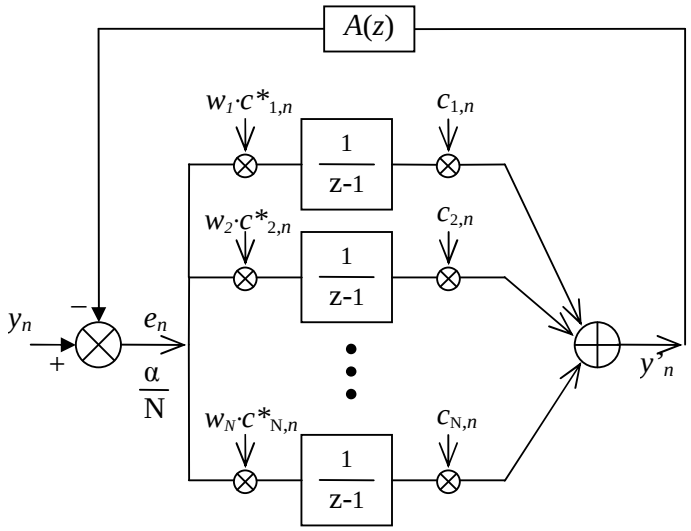
A gyakorlati alkalmazás során problémát jelent, hogy a bemeneti jel frekvenciája nem konstans, és gyakran változhat. Ha azonban a harmonikusok nem a rezonátorpozícióban vannak, akkor a DFT-ből már ismert leakage és picket-fence jelenség áll elő. Erre kínál megoldást az Adaptív Fourier Analizátor (AFA) [5]. Ebben a struktúrában a $c_{i,n}$ és $c^*_{i,n}$ tagokban a (2.11) egyenletben szereplő f_1 tagot adaptáljuk a jel frekvenciájához a (2.23) egyenlet szerint.

$$f_{1,n+1} = f_{1,n} + \frac{1}{2\pi N}\,\mathrm{angle}(x'_{1,n+1}, x'_{1,n}) \qquad (2.23)$$

(2.23) azt fejezi ki, hogy az állapot-megfigyelőben a frekvenciaadaptációt az alapharmonikushoz tartozó x'_1 állapotváltozó pillanatnyi- és azt megelőző értéke közti szögeltérés alapján végezzük. A frekvenciaadaptációval kapcsolatos egyik probléma, hogy az alapharmonikus frekvencia változásával adott rezonátorszám esetén nem biztosítható, hogy a (2.12) egyenlet teljesül. Emiatt folyamatosan figyelni kell, hogy mekkora rezonátorszám tesz eleget (2.12) feltételnek. Amennyiben több rezonátorunk van, mint megengedhető, akkor a nem használható rezonátorokat megszüntetjük, ellenkező esetben új rezonátorokat léptetünk be a rendszerbe az adott rezonátorhoz tartozó állapotváltozó nulla értékkel történő inicializálását követően. Felmerülhet még az a kérdés, hogy az adott rezonátor-elhelyezkedéshez megtervezett **G** becsatoló mátrixot nem kell-e újra meghatározni. Elvileg minden alapfrekvenciához más **G** tartozik, az eredeti **G**-hez képest azonban ezek egy konstans szorzóban térnek el, ami nem okoz jelentős sebességcsökkenést.

2.2.2. Zajcsökkentő rendszer felépítése

A rezonátoros struktúra zajcsökkentő rendszerekben való alkalmazásakor azt használjuk ki, hogy a struktúra a bemenő jelet zérus hibával képes előállítani, amennyiben a bemenő jel harmonikus komponensei rezonátorpozícióba esnek. Ezen feltétel biztosítására használjuk a frekveniaadaptációs mechanizmust (AFA). A fizikai rendszerben a 2.12. ábrán látható rezonátoros struktúrát használhatjuk zajelnyomásra.



2.12. ábra. Egycsatornás zajelnyomó elrendezés

A zajcsökkentő rendszer úgy valósul meg, hogy az y'_n jel –1 -szeresét hangszórón kiadjuk, az y_n zaj valamint a $-y'_n$ eredőjeként adódó e_n hibajelet pedig mikrofonnal érzékeljük. Így jön létre a 2.2. ábrának megfelelő elrendezés. A bemutatott visszacsatolt rezonátoros megfigyelő tulajdonképpen ennek oly módon feleltethető meg, hogy $R(z)$ a visszacsatolatlan rezonátorok, $A(z)$ pedig a másodlagos út átviteli függvényét jelöli. A valódi rendszerbe tehát beépül egy nem egységnyi $A(z)$ akusztikus átviteli függvény. Ennek kompenzálásaként a rezonátorok **G** becsatoló-mátrixát úgy kell módosítani, hogy felhasználjuk benne az $A(z)$ akusztikus átviteli függvény inverzét is, így biztosítva a hurok eredő egységnyi visszacsatolását. A **G** egycsatornás esetben tulajdonképpen egy vektor, mely a (2.24) szerinti értéket veszi fel az egycsatornás zajcsökkentő rendszerek esetén:

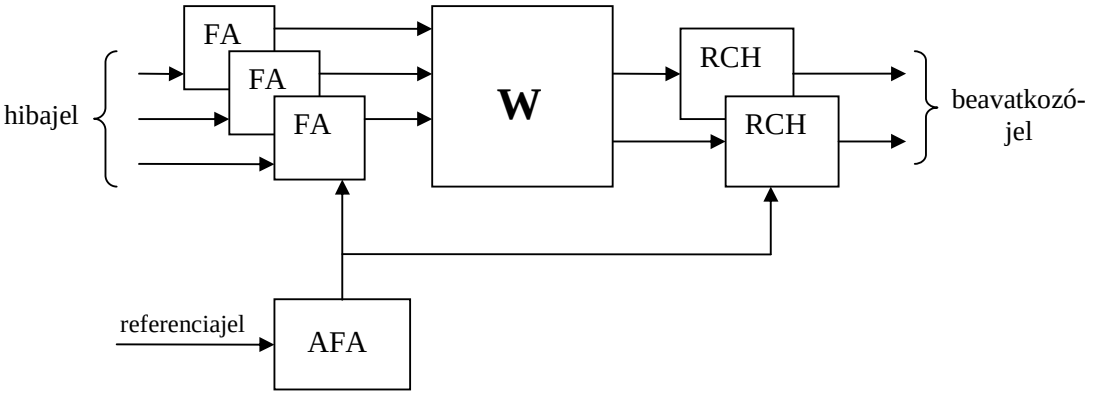
$$\mathbf{g} = \frac{\alpha}{N}\begin{bmatrix} w_1 c^*_{1,n} & w_2 c^*_{2,n} & \dots & w_N c^*_{N,n} \end{bmatrix} \tag{2.24}$$

ahol w_i jelöli az adott rezonátorfrekvencián az $A(z)$ akusztikus átviteli függvény inverzét. A rendszer stabilitási vizsgálatával [6] meghatározható, hogy a rendszer stabilizálható, amennyiben $A(z)$ fáziskarakterisztikáját legalább 90°-os pontossággal ismerjük. Ez a (2.25) képlettel fejezhető ki:

$$-\pi / 2 < \arg(w_i) + \arg(A(\omega_i)) < \pi / 2 \tag{2.25}$$

A stabilitás eléréséhez szükség van még csatornánként egy $\alpha \leq 1$ szorzótényezőre is, amennyiben $A(z)$ nem pontosan ismert. α szorzótényezővel a rendszer tranziens tulajdonságai, és a külső zavarok iránti érzékenység is befolyásolható.

A frekvenciaadaptáció az alapharmonikus frekvenciához tartozó első csatorna x_1 állapotváltozója segítségével elvégezhető. Ezzel kapcsolatban problémát jelent, hogy a periodikus zajt elnyomva a külső zavarhatások dominánssá válnak, és módosíthatják az alapharmonikus frekvenciát. Ennek kiküszöbölésére léteznek különféle módszerek, azonban a gyakorlatban inkább a 2.12. ábrán látható rendszer egy módosított, nagyobb sebességű változatát alkalmazzák, melynek több csatornára általánosított blokkdiagramja a 2.13. ábrán látható.



2.13. ábra. Megnövelt sebességű zajcsökkentő rendszer

Egyes blokkok jelentése:

- FA: a 2.8. ábrán látható visszacsatolt rezonátorral megvalósított Fourier-analizátor
- RCH: a 2.7. ábrán látható rezonátorkészlet
- **W**: átcsatoló mátrix
- AFA: frekvenciaadaptációt megvalósító adaptív Fourier-analizátor

A rendszer működése a következő: a zajforrástól érkező referenciajel segítségével az AFA előállítja a bemenő és kimenő FA és RCH rezonátorfrekvenciáit. Mivel ebben az esetben az AFA nem az elnyomandó zaj alapján állítja elő az alapharmonikus frekvenciát, ezért a frekvenciaadaptáció igen pontosan működik a zaj teljes elnyomása esetén is. A mikrofonokról érkező bejövő hibajelből a bemeneten található FA blokkok meghatározzák a hibajel harmonikus komponenseit, melyek a rezonátorok állapotváltozói. Ezen komponenseket a **W** átcsatoló mátrixszal csatoljuk a kimenő rezonátorokhoz, melyek a beavatkozójelet állítják elő. Ezen beavatkozójelet DA- átalakító segítségével a hangszórókra juttatjuk. A **W** átcsatoló mátrix feladata az egységnyi átvitel biztosítása a visszacsatoló hurokban. Meghatározása a (2.26) egyenlet alapján történik:

$$\mathbf{W} = \mathbf{A}(z_i)^{\#} \tag{2.26}$$

ahol $\mathbf{A}(z_i)^{\#}$ az átviteli függvény mátrix pszeudoinverzét jelöli a rezonátorfrekvenciákon, amely frekvenciát az AFA segítségével határozzuk meg. A **W** mátrixot tehát a működési frekvenciának megfelelően kell frissíteni. A **W** inverz átviteli függvény mátrixot táblázatosan tároljuk, mivel számítástechnikai okok miatt ez felel meg legjobban a struktúrának. Természetesen nem tartozik minden lehetséges frekvenciaértékhez külön **W** mátrix, csupán bizonyos frekvenciafelbontással. Azt, hogy milyen sűrűséggel kell ismerni az átviteli függvényt, az határozza meg, hogy milyen meredekséggel változik az átviteli függvény, és az, hogy milyen pontossággal szeretnénk ismerni azt. $\mathbf{A}(z)$ identifikálása után **W** off-line meghatározható, és felhasználható a rendszerben. A **W** mátrixszal való szorzás után minden kimeneti csatornán található egy közös α szorzótényező, mellyel a stabilitás biztosítható.

A stabilitás feltételét többcsatornás rendszerekben (2.27) egyenlet írja le:

$$\begin{gathered} -\pi / 2 < \arctan(\lambda_{l,i}) < \pi / 2 \\ \lambda_{l,i} = \lambda_l(\mathbf{A}(z_i)\mathbf{W}_i);\ l=1\ldots L,\ i = 1\ldots N \end{gathered} \tag{2.27}$$

ahol N a rezonátorok számát, L az elnyomandó jelek számát jelöli. (2.27) azt fejezi ki, hogy a $(\mathbf{A}(z_i)\mathbf{W}_i)$ mátrix sajátértékeinek valós része legyen pozitív minden rezonátorfrekvencián. Egycsatornás esetben a (2.25) egyenletet adja vissza.

2.3. Identifikációs eljárások

Előzőekben láthattuk, hogy az aktív zajcsökkentő eljárásokban igen fontos szerepet játszik a zajcsökkentő rendszerek visszacsatoló ágának, az úgynevezett másodlagos út átviteli függvényének minél pontosabb ismerete.

Az átviteli függvények meghatározására többféle identifikációs eljárás létezik. Az, hogy az adott feladathoz melyiket használjuk, meghatározhatja például az identifikációs eljárás pontossága, vagy kivitelezési nehézségei. Esetünkben a választás főleg az alapján történt, hogy a létező eljárások közül melyik illeszkedik legjobban az adott struktúrához, természetesen mindamellett, hogy megfelelő pontosságot eredményez. Gondolunk itt arra, hogy a zajcsökkentéshez kialakított programok mennyire használhatóak fel identifikációra, és az identifikáció eredményeként kapott paraméterek mennyire egyszerűen építhetőek be a zajcsökkentő programba.

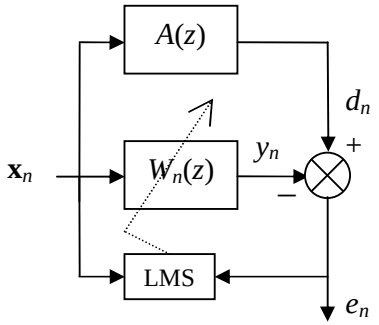
Az identifikációra többféle lehetőség kínálkozik, ezek közül összefoglaltunk néhány, gyakran használt módszert:

1. gerjesztés fehér zajjal, kiértékelés FFT-vel
2. gerjesztés multiszinuszos jellel, kiértékelés FFT-vel

3. adaptív szűrő paramétereinek beállítása, gerjesztés zajjal
4. gerjesztés léptetett frekvenciájú szinusszal, átvitel mérése pontonként

Az 1. módszer hátránya, hogy ahol $A(z)$ átvitele kicsi, ott nem gerjeszti eléggé a rendszert, így a külső zajok jobban befolyásolják a mérési eredményt. A mérés varianciája igen nagy a gerjesztő jel tulajdonságai miatt, de ez a variancia átlagolással csökkenthető. A 2. módszer igen gyors identifikációt tesz lehetővé, viszont a gerjesztő jel előállítása nehéz, ugyanis az azonos fázishelyzetbe kerülő komponensek miatt igen nagy értékű gerjesztőjel is kialakulhat, ami pl. az AD, DA átalakító telítése miatt torzulhat. Az előbb leírt, és a rendszer más tagjai által előidézett intermodulációs torzítások az átviteli függvény mérésének hibáját növeli.
A megvalósított rendszerekben az utóbbi két algoritmust használtuk fel.

LMS algoritmus alapú zajcsökkentő rendszer esetén igen jól használható a 3. módszer. A gyakorlatban igen fontos szempont, hogy identifikációs program a zajcsökkentő program kis módosításával egyszerűen kialakítható, és az identifikáció eredményeként adódó $A'(z)$ közvetlenül, előfeldolgozás nélkül felhasználható a zajcsökkentő programban. Az identifikációs rendszer felépítése a 2.14. ábrán látható.

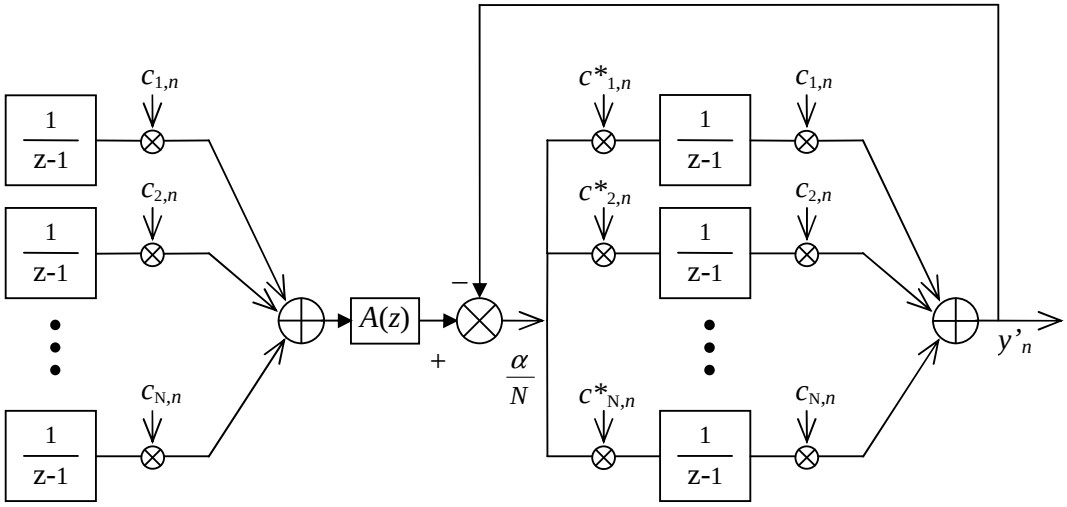


2.14. ábra. LMS alapú identifikációs eljárás blokkvázlata

Állandósult állapotban a hiba nulla, tehát azonos gerjesztésre $A(z)$ illetve $W(z)$ kimenete megegyezik, ez pedig csak $A(z) = W(z)$ esetén lehetséges, tehát a $W(z)$ szűrőt reprezentáló **w** együtthatójú FIR szűrő felhasználható $A(z)$ modelljeként. **w** méretét úgy kell meghatározni, hogy $W(z)$ elegendően jól tudja közelíteni $A(z)$-t. Amennyiben $A(z)$-ről semmilyen információnk nincsen, ennek meghatározása iteratív teszteléssel lehetséges. A μ konvergencia-paramétert célszerű minél kisebbre választani, ugyanis így a külső zajok kevésbé befolyásolják az identifikáció pontosságát. Alsó korlátot az identifikáció időtartama – ami egyébként, off-line műveletről lévén szó, nem kritikus – és a DSP számábrázolási pontossága szab. x_n gerjesztőjelként minél szélesebb sávú jelet célszerű alkalmaznunk, ugyanis a $W(z)$ csupán olyan frekvenciákon adaptálódik, amelyben a gerjesztőjelnek is van komponense. Ezért gyakorlatban gerjesztésként fehérzajt alkalmazunk. A zaj fehérsége, tehát az, hogy az egész frekvenciatartományban

egyenletes a teljesítmény-eloszlása, nehezen biztosítható, de gyakorlatban elegendő, ha csupán a rendszer sávszélességében nagyjából egyenletes a gerjesztés spektruma.

A rezonátoros struktúrához a 4. identifikációs eljárás [7] illeszkedik a legjobban. Ezen módszer blokkdiagramja a 2.15. ábrán látható.



2.15. ábra. Identifikáció rezonátoros struktúrával

Mérés során egyetlen, léptetett frekvenciájú szinuszjellel gerjesztjük a rendszert, és az erre adott válaszból meghatározható az átviteli függvény értéke az adott frekvencián. Mivel a gerjesztés keskeny sávú, így a mérés jel-zaj viszonya igen jó. Az $\frac{\alpha}{N}$ becsatoló paraméterben α csökkentésével javítható a mérés jel-zaj viszonya. α = 1 esetén a rendszer FIR beállású, amennyiben egyenletes a rezonátor elhelyezkedés, tehát: $f_1 \cdot N = f_s$. A jelölések a (2.11) képlethez tartozó jelöléseknek felelnek meg. α csökkentésével a beállás lassul, de a rendszer kevésbé érzékeny a zajjal szemben. Ez tulajdonképpen egy exponenciális átlagolásnak felel meg az adott csatornán, mellyel javítható a mérés jel-zaj viszonya, viszont nő a mérési idő, hiszen a rendszer beállása lassabb. A mérési eredmények utófeldolgozást igényelnek, ugyanis meg kell határozni minden mért frekvencián a $\mathbf{W} = \mathbf{A}(z_i)^{\#}$ mátrixot.

3. Szenzorhálózatok

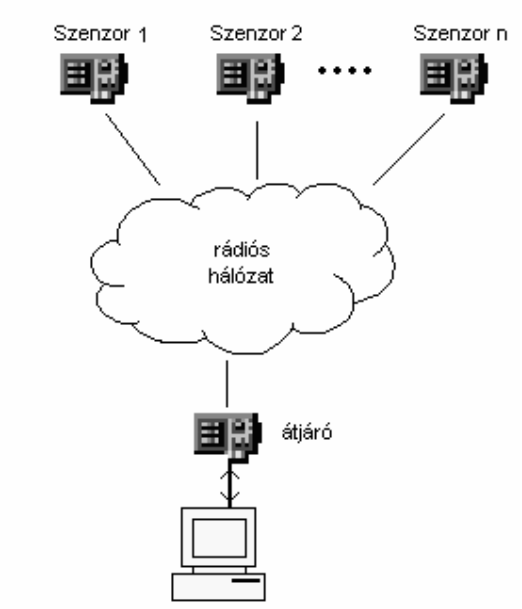
Ezen fejezetben a manapság igen széles körben alkalmazott szenzorhálózatok bemutatására kerül sor, szem előtt tartva az aktív zajcsökkentésben történő felhasználásukat is.

Az aktív zajcsökkentő rendszerek kialakításakor az egyik megoldandó feladat a mikrofonok és hangszórók kiválasztása, valamint meg kell oldani a hibamikrofonok jeleinek, valamint a beavatkozó jelek átvitelét. Hagyományosan ezeket a jeleket vezetéken továbbítjuk.

Manapság azonban a technológia fejlődése lehetővé tette az olcsó rádiós digitális adattovábbító berendezések kialakítását, melyeket különféle szenzorokkal és vezérlő logikával ellátva mérésadatgyűjtő és továbbító hálózatokat, elterjedt nevükön szenzorhálózatokat alakíthatunk ki. Ezen hálózati elemeket akár készen is megvásárolhatjuk, és megfelelő szenzorok illesztésével, melyeket a gyártók szintén készen kínálnak, viszonylag egyszerűen megoldható saját hálózat kialakítása.

A szenzorhálózatok elterjedésében nagy szerepet játszott könnyű kezelhetőségük, mobilitásuk. A hagyományos rendszerekkel ellentétben telepítésük nem igényel meglévő hálózati infrastruktúrát, amely jelentős költségmegtakarítást jelent. Fokozottan igaz ez például azokra az esetekre, melyekben a hálózatot csak ideiglenesen, egyszeri mérés elvégzésére alakítjuk ki.

A szenzorhálózatok általános felépítése a 3.1. ábrán látható:



3.1. ábra. Szenzorhálózat általános felépítése

Látható, hogy az adatok begyűjtését végző szenzorok a mérési eredményeiket, esetleges alapvető adatfeldolgozást követően, továbbítják a hálózaton keresztül egy

átjáróként szolgáló szenzorhoz, mely a rádiós hálózat adatait továbbítja az adatokat feldolgozó berendezéshez.

A szenzorhálózatokat manapság már az élet számos területén alkalmazzák [8]. Néhány felhasználási példát itt is megemlítünk. Gyakori felhasználási terület a különböző környezeti és meteorológiai jellemzők, mint például hőmérséklet, fényerősség, légnyomás vagy páratartalom mérése. Találkozhatunk különböző rezgésanalízisben történő alkalmazással, például szeizmológiai megfigyelés, különböző szerkezetek, mint például épületek, hidak rezgéseinek megfigyelése. Segítséget jelenthetnek ezentúl egészségügyi, épületinformatikai és még számos alkalmazás fejlesztésében.

A fent említett felhasználások jellemzői általában, hogy az átvitt adatok mennyisége csekély a hálózat áteresztőképességéhez képest, illetve nagy mennyiségű adat továbbítása esetén az adatok átvitelére fordítandó idő nem kritikus, feldolgozásuk off-line történik.

Előfordulhatnak azonban olyan esetek, amikor nagy mennyiségű adatot határidőn belül továbbítani kell a hálózaton keresztül, aminek biztosítása igen nehézkes a rádiós kapcsolat viszonylag alacsony adatátviteli sebessége és bizonytalan volta miatt. Tipikusan ilyenek a szabályozástechnikai alkalmazások. A fenti okok miatt a szenzorhálózatok felhasználása a szabályozási feladatokban még nem elterjedt [9]. Általában a meglévő alkalmazásokban sem közvetlenül a folyamat kimenő jelének érzékeléséhez használják, hanem bizonyos környezeti paraméterek mérésével segítik a szabályzó optimális beállítását. Ha mégis kimenőjel érzékelőként használják a hálózat elemeit, akkor is lassú folyamatok szabályozásában, ahol nem játszik akkora szerepet az adatátvitel időbeni bizonytalansága.

A fent említett problémák miatt érdekes feladatnak bizonyult egy viszonylag nagy sebességgel működő aktív zajcsökkentő rendszer megvalósítása, mely bizonyos szemszögből szabályozástechnikai feladatnak is tekinthető, és igen érzékeny a visszacsatolás minőségére. Mivel az aktív zajcsökkentő rendszerek felhasználási tartománya néhány kHz-ig terjed, ezért a hálózat sebességét figyelembe véve még reménytelinek látszik a rendszer megvalósítása, bár nyilvánvalóan bizonyos kompromisszumok megkötése szükséges.

4. A szinkronizáció alapjai

A szinkronizáció kérdése már a számítógépes hálózatok megjelenése óta létező probléma, ennek megfelelően a téma már elég jól kidolgozott, bár a szenzorhálózatban történő szinkronizáció valamelyest eltérő felfogást igényel. Az alapvető fogalmak és összefüggések azonban a szenzorhálózatok esetén is fennállnak, ezért ebben a fejezetben ezek bemutatására kerül sor. Az itt közölt adatokról részletes leírás található [10]-ben.

4.1. Szinkronizáció szerepe

Ugyan a hálózatokban lévő egységek a kommunikáció révén kapcsolatban állnak egymással, ez a kapcsolat igen laza, és így az egységeket autonóm rendszereknek tekintjük bizonyos szempontból. Az idő ismerete sok esetben kulcsfontosságú kérdés, ezért általában minden hálózati egység rendelkezik önálló órával. Ezen órák – habár névlegesen megegyező értéket mutatnak – természetesen nem járhatnak teljesen pontosan, hiszen az őket működtető órajel-generátorok frekvenciái általában nem egyeznek meg teljesen pontosan. Ennek okai lehetnek például a gyártási bizonytalanságok, eltérő környezeti viszonyok, pl. hőmérséklet. Természetesen az órajel-generátorok pontossága különleges technikákkal növelhető, de ezzel általában az áruk is növekszik, így ezen eszközök alkalmazása olyan helyen, amely nem igényli kifejezetten a megnövelt pontosságot, nem gazdaságos. Az alkalmazások nagy része ráadásul nem is kíván ilyen pontos órajelet, hiszen az egyre növekvő pontosság nem szünteti meg a szinkronizálatlanság kérdését, csupán a megfelelő pontosság tartásához szükséges szinkronizációs intervallumok növekednek meg.

Az, hogy miért is szükséges a hálózatokban az órák szinkronizálása, többféle aspektusból is vizsgálható:

- **Adatfeldolgozás**: Sok esetben találkozunk olyan feladattal, ahol az adatok gyűjtése, vagy valamilyen esemény figyelése több helyen, elosztottan, a hálózati csomópontokban történik, a feldolgozást viszont központilag végezzük. Amennyiben az adatok nem függetlenek, fontos lehet azok sorrendisége. Konzisztens adatok gyűjtéséhez tehát megfelelő pontossággal kell ismernünk az időt. Ugyanez mondható bizonyos események figyelésekor. Szintén gyakorlati probléma az is, amikor az adatok keletkezésének ideje fontos szerepet játszik azok feldolgozásában (például a sebesség számítása elmozdulás és az ahhoz szükséges idő felhasználásával).

- **Megfigyelés:** A megfigyelő oldalon is fontos, hogy adott jelenségről megfelelő időpontokban készítsünk megfigyeléseket, méréseket, ezzel lehetővé tehetjük a feldolgozást. Különösen fontos ez, ha egy adott jelenségről egyszerre több helyről

is gyűjtünk adatokat, hiszen ezen struktúrák célja általában az, hogy több megfigyelés, és ezen adatok kiértékelése javítja a megfigyelés pontosságát. Nyilvánvaló, hogy ebben az esetben is fontos az idő megfelelő pontosságú ismerete. A pontosság attól függ, hogy milyen gyorsan változik a megfigyelt jellemző.

● **Hálózat**: A hálózati működés és a különféle protokollok szempontjából is fontos lehet a megfelelően pontos időzítési lehetőség, mely a csomópontok szinkronizálását igényli. Különösen fontos lehet ez például szenzorhálózatok esetén, ugyanis annak ismerete, hogy mikor kell számítanunk adatok érkezésére és mikor kell adatokat továbbítani, jelentős energia-megtakarítást eredményezhet, hiszen a kommunikáció szempontjából haszontalan időintervallumokban a rádió kikapcsolható, mely csökkenti az energiafelhasználást. Emiatt is kedveltek a szenzorhálózatokban a különféle időosztásos (TDM) protokollok.

Láthatjuk, hogy a szinkronizáció igen sok téren fontos kritérium, bár ezen követelmények sokszor igen hasonlóak, esetleg egy adott probléma más-más megvilágítása során merülnek fel.

4.2. Óramodell

A különböző szinkronizációs mechanizmusok és problémák megértéséhez szükség lehet néhány, az órák működését, azok hibáit jellemző összefüggésre. Ebben az alfejezetben ezeket foglalom össze.
Felsorolásszerűen ismerkedjünk meg néhány jelöléssel, definícióval:

- t: referencia idő, melyet a valódi időként fogadunk el
- N_i: i. node-ot (hálózati elemet) jelöli
- $h_i(t)$: az N_i node lokális ideje t referenciaidőben
- t_a: a esemény időpontja
- h^i_a: a esemény időpontja az N_i node lokális ideje alapján
- ofszet: $[h_i(t) - t]$, tehát a lokális óra eltérése a referencia órához képest

$$f_i(t) = \frac{dh_i(t)}{dt} \tag{4.1}$$

$$\rho_i(t) = f_i(t) - 1 \tag{4.2}$$

ahol $f_i(t)$ az óra gyorsaságát jellemzi, $\rho_i(t)$ mennyiséget pedig driftnek nevezzük, és azt határozza meg, hogy az óra milyen gyorsan távolodik a referenciaidőtől, hiszen $\rho_i(t)$ az ofszet deriváltjaként is kifejezhető.

A fenti jelölések ismeretében megfogalmazhatóak az órák jellemzésére szolgáló alapvető definíciók:

● **referencia óra**: $h_i(t) = t$, $\forall\ t$ esetén. Referencia óra esetén nyilvánvalóan $f_i(t) = 1$ teljesül, így $\rho_i(t) = 0$.

● **pontos óra**: $f_i(t) = 1$, $\forall\ t$ esetén. Ez azt fejezi ki, hogy az adott óra a referencia órával azonos sebességgel jár, viszont az általuk mutatott értékek nem feltétlenül egyeznek meg. Ekkor a lokális és referencia óra közötti különbség (ofszet) állandó, tehát: $h_i(t) - t = const.$

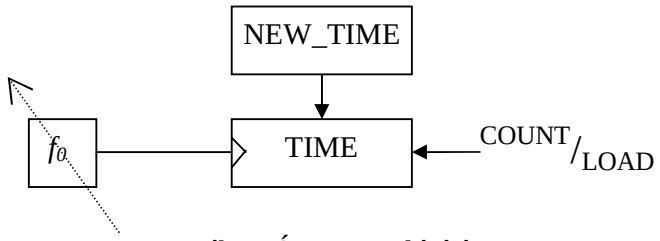
● **helyes óra**: $h_i(t_0) = t_0$, egyetlen t_0 esetén. Megfelel ez annak az esetnek, ha az órát t_0 időpillanatban szinkronizáltuk, és ezután $t > t_0$-ra:
a) amennyiben $f_i(t_0) < 1$, az óra lassabban jár, mint a referencia óra, így $h_i(t) < t$
b) amennyiben $f_i(t_0) > 1$, az óra gyorsabban jár, mint a referencia óra, így $h_i(t) > t$

● **korlátozott driftű óra**: $|\rho_i(t)| < \rho_{max}$. Ez a fizikailag megvalósított órákra általában igaznak tekinthető, és reális feltétel. Egy átlagos minőségű kvarc-oszcillátor által szolgáltatott órajel driftje általában 10 ppm…100 ppm nagyságrendbe esik. Ez meghatározza a referenciaórától viszonyított elcsúszás mértékét. Adott ΔT idő alatt ρ_{max} abszolút értékű drift esetén a referenciaórától való elcsúszás: $\Delta\tau = \pm\rho_{max} \cdot \Delta T$. Az óra megkívánt pontosságának ismeretében ($|\Delta\tau_{max}|$) tehát korlátozott driftű óra esetén meghatározható a szinkronizálási időpontok közötti maximális időtartam (ΔT_{max}). ρ_{max} paraméterrel jellemezhető az oszcillátorok pontossága, a drift ugyanis meghatározza a valódi és névleges frekvencia viszonyát.

● **korlátozott driftváltozású óra**: mivel a drift értéke a gyakorlatban ugyan nem állandó, viszont ingadozása általában lassan változó környezeti hatások miatt következik be, ezért ezen megkötés is jól jellemzi az órákat. Gyakorlati jelentősége az, hogy amennyiben tudomást szerzünk a drift értékéről (amire léteznek eljárások [11]), akkor biztosnak vehető, hogy a drift az így meghatározott érték adott környezetében marad megadott idő elteltéig. Ennek felhasználásával viszont jól becsülhető a folytonosan változó ofszet, és növelhető a szinkronizációs pontok között megengedhető maximális idő, ami például a szinkronizációval járó extra hálózati forgalom csökkentését segíti. A korlátozott driftváltozás az órajel-generátor oszcillátorának frekvenciastabilitására tesz megszorítást.

Látható, hogy az óra minőségi jellemzésére sokféle módszer kínálkozik. Ezen jellemzők lehetőséget nyújtanak bizonyos következtetések levonására, melyek a későbbiekben a szinkronizáció során hasznosak lehetnek.

Az eddigi fogalmak tárgyalása során az óra mint absztrakt dolog jelent meg. A 4.1. ábrán láthatunk egy tipikus példát az óra fizikai megvalósítására.



4.1. ábra. Óra megvalósítása

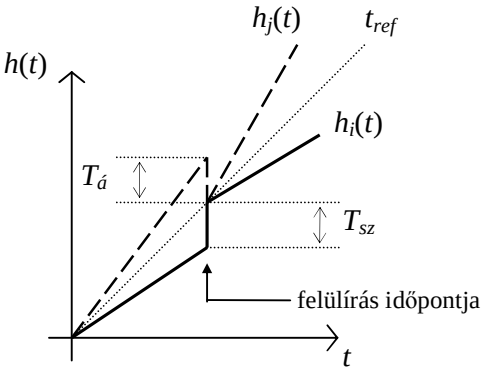
Az ábra alapján a működés a következő: a TIME címkével jelölt számláló blokk az f_0 frekvenciájú órajel-generátor ütemében lép felfele, így frissítve az általa „mutatott” időt. Ez a normál működési állapot.

Ezen folyamatba kétféleképpen avatkozhatunk be. Egyrészt külső NEW_TIME értéket töltünk a számlálóba annak frissítésekor, vagy változtatjuk az órajel-generátor eredetileg f_0 értékű frekvenciáját.

Általában az idő direkt frissítése – tehát új idő beírása – egyszerűbb, és az óra frissítése után az egyből szinkronizált állapotba kerül. Hátrányos tulajdonsága viszont, hogy az óra frissítéséhez szükséges idő néha kiszámíthatatlan, így nem biztos, hogy amire frissül az óra, még akkor is aktuális a beírt idő. Másik veszélyes dolog ezzel a módszerrel kapcsolatban, hogy a lokális idő így nem lesz folytonos, hiszen ha nem ugyanazt az értéket írjuk a TIME regiszterbe, mint amit eredetileg is tartalmazott – ami az esetek többségében teljesül – akkor vagy kétszer is ugyanazt mutatja az óra, vagy egyáltalán nem mutat bizonyos értékeket. Ez több szempontból is problematikus. Bizonyos feladat időzítésekor előfordulhat, hogy az adott feladat vagy nem hajtódik végre, vagy kétszer, esetleg többször is végrehajtódik. Események sorrendiségének mérésekor viszont megtörténhet, hogy két esemény sorrendjét jelző időbélyegek nem a helyes sorrendet jelzik. Az idő felülírásával kapcsolatos problémákat a 4.2.a ábrán vizsgálhatjuk.

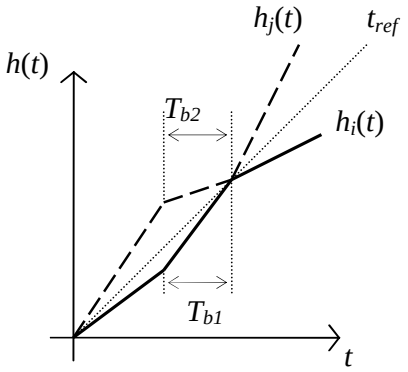
Amennyiben f_0 értéket változtatjuk meg, azzal gyorsítjuk, vagy lassítjuk az órát, így közelítjük meg a referencia időt. Ha az óra „siet”, akkor f_0-t csökkentjük, ha „késik”, akkor f_0-t növeljük, amennyiben a helyes értéket mutatja – vagy annak valamilyen hibasávján belül van –, f_0-t változatlanul hagyjuk. Ezen módszerrel biztosítható a lokális idő folytonossága. Hátránya, hogy az óra nem kerül egyből szinkronizált állapotba. A beállás folyamatát a 4.2.b ábrán láthatjuk.

N_i: referencia időhöz viszonyítva lassú node. Lokális ideje: $h_i(t)$
N_j: referencia időhöz viszonyítva gyors node. Lokális ideje: $h_j(t)$



$T_á$: átlapolódás az időben
T_{sz}: szakadás az időben

4.2.a ábra. Idő felülírása



T_{b1} : szinkronizálódás ideje
T_{b2}: szinkronizálódás ideje

4.2.b ábra. Óra gyorsítása/lassítása

Bármely beavatkozás lehetőséget ad az óra szinkronizálására. Az hogy melyik módszert választjuk, az alkalmazástól függ.

4.3. Kommunikációs modell

Mivel a szinkronizáció a hálózat csomópontjai közti kommunikáció révén valósul meg, így fontos szerepet játszik a szinkronizáció témakörében a kommunikáció jellemzőinek vizsgálata.

A kommunikáció a hálózatokban általában több bájtnyi adatot tartalmazó csomagokkal történik, melyek egy üzenetegységnek tekinthetők. Annak oka, hogy általában nem elegendő csupán egyetlen bájtnyi adat továbbítása – még ha ekkora is a teljes hasznos üzenet – az, hogy a hálózati protokollok megkívánnak bizonyos overhead-et, például forrás- és célállomás címe.

A szinkronizáció természetesen kommunikáció segítségével jön létre. Ez nyilvánvalóan ugyanúgy terheli a hálózatot, mint akármilyen üzenettovábbítás. Eredményes módszer lehet, ha a hasznos üzenetek továbbítását használjuk fel szinkronizációs célra, így kevésbé csökken a hálózat hatékonysága. Ezeket nevezzük implicit szinkronizációs üzeneteknek. Természetesen nem minden alkalmazás kínál lehetőséget implicit szinkronizációs üzenetek küldésére, hiszen ha az adatok továbbításának gyakorisága kisebb, mint a szinkronizációs üzenek küldésének gyakorisága, abban az esetekben explicit szinkronizációs üzeneteket kell a hálózatban elküldeni. Ezek az üzenetek csak szinkronizációs célt szolgálnak, és plusz növekedést okoznak a hálózat terhelésében, valamint az energiafogyasztásban.

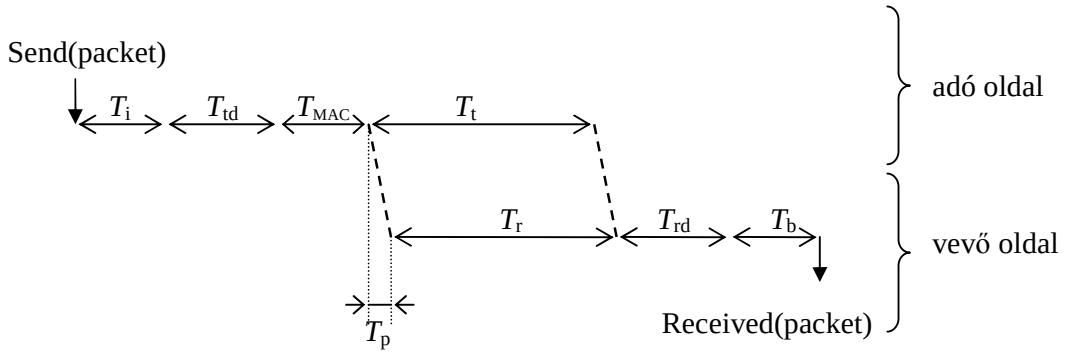
Annak alapján, hogy egy-egy nodepárban az üzenettovábbítás mindkét irányba lehetséges-e, vagy csupán az egyik irányba, megkülönböztetünk szimmetrikus és

aszimmetrikus kapcsolatot. Ha két azonos szerepű node-ról van szó, akkor általában szimmetrikus kapcsolat alakul ki. Szenzorhálózatokban az aszimmetrikus kapcsolat általában úgy jön létre, hogy léteznek olyan állomások, melyek nem telepes táplálásúak, és így az energiagazdálkodás nem játszik fontos szerepet. Ennek következtében nagyobb teljesítménnyel képesek üzeneteket sugározni. A célállomások viszont kis teljesítménnyel sugároznak az energiatakarékosság miatt, így az ő üzeneteik nem biztos, hogy eljutnak a célállomáshoz. Az aszimmetrikus kapcsolat sokszor megnehezíti a szinkronizációt, ugyanis az adatátvitel idejének, illetve annak bizonytalanságainak meghatározásához sokszor használnak fel az egyes szinkronizációs protokollokban üzenetváltásokat.

A szinkronizáció megvalósításában nagy nehézséget okoz a kommunikáció időzítési adatainak meghatározása, és azok bizonytalanságainak kezelése. A következőkben a csomagok küldésével kapcsolatos időzítési paramétereket ismerhetjük meg. Az időzítési értékek a 4.3. ábrán is nyomon követhetőek.

- adó oldal:
  - ► T_i: csomag elküldését kezdeményező parancs kiadása, és annak elfogadása közt eltelt idő. Nagysága függ a processzor terheltségétől, az azon futó folyamatok számától, így ingadozása nagy lehet.
  - ► T_{td}: kommunikációs eszköz és a vezérlő közti kommunikáció. Az adást kezdeményező parancs elfogadása után a processzor elkezdi a kommunikációt megvalósító eszköznek átadni az adatokat, melynek időtartama T_{td}. Értéke viszonylag fix.
  - ► T_{MAC}: közeg-hozzáférési idő. Amennyiben a csatorna foglalt, várni kell míg felszabadul. Értéke függ a hálózat terheltségétől.
  - ► T_t: adatátviteli idő. Az az idő, míg az adó továbbítja az adatot a vevő felé.
- jelterjedési idő: T_p: az az idő, míg a jel az adótól a vevőhöz érkezik. Értéke általában maximum néhány μsec.
- vevő oldal:
  - ► T_r: vételi idő. Az az idő, míg a vevő fogadja az adó által sugárzott adatot. Ez átlapolódik az adatátviteli idővel, hiszen az adás és a vétel egy T_p eltolástól eltekintve egyszerre történik.
  - ► T_{rd}: a kommunikációs eszköz és a vezérlő közti adatátvitel. A kommunikációs eszköz jelzi, hogy adat érkezett. Ezután a processzor elkezdi a vett adatok beolvasását. A jelzés kezdete és beolvasás vége között eltelt idő T_{rd}.
  - ► T_b: a megfelelő alkalmazási réteg értesítéséhez szükséges idő. Nagysága függ a processzor terheltségétől, az azon futó folyamatok számától.

Eszköztől függően T_{td} és T_t, valamint T_{rd} és T_r is átlapolódhat, tehát a kommunikációs eszköz nem biztos, hogy az adás kezdésével megvárja, míg minden továbbítandó adatot rendelkezésére bocsát a CPU, illetve a CPU nem biztos, hogy megvárja az adatok kiolvasásával, míg az összes fogadni kívánt adat beérkezik.



4.3. ábra. Üzenetküldés –fogadás közti késleltetések

Látható, hogy a kommunikáció lefolyása során igen sokfajta késleltetési idő jelentkezik a kommunikációs csatornában. Sokszor nem is maga a késleltetés nagysága, hanem annak ingadozása okozza a legnagyobb problémát, melynek kiküszöbölése nehéz feladat.

4.4. Szinkronizáció típusai

A szinkronizációt lehetővé tevő kommunikáció, illetve a szinkronizáció céljául szolgáló óra tulajdonságainak megismerése után ebben az alfejezetben a szinkronizáció különböző eseteit taglaljuk. A szinkronizáció természetesen soha nem lehet tökéletes, ezért a továbbiakban, ha a szinkronizált állapotra hivatkozom, akkor abba beleértendő a hozzá tartozó hiba is, melynek értéke alkalmazástól függően széles tartományban mozoghat.

- **külső/belső szinkronizáció**: amennyiben a hálózat elemei egy központilag meghatározott időhöz szinkronizálódnak, akkor külső szinkronizáció valósul meg. A központi idő lehet egy master saját maga által fenntartott lokális idő, vagy egyéb módszerrel (pl. GPS) a globális időhöz igazított óra. Ha a feltétel csupán a hálózatban lévő node-ok konzisztenciája, akkor belső szinkronizációról beszélünk. Ebben az esetben nincsen egy referenciának tekintett idő, lényeg, hogy minden óra egymáshoz képest bizonyos hibahatáron belül mozogjon.

- **folytonos/igény szerinti**: folytonos szinkronizációról akkor beszélünk, ha az adott órának minden időpillanatban szinkronizálva kell lennie. Előfordul azonban sok olyan alkalmazás, mely nem kíván meg folyamatos szinkronizált állapotot, így a szinkronizáció fenntartása erőforrás-pazarlás lenne. Ekkor kap létjogosultságot az igényre történő szinkronizáció, mely lehet akár utólagos is. Ez alatt azt kell érteni, hogy ha például egy esemény időpontjának meghatározása a cél, akkor elég utólag szinkronizálni az órákat, és mérni az esemény valamint a szinkronizációs pont között eltelt időt, így az esemény valódi időpontja kiszámítható. Ez a módszer

természetesen nem alkalmazható akkor, ha célunk valamely feladat időzítése, ekkor a feladat végrehajtási időpontja előtt már ismerni kell a valós időt.

● **abszolút/relatív**: abszolút szinkronizációról akkor beszélünk, ha a szinkronizált órák ugyanazt az értéket mutatják, tehát cél: $h_i(t) = t$. A relatív szinkronizáció, vagy más néven együttfutás azt jelenti, hogy az óráknak nem kell ugyanazt az értéket mutatni, de ugyanolyan gyorsan kell járni, tehát a cél: $f_i(t) = 1$ biztosítása. Relatív szinkronizációt főleg ott alkalmaznak, ahol a feladat időintervallumok mérése. Ez a módszer ekkor előnyösebb, mivel megvalósítása általában egyszerűbb.

● **valós/virtuális**: a valós szinkronizáció azt jelenti, hogy a szinkronizált órák minden esetben ugyanazt az értéket mutatják: $h_i(t) = t$. Megvalósítása kétféleképpen lehetséges, a két módszer eredménye megegyezik:

a) folyamatos ofszetkiegyenlítés: $h_i(t_k) = t_k$; $\forall t_k$ esetén

b) egyszer ofszetkiegyenlítés, aztán együttfutás: $h_i(t_0) = t_0$; $f_i(t_k)=1$ $\forall t_k \neq t_0$ esetén

Virtuális szinkronizáció esetén is minden egyes node ismeri a referencia időt, óráik viszont nem azt mutatják. Ez úgy lehetséges, hogy a node-ok lokális órái a saját ütemükben járnak, viszont ismerik a referencia órához viszonyított ofszet és drift értékét. Ezek már elegendőek egy lineáris modell felállításához, mely egy c transzformációt valósít meg a lokális és referencia idő között, tehát: $c(h_i(t)) = t$. Az első módszer inkább kommunikációigényes, míg a második főként memória- és számításigényes.

A fent bemutatott elvek közül az adott probléma elemzésével az alkalmazásnak megfelelőt célszerű felhasználni feladatunk megoldása során.

4.5. Hagyományos és szenzorhálózatban használt algoritmusok

A szinkronizációs problémák már régóta, a számítógép-hálózatok megjelenése óta ismertek, és így számtalan szinkronizációs megoldás született az idők során. A szenzorhálózatok alkalmazása során is nagyon gyakran felmerül ez az igény. A feladat ugyan megegyezik mindkét esetben, hiszen cél az, hogy az elosztottan működő órákat egymással összhangban tartsuk. Az eltérő eszközök használata azonban a probléma más szemszögből történő megközelítését igényli. Az eddig kifejlesztett algoritmusok nem minden esetben ültethetőek át, persze bizonyos alapelvek átvétele hasznos lehet.

Az eltérő felépítés több ponton is megnyilvánul. Először is a szenzorhálózatban található node-ok jóval szerényebb erőforrásokkal rendelkeznek, mint a hagyományos hálózatokban lévő számítógépek. Ennek következménye, hogy az algoritmusok bonyolultságát, számítás- és memóriaigényét ezekhez a szerény erőforrásokhoz kell igazítani. A szenzorhálózatok kommunikációs csatornája ráadásul sokkal bizonytalanabb, mint a hagyományos rendszerekben, és minősége is gyorsan változhat

az időben. Ugyan a hálózat kiterjedése meg sem közelíti például a világot beszövő Internet méretét, de topológiája igen dinamikusan változhat.

A hagyományos szinkronizációs eljárások általában a maximális szinkronizációs pontosságra törekszenek. Az alkalmazások egy része viszont nem biztos, hogy igényli ezt a precizitást, és ezzel a megnövelt erőforrás-felhasználást. Emiatt többféle minőségű szinkronizációs protokoll kidolgozására is igény lehet.

Nem szabad arról sem megfeledkeznünk, hogy a szenzorhálózatok elemei telepes táplálásúak, és egy telepnek akár igen hosszú ideig is ki kell tartania. Ennek érdekében célszerű mind a processzort, mind a rádiót kis fogyasztású nyugalmi állapotban tartani. Ezt viszont csak úgy lehet megtenni, ha például szinkronizáció esetén nem kell véletlenszerű szinkronizációs üzenetek fogadására felkészülni.

Az eddig felsorolt tulajdonságok főleg megszorításokat vetnek fel a szenzorhálózatok alkalmazásával kapcsolatban. Jelentős előnyt hozhat viszont, hogy az itt használt eszközök felépítése jóval egyszerűbb, átláthatóbb, és nagyobb rugalmassággal kezelhetők, mint a hagyományos számítógépes hálózatok igen bonyolult elemei. Ennek következményeként a megvalósítandó algoritmusok kidolgozásakor bizonyos szempontból kevésbé kötött a tervezők keze.

4.6. Szinkronizációs algoritmusok

Az előző alfejezetekben láthattuk, hogy milyen szempontokat kell szem előtt tartani a szinkronizációs algoritmus tervezésekor, és hogy milyen főbb irányelveket lehet követni. Ebben a részben megismerhetünk néhány, széles körben ismert, és kifejezetten szenzorhálózatokra kifejlesztett algoritmust.

4.6.1. Time-stamp Synchronization (TSS)

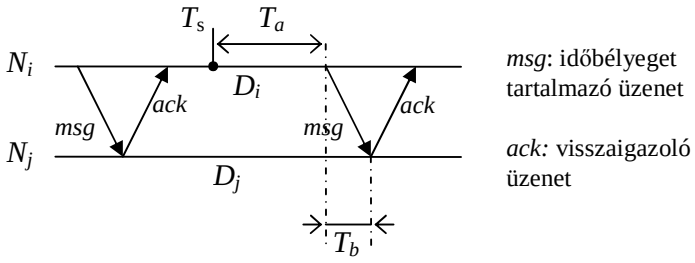
A TSS algoritmus [12] az időbélyegek konverzióján alapuló virtuális szinkronizáció. Az üzenetekben érkező időbélyegeket minden node a saját lokális idejére transzformálja, és ha nem ő a címzett, akkor ezt az időbélyeget küldi tovább. Ily módon az üzenetben található időbélyeg mindig az aktuális küldő lokális idejében értendő, tehát a végső címzettnek is a szomszédja lokális idejét kell ismerni. Egy időbélyeget tartalmazó csomag érkezésekor a vevő egy visszaigazoló üzenetet (ack.) küld.

Az üzenetet fogadó node az új időbélyeget úgy számítja, hogy az üzenet érkezési idejéből kivonja az adott időbélyeg korát. Az időbélyeg kora két adatból áll össze:

a) keletkezésétől a továbbítás időpontjáig tartó idő: T_a

b) az elküldéstől a megérkezéséig eltelt idő: T_b

Ezen két értéket külön-külön számítjuk. Ennek megértéséhez vizsgáljuk meg a 4.4. ábrát. A T_a időt maga az adó tudja meghatározni, ez egyszerű időmérés (T_s jelöli az időbélyeg keletkezési időpontját). A T_b mérése ún. round-trip módszerrel történik.

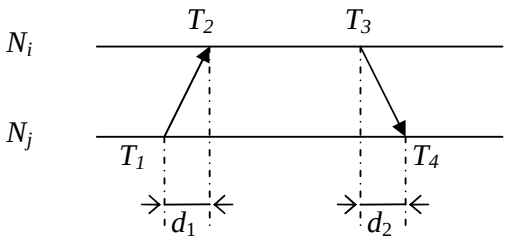


4.4. ábra. Round-trip módszer szemléltetése

T_b becsülhető a (D_j - D_i) értékből, és az üzenetküldés minimális idejéből. Dj-t és D_i-t az előző üzenetcserétől mérjük, ha ez az első adás, akkor meg kell tenni egy extra üzenetcserét. T_a és T_b ismeretében a konverzió elvégezhető. Az algoritmus ezen érték bizonytalanságát a maximális drift segítségével számolja. Energiatakarékossági szempontból fontos, hogy az ack. üzenetként az időbélyeget tartalmazó üzenet továbbítása is felhasználható, ugyanis a rádiós kommunikáció miatt ezt az előző forrás is hallja.

4.6.2. Timing-Sync Protocol for Sensor Networks (TPSN)

A TPSN protokoll [13] alapja, hogy a hálózatban található node-ok fába szerveződnek, és a fa ágain lévő node-ok egymáshoz szinkronizálódnak. A fa gyökerében lévő root kezdeményezi a szinkronizációt, és a fában egyre mélyebbre haladva sorra kerülnek a node-ok szinkronizált állapotba. A protokoll megoldást kínál a fa felépítésére, valamint azon esetek kezelésére, amikor egy-egy node kiesik, vagy újonnan belép a hálózatba. A node-ok egymás között a TEMPO algoritmus segítségével szinkronizálódnak. Az algoritmus magyarázatához tekintsük a 4.5. ábrát.



Feltételezés: $d_1 = d_2 = d$

$T_2 = T_1 + \Delta + d$
$T_4 + \Delta = T_3 + d$

d: adatátvitel ideje
$\Delta = h_i(t) - h_j(t)$
Δ : ofszet a két node között

4.5. ábra. TEMPO algoritmus szemléltetése

A szinkronizációt N_j kezdeményezi. N_i feljegyzi a csomag érkezését (T_2), aztán válaszként visszaküld egy csomagot, melyben T_3, tehát a küldés ideje is bekerül. N_j ezen csomagot T_4 időben kapja meg. A 4.5. ábrán látható egyenletek alapján meghatározható Δ paraméter, mely a két node lokális ideje közti ofszetet határozza meg. Ha N_j a $h_j(t)$

lokális idejét Δ-val módosítja, akkor szinkronizálódik N_i-hez. Az üzenetcsere segítségével sikerült eliminálni az üzenetküldés idejét.

4.6.3. Reference-Broadcast Synchronization (RBS)

Az RBS algoritmus [14] alapötlete a következő: ha egy N_k node broadcast üzenetet küld, akkor azt a hálózatban található N_i és N_j node egyszerre kapja meg, mivel két node esetén az üzenet küldése és megérkezése között eltelt időben a – 4.3. alfejezet terminológiáját követve – csupán T_{rd} és T_b különbözik, ugyanis csupán ezen két paraméter származik független körülményekből. Mivel azonos felépítésű eszközök esetén T_{rd} és T_b viszonylag álladónak tekinthető, amennyiben a processzor nem túl terhelt, így az a feltételezés, hogy a broadcast üzenet egyszerre érkezik meg a két node-on, helyesnek tekinthető. Ezt az üzenetet a két node szinkronizációs pontnak tekinti, és elküldik egymásnak azt az időpontot, amikor az üzenetet a lokális idejük szerint megkapták. Ez alapján megállapíthatják egymásról, hogy mennyiben térnek el a lokális idők, ezzel lehetőség nyílik a szinkronizációra. Az algoritmus több node-ra könnyedén kiterjeszthető. Amennyiben nem tud a hálózat minden eleme minden más elemmel kommunikálni, akkor a hálózat olyan csoportokra bomlik, ahol mindenki mindenkit hall, és a kapcsolatot a csoportok között olyan node-ok teremtik meg, mely több csoportnak is tagjai.

4.6.4. Flooding Time-Synchronization Protocol (FTSP)

Az FTPS algoritmusban [15] az RBS-hez hasonlóan broadcast üzenetek képzik a szinkronizáció alapját. Különbség viszont az, hogy a csomagban az adó elhelyezi a csomag fizikai küldésének kezdetét. A vevő szintén figyeli a csomag fizikai érkezésének kezdetét, amely nagy pontossággal megegyezik a küldés kezdetével. Ez a 4.3. ábrán látható T_t és T_r közti átlapolásból következik, csupán T_p csúszás van közöttük. Ez azt jelenti, hogy a vett csomagban lévő idő, és annak megérkezése közti különbség megegyezik a két node lokális órája közti ofszettel, ami alapján elvégezhető a szinkronizáció. A hálózatban a szinkronizációt egy root kezdeményezi. A szinkronizálódott node-ok eztán további szinkronizációs üzenetet küldenek, és így árasztják el a hálózatot szinkronizációs üzenetekkel, tehát minden, a hálózatban lévő node-nak lehetősége nyílik a szinkronizálódásra. Természetesen az algoritmus megoldást kínál arra az esetre, ha a root kiesik, ha új node lép be, és próbálja minimális szinten tartani a szinkronizációs broadcast üzenetek számát, amely az energiatakarékosság és a hálózat sávszélességének minél jobb kihasználása szempontjából fontos.

4.6.5. Értékelés

A bemutatott algoritmusokban az egyes paraméterek meghatározása pontosításának érdekében átlagolást alkalmaznak. Az algoritmusok mindegyike megegyezik abban, hogy próbálják redukálni az üzenetváltások számát. A RBS és az FTSP igen hasonló egymáshoz, hiszen a szinkronizáció broadcast üzeneteken alapul, így az üzenetet megkapó node-ok az RBS alapelvének megfelelően igen nagy pontossággal szinkronban lesznek. A különbség abban áll, hogy RBS esetén a szinkronizáció extra üzenetváltásokat kíván, míg az FTPS esetén a referencia időt is továbbítják, így az üzenet megérkezése után a szinkronizáció végrehajtható, mely nyilvánvaló előnyként értékelhető. Az FTPS-ben és a TPSN-ben viszont jelentősen építenek a kommunikációs eszközként szolgáló rádió által kínált hardveres támogatásokra az időbélyeg előállításában, aminek kihasználása jelentős szoftveres munkát igényel, így a fejlesztés ideje növekedhet. Az RBS-ben viszont a rendelkezésre álló szoftver eszközökkel próbálnak eredményeket elérni, ami megkönnyíti az algoritmus különböző platformokon történő implementálását. A két módszer tehát két különböző felfogást képvisel. A fejlesztés során át kell gondolni, hogy mennyire tudjuk kihasználni a rendelkezésre álló hardveres és szoftveres erőforrásokat, hiszen új programok kifejlesztése és tesztelése sok időt emészthet fel, viszont némely esetben elkerülhetetlen.

4.7. Szinkron mintavételezés

Az eddig bemutatott algoritmusok főleg olyan rendszerekben alkalmazhatók, ahol az adatok gyűjtése statisztikai célokból, adatbázisok létrehozásához, vagy azok utólagos feldolgozásához szükséges. Ekkor elegendő az összegyűjtött adatokhoz a keletkezésüknek megfelelő időbélyeget rendelni. Ehhez az szükséges, hogy az adatokat előállító node-ok szinkronizálva legyenek, és ismerjék a referencia időt.

Egy másik irányvonalat követnek az olyan rendszerek, amelyekben bizonyos mennyiségek folyamatos mintavételezése szükséges, és az így előálló adatokat jelfeldolgozási, vagy szabályozástechnikai célokra használják fel. Az eddig elterjedt hagyományos algoritmusok több-bemenetű rendszerek esetén azonos időpontban érkező minták felhasználásával működnek. Általános esetben az sem biztosítható, hogy a feldolgozás és az adatok keletkezésének sebessége megegyezik. Ezen problémák megoldására számos szinkronizációs lehetőség létezik. Az, hogy melyiket használjuk, sok tényező függvénye lehet, például az átviendő adatmennyiség és annak továbbítására felhasználható csatorna sávszélességének viszonya, az adatok keletkezési sebessége, vagy az egységek számítási kapacitása.

Egyik megoldási lehetőség, ha a szenzorok a feldolgozó egység kérésére állítják elő az adatokat. Ez kétféleképpen is lehetséges. Első megoldás az, hogy a kérés hatására megfigyelést végeznek az adott mennyiségen, és a megfigyelés eredményeként létrejövő adatot továbbítják a feldolgozó egységhez. Másik megoldás, hogy a szenzorok

folyamatos megfigyeléseket végeznek, és a kérés időpontjában a megfigyelések alapján becslést adnak a jel pillanatértékéről, például valamilyen interpolációs technika segítségével. Ebben az esetben tehát a szinkronizáció elosztottan, az egyes állomásoknál történik, így tehermentesítik a központi egységet a szinkronizáció feladata alól.

Amennyiben nem elegendően nagy a hálózat sávszélessége, akkor nem használható ez a módszer, hiszen a minták kérése extra terhelést jelent a hálózat számára, ráadásul előfordulhat, hogy egy adatkérő üzenet kiadása hosszabb ideig tart, mint a mintavételek közötti időintervallum. Ebben az esetben vagy más módon jelöljük ki a mintavételi időpontokat, vagy olyan elrendezést használunk, melyben nem központilag határozzunk meg a mintavételezés időzítését.

Az utóbbi esetben a megfigyelést végző csomópontok saját maguk állítják elő a mintákat, és ezeket továbbítják a jelfeldolgozó egység felé, tehát a mintavételezés időzítése lokálisan történik, és a szenzorok folyamatosan továbbítják a rendelkezésre álló adatokat. A szinkronizáció feladata tehát a feldolgozó egységre hárul. A szinkronizáció ilyen esetben úgy történik, hogy a beérkezett adatok segítségével a jelfeldolgozó-, vagy szabályzó algoritmus számára szükséges időpontban állítjuk elő az adott mennyiség pillanatértékét valamilyen interpolációs technikával. Ennek legegyszerűbb módszere, hogy mindig a legutoljára érkezett mintát tekintjük pillanatértéknek – nulladrendű tartó – , de jó megoldást jelent például a lineáris interpoláció használata is. Amennyiben jobb minőségű becslésre van szükségünk, magasabb rendű interpolációt kell használni.

A központi szinkronizáció esetében azonban problémát jelent, hogy az egyes szenzorok mintavételezése egymástól függetlenül történik. Még névlegesen megegyező mintavételi frekvencia esetén is egyes szenzorok esetén eltérő a mérési eredmények előállítási sebessége, és elcsúsznak egymáshoz képest a mintavételi időpontok. Amennyiben több szenzor adatát kell szinkronizálni a jelfeldolgozási algoritmushoz, ez jelentős számítási teljesítményt és memóriát igényelhet, ugyanis minden szenzor esetén a pontos interpoláció számításához mérni kell az adatok beérkezési idejét, melyet legegyszerűbb esetben az adatok keletkezési idejének tekinthetünk.

Egyszerűbb rendszert eredményez, ha sikerül biztosítani, hogy a szenzorokról megegyező sebességgel érkezzenek az adatok. Ez két okból is előnyös. Egyrészt, ha a szenzorokból egyszerre érkeznek az adatok, vagy legalábbis átlagosan egyenlő sebességgel, akkor nem kell a csatornák adatait különböző sebességgel feldolgozni. Másik előny a hálózat működésében mutatkozik meg. Amennyiben nagy mennyiségű adatot kell továbbítanunk egy többszörös hozzáférésű csatornán, akkor célszerű determinisztikus hozzáférési protokollt kialakítani. Ekkor – amennyiben minden állomás megegyező mintavételi frekvenciával dolgozik, tehát elvileg átlagosan megegyező számú adat keletkezik – minden node adott időben továbbíthatja adatait. Ez igen jól működik, ha minden node mindig megegyező mennyiségű adatot továbbít egy hozzáférés alkalmával, hiszen állandó méretű csomagok esetén kiszámítható az üzenetküldés ideje. Gyakorlatban viszont a névlegesen egyező mintavételi frekvencia esetén is előfordulnak eltérések a valódi mintavételi frekvenciákban. Ekkor viszont a gyorsabban mintavételező szenzorokban felhalmozódnak az adatok, hiszen a hálózat

sebességét a leglassabb mote-hoz kell igazítani, mert így lehet biztosítani a folyamatos adatáramlást. Az sem jelent megoldást, hogy minden állomás az adatok keletkezésének ütemében küldi az adatokat, hiszen ekkor a mintavételi időpontokkal az üzenetküldések is csúsznak, mely az elküldött csomagok ütközéséhez vezet, és a hálózat determinisztikusságát sérti. Szintén a determinisztikusság rovására megy, ha az eltérő mintavételi frekvencia miatt keletkező extra adatokat új csomagban, vagy a szokásosnál hosszabb csomagban továbbítja egy node.

Az átlagosan megegyező sebességgel érkező adatok biztosítására kínálkozó egyszerű lehetőség, ha minden szenzor minden küldés alkalmával a legutolsó N_u mintát továbbítja. Ez azt jelenti, hogy átlagosan megegyezik az adattovábbítás sebessége, hiszen minden állomás megegyező idejű adási jogot kap, és az adások alkalmával megegyező mennyiségű adatot továbbít. Ekkor tulajdonképpen a hálózatban a leglassabb állomásnál gyorsabban mintavételező node-ok eldobják a gyorsabb működés miatt keletkező extra mintákat. Ezzel a módszerrel az lehet a probléma, hogy a mintavételi időpontokban ±½ mintavételi időköznyi bizonytalanság lehet a vevő oldalon, hiszen az csupán egyenletesen érkező adatokat kap, a mintavételi időpontok azonban elcsúsznak egymáshoz képest. Az elcsúszás maximum 1 mintavételnyi időköz lehet, hiszen ennél nagyobb elcsúszást a „felesleges” minta eldobásával korrigálja az állomás.

Kifinomultabb módszert jelent az, amennyiben szinkronizáljuk a mintavételezést. Ez azt jelenti, hogy valamilyen módszerrel biztosítjuk, hogy a szenzorokon a mintavételezés mindig egyszerre történjen, de akár az is megengedhető, hogy a mintavételezések egymáshoz képest mindig fix különbséggel történjenek, hiszen az adatok keletkezési sebessége ebben az esetben is megegyezik.

Az ebben az alfejezetben ismertetett irányelvek általános módszereket mutatnak be arra az esetre, amennyiben célunk több forrásból érkező adatok szinkronizálása. Az alkalmazott módszer kiválasztása a konkrét alkalmazás által támasztott követelmények vizsgálatával lehetséges, implementálási módjuk pedig szintén a kialakított rendszer felépítésétől függ.

5. Zajcsökkentő rendszer szinkronizációs kérdései

Az eddigiekben áttekintettük a szinkronizáció általános kérdéseit. Ebben a fejezetben a megtervezendő zajcsökkentő rendszerben jelentkező szinkronizálási problémák és a szinkronizációra felhasznált algoritmus ismertetése történik, különös hangsúlyt fektetve a szenzorhálózatra.

5.1. Szinkronizálatlanság hatásai

Az alfejezetben egy olyan zajérzékelő hálózattal foglalkozunk, melyben a zajjelet mintavételező mote-ok a zajérzékelő mikrofon kimenetét a névlegesen megegyező mintavételi frekvenciával mintavételezik, az órajel-generátorok névlegestől eltérő frekvenciája miatt azonban a fizikailag megvalósuló mintavételi frekvenciák eltérnek. Ha nem végzünk semmilyen szinkronizációt, akkor ez többek között az 5.1.1.-ben és 5.1.2.-ben ismertetett problémákhoz vezet. Ezekben az esetekben két darab moteból álló hálózat működik, a mintavételezés két helyen valósul meg: egy gyorsabban, f_{gy} mintavételi frekvenciával – T_{gy} mintavételi időközzel –, és egy lassabban, f_l mintavételi frekvenciával – T_l mintavételi időközzel – dolgozó mote-on, melyek névleges mintavételi frekvenciája megegyezik. Ekkor a gyorsabb mintavételi frekvenciával üzemelő mote f_{gy} frekvenciával mintavételezi az adott jelet, a keletkező adatok feldolgozása viszont f_l frekvenciával történik. Ez azért szükséges, hiszen a folyamatos adattovábbítás csupán így lehetséges mindkét csatorna esetén. f_l-nél gyorsabb feldolgozás esetén a lassabb mote nem tud elegendő adatot szolgáltatni.

5.1.1. Frekvenciabecslési hiba

Mivel a jelenség periodikus jelek esetén lép fel, ezért indokolt az eltérő mintavételi és feldolgozási frekvencia periodikus jelek esetén fellépő hatásának vizsgálata. Legyen a mintavételezett jel az egyszerűség kedvéért egy A amplitúdójú és f frekvenciájú szinuszjel. Mivel a gyorsabb mote esetén a mintavételi időköz T_{gy}, így a jel időfüggvénye:

$$g(n) = A \cdot \sin(2 \cdot \pi \cdot f \cdot n \cdot T_{gy}) \tag{5.1}$$

ahol f a vizsgált szinuszos jel frekvenciája, A pedig az amplitúdója.

Az adatfeldolgozás, mint már említettük, T_l-nek megfelelő mintavételi időköz szerint történik, tehát a $g(n)$ függvényű jelből érkező n-edik minta a függvény $n \cdot T_l$ időpontbeli értékének felel meg a feldolgozás szempontjából, míg a valóságban a gyorsabb mintavételezés helyén ez az $n \cdot T_{gy}$ időpontot jelöli.

A jelenség értékeléséhez végezzük el a következő átalakítást:

$$g(n) = A \cdot \sin(2 \cdot \pi \cdot f \cdot n \cdot T_{gy})$$

$$g(n) = A \cdot \sin(2 \cdot \pi \cdot f \cdot n \cdot T_{gy} \cdot \frac{T_l}{T_l})$$

$$g(n) = A \cdot \sin(2 \cdot \pi \cdot f \cdot \frac{T_{gy}}{T_l} \cdot n \cdot T_l)$$

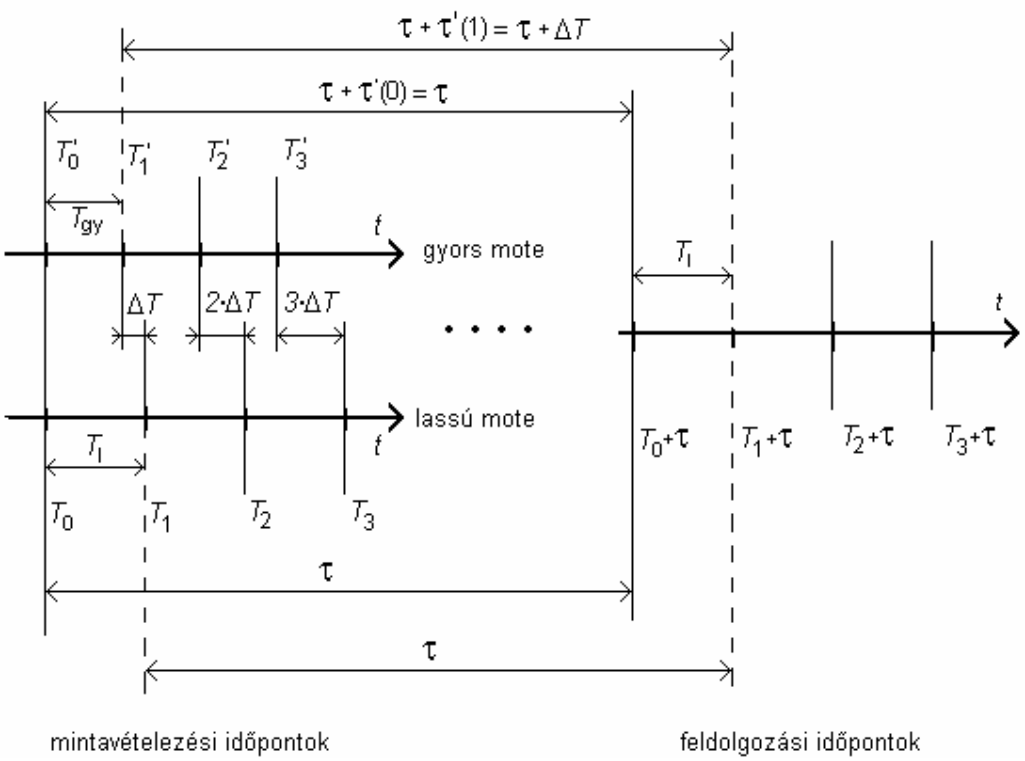
$$g(n) = A \cdot \sin\left[2 \cdot \pi \cdot \left(f \cdot \frac{T_{gy}}{T_l}\right) \cdot n \cdot T_l\right] = A \cdot \sin(2 \cdot \pi \cdot f_2 \cdot n \cdot T_l) \tag{5.2}$$

Ha tehát a mintavételi időköz T_{gy}, de mi ehelyett T_l-t feltételezünk, akkor a valójában f frekvenciájú jelet f_2 frekvenciájúnak mérjük, ahol $f_2 = \frac{T_{gy}}{T_l} \cdot f$. Mivel $T_{gy} < T_l$ ezért az így előálló jel frekvenciája csökken az eredeti jelhez képest.

A frekvenciabecslés hibájának hatása rezonátoros zajcsökkentő rendszer esetén szemléltethető a legjobban. Ebben a rendszerben a zajra vonatkoztatott átviteli függvény zérusai a rezonátorfrekvenciákon vannak, amelyet a referenciajel segítségével lehet meghatározni. Amennyiben zaj becsült frekvenciája – így a rezonátorpozíciók is – eltérnek a valós frekvenciától, abban az esetben a rendszer zérusai nem esnek az elnyomandó jel frekvenciájára, tehát nem érhető el teljes zajelnyomás. (5.2) alapján viszont láthatjuk, hogy szinkronizálatlan esetben ilyen frekvenciabecslési hiba jöhet létre, tehát ez a zajcsökkentő rendszer hatékonyságának csökkenéséhez vezet. Gyakorlatban a mintavételi frekvenciák eltérése oly csekély, hogy nem jelentős a hatékonyság csökkenése.

5.1.2. Átviteli függvény megváltozása

Mint már ismertettük, az aktív zajcsökkentő rendszerek igen érzékenyek a másodlagos átviteli függvény megváltozására. Az adatgyűjtő hálózat is részt vesz ezen átviteli függvény kialakulásában az adat keletkezése és feldolgozása között eltelt idő miatt megjelenő késleltetéssel. Ebben a szakaszban azt vizsgáljuk, hogy milyen hatással van a szinkronizálatlan mintavételezés a szenzorhálózat által reprezentált késleltetés megváltozásában. A folyamatot az 5.1. ábrán vizsgálhatjuk. Tételezzük fel, hogy a hálózatban két mote található, egyik T_{gy}, másik T_l mintavételi időközzel dolgozik, és az adatok feldolgozása T_l időközönként történik. Az ábrán az egyszerűség kedvéért olyan helyzetből indultunk ki, melyben két mintavételi időpont megegyezik, ez szinkronizált esetben sem feltétlenül szükséges. τ-val jelöltük a minták keletkezése és feldolgozása közötti időkülönbséget a lassabb mote esetén.



5.1. ábra. Átviteli függvény megváltozásának magyarázata

az ábrán:

$$T'_{n+1} - T'_n = T_{gy} \tag{5.3}$$

$$T_{n+1} - T_n = T_l \tag{5.4}$$

$$\Delta T = T_l - T_{gy} \tag{5.5}$$

T_i jelöli a lassú mote-on, T'_i pedig a gyors mote-on a mintavételi időket. A T_i-ben, és T'_i-ben vett adatok feldolgozása pedig a $T_i+\tau$ időpontokban történik.

A mintavételezést T_0 időponttól figyeljük mindkét mote-on. A beolvasási időköz megegyezik a lassabb mote mintavételi időközével, azaz T_l-lel, míg a gyorsabb mote T_{gy} időközönként vesz mintát, de azt a vevő szintén csak T_l időnként dolgozza fel.

A feltételekből adódik, hogy a lassabb mote és a fogadó állomás közti késleltetés állandó, de vizsgáljuk meg, hogy a gyorsabb mote és a fogadó állomás közti késleltetés hogyan változik. Jelöljük ezt a változást τ'-vel. Mivel $T_n \neq T_n'$, ami az ábrából leolvasható, emiatt a gyorsabb mote és a fogadó állomás közti késleltetés minden mintavétel alkalmával ΔT-vel nő. Tehát megállapítható, hogy a késleltetés változása az (5.6) képlet szerint alakul, és átviteli függvényét az (5.7) képlet adja meg, mivel a késleltetés a minták időbeni eltolásának felel meg.

$$\tau'(n) = n \cdot \Delta T \tag{5.6}$$

$$W_{kesl}(f) = e^{-j2\pi\tau'(n)f} \tag{5.7}$$

A növekvő késleltetés tehát az átviteli függvény fázistolásában jelenik meg, mivel egy $\tau'(n)$ késleltetést az (5.7) szerinti szorzóval vehetünk figyelembe. Amennyiben a mintavételi frekvenciák megegyeznek, akkor a $\tau'(n)$ állandó, és az identifikáció során meghatározható, tehát nem okoz zavart. Ennek ellenére a minél gyorsabb működés érdekében érdemes a késleltetést lehető legkisebb értéken tartani, hiszen egy szabályozási kör visszacsatoló ágában lévő késleltetés lassabb szabályozó tervezését teszi csak lehetővé. Így van ez a zajcsökkentő rendszereknél is. A valóságban azonban, mint láttuk, $\tau'(n)\cdot$nem állandó, ezért működés közben az általunk identifikált átviteli függvénytől el fog térni a valódi átviteli függvény. A második fejezet alapján kijelenthető, hogy ez akkor okoz instabilitást, ha a valóságos és mért átviteli függvény fáziskarakterisztikája közti különbség a működési frekvencián meghaladja a 90º-ot.

Igen informatív lehet, ha meghatározzuk, hogy ez az eset egy átlagos rendszerben mikor következik be. Ez az idő ugyanis elég nagy lehet ahhoz, hogy bizonyos esetekben ne kelljen foglalkozni a problémával, például azért, mert az instabilitás elérése sokkal tovább tart, mint a berendezés egy bekapcsolása alkalmával a folytonos működési idő. Nézzük, hogyan becsülhető az instabilitás eléréséhez szükséges idő. A mintavételt időzítő kvarcok által szolgáltatott mintavételi frekvencia eltérése a gyakorlatban normál kristályok esetén körülbelül h=10 ppm, tehát $h= 10^{-5}$, ahol h a hibát jelöli. Az eddigi jelölésekkel:

$$f_{gy} = \frac{1}{T_{gy}} \tag{5.8}$$

$$f_l = \frac{1}{T_l} \tag{5.9}$$

ahol f_{gy} és f_l a két mintavételi frekvenciát jelöli, melyeknek hibája megegyezik az órajel-generátorok hibájával, ugyanis a mintavételi frekvenciák és az órajel-generátorok frekvenciája között csupán egy konstans osztószám teremt kapcsolatot. Ekkor tehát felírható a két mintavételi frekvencia közti kapcsolat a hiba értékét felhasználva:

$$f_{gy}=(1+h)\cdot f_l \tag{5.10}$$

Behelyettesítve (5.8)-at és (5.9)-et, valamint (5.5)-öt felhasználva:

$$T_l=(1+h)\cdot T_{gy} \rightarrow \Delta T= T_l - T_{gy} = h\cdot T_{gy} \tag{5.11}$$

Tehát minden mintavétel alkalmával ekkora késleltetés jelenik meg a gyorsabb csatornában. Az instabilitás eléréséhez a késleltetés növekedésének el kell érnie az üzemelési frekvencián a 90º-ot. Legrosszabb esetben ez a frekvencia megegyezik a mintavételi tétel által megengedett legnagyobb mintavételezhető frekvenciával, azaz $f=f_{gy}/2$-vel, ugyanis adott késleltetés itt okozza a legnagyobb fázistolást. Nem fontos, hogy ez a frekvencia megegyezzen a zaj frekvenciájával, az is elegendő, ha annak valamely felharmonikusa, amelyet még el kívánunk nyomni.

Ha tehát a jel frekvenciája $f=f_{gy}/2$, az azt jelenti, hogy $f = \frac{1}{2 \cdot T_{gy}}$.

$\tau'(n)$ késleltetés f frekvencián:

$$\Delta\varphi=360^{o}\cdot f\cdot\tau'(n) \tag{5.12}$$

értékű fázistolást okoz. Stabilitás határhelyzetében $\Delta\varphi=90^{o}$, tehát (5.12)-be (5.6)-ot, aztán (5.11)-et behelyettesítve, az egyenletet $\tau'(n)$-re megoldva:

$$\tau'(n)=\frac{\Delta\varphi}{360^{\circ}\cdot f}=\frac{90^{\circ}}{360^{\circ}\cdot f}=\frac{1}{4\cdot f}=\frac{1}{4\cdot\frac{1}{2\cdot T_{gy}}}=\frac{T_{gy}}{2} \tag{5.13}$$

melyből (5.6) alapján n meghatározható:

$n=\frac{1}{2\cdot h}=\frac{1}{2\cdot 10^{-5}}=50.000$.

Ezek alapján 50.000 minta után érjük el a stabilitási határhelyzetet a feltételezett mértékű, de reálisnak mondható h hiba esetén. Amennyiben az instabilitás bekövetkeztének idejére vagyunk kíváncsiak, akkor ezt az értéket meg kell szoroznunk a mintavételi időközzel, mely pedig T_{gy} . Ez például 2 kHz-es mintavételi frekvencián 25 másodpercet eredményez, ami tehát mindenképpen szükségessé teszi a szinkronizációt, hiszen ez idő alatt még az identifikáció elvégzése sem lehetséges, tehát már az átviteli függvény mérésekor is hibás eredményt kapunk.

5.2. Szinkronizációs algoritmus a szenzorhálózatban

Az adatgyűjtő hálózatban felhasznált szinkronizációs algoritmus kidolgozása során arra törekedtünk, hogy a hálózat által támasztott követelmények kielégítése mellett az algoritmus implementálása minél egyszerűbb legyen, és reális számítási követelményeket támasszon egy közepes teljesítményű mikrokontrollerrel szemben.

A mintagyűjtő hálózat szinkronizációjának elvi lehetőségeivel a 4.7. alfejezet foglalkozik, ezen alternatívák alapján történt a feladat elemzése.

Fő irányelv az volt, hogy olyan adatgyűjtő hálózatot alakítsunk ki, melyben azonos sebességgel érkeznek a minták az egyes node-októl, hiszen ez igen egyszerű hálózati és adatfeldolgozási struktúrát eredményez, és ezen esetben jól kihasználható a hálózat adatátviteli sebessége.

Kérdéses lehet ebben az esetben, hogy elképzelhető-e, és lehet-e létjogosultsága egy olyan rendszer kialakításának, ahol a minták gyűjtése a feldolgozó egység parancsára történik. Mivel a megcélzott mintavételi frekvencia körülbelül 1-2 kHz, így a mintavételi időközök hossza maximum 1 ms. Ez esetleg elég lehet egy mintavételezési parancs elküldésére, de már ebben az esetben is teljesen leterhelné a hálózatot a parancsok kiadása, így az összegyűjtött adatok továbbítására nem lenne lehetőség. Felmerülhet annak a lehetősége, hogy a szenzorokon aszinkron mintavételezés történik, és valamilyen módszerrel kijelöljük a valódi mintavételi időpontokat, majd interpolációval a mote-ok előállítják a mintavételi időpontokban a zajminták pillanatértékét az összegyűjtött minták alapján. Ez a lehetőség azért nem optimális,

mivel a mote-oknak ebben az esetben is mérniük kell az egymástól való elcsúszás mértékét, mint a későbbiekben bemutatásra kerülő algoritmusban is, viszont már egy igen egyszerű lineáris interpolációs technika megvalósítása is jóval bonyolultabb programkódot eredményezne. Az a módszer, miszerint egyszerűen „eldobjuk” az eltérő mintavételi frekvenciák miatt keletkező extra mintákat, nem túl jó megoldás, mivel a mintavételi időpontok egymáshoz képest akár egy mintavételi időköznyi távolságra is elcsúsznak, mely, mint megmutattuk, az átviteli függvény jelentős megváltozását vonja maga után. Ez a rendszer instabilitását okozhatja, alacsonyabb frekvenciákon pedig, ahol a késleltetés miatt bekövetkező fázistolás nem haladja meg a 90°-t, a rendszer jelentős lassulásához vezet.

Ennek tudatában azt a megoldást választottuk, miszerint a mote-ok mintavételi időpontjai valóban szinkronizálva vannak, így biztosítjuk az egységes adatáramlást a jelfeldolgozó egységhez. Másik nyomós érv a valódi szinkronizáció megvalósítására az, hogy így egyszerűbben megvalósítható a hálózati folyamatok egységes kezelése, és egy átláthatóbb rendszer alakul ki.

Az előzőekben leírtak alapján tudjuk, hogy a hálózatban folyamatos adatgyűjtés folyik, mely folyamatos adattovábbítást kíván meg. Mivel az adatgyűjtés frekvenciája a tervek alapján 1-2 kHz-es tartományban van, így az adatok továbbítása nagy valószínűséggel megfelelő gyakorisággal történik ahhoz, hogy implicit szinkronizációs üzeneteket használhassunk.

A szinkronizációt egyszerűsíti, hogy nem kell abszolút szinkronizációt megvalósítani, tehát nem szükséges egy egységes referenciaidőt ismerni, elegendő a mote-ok együttfutásának biztosítása. Sok szabályzási rendszerben követelmény, hogy időben összetartozó minták alapján történjen a beavatkozó jel számítása. Esetünkben viszont nem követelmény, hogy a mote-okon a mintavételezés egyszerre történjen – persze nem is probléma –, elegendő, ha mindig fix időkülönbséggel követik egymást a mintavételi időpontok. Ez azért lehetséges, mert a zajcsökkentő rendszerekben az egyes csatornákat külön-külön identifikáljuk.

Az algoritmus arra alapul, hogy a hálózatban lezajló folyamatok (mintavételezés, adattovábbítás, számítások …) periodikusak, és folyamatosan végrehajtódnak. A hálózatban ki kell jelölni egy referenciaként szolgáló mote-ot, melyhez a többi mote szinkronizálódni fog. Ennek kijelölése tetszőleges, gyakorlati megfontolások alapján történhet. A többi mote mindegyike ezen referenciaállomás csomagjainak érkezését használja szinkronizációs célra. Tekintsük az 5.2. ábrát. Amennyiben a referencia mote az adatokat tartalmazó csomagot mindig egy $h_{\mathrm{ref}}(t_{samp\_r})$ tetszőleges mintavételi időpont után fix idővel, $h_{\mathrm{ref}}(t_{send})$ időpontban küldi el, és ez a többi mote-on egy $h_{\mathrm{i}}(t_{samp\_i})$ mintavételi időpont után egy tetszőlegesen meghatározott időt követően, $h_{\mathrm{i}}(t_{rec})$ időpontban érkezik meg, akkor a két mote együttfutása biztosított abban az esetben, ha a csomag küldése és megérkezése közt eltelt idő állandónak tekinthető. Ekkor a mintavételi időpontok között mindig fix az eltérés. A szöveg és az ábra alapján a következő egyenletek írhatók fel:

$$h_{\mathrm{ref}}(t_{send}) - h_{\mathrm{ref}}(t_{samp\_r}) = T_{loc1} \tag{5.14}$$

$$h_i(t_{rec}) - h_i(t_{samp\_i}) = T_{loc2} \quad (5.15)$$

$$h_i(t_{rec}) - h_{ref}(t_{send}) = T_{Send} \quad (5.16)$$

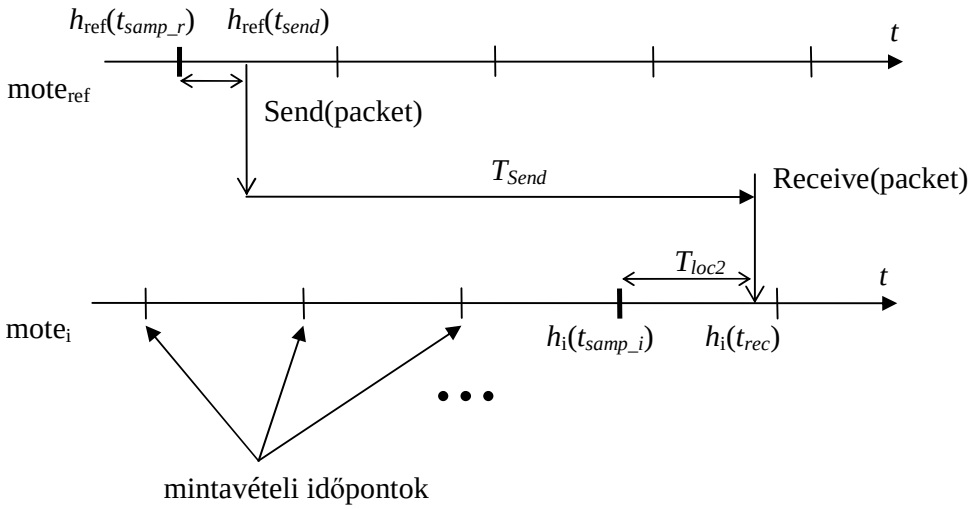
melyek alapján a mintavételi időpontokra felírható összefüggés:

$$h_i(t_{samp\_i}) - h_{ref}(t_{samp\_r}) = T_{ofs} = T_{Send} + T_{loc1} - T_{loc2} \quad (5.17)$$

Az egyenlet mindkét oldalát deriválva, mivel az egyenlet jobb oldala konstans, és feltételezzük, hogy a mote-ok szinkronizált állapotban vannak:

$$f_i(t) = f_{ref}(t) \quad (5.18)$$

(5.18) azt fejezi ki, hogy a referencia- és a többi mote órái ugyanolyan gyorsan járnak, tehát az együttfutás megvalósul.

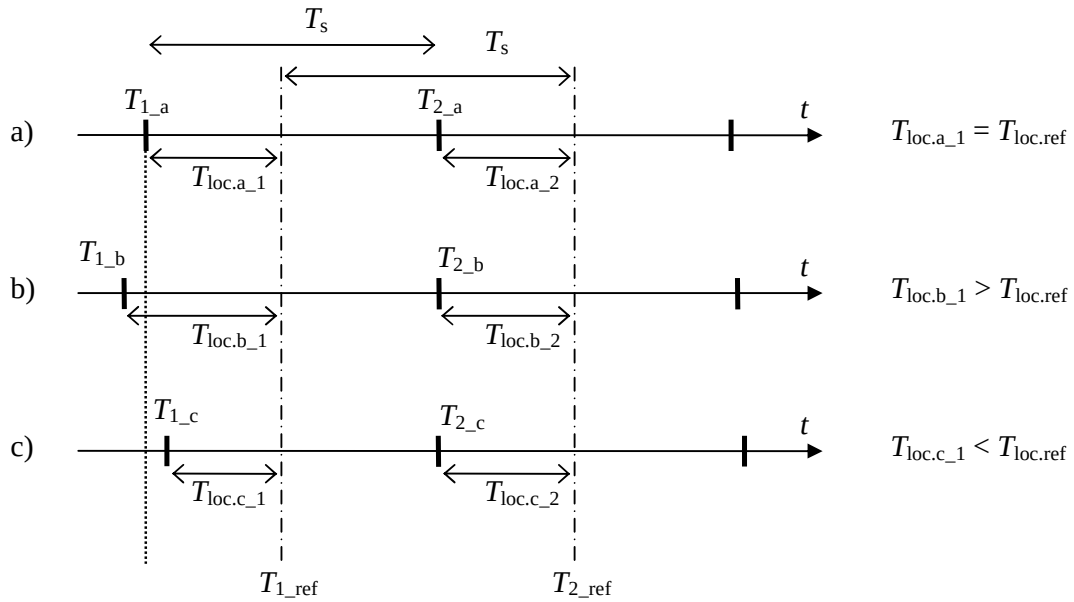


5.2. ábra. Szinkronizáció helyességének igazolása

Az a feltételezés, miszerint T_{Send} idő állandó, azzal magyarázható, hogy a mote-okon futó programok periodikus működésűek, a hálózatban a csomagok küldése pedig determinisztikus, így a 4.3. alfejezetben felsorolt időzítési adatok jó közelítéssel állandónak tekinthetőek. Ugyan a referencia csomag nem tartalmaz időinformációt, az implicit mégis benne foglaltatik azáltal, hogy a csomag elküldése mindig meghatározott időpontban történik.

A szinkronizáció, tehát annak biztosítása, hogy a referencia mote csomagja mindig egy mintavételi időpont után adott idővel érkezik, az 5.3. ábrán vázolt módszerrel biztosítható. Az ábrán az időtengelyen a függőleges vonalak a mintavételi időpontokat jelölik. A pontvonallal jelölt időpontok a referencia mintavételi időpontokat jelölik, amelyhez a mintavételi időpontokat igazítani kell. Ezek a referencia csomag érkezési ideje alapján jelölhetőek ki. Amennyiben a referencia mote csomagja a megfelelő időpontban érkezik, akkor változatlan mintavételi frekvenciával tovább

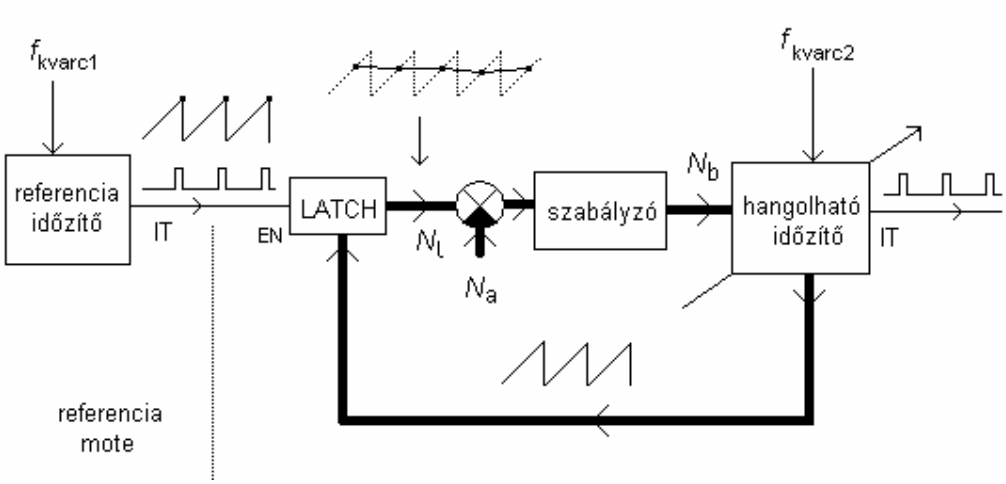
működve megmarad a szinkronizált állapot. Ezt mutatja az a) eset. A b) esetben, amikor a referencia mote csomagja később érkezik az előírtnál, akkor a T_s mintavételi időközt megfelelő mértékben növelve – tehát a mintavételi frekvenciát csökkentve – a következő referencia csomag érkezésére $T_{loc.b\_2} = T_{loc.ref}$ biztosítható. Amennyiben viszont korábban érkezik a csomag az előírtnál, a c) eset alapján a mintavételi frekvencia növelésével teljesíthetjük $T_{loc.c\_2} = T_{loc.ref}$ feltételt.



$T_{loc.ref}$: a $T_{loc.x\_y}$, tehát a mintavétel és a referencia csomag érkezése közt eltelt idő előírt követendő értéke

5.3. ábra. Szinkronizáció megvalósítása

A fent leírt algoritmus igen jól szemléltethető egy PLL analógiával. A PLL-ben mint tudjuk cél az, hogy a fázisdetektor bemenetén lévő jelek fázismerev kapcsolatba kerüljenek. Esetünkben ezen két jel az egyes szenzorokon elhelyezkedő, a mintavételezési időpontokat meghatározó időzítőkben lévő számlálók értéke, melyek időfüggvénye egy fűrészjellel írható le. A fűrészjelek lefutó élei a mintavételezési időpontoknak felelnek meg. Ha tehát sikerül az időzítők számlálója által előállított fűrészjeleket fázismerev kapcsolatba hozni, akkor a szinkron mintavételezés megvalósul. Egy ilyen, PLL elven működő szinkronizációs mechanizmus blokkdiagramját mutatja az 5.4. ábra, mely egyébként egyfajta segítséget is nyújt az algoritmus későbbi implementálásához, hiszen működése teljes mértékben megfelel az 5.3. ábrán vázolt szinkronizációs algoritmusnak. Előnye, hogy szemléletesebb képet nyújt a feladat megoldásáról.



5.4. ábra. PLL analógia

A struktúra működése a következő: a referencia időzítő előállítja a saját mintavételi időpontjait. Az időzítőt általában egy hangolható órajelosztó valósítja meg. Ezekben egy számláló működik, mely az eredeti órajel ütemében – esetünkben f_{kvarc1} és f_{kvarc2}, melyek névleges értéke megegyezik – számlál fölfelé, és egy aktuálisan beállított N_{div} érték elérésekor a számláló nullázódik, és megszakítást generál. A számlálók értékét az idő függvényében ábrázolva egy fűrészjelet kapunk, melynek frekvenciája N_{div}-vel vezérelhető. A szinkronizáció tehát úgy is értelmezhető, hogy ezen fűrészjeleket kell fázismerev kapcsolatba hozni, hiszen ez azt eredményezi, hogy a mintavételi időpontok egymáshoz képest állandó távolságra kerülnek. Ez alapozza meg a PLL analógia használatát is. A PLL egyes részeinek megvalósításával a struktúra biztosítja a szinkronizációt. A VCO funkciót a már bemutatott hangolható időzítő reprezentálja. A VCO frekvenciáját meghatározó N_b beavatkozójel megegyezik az időzítő N_{div} értékével, így hangolható a frekvencia. A fázisdetektort úgy valósítjuk meg, hogy a referencia mote jelzéseikor, tehát a referencia csomagok megérkezésekor, a hangolható időzítőben található számláló értékét mintavételezzük. A mintavételezett érték arányos a fűrészjel fázisával. Hurokszűrőként az 5.3. ábrán bemutatott szinkronizálási elvet megvalósító szabályzót alkalmazzuk. Ennek egy konkrét megvalósítási lehetőségére a későbbiekben térünk vissza.

Az ilyen, PLL elvet követő, szinkronizációban felhasznált algoritmusok igen elterjedtek. Jó példa erre a számítógép hálózatokban elterjedt Network Time Protocol (NTP) is [16].

Az algoritmus megvalósítási kérdéseivel és teszteredményeivel későbbi fejezetekben foglalkozunk.

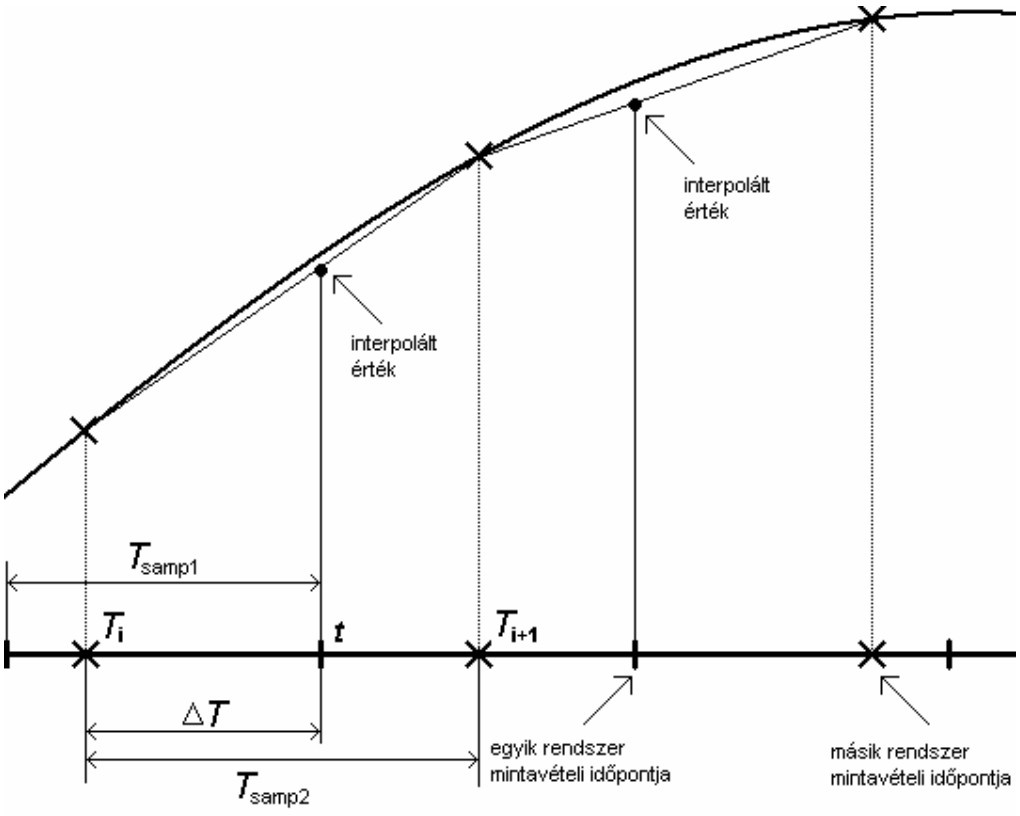
5.3. Szinkronizáció interpolációval

Az 5.2. alfejezetben egy adatgyűjtő hálózatokban felhasználható szinkronizálási algoritmust mutattunk be, amely a mintavételi frekvenciák PLL-szerű összehúzásán alapszik. Előfordulnak a gyakorlatban olyan helyzetek is, amikor a valódi szinkronizáció nem valósítható meg, mert például a mintavételi frekvencia nem módosítható. Ezekben a helyzetekben alkalmazható a bemutatásra kerülő, a mintavételezett jelek interpolációján és újra-mintavételezésén alapuló algoritmus [17]. Esetünkben ilyen algoritmust alkalmazunk a szenzorhálózat és DSP illesztésére.

Az interpolációra többféle megoldás is lehetséges. Jó kompromisszumot jelent a számításigény és az interpoláció minőségi jellemzői között a lineáris interpoláció alkalmazása. Ennek során egy [T_i, T_{i+1}] intervallumban egy $f(t)$ függvény értékét a következő képlet szerint becsüljük:

$$f(t) = f(T_i) + \left[f(T_{i+1}) - f(T_i) \right] \cdot \frac{t - T_i}{T_{i+1} - T_i} \tag{5.19}$$

(5.19) az 5.5. ábra alapján írható fel, mely két eltérő mintavételi frekvencián működő rendszer interpolációval történő szinkronizálását mutatja.



5.5. ábra. Lineáris interpoláció

Az ábra jelöléseinek magyarázata:

- T_{samp1}, T_{samp2} : egyik és másik rendszer mintavételi időközei
- t : a t időpontban szeretnénk a jel értékét becsülni
- ΔT: az utolsó beérkezett minta és t között eltelt idő

Az interpoláció implementálása nem ebben a formában történik, hiszen t időpontban az $f(T_{i+1})$ még nem ismert, így az interpolációba egy mintányi késleltetést viszünk be, ezzel biztosítva a kauzalitást.

Annak ellenére, hogy a lineáris interpoláció sok esetben igen torz jelet eredményez, a zajcsökkentő rendszer esetén alkalmazása mégis jó megoldás. Ebben az esetben a célunk ugyanis a nullszint érzékelése, mely lineáris interpoláció esetén is kifogástalanul elvégezhető.

6. Felhasznált hardver és szoftver eszközök

A zajcsökkentő rendszerben két fő hardveres egység található:

- Jelfeldolgozó processzor (DSP), mely a zajcsökkentő algoritmusok futtatását végzi.
- Berkeley-mote-ok, melyek a zajérzékelést végzik.

6.1 ADSP-21061EZ-KIT Lite fejlesztői kártya

Az ADSP-21061EZ-KIT Lite fejlesztői kártya [18] az ADSP-21061-es típusú jelfeldolgozó processzor köré épített, a processzorhoz tartozó perifériákat is magába foglaló áramkör. A fejlesztői kártya segítségével tesztelhetőek a különböző jelfeldolgozói algoritmusok új hardver megtervezése nélkül. Ez kényelmes és gyors lehetőséget biztosít a programok fejlesztésére.
A fejlesztői kártya a DSP-n kívül következő fő perifériákat tartalmazza:

- **EEPROM**: a DSP boot programját tartalmazza. Ebben található egy kernel program is, mely kapcsolatban áll a fejlesztői környezettel.
- **UART illesztő IC**: a PC16550D típusú IC segítségével lehetőség nyílik soros porton történő kommunikációra. Amennyiben a programfejlesztés, tehát a programok letöltése, és a debuggolás soros porton keresztül történik, akkor ezen IC segítségével csatlakoztathatjuk a kártyát a PC soros portjára.
- **JTAG csatlakozó**: lehetőséget kínál a JTAG-en keresztül történő programfejlesztésre.
- **AD1847-es sztereo kodek:** ennek segítségével történik az analóg-digitális és digitális-analóg átalakítás. Ez azért szükséges, mivel sok alkalmazásban szükséges analóg jelek feldolgozása, így ezen keresztül kapcsolódik a DSP az analóg világhoz. A DSP a nullás soros portján keresztül csatlakozik a kodekhez.
- **csatlakozók**: a DSP-nek a fejlesztői kártya által nem használt lábaihoz férhetünk hozzá a csatlakozókon keresztül. Így illeszthető a kártyához a felhasználó által tervezett periféria.
- **gombok**: a kártyán három gomb található. Ezek segítségével resetelhető a kártya, lehetőség nyílik megszakítás generálására, illetve harmadik digitális bemenetként használható.
- **LED-ek**: három, a DSP digitális kimeneteire csatlakoztatott LED segítségével lehetőség nyílik a DSP-n futó szoftver állapotáról képet kapni. A programfejlesztés során ezek bármilyen célra felhasználhatók.

Az ADSP-21061EZ-KIT Lite fejlesztői kártya kezelése, és a programfejlesztés az Analog Devices által kínált VisualDSP++ fejlesztői környezet segítségével történik, mely támogatja mind a soros porton, mind a JTAG segítségével történő

programfejlesztést is. Soros port segítségével történő programfejlesztés esetén a fejlesztői környezet a DSP-n futó kernel programmal kommunikál, a kommunikáció révén válik lehetővé a programok letöltése, futtatása és debugolása. A JTAG segítségével történő fejlesztés során szabványos interfészen keresztül, egy programozó áramkör felhasználásával férünk hozzá a processzor különböző egységeihez. Mindkét fejlesztési mód esetén elvégezhető a program letöltése, elindítása, megállítása, különböző regiszterek és perifériák állapotának lekérdezése. Mivel a programokat a DSP SRAM memóriájába töltjük be, melynek tartalma a tápfeszültség megszűnte után elvész, ezért a programokat minden bekapcsoláskor újra be kell tölteni, hiszen sem a DSP, sem a fejlesztői kártya nem rendelkezik program flash-sel.

A fejlesztői környezet segítségével lehetőség nyílik C, illetve assembly nyelven történő programfejlesztésre.

6.2. ADSP-21061 (SHARC) jelfeldolgozó processzor

Az ADSP-21061-es processzor [19] az Analog Devices (ADI) által kínált ADSP-21000-s család tagja. A családon belül a felhasznált processzormag megegyezik, csupán a maghoz kapcsolt perifériákban van különbség. Az eszköz CMOS technológiával készült, maximum 50 MHz-es órajel-frekvenciával képes működni. A SHARC elnevezés a Super Harvard ARchitecture Computer rövidítéséből származik, mely alapján a processzor felépítése nem a személyi számítógépekben megszokott Neumann-architektúrát követi. A DSP a Harvard-architektúrának megfelelően külön adat- és programmemóriát használ, mely lehetővé teszi a két memória párhuzamos elérését. A super Harvard-architektúra azzal a kiegészítéssel jár, hogy egy tokon belül két adatbusz, egy utasítás- és egy IO busz található. Az utasítás-cache és a négy független busz segítségével lehetővé válik, hogy a processzor egy órajel ütem alatt egy utasítást hajtson végre.

A párhuzamosan működtethető egységek segítségével lehetőség nyílik egy utasításon belül két memória és két aritmetikai művelet elvégzésére bizonyos kötöttségek mellett. Ez jól igazodik az általános jelfeldolgozó algoritmusokhoz. Ennek köszönhetően például egy FIR szűrő algoritmus lefuttatásához szükséges órajelek száma körülbelül megegyezik a szűrő együtthatóinak számával, ugyanis egy órajel alatt lehetséges két adat lehívása a memóriából, valamint egy szorzás és egy összeadás elvégzése (MAC művelet). A két adat egyike a program-, míg a másik az adat-memóriában található (a programmemóriában is tárolható adat).

A processzormaghoz tartozó műveletvégző egység lehetővé teszi 32 bites IEEE single-precision floating-point szabványos formátumú, illetve 40 bites kiterjesztett (extended precision) formátumú lebegőpontos számok kezelését. Lehetőség van ezen kívül 32 bites fixpontos formátum használatára is. A műveletvégző egységben található ALU (Arithmetic and Logic Unit), szorzó egység és shifter párhuzamos működésre is képesek, így lehetővé válik egy órajel alatt akár két számítási művelet elvégzése is.

A processzormag 16 általános regisztert tartalmaz. A műveletvégző egység ezeket használja a műveletek forrás- és célregisztereiként. Párhuzamos műveletvégzés esetén bizonyos műveletekhez csak adott regiszterek használhatók.

Az adatok és a programkód tárolására a processzorhoz a tokon belül csatlakoztatott SRAM szolgál, mely az ADSP-21061-es esetén 1 Mbit méretű. Az SRAM bizonyos kötöttségekkel feloszthető adat- és programmemória blokkokra, melyekhez párhuzamos hozzáférés lehetséges. Amennyiben 32 bites adatokat tárolunk, akkor ez 16 kiloszó méretű adat- és programmemóriát jelent. A memória felbontása a műveletvégzések párhuzamosítását teszi lehetővé, mivel a programmemóriában is lehetőség van adat tárolására. Amennyiben ez a memória kevés, lehetőség van külső memória csatlakoztatására is, a külső memória és periféria interfész segítségével. A külső memória területre memóriába ágyazott periféria is illeszthető.

A memóriacímző egységek hatékony memóriakezelést tesznek lehetővé. A segítségükkel létrehozhatóak a memóriában cirkuláris bufferek, lehetséges a mutatók automatikus növelése/csökkentése a memóriaműveletet követően, illetve megelőzően, és támogatja az FFT algoritmusban fontos szerepet játszó bitreverse címzési módot.

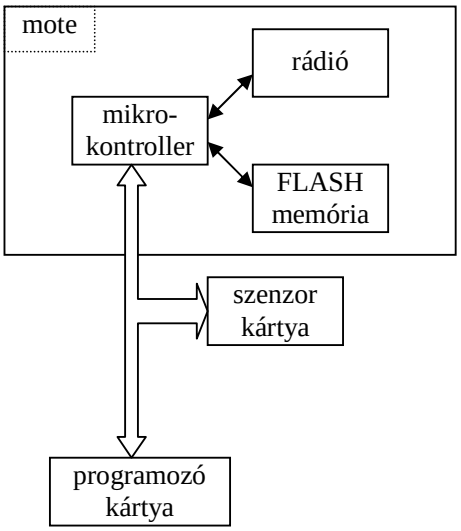
A DSP két szinkron soros porttal rendelkezik. A potrok akár a DSP órajelének megfelelő adatátviteli sebességgel is képesek működni, mellyel igen gyors adatátvitel valósítható meg az egységek között.

Összességében megállapítható, hogy a processzor igen hatékonyan, hardveresen támogatja az általános jelfeldolgozási algoritmusokhoz szükséges alapvető műveleteket. Ez jelentősen megkönnyíti az algoritmusok implementálását.

6.3. A Berkeley mote-ok

6.3.1. Hardveres felépítés

A zajcsökkentő rendszerben zajérzékelésre Berkeley mote-okat [20] használtuk, melyekhez tartozik mikrofont is tartalmazó szenzorkártya; ennek típusa MTS310. A Berkeley-mote-ok tulajdonképpen egy családot jelölnek, melyeknek különböző tagjai vannak, mi a MICAz típust használjuk. Blokkvázlat szintjén a mote a 6.1. ábrán látható módon épül fel.



6.1. ábra. A MICAz mote-ok felépítése

A következőkben a különböző részegységeket mutatjuk be.

- **Flash memória:** A mote-on található egy AT45DB041B típusú memória, mely alkalmas a mérések során keletkezett adatok tárolására. Ennek köszönhetően 512 kByte-nyi adatot lehet hosszú távon is tárolni, ha nincs lehetőség az adatok azonnali továbbítására.

- **Rádiós IC:** A MICAz mote tartalmaz egy CC2420 típusú integrált adó-vevő áramkört, melynek segítségével lehetőség nyílik a mote-on futó alkalmazások számára az adatok vezeték nélküli továbbítására. A ZigBee szabványnak megfelelő rádió a 2.4 GHz-es ISM sávban működik, és adatátviteli képessége 250 kbps, melyet alkalmazásainkban természetesen nem lehet teljes mértékben kihasználni a kommunikációs overhead miatt. Ez a rádió az IEEE 802.15.4 szabvány szerint [21] definiált fizikai és MAC réteget tartalmazó integrált áramkör. Ezt a szabványt célirányosan a kisméretű helyi hálózatokkal szemben támasztott követelményeket szem előtt tartva alakították ki. Ilyen szempont például a nagy csomópontszámú hálózatok támogatása, amit a nagy címtartománnyal tesznek lehetővé. Fontos szempont a kis fogyasztású eszközök kialakításának támogatása, melyet a CC2420 például célirányos hardveres moduljaival is segít, mivel így nem szükséges a mikrokontroller erőforrásait bizonyos feladatokra feleslegesen használni.

- **Mikrokontroller:** A MICAz mote egy ATmega128 típusú mikrokontrollert tartalmaz. Ez egy általános célú 8 bites RISC architektúrájú mikrovezérlő. A következőkben felsorolásszinten bemutatjuk, hogy a mikrokontroller milyen fontosabb perifériakészlettel rendelkezik, és milyen szokásos paraméterekkel írható le:
  - 128 kB program flash
  - 4 kB SRAM
  - 4 kB EEPROM

- szinkron és aszinkron soros interfész
- 4 db időzítő/számláló, melyeket PWM módban is használhatunk
- analóg komparátor
- 8 csatornás multiplexelt 10 bites AD-átalakító
- többféle energiatakarékos üzemmód
- legfeljebb 16 MHz-es órajel-frekvencia, az általunk használt mote órajel-frekvenciája 7.3728 MHz.

Már ezen adatokból is látható, hogy bonyolultabb jelfeldolgozási algoritmusok futtatására nem képes a mikrokontroller, a mote-on található perifériák kezelésére és alapvető adatfeldolgozásra viszont kitűnően használható.

- **Szenzorkártya:** A mote-okhoz, a rajtuk elhelyezett csatlakozón keresztül különféle szenzorkártyák csatlakoztathatók. Számunkra a legmegfelelőbb az MTS310-es típusú kártya, mivel ez tartalmazza a feladathoz szükséges mikrofont, valamint az ahhoz tartozó analóg jelkondicionáló áramköröket. A kártya lehetőséget kínál a mikrofonhoz tartozó erősítő erősítési tényezőjének bizonyos szintű állítására, mely egy a kártyán elhelyezkedő I^2C porton programozható ellenállás segítségével valósul meg. Ez a programozható értékű ellenállás egy műveleti erősítő visszacsatoló ágában foglal helyet, így teszi lehetővé az erősítés beállítását.

- **Programozói kártya:** Az általunk használt MIB510 típusú programozói kártya lehetővé teszi a mote-ok programozását a PC soros portja segítségével. A PC-n a fordító előállítja a forráskódból a mote-ra letölthető gépi kódú programot, mely az erre szolgáló szoftver segítségével, soros porton, a mote-ot a programozó kártyára helyezve letölthető a mote-ok programmemóriájába. A letöltött program a következő újraprogramozásig bennmarad a programmemóriában. A mote-ok hálózati azonosítóját programozás során kell megadni. A kártya üzemeltethető a mote-ok telepeiről, illetve lehetőség van tápegység csatlakoztatására is. Ez az eszköz azonban nem csak a programozásban játszik szerepet, ugyanis a rajta található áramkörök segítségével a mote-ok soros vonalán megjelenő logikai jelszinteket RS232-es jelszintekké alakítja, így lehetővé teszi más RS232-es soros interfészt használó eszközhöz való egyszerű csatlakoztatást. Ennek segítségével tehát egyszerűen kialakíthatunk olyan bázisállomásokat, melyek képesek továbbítani a rádiós hálózaton keresztül küldött adatokat az azokat feldolgozó egységhez.

6.3.2. TinyOS és NesC nyelv

A Berkeley mote-ok programozása egy TinyOS nevű operációs rendszerben NesC nyelven történik, ezekről részletesebben [22]-ben olvashatunk. A NesC nyelvű kódból az előfordító egy C nyelvű kódot generál, melyből aztán a fordító a mikrokontrollerre tölthető kódot állít elő.

A TinyOS kifejezetten a vezeték nélküli szenzorhálózatokhoz kifejlesztett operációs rendszer, mely nyílt forráskódú, ingyenesen letölthető és felhasználható. Alapvető kialakítási szempontjai az eseményvezérelt működés és a komponens alapú felépítés. Az operációs rendszert NesC nyelven fejlesztették ki, és ez is a hozzá kapcsolódó programozási nyelv, mely jól illeszkedik az operációs rendszer alapkoncepcióihoz.

A NesC alapvetően C szintaxisú programozási nyelv. A NesC-ben megírt programok komponensekből épülnek fel, ez egyszerűbbé teszi a program tervezését és tesztelését, hiszen a hierarchikus szemlélet megkönnyíti a rendszer áttekintését, az egymástól nagyjából függetlenül fejleszthető és tesztelhető komponensek pedig csökkentik a hibalehetőségeket. A komponensek összekötése interfészeken keresztül lehetséges, melyek parancsokat (command) és eseményeket (event) tartalmaznak. A command-ok segítségével egy feladat végrehajtására utasítjuk az adott komponenst, amely az event-ek segítségével értesít egy esemény bekövetkeztéről.

A TinyOS alapvető ütemezési entitásai a következő típusúak lehetnek:

- **hardver megszakítás**: Ez a legmagasabb prioritású kódrészlet. Azokon a szakaszokon kívül, ahol a megszakítások tiltva vannak, bármilyen kódrészletet megszakíthat.
- **aszinkron command, aszinkron event**: olyan parancsok és események, melyeket megszakításrutinokban generálunk.
- **taszk**: taszkokat a program bármely részéből indíthatjuk. A taszkok elindításuk kezdeményezésekor egy FIFO tárba kerülnek, amelyben bekerülésük sorrendjében végrehajtódnak, amint lehetséges. Egy új taszk indítása csak az előző taszk befejezésekor kezdődhet. Taszk nem szakíthat meg semmilyen folyamatot.
- **szinkron command, szinkron event**: olyan parancsok és események, melyeket taszkokban generálunk.

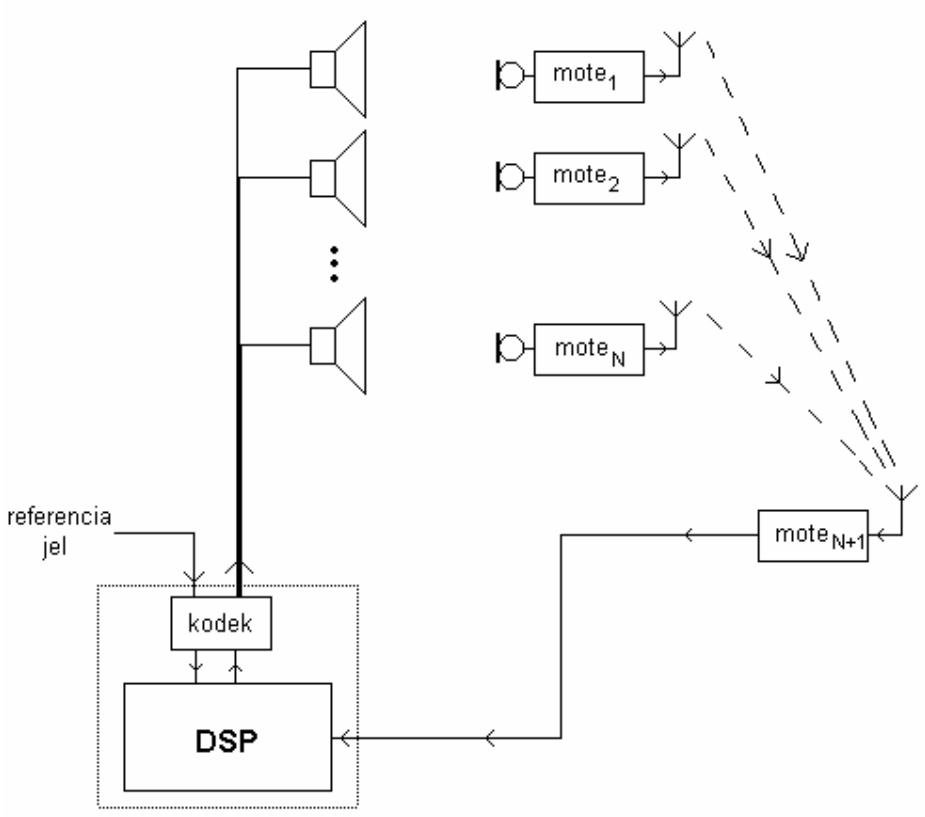
A kölcsönös kizárás megoldására a TinyOS két lehetőséget kínál. Az első lehetőség, hogy a közös erőforrás kezelése csak taszkok használatával valósul meg. Mivel a taszkok nem szakíthatják meg egymást, a kritikus szakaszban csak egy taszk férhet hozzá az erőforráshoz. Második lehetőség az, hogy a kritikus szakaszokat atomikusan hajtsuk végre, mely a TinyOS atomic kulcsszava segítségével lehetséges. Az atomikus műveletek során a mikrokontroller megszakításai tiltva vannak, ez biztosítja a kölcsönös kizárást.

A TinyOS kész komponenseket kínál a mote-ok perifériáinak kezelésére. Ilyenek például a rádió, a mikrofon és az időzítő kezelését segítő komponensek. Ezek az általános felhasználói célnak megfelelnek, alaposan tesztelt egységek, melyek sokrétű szolgáltatást nyújtanak. A komponensek forráskódja nyilvános. Ez azért fontos, mert előfordulhatnak olyan alkalmazások, melyekben nem elégítenek ki minden követelményt a kész egységek, ilyenkor ezek módosításával tehetjük alkalmassá az adott komponenst a feladat elvégzésére.

7. Adatgyűjtő hálózattal megvalósított zajcsökkentő rendszer bemutatása

7.1. Rendszer felépítése

A szenzorhálózattal felépített zajcsökkentő rendszer a 7.1. ábrán látható terv alapján épül fel.



7.1. ábra. Rendszerterv

A rendszerben a zajérzékelést a szenzorhálózat segítségével végezzük. Az 1…N mote-ig a rajtuk található mikrofonok jeleit 8 bites felbontással mintavételezik a mikrokontrolleren található AD- átalakító segítségével. Az összegyűjtött mintákat a mote-ok a rádiós hálózaton keresztül továbbítják az N+1. mote felé, amely a szenzorhálózat és a DSP között szolgál átjáróként. Az átjáró mote, vagy más néven bázisállomás a hálózatban összegyűjtött mintákat soros port segítségével továbbítja a DSP felé, mely a zajelnyomó algoritmus segítségével a mintákat felhasználva kiszámítja a beavatkozó jelet, amelyet a hozzá csatlakoztatott kodek segítségével kiad a

hangszórók felé. A zajcsökkentő algoritmusokhoz szükséges referenciajel bevitele történhet az ábrán látható módon a kodek segítségével, de akár a szenzorhálózat valamely elemének felhasználásával is.

A rendszer kialakításakor azzal a feltételezéssel éltünk, hogy a mote-ok mindegyike közvetlen kapcsolatban áll a bázisállomással, egy valamilyen algoritmus alapján kijelölt referencia mote-tal, illetve még legalább egy másik mote-tal úgy, hogy az egymást halló mote-ok egy zárt körbe szervezhetőek. Ezek a feltételezések azért tekinthetőek reálisnak, mert az aktív zajcsökkentő berendezések sokszor csupán néhány, vagy néhányszor tíz négyzetméteres területen helyezkednek el, így valósítható meg a közvetlen rádiós kapcsolat.

A tervezés során olyan rendszer kialakítását tűztük ki célul, melyben körülbelül 2 kHz-es mintavételi frekvencia érhető el a mote-okon, és képes periodikus jelek első néhány harmonikusára nézve – mely általában a domináns harmonikus komponens – mintegy 20-30 dB-es elnyomásra.

7.2. A szenzorhálózat működése

A hálózat kiépítésekor a cél az volt, hogy minél hatékonyabban használjuk ki a nem túl nagy sávszélességű rádiós csatorna (250 kbps) kapacitását. Ennek érdekében minimálisra kellett csökkenteni az adott idő alatt elküldött üzenetek számát.

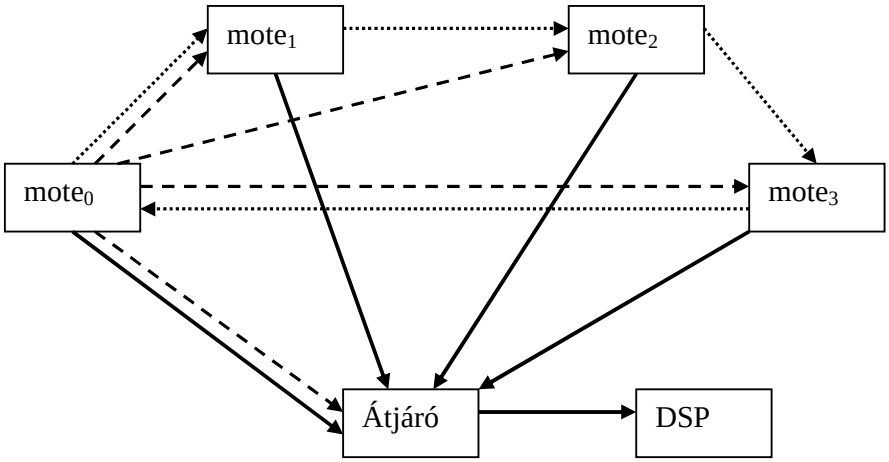
Másik fő irányelv az volt, hogy determinisztikusan működő hálózatot alakítsunk ki. Erre az 5.2. alfejezetben, a szinkronizációs algoritmus bemutatásánál tett kikötések miatt van szükség, tehát azért, hogy a közeg-hozzáférési idő állandó legyen, így az üzenetek küldésekor fellépő legnagyobb bizonytalansági tényezőt kiküszöbölhetjük.

A fenti megfontolásokat figyelembe véve egy token ring hálózatot alakítottunk ki a mote-okból. Ennek vázlata a 7.2. ábrán látható.
A nyilak a különböző logikai kapcsolatot jelölik. A nyíltípusok jelentése a következő:

- Folytonos vonal: adattovábbítás
- Pontozott vonal: adatküldés engedélyezése; a gyűrűs hálózat kialakításához kell
- Szaggatott vonal: szinkronizáció

A hálózatban az összegyűjtött adatokat a mote-ok egymás után megadott sorrendben továbbítják az átjáró mote felé, mely a következő módon zajlik le ideális esetben: tegyük fel, hogy a token, tehát az adás joga a 0-s azonosítójú $mote_0$-nál van. A $mote_0$ ekkor zajmintákat tartalmazó csomagot küld az átjáró felé. Ez egy broadcast üzenet, melyet a többi mote is hall. Ezen csomag elküldésével ad engedélyt a $mote_1$-nek a saját csomagja elküldésére. A $mote_1$ a saját mintáit tartalmazó csomag elküldésével engedélyt ad $mote_2$-nek, és így tovább, míg az adás engedélye visszajut a $mote_0$-hoz, amely újabb csomag összeállítása után újrakezdi a kört. Az adatokat tartalmazó csomag elküldésével tehát minden mote átadja az adás jogát a szomszédjának. Egy x azonosítójú $mote_x$ szomszédjának azonosítója x+1, kivéve a $mote_N$, melynek szomszédja $mote_0$. A

hálózat mérete egyelőre még nem kezelhető dinamikusan, az algoritmus csupán adott méretű hálózat esetén működik, mely méretet a programozás során adjuk meg.



7.2. ábra. Hálózat felépítése

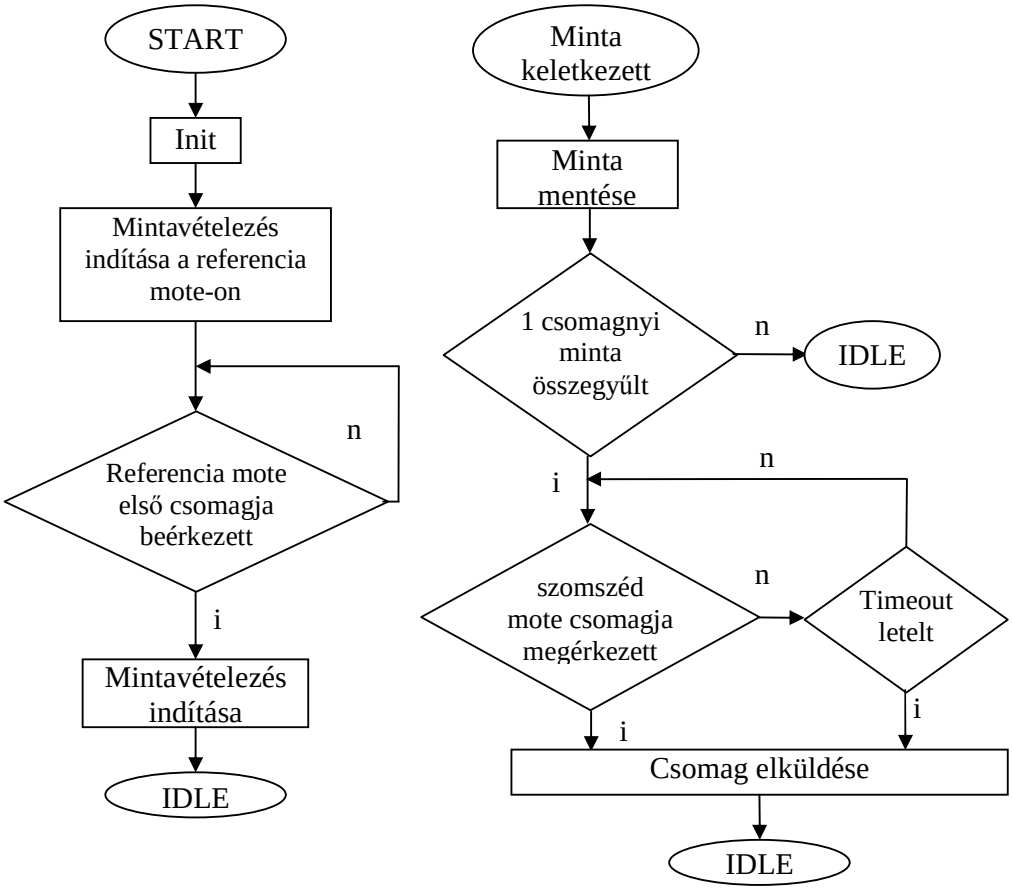
Az algoritmus alapján megállapítható, hogy a logikai kapcsolatokat egyazon üzenet valósítja meg. A hálózat ilyen kiépítésben még nem elég biztonságos, ugyanis ha egy csomag nem érkezik meg, akkor megszakad a kör, tehát elvész az adás jogát jelző token. Ennek megoldására egy timeout algoritmust is beépítettünk, melynek alapja, hogy az üzenetek megérkezésére megállapítunk egy maximális korlátot, így ezen időkorlát leteltével a soron következő mote automatikusan átveszi az adás jogát.

A hálózati folyamatok állandósult állapotban történő tervezése mellett fontos kérdés azok beindítása is. Esetünkben a referencia mote-on, annak bekapcsolása után azonnal beindul a mintavételezés, és mivel inicializáláskor hozzá kerül a token, így egy csomag összeállítása után kezdi az adást is. A hálózat többi mote-ján a mintagyűjtés a referencia mote első csomagjának hatására indul el, és így kezdődik a fentebb már vázolt normális működés.

Látható a 7.2. ábrán, hogy a csomagok megérkezéséről az átjáró mote semmilyen megerősítést nem küld vissza, ez ugyanis a hálózati forgalom növekedését okozná, és kiszámíthatatlanabbá tenné az időzítési értékeket, ha az üzeneteket ismételni kellene. Tapasztalatok szerint a mote-ok néhány méteres távolsága esetén néhány percenként tapasztalható csomagvesztés. Mivel a zajcsökkentő rendszer egy zárt szabályzási kör, a csomagvesztés miatt fellépő zavart a rendszer automatikusan korrigálja, nem okoz maradó hibát. Egy-egy csomag kiesése ráadásul olyan rövid idejű zavart okoz, hogy az a rendszer gyorsaságához mérve elhanyagolható, viszont a programfejlesztés során kiemelt figyelmet kell fordítani a csomagvesztés kezelésére.
Az eddig vázolt algoritmusokat megvalósító folyamatábrák a 7.3. ábrán láthatók.

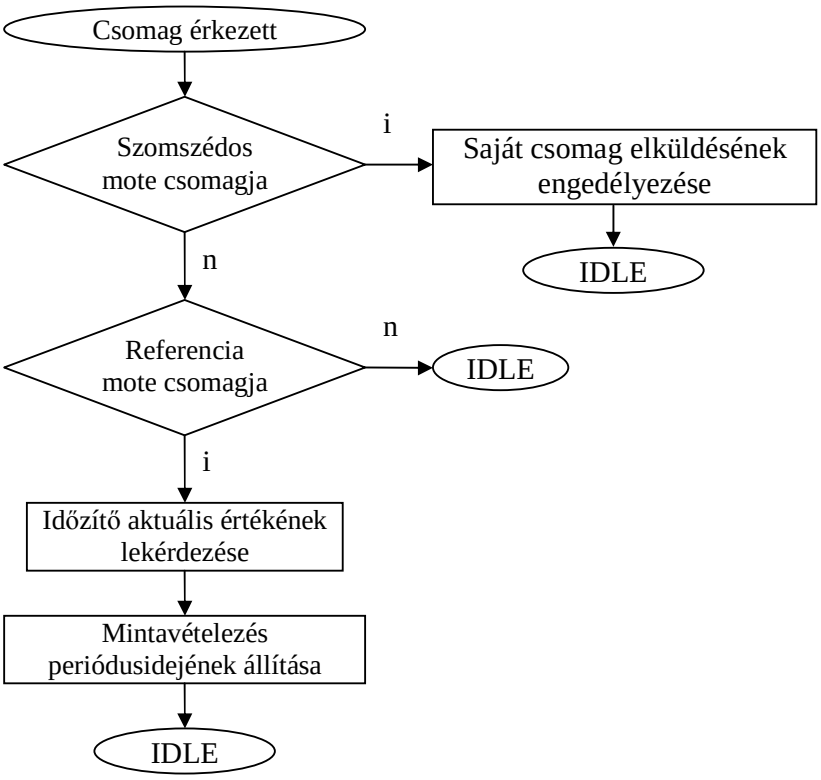
Nem esett még szó a hálózatban küldött csomagok felépítéséről. A maximális hatékonyság érdekében a lehető legtöbb adatot továbbítjuk egy csomagban, így legkisebb ugyanis egy minta továbbítására jutó overhead – mely a csomagok keretezése, és szoftveres késleltetés miatt lép fel – tehát jobb a hatékonyság. Ennek alapján egy

csomagban a TinyOS által maximálisan megengedett 29 byte-ot továbbítunk: 25 db 8 bites mintát, a mote saját azonosítóját, és a csomag sorszámát. Utóbbi kettő a hálózat kialakítása, az adatok DSP felé történő továbbítása, és a csomagvesztés észlelése miatt szükséges. A 25 mintát tartalmazó csomagok miatt a mote-ok átlagosan 25 mintavételi időközönként küldik el a mintákat tartalmazó csomagokat.



7.3. ábra. A program inicializációs szakasza és a mintagyűjtés folyamata

A 7.2. ábrán látható, hogy referencia mote-ként a $mote_0$ állomás szolgál, tehát ez a mote szolgál referencia állomásként a mintavételi időpontok szinkronizálásához. Azért jelöltük ki ezt a mote-ot, ugyanis a mote-ok programozásakor az azonosítójukat nullától kezdjük kiosztani, így a $mote_0$ bármekkora méretű hálózatban szerepel. Az adatot tartalmazó csomagot használjuk fel szinkronizációs célra is, mely az 5.2. alfejezetben leírt algoritmus alapján történik: a referencia mote csomagjának megérkezésekor minden mote kiolvassa a mikrofon kimenetének mintavételezését időzítő számláló értékét, és megvizsgálja, hogy a kiolvasott érték megfelel-e egy referencia értéknek – melyet a programban rögzítünk – és ez alapján módosítja a mintavételi frekvenciát. A mintavételi frekvencia a következő csomag érkezésekor újra meghatározásra kerül. Ezt a folyamatot szemlélteti a 7.4. ábra.



7.4. ábra. A hálózati működés időzítése és szinkronizálás

A mintavételi frekvencia meghatározását az 5.4. ábrán látható „szabályzó” blokk végzi az 5.3. ábrán látható alapelv szerint. A szabályzó algoritmus:

$$N_b = N_s + (N_l - N_a) / 32 \qquad (7.1)$$

Jelölések magyarázata:

- N_s: a névleges f_s mintavételi frekvencia, és a mote 7.3728 MHz-es órajel frekvenciájának aránya által meghatározott osztószám. Konkrét megvalósításban 1.8 kHz-es mintavételi frekvencián $N_s = 4096$.
- $N_a = 1700$, megválasztása a teszteredmények értékelésével lehetséges, lsd. később.

A 32-es osztószám meghatározása a következő gondolatmenet alapján történt: amennyiben N_s-t ΔN-nel megváltoztatjuk, akkor a következő csomag megérkezésekor N_l értéke kb. $25 \cdot \Delta N$-nel változik, mivel a referencia mote csomagjai 25 mintavételi időközönként érkeznek, és ekkora időköz alatt a drift hatása elhanyagolható. Ha ismerjük N_l -nek a következő csomag beérkezésekor megkívánt értékét (N_a), akkor $\Delta N = (N_l - N_a) / 25$ adódik. Az egyszerű megvalósítás miatt a 25-tel történő osztás helyett 32-vel osztunk, hiszen ez egy egyszerű 5 bittel történő aritmetikai shifteléssel egyenértékű. Ez csupán azt eredményezi, hogy kissé lassabb lesz a szinkronizáció beállása.

A szabályzó a (7.1) algoritmus végrehajtásán kívül ki van egészítve egy biztonsági megoldással is. Amennyiben $N_b \neq N_s$, tehát beavatkozás szükséges a mintavételi frekvenciába, akkor N_u db ütemen keresztül fenn kell állnia a feltételnek,

hogy a végrehajtsa N_s módosítását. Ez azért szükséges, mert az esetek többségében ugyan teljesül az, hogy a szinkronizációs csomagok továbbításának ideje állandó, azonban tapasztalatok szerint vannak olyan esetek, melyekben ez nem teljesül. Ennek oka lehet például, hogy a közeg-hozzáférési idő változik a normális működéshez képest, mert az eszköz a küldés pillanatában foglaltnak észleli a csatornát valamilyen rádiós zavar következtében (pl. rádiótelefon, WLAN …). Mivel ez igen ritkán fordul elő, N_u-t nem szükséges túl nagyra választani; esetünkben értéke: $N_u = 3$. Ezen holtidő alatt a két eszköz alig csúszik el egymáshoz képest, tehát a szinkronizáció minősége nem romlik.

Az eddigieket összefoglalva megállapíthatjuk, hogy a hálózatban található mote-ok a szinkron mintavételezett zajmintákat 25 mintát tartalmazó csomagokban továbbítják meghatározott sorrendben egymás után. A minták a csomagokban található azonosító révén rendelhetők a mote-okhoz.

7.3. A szenzorhálózat és a DSP illesztése

7.3.1. Fizikai illesztés

A szenzorhálózat és a DSP között az átjáró mote teremt kapcsolatot. Az átjáró a hálózat felől rádión keresztül kapja az adatokat, míg a DSP felé soros porton továbbítja. A soros porton történő kommunikációt azért választottuk, mivel sebessége megfelelő, és az illesztés megvalósítása vizsgálataink szerint így a legegyszerűbb. Habár a rádió adatátviteli sebessége 250 kbps, ennek kihasználása 25 adat továbbítása esetén a keretezés miatt alig 60%, tehát a hasznos sávszélesség kb. 150 kbps. A soros port akár 115.2 kbps-al is működhet, ráadásul az egyszerűbb kezelhetősége miatt elég jó kihasználtsággal, tehát nem jelent szűk keresztmetszetet a rádió mellett. A rádió kezelése ugyanis akár jelentős szoftveres overhead-del járhat, ami a hasznos sávszélességet csökkenti.

A DSP és a mote soros portja közvetlenül nem kapcsolható össze, két ok miatt:

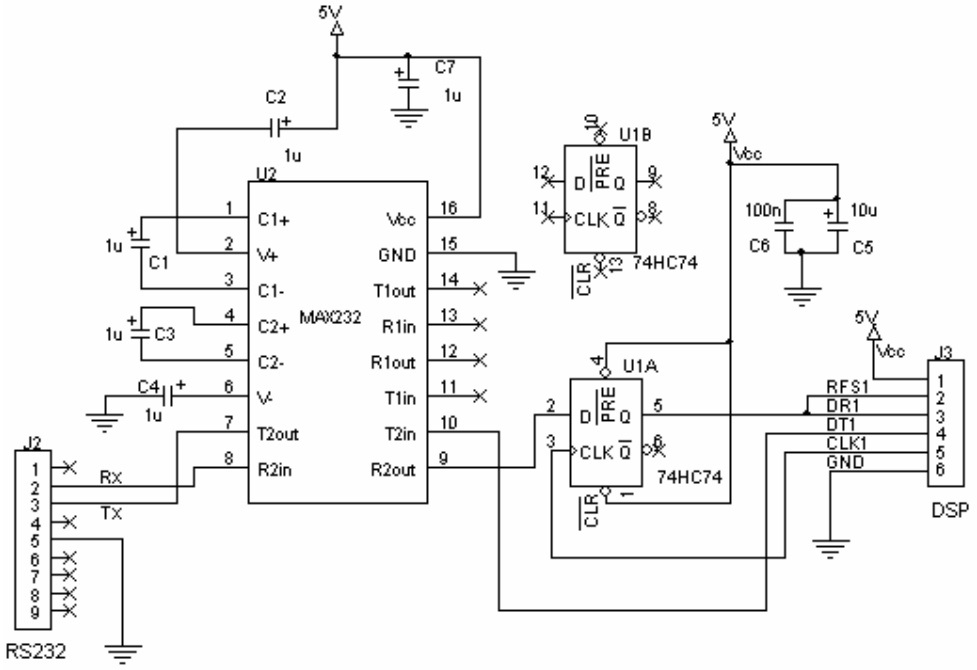
a) a DSP szinkron soros porttal rendelkezik, míg a mote szinkron soros portját a rádiós IC használja, így csupán az aszinkron soros port használható

b) a DSP 5 V-os, míg a mote 3 V-os tápfeszültségről üzemel

Az a) probléma megoldására a [23]-ban kínál a gyártó megoldást. Lényege, hogy a DSP szinkron soros portja képes saját maga számára előállítani az órajelet. Amennyiben ezt az aszinkron soros port frekvenciájának többszörösére állítjuk, akkor az aszinkron port jelét túlmintavételezve megbízható kommunikáció lehetséges.

A b) probléma megoldása közvetett úton történik. Nem a mote 3 V-os jelszintjét illesztjük az 5 V-os jelszinthez, hanem felhasználjuk a mote-ok programozó kártyáját. A kártyára a mote-ot ráhelyezve, a mote soros portjának jeleihez szabványos RS232-es csatlakozón és jelszinten hozzáférhetünk, így egyszerűbb az illesztés. Másik előnye ennek a megoldásnak, hogy a programozó kártya segítségével megoldható a mote

hálózatról történő táplálása. Mivel az átjáró mote a jelfeldolgozó egység mellett foglal helyet, ahol egyébként is szükséges a hálózati táplálás, így ez nem jelent jelentős plusz költséget, és mivel nem telepes a táplálás, kevésbé fontos az energiatakarékosság.
A programozó kártya és a DSP illesztéséhez szükséges áramkör kapcsolási rajza a 7.5. ábrán látható.



7.5. ábra. RS-232 ⇔ DSP illesztő áramkör

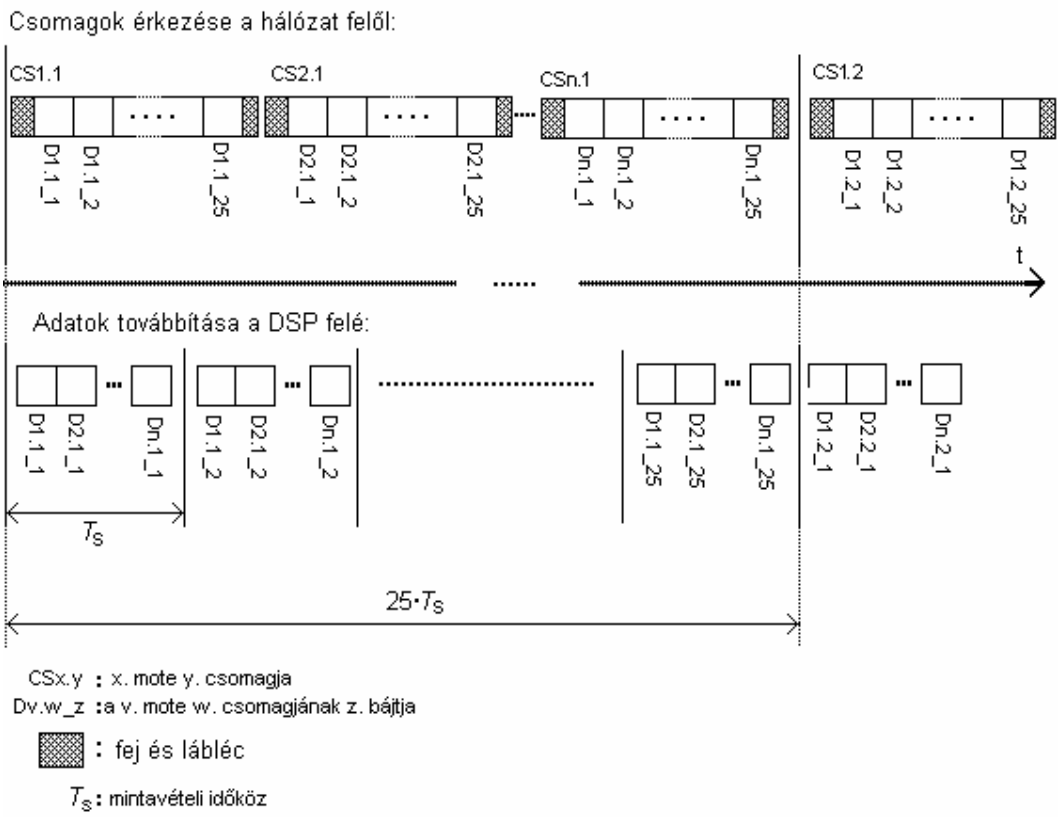
Ez az áramkör egy RS232-es és TTL jelszint konverziót megvalósító MAX232-es IC-t, a DSP soros porti órajelével működő dual D-flip-flopot tartalmazó SN74HC74-es IC-t, és az azokhoz tartozó passzív alkatrészeket tartalmazza. A MAX232-es IC szerepe a jelszint illesztés, a flip-flop pedig a DSP soros porti órajeléhez szinkronizálja a jeleket. Az illesztő áramkör a tápfeszültséget a DSP csatlakozóján keresztül kapja. A DSP felől található csatlakozón CLK1 a szinkron soros port órajelét jelöli. DT1 (Data Transmit) és DR1 (Data Receive) a soros port adó és vevő ki- és bemenete. Az RFS1 (Receive Frame Signal) lábon a soros porton érkező keretek elejét kell jelezni egy lefutó éllel. A gyártó ajánlása alapján, ezt a DR1 bemenettel kapcsoljuk össze, ugyanis így az RS-232-es vonalon a bájtok elejét jelző START feltételt (lefutó él) használjuk fel keretezésre, így egy keret egy bájtot tartalmaz.

A szoftveres beállítások szerint a soros kommunikáció 9 biten történik. 8 biten az adatokat továbbítjuk, a 9-edik bit keretezésre szolgál, így szinkronizálható a kommunikáció, mivel semmilyen handshake jelet nem használunk.

Bár lehetőség lenne rá, a DSP-n vétel módban nem használunk DMA-t, mivel úgy nem lenne hatékonyan biztosítható a keret felismerése. Adó módban viszont, mivel egész kereteket küldünk, jól használható DMA módban a soros port.

7.3.2. Adatátvitel

A szenzorhálózat által gyűjtött mintákat az átjáró mote fogadja, és továbbítja a DSP felé. Az adatok továbbítása viszont nem egyszerűen a csomagok továbbküldését jelenti. Az átjáró mote a szenzorhálózatból érkező blokkos adatokat újra felbontja és mintavételi időközönként továbbítja a mintákat, ahogyan azt a 7.6. ábra mutatja.

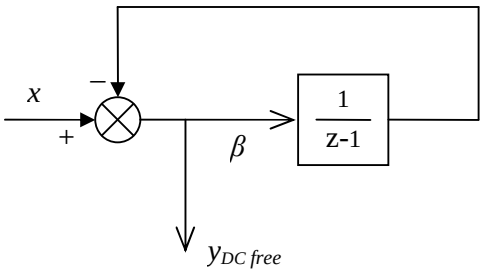


7.6. Az adatformátum átalakítása

Ahhoz, hogy a minták érkezésének és továbbításának sebessége megegyezzen az átjáró mote-on, azt a már leírt módszerrel szinkronizáltuk a referencia mote-hoz, így a minták továbbítása szinkronizáltan történik, nem fordulhat elő adat alul- és túlcsordulás. Ezzel a módszerrel az átjáró mote a DSP szempontjából gyakorlatilag úgy tekinthető, mint egy sokcsatornás AD átalakító, mely az egyes csatornák mintáit adott mintavételi frekvenciával továbbítja soros csatornán. Az átjáró T_s mintavételi időközönként egymás után továbbítja a mote-októl származó adatokat a mote-ok azonosítójának sorrendjében. A nulladik mote adatának továbbításakor a 9. bit egyes, a többi esetben nulla. Ezzel a módszerrel lehet biztosítani, hogy akármikor is kezdi a DSP figyelni az átjáró által szolgáltatott adatokat, mindig meg tudja állapítani, hogy melyik adat melyik mote-hoz rendelhető, hiszen ezen kívül semmi sem különbözteti meg egymástól a továbbított bájtokat.

Mivel a DSP-n és a szenzorhálózatban nem egyezik meg a mintavételi frekvencia, és a DSP mintavételi frekvenciáját a hozzá csatlakoztatott kodek határozza meg, a mote-ok mintavételi frekvenciája pedig nem biztos, hogy el tudja érni a DSP mintavételi frekvenciáját, ezért a DSP-t az 5.5. ábrán látható lineáris interpolációs módszerrel szinkronizáljuk a hálózathoz. A T_{samp1} ebben az esetben a DSP mintavételi időközeit, míg a T_{samp2} a mote-ok mintavételi időközeit jelöli. Az (5.19) képletben szereplő $(t–T_i)$ értéket – melyet az 5.5. ábrán ΔT-vel jelöltünk – a DSP a nullás azonosítójú mote utolsó adatának beérkezésétől méri a rajta található időzítő segítségével. Mivel a mote-ok mintavételi időpontjai szinkronizálva vannak, így minden mote esetén ugyanazon ΔT és $(T_{i+1} – T_i)$ értékek használhatók. Az interpolációs szinkronizáció segítségével a DSP-n futó jelfeldolgozó algoritmusok számára folyamatosan rendelkezésére áll a mote-okon mintavételezett zajminta, persze csupán adott késleltetéssel.

Az így érkező minták azonban még nem minden esetben használhatók fel közvetlenül a zajelnyomó algoritmusban. A mote-okon lévő AD- átalakító ugyanis unipoláris, így a mikrofonról érkező hangjelek egy adott DC szintre szuperponálódnak. A rezonátoros struktúránál ez nem jelent problémát, ugyanis abban a DC jelet nem csatoljuk tovább a kimenethez, így a rendszer nem próbálja kioltani azt. Az LMS algoritmus viszont nem szelektív semmilyen frekvenciára, tehát az egyenszint kikompenzálására is törekszik. Az akusztikus rendszer DC átvitele viszont zérus, így a hangszóróra kiadott egyre növekvő egyenszintű jel nem kompenzálja a DC hibát, és egy idő után telítésbe vezérli a DA- átalakítót, mely a rendszer működésképtelenségét okozza. Ezen probléma megoldására a mote-októl érkező jelek DC komponensét el kell távolítani. Erre a 7.7. ábrán látható struktúrát használtuk.



7.7. ábra. DC kompenzálás

Ez a rendszer tulajdonképpen olyan rezonátoros megfigyelő struktúra, mely csak DC csatornát tartalmaz. A zajcsökkentő rendszereknél bemutatottak alapján a megfigyelő hibajelében a megfigyelt jelekre vonatkoztatott átvitel nulla, így a hibajel – melyet az ábrán $y_{DC\ free}$-vel jelöltünk – nem tartalmaz DC komponenst. Ez tehát a zajcsökkentő rendszerek egy olyan alkalmazása, melyben cél a jelek DC szintjének elnyomása. A rezonátor állapotváltozójának kezdeti értéke 0.5, mivel az állapotváltozó a DC szintet tartalmazza, és ideális esetben körülbelül ekkora a jelek egyenszintje – a teljes dinamikatartomány 0-tól 1-ig terjed. A becsatoló β paraméter megválasztása

szimulációk alapján történt. A paraméter nem állandó értékű, kezdetben 10^{-3}, és exponenciálisan 10^{-6}-ra áll be. Azért kell ilyen kicsire választani a becsatoló együttható végértékét, mert így kevésbé befolyásolják a DC szintre szuperponált jelek a DC szintet tartalmazó tároló értékét, viszont a DC szint lassú változását – amit például az analóg áramköri elemek ofszetvándorlása okoz – követni tudja. A kezdeti érték viszont azért nagyobb ennél, mert így gyorsabban áll be a megfigyelő a DC szint közelébe, hiszen a 0.5-ös inicializált érték csupán becslés, a valódi DC szint eszköztől függő paraméter lehet.
A fenti struktúra átviteli függvényét a (7.2) egyenlet írja le:

$$W(z) = \frac{Y_{DCfree}(z)}{X(z)} = \frac{z-1}{z-(1-\beta)} \tag{7.2}$$

A $W(z)$ átviteli függvénynek tehát nulla frekvencián zérusa van, így a DC szintre nulla az átvitele. Minél kisebb a β paraméter, annál közelebb kerül a zérus és a pólus, így $W(z)$ átvitele annál alacsonyabb frekvencián éri el az egységnyi átvitelt.

Az illesztések megvalósítása lehetővé tette a DSP-n a 2.4. és a 2.13. ábrán látható, előrecsatolt FxLMS és rezonátoros struktúrák implementálását, melyek a mote-ok által küldött zajmintákat feldolgozva üzemelnek.

8. Rezonátoros zajelnyomás elosztott jelfeldolgozással

A 7. fejezetben bemutatott rendszer hátrányos tulajdonsága, hogy a hálózatban lévő zajérzékelő mote-ok számával fordítottan arányos a rajtuk elérhető maximális mintavételi frekvencia. Ennek oka, hogy minél több mote van a rendszerben, annál több minta keletkezik adott időintervallumban állandó mintavételi frekvencia mellett. A keletkezett mintákat viszont ugyanakkora sávszélességű csatornán kell átvinni, amely egy bizonyos adatmennyiség esetén betelítődik, így a minták nem vihetőek át folyamatosan.

Erre a problémára kínál megoldást az az alapgondolat, hogy a zajmintákon bizonyos előfeldolgozást végezve, olyan, a jelet leíró paramétereket állíthatunk elő, melyek jóval lassabban változnak, mint a jel időfüggvénye. Ezeknek a paramétereknek a továbbítása már kisebb gyakorisággal is lehetséges, így kevésbé foglaljuk a rádiós hálózat sávszélességét, viszont az állandó mintavételi frekvencia tartásával a zajcsökkentő rendszer sávszélessége nem csökken. Ennek persze feltétele az, hogy találjunk ilyen paramétereket. Kézenfekvőnek tűnik a jelek 2.2.1. alfejezetben bemutatott Fourier-együtthatóit meghatározni, melyek periodikus jelek leírására alkalmasak. Ezen paraméterekkel sztochasztikus jelek nem jellemezhetőek, a sztochasztikus jelek olyan paramétereinek megtalálása, mely a zajcsökkentő rendszerekben is felhasználható, igen bonyolult, ha nem reménytelen feladatnak látszik. A felsorolt okok miatt ezen fejezetben csupán periodikus jelek elnyomására alkalmas rendszer bemutatására kerül sor.

8.1. Zajcsökkentő rendszer felépítése

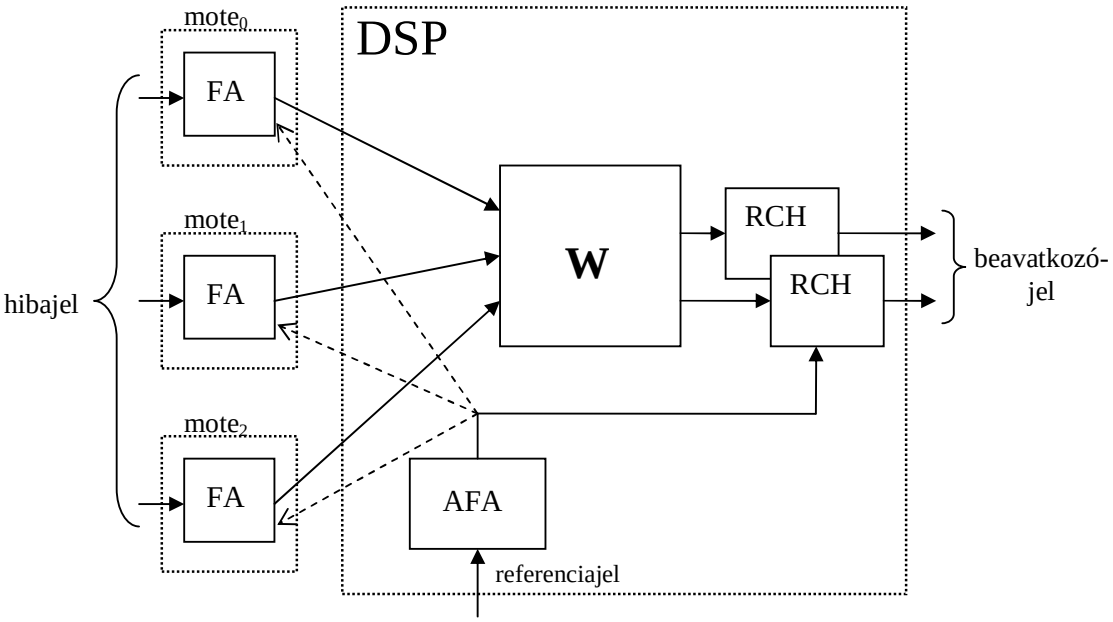
A rendszer sémája megegyezik a 7.1. ábrán vázolt elrendezéssel. Változás az, hogy a mote-ok nem a jel időfüggvényét, hanem a jelet leíró Fourier-együtthatókat küldik a bázisállomásnak, mely a DSP felé továbbítja azokat. Ezzel kapcsolatban fellépő új probléma, hogy a mote-oknak ismerniük kell a (2.11) képletben látható $c_{k,n}$ bázisfüggvényeket, hiszen a Fourier-együtthatók meghatározásához ez nélkülözhetetlen. Mivel ezeket a bázisfüggvényeket a DSP-n állítjuk elő, így a rendszerben egy DSP → mote irányú kapcsolatot is fel kell építeni, mely a hálózati struktúra megváltozását is magával vonja.

A DSP-n a jelparaméterek feldolgozására két lehetőség kínálkozik:

- a paraméterekből a jel rekonstruálása, és továbbítása a zajelnyomó algoritmushoz
- a jelparaméterek közvetlen felhasználása zajcsökkentésre

Rendszerünkben a második megoldást alkalmaztuk, ugyanis ez kiválóan illeszkedik a rezonátoros zajcsökkentő struktúrába, gyakorlatilag azt jelenti, hogy a 2.13. ábrán látható struktúrában a Fourier-együtthatókat előállító blokkok nem a DSP-n, hanem a

mote-okon foglalnak helyet. Ez alapján alakult ki a 8.1. ábrán látható elosztott rezonátoros struktúra.

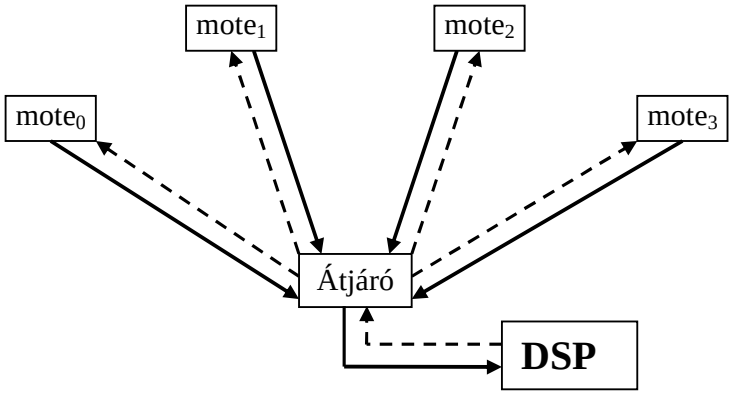


8.1. ábra. Elosztott rezonátoros struktúra

Kis különbséget jelent még az eredeti struktúrához képest, hogy míg ott az FA blokkok minden mintavételi ütemben az AFA-tól kapták a $c_{k,n}$ együtthatókat, ez ebben az esetben nem lehetséges, hiszen a hálózat nem elég gyors ahhoz, hogy a bázisfüggvényeket minden ütemben továbbítsa. Ehelyett a mote-ok megkapják az AFA-tól – természetesen közvetetten az átjáró mote-on keresztül – bizonyos időközönként a jel frekvenciáját és pillanatnyi fázisát, ezek alapján pedig a bázisfüggvények meghatározhatók, mivel a jel frekvenciájának és egy adott pillanatban felvett fázisának ismeretében bármilyen időpontban kiszámítható fázisának új értéke is.

8.2. A hálózati felépítés

A 8.1. alfejezetben említett okok miatt a hálózat kialakításakor ebben a rendszerben a bázisállomás nemcsak adatgyűjtő szerepet játszik, hanem maga is küld csomagokat a hálózat többi tagja felé. Ez lehetőséget ad bizonyos szempontokból tekintve jobb tulajdonságokkal rendelkező hálózat kialakítására. Emiatt ebben az esetben a hálózat nem a már bemutatott token ring algoritmus alapján működik, hanem időosztásos (TDM) rendszerben. Ez azt jelenti, hogy minden egyes mote kap egy saját időszeletet, amit a bázisállomás csomagjainak érkezéséhez viszonyítunk. A hálózati kapcsolatokat a 8.2. ábrán láthatjuk. A folytonos vonal az adattovábbítást, a szaggatott vonal a szinkronizációt jelöli.



8.2. ábra. TDM hálózat

Ez a token ring hálózattal szemben több előnyös tulajdonsággal is rendelkezik:

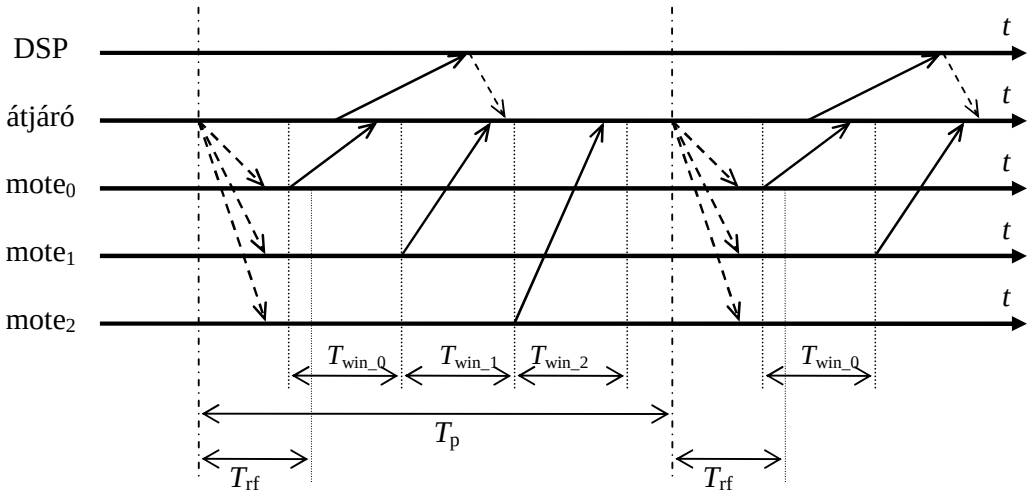
- A mote-oknak csupán a bázisállomással kell kapcsolatban lenniük. A többi mote üzeneteit nem kell figyelni, így azok feldolgozása nem foglalja a mote-ok erőforrásait. Ezzel kapcsolatban segítséget jelent a mote-okon található rádiós IC, amely hardveresen támogatja a címfelismerést, így azok az üzenetek, melyeket nem az adott mote-nak címeztek, el sem jutnak a processzorhoz.
- Mivel minden mintavételező mote közös szinkronjelet kap a bázisállomástól, így az RBS szinkronizációnál leírt elv szerint ezek igen nagy pontossággal szinkronban lesznek. Ebben az esetben ez ugyan nem számít, de lehetnek olyan alkalmazások, ahol fontos.
- Nincs kitüntetett szerepű mote a telepes táplálású mote-ok között, mivel a szinkronizációt és a hálózat időzítését a bázisállomás végzi, így a hálózat kevésbé sérülékeny abban az esetben, ha valamelyik eszköz telepe lemerül.
- A rendszer skálázhatósága jóval egyszerűbb, mivel a mote-ok függetlenek egymástól.

Hátrányos tulajdonságként róható fel a token ring hálózattal szemben, hogy 8.2. ábrán látható elrendezésben az állomásoknak úgy kell kiosztani az időszeleteket, hogy biztosan ne történjen ütközés az adatok továbbításakor Ez az időzítési adatokba vitt biztonsági szakaszokkal lehetséges, mely csökkenti a hasznos adatátviteli időt. Mivel a token ring hálózatban egymást indítják a szomszédos állomások, tehát nem kell biztonsági korlátokat hagyni, jobb csatornakihasználás lehetséges. Az, hogy melyik módszert használjuk, alkalmazástól függ. Mivel az elosztott rezonátoros struktúránál kevésbé számít a csatorna kihasználtsága, ezért alkalmazzuk a bemutatott TDM rendszert.

A hálózat működését a 8.3. ábra szemlélteti. Az egyes nyilak kezdete az üzenetküldés kezdetét jelzi, tehát amikor kiadjuk az adott eszközön a küldést kezdeményező parancsot, a nyíl vége pedig az üzenet megérkezését jelöli, tehát amikor

az alkalmazás számára elérhető a teljes üzenet. Látható módon a hálózat működése periodikus. A periódusidőt úgy választjuk meg, hogy a hálózat minden üzenetének elküldésére legyen elegendő idő.

Egy periódus az árjáró mote szinkronizációs üzenetével kezdődik. Ezen üzenet megérkezésétől számítják a mote-ok a saját időkereteiket, amelyet az azonosítóik sorrendjében kapnak meg. Ebből következik, hogy minden mote-nak egyedi azonosítóval kell rendelkeznie. Ez nem különösen szigorú feltétel, gyakorlatilag minden hálózatban megkövetelik. Az időkeret akkora, hogy abba beleférjen egy üzenet továbbítása. Meghatározása mérések segítségével történt. Természetesen az időkeret tartalmaz egy biztonsági időrést is, mely körülbelül a szükséges idő 30%-a.



T_{win_i}: i. mote időkerete — : adatot tartalmazó üzenet
T_p: egy hálózati periódus - - - : szinkronizációs üzenet

8.3. ábra. Adattovábbító hálózat működése

A mote-ok előző periódusban továbbított üzeneteit a bázisállomás az új periódus elején küldi tovább a DSP felé. Azért nem egyből, mert a rádiós szinkronizációs üzenet küldésében esetleg bizonytalanságot okozhat a soros porton történő kommunikáció. Mivel a rádiós üzenet továbbítási ideje szinkronizációs szempontból fontos, így érdemes annak elküldésének idejére minél kevesebb feladattal terhelni a processzort. Célszerű viszont a DSP felé minél hamarabb továbbítani az adatokat, mivel ez a szabályzási hurokban lévő késleltetést csökkenti.

A hálózat indítása, az egyszerű mintavételezést megvalósító hálózathoz hasonlóan, a nullás azonosítójú mote bekapcsolásával végezhető, miután már a hálózat összes többi mote-ját bekapcsoltuk. Ez egy start üzenetet küld a bázisállomás felé, de még nem indul meg az FA és a mintavételezés. A bázisállomás a start üzenet hatására lép működésbe, a hálózat tagjai pedig a bázisállomás első szinkronizációs üzenete hatására, így minden zajérzékelő mote egyszerre indul el.

8.3. A többszintű szinkronizáció megvalósítása

Az elosztott rezonátoros zajcsökkentő rendszerben a szinkronizáció nem csupán a mintavételi időpontok szinkronizálását jeleni, ami az átviteli függvény állandósága miatt szükséges. A mote-okon futó rezonátorok ugyanis a mintavételezett zaj Fourier-együtthatóit állítják elő, mely a periodikus zaj harmonikus komponenseinek amplitúdóját és fázisát határozza meg. Ezen két adat közül a fázis a kritikus, ugyanis a fázis megadásához szükség van egy referencia értékre is, amihez a viszonyítás történik. Viszonyítási alapként a mote-ok által futtatott rezonátoros megfigyelőben a $c_{i,n}$ bázisfüggvények szolgálnak. Ahhoz tehát, hogy értelmezni lehessen a mote-ok által szolgáltatott Fourier-együtthatókat, szinkronizálni kell a mote-okat a DSP-n futó rezonátorokhoz, hiszen végeredményben itt történik az együtthatók felhasználása. A szinkronizáció két szinten történik. Első lépésben az átjáróként szolgáló mote szinkronizálódik a DSP-hez, majd a hálózat mote-jai szinkronizálódnak az átjáróhoz. A két szinkronizációs lépcső megvalósítása két különböző módszerrel történik.

A leírtak alapján a bázisállomás szolgál referenciaként a hálózat mote-jai számára. Ez igaz mind a mintavételi időpontok, mind a bázisfüggvények tekintetében. A mintavételi időpontok kijelölése az átjáró saját időzítője segítségével történik, mely adott mintavételi frekvenciával megszakításokat generál, ezek jelölik a referencia mintavételi időpontokat, ehhez szinkronizálja a hálózat mote-jait. A rezonátorok szinkronizálásához a bázisállomáson is fenn kell tartani egy referencia rezonátorkészletet. A rezonátorkészlet itt nem azt jelenti, hogy ismeri minden $c_{i,n}$ együttható értékét, elég, ha az alapharmonikus fázisát és frekvenciáját ismerjük, a többi ebből származtatható. A rezonátorok szinkronizálása a következő módszerrel történik. A 8.3. ábrán a periódusok elején jelölt időpontban a bázisállomás elküldi az alapharmonikus bázisfüggvény pillanatnyi fázisát (φ) és frekvenciáját (f_1). Ezt az üzenetet használjuk egyébként a mintavételi időpontok szinkronizációjához is, mely az 5.2. alfejezetekben leírt módszerrel történik a hálózatban található mote-okon. Eltérést jelent az elosztott rezonátoros rendszerben a „hagyományos” rezonátoros rendszerekhez képest, hogy itt az AFA nem tudja minden mintavételi időpontban szolgáltatni az FA blokkok számára a bázisfüggvényeket, mivel azt nem engedi meg a hálózat adatátviteli sebessége. Amennyiben lehetőség lenne a folyamatos szinkronizációra, akkor elegendő lenne minden esetben csak a fázis továbbítása. Ebben az esetben viszont az FA blokkoknak az AFA nélkül is elő kell tudniuk állítani a bázisfüggvényeket az új szinkronizációs üzenet megérkezéséig, melyhez egy adott pillanatban felvett φ fázison kívül az f_1 frekvencia ismerete szükséges.

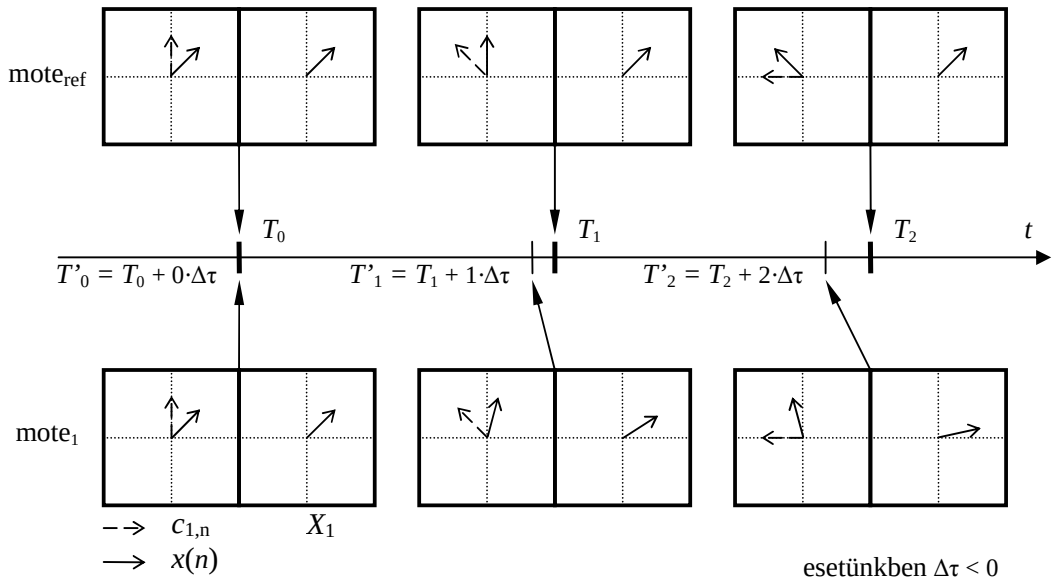
A 8.3. ábrán látható módon a szinkronizációs csomag elküldése után a mote-ok a csomag elküldése után T_{rf} idővel tekintik érvényesnek a benne lévő szinkronizációs adatokat, akkor használják fel a rezonátorok paramétereinek frissítésére.
Ezek után a rezonátorok számítása a következő módon történik:

$$c_{k,n} = e^{\mathrm{j}k(2\pi f_1 \Delta n + \varphi)} \tag{8.1}$$

ahol Δn a φ és f_1 paraméterek frissítése óta eltelt mintavételi időpontok számát jelöli. A mote-okon található rezonátorok csupán akkora $\Delta\varphi$ fázisszöggel vannak lemaradva a referencia értéktől, mint amennyit a referencia rezonátor fázisa változik a szinkronizációs üzenet elküldésétől a T_{rf} idő alatt, amikor is φ értéket aktualizálják. Ez a fáziskésés adott frekvencián állandó, mivel a bázisállomás és a mote-ok mintavételi időpontjai szinkronizálva vannak, és a csomagtovábbítási idő minden esetben állandónak tekinthető. A konstans fáziskésés az identifikáció során a másodlagos út átviteli függvény részeként szerepel, és egy állandó fázistolást okoz.

Megjegyezzük, hogy a rezonátorok szinkronizálására nem ez az egyetlen módszer. Ebben az esetben ugyanis feltételeztük, hogy a mintavételi időpontok szinkronizálva vannak, mely egyébként az 5.2. alfejezetben leírtak alapján egy lehetséges, és a felmerülő alternatívák közül a legjobbnak vélt megoldás az átviteli függvény állandóságának biztosítására.

A 8.4. ábra alapján beláthatjuk, hogy a mintavétel szinkronizációja szükséges, bár nem kizárólagos megoldás. Ehhez tételezzük fel, hogy a mote-ok mintavételi időpontjai nincsenek szinkronizálva.



8.4. ábra. Szinkronizáció szemléltetése

Tegyük fel, hogy a Fourier-együtthatót csupán a (8.2) egyenlettel leírt jelre kell meghatároznunk. Ez tulajdonképpen egy alapharmonikus frekvenciával forgó komplex vektor.

$$x(n) = X_1 \cdot e^{\mathrm{j}2\pi f_1 n} \tag{8.2}$$

vagy

$$x(t) = X_1 \cdot e^{\mathrm{j}2\pi f'_1 t} \tag{8.3}$$

ahol X_1 az alapharmonikus Fourier-együtthatót, f_1 a mintavételi frekvenciához viszonyított frekvenciát, f'_1 pedig a tényleges frekvenciát jelöli. $x(n)$ előállításához (2.10) és (2.11) alapján csupán $c_{1,n}$ bázisfüggvény szükséges: $x(n) = X_1 \cdot c_{1,n}$, így a többi bázisfüggvénnyel nem foglalkozunk, mivel az azokra vonatkoztatott együttható nulla. A vizsgálat a többi bázisfüggvényre is elvégezhető több harmonikus esetén.

A 8.4. ábrán a dupla négyzetek bal oldalán egy olyan koordináta-rendszer látható, melyben szaggatott vonallal a $c_{1,n}$ bázisfüggvény, míg folytonos vonallal az $x(n)$ jel látható. A jobb oldali koordináta-rendszerben X_1 komplex Fourier-együttható látható. A koordináta-rendszerekben ábrázolt mennyiségek az időtengelyen jelölt időponthoz tartoznak. T_i időpontok jelölik a referencia időpontokat. A referencia mintavételi időköz: $T_s = T_{i+1} - T_i$, míg $mote_1$ mintavételi időközei: $T_s + \Delta\tau$. Az időtengely feletti ábrák egy referencia sebességgel mintavételező mote-hoz, míg az időtengely alattiak egy gyorsabban mintavételező mote-hoz tartoznak. Mintavételezett rendszer lévén mindkét mote-on a mintavételi időpontokban a $c_{1,n}$ bázisfüggvény fázisa $2\pi f_1$-gyel nő. Esetünkben $f_1 = {}^1/_8$. A megfigyelt $x(t)$ jel azonban a rendszertől független jel, mely a valós idő alapján változik. Frekvenciája szintén $f_1 = {}^1/_8$. Ennek következtében a felső ábrasorozaton $x(t)$ vektor együtt forog $c_{1,n}$-nel, míg az alsó ábrákon fokozatosan lemarad attól:

$$x(T_i + i\cdot\Delta\tau) = X_1 \cdot e^{j2\pi f'_1 (T_i + i\cdot\Delta\tau)} = X_1 \cdot e^{j2\pi f'_1 T_i} \cdot e^{j2\pi f'_1 i\cdot\Delta\tau} \quad (8.4)$$

$$x(T_i + i\cdot\Delta\tau) = [X_1 \cdot e^{j2\pi f'_1 i\cdot\Delta\tau}] \cdot e^{j2\pi f'_1 T_i} = X'_1 \cdot c_{1,n} \quad (8.5)$$

(8.5) alapján látható, hogy míg az első esetben az X_1 állandó értékű ($mote_{ref}$), mint ahogyan azt (8.2) előírja, a második esetben egy X'_1 változó együtthatót kapunk ($mote_1$), mely a csúszás mértékétől függő sebességgel forog. Mivel X_1 illetve X'_1 a megfigyelt jel fázisát és amplitúdóját adja meg, és ezeket a 8.1. ábrán látható FA blokkok állítják elő a mote-okon, így a konkrét zajcsökkentő rendszerben f_1 frekvenciájú, adott fázisú jel kiadása esetén, hiába vannak az AFA által szinkronizálva a rezonátorok fázisban és frekvenciában, a mintavétel csúszás miatt mégis folyamatosan változó fázisú Fourier-együtthatók érkeznek a szinkronizálatlan mote-okról. Mivel az FA blokkok a zajcsökkentő rendszer visszacsatoló ágában vannak, ez azt jelenti, hogy a másodlagos út átviteli függvénye folyamatosan változik, mely instabilitáshoz vezet.

A probléma a 8.4. ábra alapján kétféle módszerrel oldható meg:

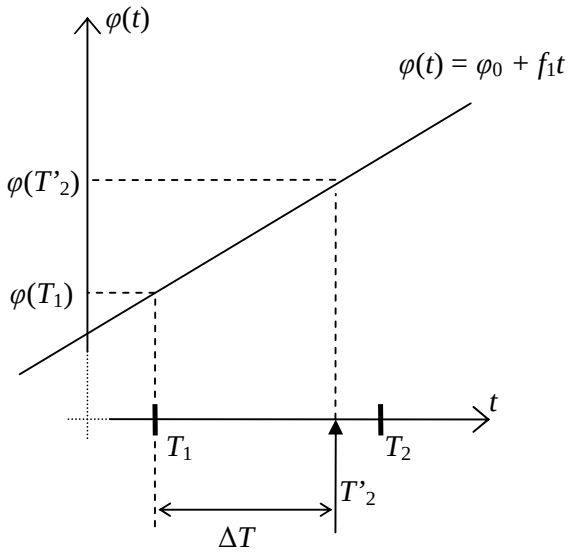
a) A csúszás mértékét figyelve a $c_{i,n}$ együtthatók fázisát úgy módosítjuk, hogy az megfeleljen azok T'_i-nek megfelelő értéknek.

b) A mintavételi időpontok szinkronizálásával $T'_i = T_i$ biztosítása.

A megvalósított rendszerben az utóbbi megoldást alkalmaztuk, ugyanis a csúszás mérése mindkét esetben szükséges, viszont az 5. fejezetben leírt algoritmus segítségével a szinkronizáció egyszerűen biztosítható.

Az eddig leírtak alapján látható, hogy hogyan történik a szinkronizáció az átjáró mote és a hálózat között. A referencia mintavételi időpontokat az átjáró állítja elő, viszont a rezonátorokat a DSP-n lévő rezonátorokhoz kell szinkronizálni, ezért szükség van az átjáró DSP-hez történő szinkronizálására is. Ebben az esetben viszont nem lehet

szó a DSP és az átjáró mintavételi időpontjainak szinkronizálásáról, ugyanis a DSP és a mote-ok mintavételi frekvenciája még névlegesen sem egyezik meg. Előfordulhat, hogy a mote-ok nem is tudnak akkora mintavételi frekvencián dolgozni, mint a DSP, aminek a mintavételi frekvenciáját a hozzá csatlakoztatott kodek határozza meg. Ebben az esetben tehát a szinkronizáció azon módját alkalmaztuk, melyről a 8.4. ábra elemzése alkalmával már szó esett a szinkronizációs alternatívák ismertetésénél – a) módszer –, nevezetesen: amennyiben nincs lehetőség a mintavételi időpontok egyeztetésére, akkor a mintavételi időpontok elcsúszása alapján előállítjuk a fázis pillanatértékét a kívánt időpontban. Ennek megvalósításához vizsgáljuk meg a 8.5. ábrát.



8.5. ábra. Szinkronizáció fázisbecsléssel

T'_2 időpontban a fázis meghatározása f_1 frekvencia esetén a következő módon történik:

$$\varphi(T'_2) = \varphi(T_1) + f'_1 \cdot \Delta T \tag{8.6}$$

Ez tulajdonképpen megegyezik egy lineáris interpolációs módszerrel, viszont az 5.3. fejezetben bemutatott technikával ellentétben a jel interpolációja nem egy intervallum két végpontjában felvett értékeinek ismeretében történik. Ebben az esetben ugyanis ismerjük az interpolálandó jel intervallum elején felvett értékét és annak deriváltját, hiszen a frekvencia a fázis deriváltja. A becslést ezen értékek alapján végezzük.

Az ábrán T_1 és T_2 a DSP mintavételi időpontjait jelenti. Mivel a $c_{k,n}$ bázisfüggvények fázisának frissítése a mintavételi időpontokban történik, így a $\varphi(T_1)$ az utoljára frissített fázist jelöli. T'_2 időpont azt jelöli, amikor az átjáró mote szinkronizálódni szeretne a DSP rezonátoraihoz. A szinkronizáció kérése a konkrét rendszerben implicit szinkronizációs üzenettel történik, ugyanis a mote-ok rezonátorainak értékét tartalmazó csomag elküldésével az átjáró mote a szinkronizációt is kezdeményezi. Ennek eredményeképpen a DSP (8.6) alapján kiszámítja a kérés

időpontjában az alapharmonikus rezonátor fázisának értékét. Ezt, valamint az alapharmonikus frekvenciájának transzformált értékét soros porton továbbítja az átjáró felé, melyek alapján az átjáró mote frissíti saját rezonátorának paramétereit, így megvalósul a szinkronizáció. Az itt vázolt üzenetcserét a 8.3. ábrán is tanulmányozhatjuk.

Az alapharmonikus frekvenciájának transzformációja azért szükséges, mivel a két rendszer nem ugyanazzal a mintavételi frekvenciával dolgozik. Mivel mind a DSP-n lévő algoritmus, mind az átjárón lévő algoritmus a saját mintavételi frekvenciájához viszonyított mintavételi frekvenciát használja fel, ezért ugyanazt a frekvenciát a két rendszerben eltérő számmal jellemezhetjük. A relatív frekvenciák transzformálására az ad lehetőséget, hogy a két különböző számmal jellemzett frekvencia valójában megegyezik. Jelölje az f'_1 valódi frekvenciát a DSP-n $f_{1\mathrm{DSP}}$ a mote-on pedig f_{1mote}, melyeket a következő két képlet alapján számítunk:

$$f_{1_{DSP}} = \frac{f'_1}{f_{s_{DSP}}} \tag{8.7}$$

$$f_{1mote} = \frac{f'_1}{f_{s_{mote}}} \tag{8.8}$$

ahol $f_{s_{DSP}}$ a DSP, $f_{s_{mote}}$ a mote mintavételi frekvenciáját jelöli. A (8.7) egyenletből f'_1-t kifejezve és (8.8)-ba behelyettesítve megkapjuk a mote-nak küldendő transzformált frekvenciát:

$$f_{1mote} = \frac{f_{s_{DSP}}}{f_{s_{mote}}} \cdot f_{1_{DSP}} \tag{8.9}$$

Látható, hogy a transzformáció csupán egy szorzást jelent. Ezt célszerű a DSP-n elvégezni, ugyanis itt jóval kisebb ennek a számításigénye.

Az hogy az átjáró mote-ot használtuk a mintavételi időpontok szinkronizációs referenciájaként, nem például a hálózat valamely elemét, több szempontból előnyös. A zajérzékelő mote-oknak nem kell egymást hallani, csupán a bázisállomást, így a hálózat nagyobb területre kiterjeszthető. A másik nyomós érv az, hogy a szinkronizációs algoritmusunkban azt használjuk ki, hogy a csomagok átviteli ideje minden esetben állandó. Amennyiben a processzor terhelt, ez nehezen biztosítható. A hálózatban lévő mote-okon futó rezonátorok igen számításigényesek, és ráadásul futási idejük is függ az üzemi frekvenciától, hiszen frekvenciatartományonként (2.12) alapján változik a rezonátorszám. Ez nagyon megnehezíti az állandó csomagátviteli idő biztosítását. Gyakorlatban viszont könnyebb biztosítani a vevő oldalon az időzítések determinisztikusságát, hiszen elég csupán a csomag megérkezésének időpontját feljegyezni, amely egy rövid megszakítási rutinban elvégezhető, a feldolgozás már nem olyan sürgős. Küldésnél viszont több feladatot kell elvégezni, mely a TinyOS részletes ismeretét feltételezi, ezért erre nem térek ki.

8.4. Megvalósítási nehézségek

A fejezet bevezetőjében leírtak alapján ismeretes, hogy a mote-ok a zajjel Fourier-együtthatóit állítják elő és továbbítják a DSP felé, így csökkentve az átviendő adatok mennyiségét. A megoldás elméletileg teljesen kézenfekvő, hiszen ez nagyon jól illeszkedik a periodikus zajok elnyomására kidolgozott zajcsökkentő rendszerbe.

Gyakorlati oldalról tekintve azonban annál problematikusabb, hiszen egy FA blokk megvalósítása elég számításigényes. A DSP-n történő implementálás esetén ez nem jelent igazán nehézséget, hiszen azon a programfejlesztés assembly nyelven történik, mely segítségével jól kihasználhatóak a processzor erőforrásai. A lebegőpontos 32 bites aritmetika és a párhuzamos műveletvégzés lehetővé teszi igen hatékony kód megírását. A mote-okon ellenben NesC-ben történik a programfejlesztés, ráadásul egy általános célú 8 bites mikrokontrolleren fut a program, és csupán egész aritmetika használata lehetséges. Mindezek miatt igen alaposan át kell gondolni a megvalósítás részleteit, ahhoz, hogy a megírt rezonátoros megfigyelő struktúra megfelelően gyors legyen, és a számítási ideje beleférjen egy mintavételi intervallumba a mintavételezés és kommunikáció kezelése mellett is.

A programozástechnikai nehézségeken kívül problémát okoz sok esetben az algoritmusok működőképességének tesztelése. A jelfeldolgozási algoritmusok eredményeként előálló kimeneti jelek megfigyelése ugyanis nem történhet analóg jeleket real-time feldolgozó és megjelenítő műszerekkel (oszcilloszkóp, spektrumanalizátor, hálózatanalizátor), mint a DSP-n implementált jelfeldolgozó algoritmusok tesztelésénél. Ennek oka, hogy a mote-ok nem tartalmaznak DA-átalakítót. Az algoritmusok teszteléséhez az algoritmus által generált eredményeket rádiós csatornán aztán soros porton a PC-re továbbítottuk, ahol lehetőség nyílik az eredmények kiértékelésére.

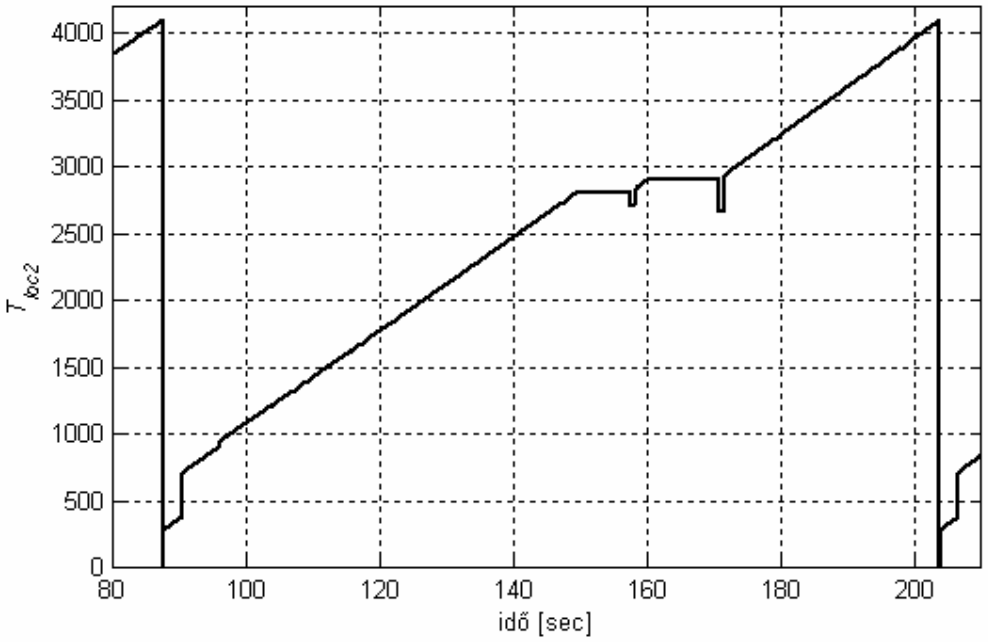
Ezen megkötések figyelembevételével egy olyan FA blokk kialakítására került sor, mely maximum öt rezonátorcsatornát tartalmaz. Ez azt jelenti, hogy a zajcsökkentő rendszer a periodikus zaj első öt harmonikusát képes elnyomni.

9. Zajcsökkentő rendszer teszteredményei

Ebben a fejezetben az elkészült zajcsökkentő rendszerek teszteredményei találhatóak. Mivel a szinkronizáció kulcsfontosságú szerepet játszik, ezért az azzal kapcsolatos teszteredményeket külön alfejezetben tárgyaljuk.

9.1. Szinkronizáció vizsgálata

Az elosztott mintavételezési helyeken a mintavételezési időpontok összehangolása a zajcsökkentő rendszer stabilitása szempontjából fontos. A szinkronizációs algoritmus azon alapszik, hogy amennyiben periodikus működésű a rendszerünk, akkor egy referencia node üzemi periódusian belül – ami esetünkben egy mintavételi időköz – meghatározott időpontban elküldött üzeneteknek szintén meghatározható állandó időpontban kell megérkezni egy másik állomás mintavételi időpontján belül, amennyiben a két rendszer szinkronizált, és az üzenettovábbítási idő állandó. Ez látható az 5.2. ábrán. A szinkronizáció biztosítása az üzemi periódusok, tehát a mintavételi frekvencia változtatásával lehetséges. A 9.1. ábrán egy olyan rendszer mérési eredménye látható, amelyben egy referencia mote folyamatosan szinkronizációs üzeneteket küld, melyet egy másik állomás fogad. A rendszer szinkronizálatlan, ezért a két rendszer mintavételi időpontjai elcsúsznak egymáshoz képest.



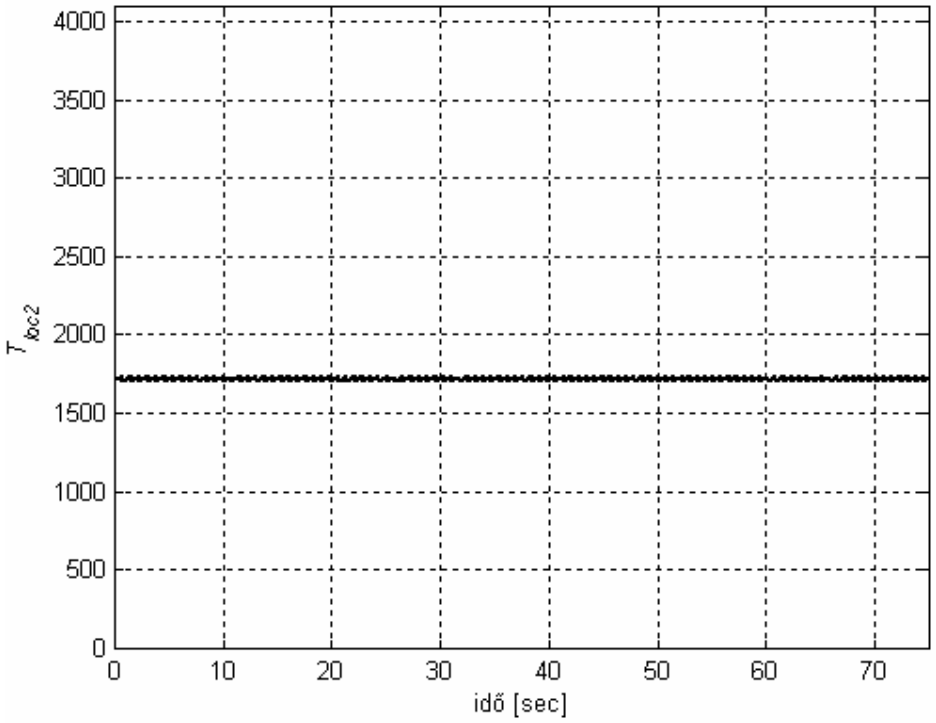
9.1. ábra. Szinkronizálatlan állapot

A mintavételi időpontok elcsúszása úgy detektálható, hogy a 5.2. ábrán T_{loc2}-vel jelölt érték folyamatosan változik. Az ábrán T_{loc2} a mintavételezés időzítésére használt időzítőből kiolvasott érték, így egy egység megfelel a mote-okon található mikrokontroller órajelének periódusidejével, ami körülbelül $T_{\text{clk}} = 0.136$ μsec. Látható, hogy a grafikonban több helyen ugrás található. Ez azért van így, mert amennyiben az üzenet olyankor érkezik, amikor a processzor terhelt, akkor az üzenetről szóló értesítés nem tud egyből érvényre jutni. A. 9.1. ábra alapján meghatározható az adott mote órájának elcsúszása, vagy driftje a referenciához képest. A drift ugyanis megegyezik a görbe meredekségével, mely például a 100 sec és 140 sec közötti egyenes szakasz alapján meghatározható:

$$\rho = \frac{(2469 - 1079) \cdot T_{clk}}{140 - 100} = 4.7 \cdot 10^{-6} = 4.7 \text{ ppm} \qquad (9.1)$$

Ez megegyezik a két mote órajel-frekvenciájának hibájával.

Szinkronizált állapot eléréséhez T_{loc2} = állandó= T_{loc2_ref} biztosítása szükséges. Ez az 5. fejezetben ismertetett algoritmus segítségével történik, melynek teszteredménye a 9.2. ábrán látható.

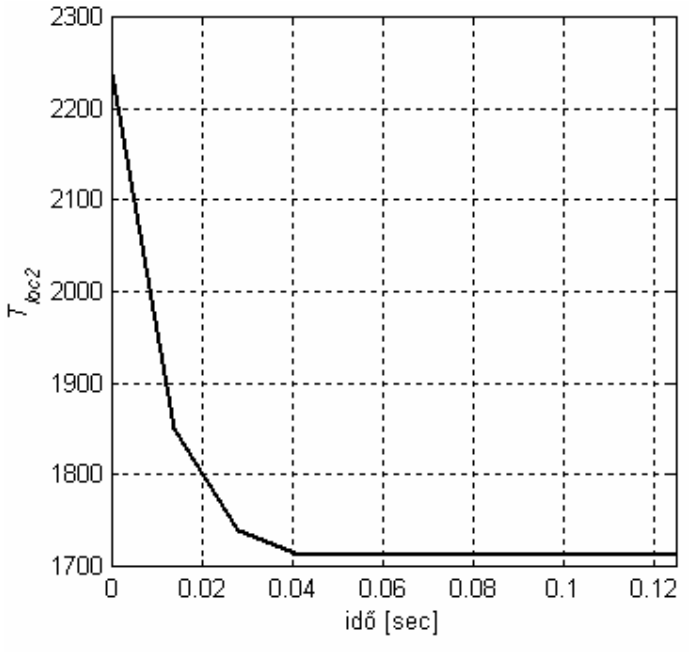


9.2. ábra. Szinkronizált állapot

Megállapíthatjuk, hogy a T_{loc2} igen jó közelítéssel tart egy állandó értéket, melynek megválasztása a 9.1. ábra alapján történt. A választott érték $T_{loc2\_ref} = 1700$, ugyanis ez körülbelül a 95 sec és 150 sec közötti lineáris szakasz közepén található. Tapasztalatok

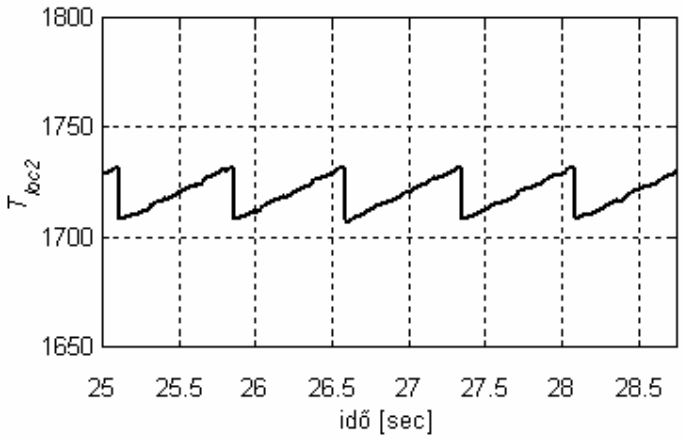
szerint a szinkronizáció azon esetben nem működik kifogástalanul, amennyiben olyan intervallumban választjuk meg a követendő értéket, ahol ugrások vannak a 9.1. ábrán látható görbén. Ilyen például $T_{loc2\_ref} = 2700$ eset.

A szinkronba kerülés folyamata a 9.3. ábrán látható. A beállás 40 msec alatt megtörténik.



9.3. ábra. Beállási szakasz

A 9.4. ábrán a 9.2. ábra egy kinagyított részlete látható.

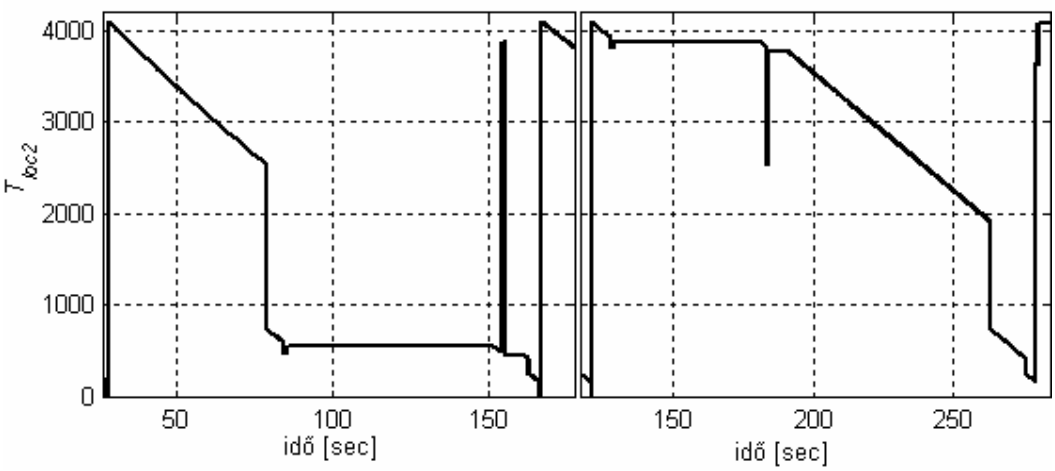


9.4. ábra. Tartási szakasz

Megállapíthatjuk, hogy a szinkronizáció során nem teljesül pontosan a $T_{loc2} = T_{loc2\_ref}$ feltétel. Ennek oka az, hogy a (7.1.) képlet alapján működő algoritmus nem észleli, ha T_{loc2} kevesebb, mint 32 egységre csúszik el a referencia értéktől, hiszen egész osztást

alkalmazunk. Ennek következtében a $(N_l - N_a) / 32$ kifejezés csak akkor ad nullától különböző értéket, tehát csak akkor történik beavatkozás, ha $|N_l - N_a| \geq 32$. Mindez azonban nem okoz számottevő problémát, ugyanis a szinkronizált mote-ok mindegyike $32 \cdot T_{clk} = 4.3$ μsec határon belül mozoghat a referencia időponthoz képest. Ez például egy 2 kHz-es mintavételi frekvencia, tehát 500 μsec mintavételi időköz esetén elhanyagolható. A 9.4. ábrán a meredek lefutó élek jelölik azokat az időpontokat, amikor az adott mote mintavételi időpontja legalább 32 órajellel elcsúszik a referenciához képest. Ilyenkor a mintavételi frekvencia változása miatt újra bekerülünk a szinkronizált állapotba.

A fenti mérések az egyszerű, zajminták gyűjtését és továbbítását végző hálózatban történtek. A rezonátorokat futtató mote-ok esetén azonban a 9.1. ábrán látható grafikon módosul, mint ahogyan a 9.5. ábrán megfigyelhető.

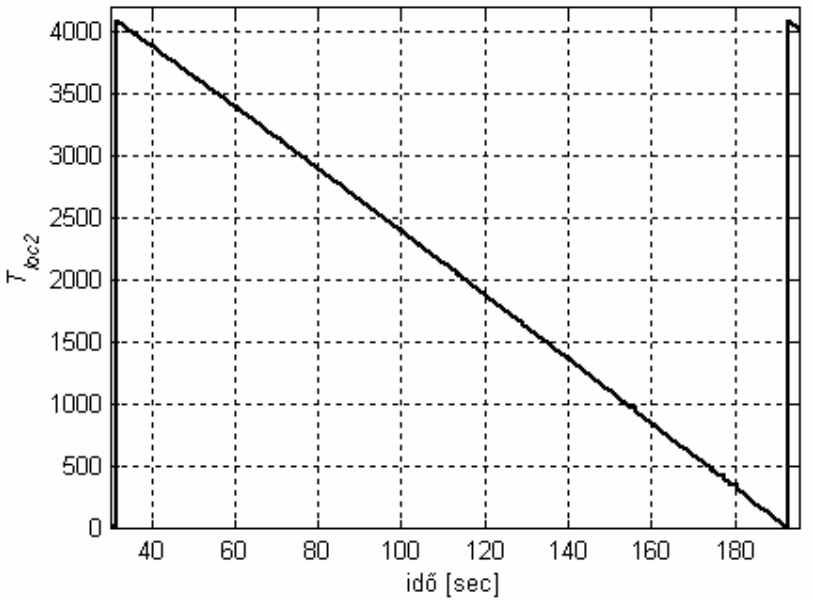


a) ábra b) ábra

9.5. ábra. T_{loc2} rezonátort futtató mote-oknál

Az a) és b) ábra kétféle alapharmonikus frekvenciához tartozik. Mivel a frekvencia függvényében változik a rezonátorszám, és ezzel a rezonátorok számításával foglalkozó taszk futási ideje, ezért tér el a két ábra egymástól. Sajnálatos módon a tesztelések során nem sikerült olyan T_{loc2_ref} munkapontot találni, melyben minden rezonátorszám esetén biztonságosan beállt volna T_{loc2} a megkívánt értékre. Emiatt változtatni kellett az eddigi koncepción, miszerint a referencia csomag megérkezésekor kiolvassuk a mintavételezést időzítő számláló értékét, és ez alapján történik a szinkronizáció. Az új rendszerben a csomag megérkezését jelző megszakítás rutinban történik a számláló kiolvasása, a feldolgozás viszont az eredeti helyen zajlik, mivel az már kevésbé időkritikus szakasz. Az ehhez az elrendezéshez tartozó mérés a 9.6. ábrán látható. Mivel a megszakítást csak a megszakításrutinok, illetve az atomikusan végrehajtandó parancsok blokkolhatják, melyek viszont igen rövidek, így ez a mérési módszer igen jól alkalmazható a csomagok érkezésének pontos érzékelésére. Láthatóan a mintavételi időpontok ebben az esetben is egyenletesen távolodnak egymástól, hiszen a $T_{loc2}(t)$ egy egyenes. Mivel ezen a grafikonon nincsenek jelentős szakadások, így a T_{loc2_ref} szinte

bármekkora értékre megválasztható. Célszerű a számlálóból kiolvasható értékek által meghatározott intervallum közepére választani.



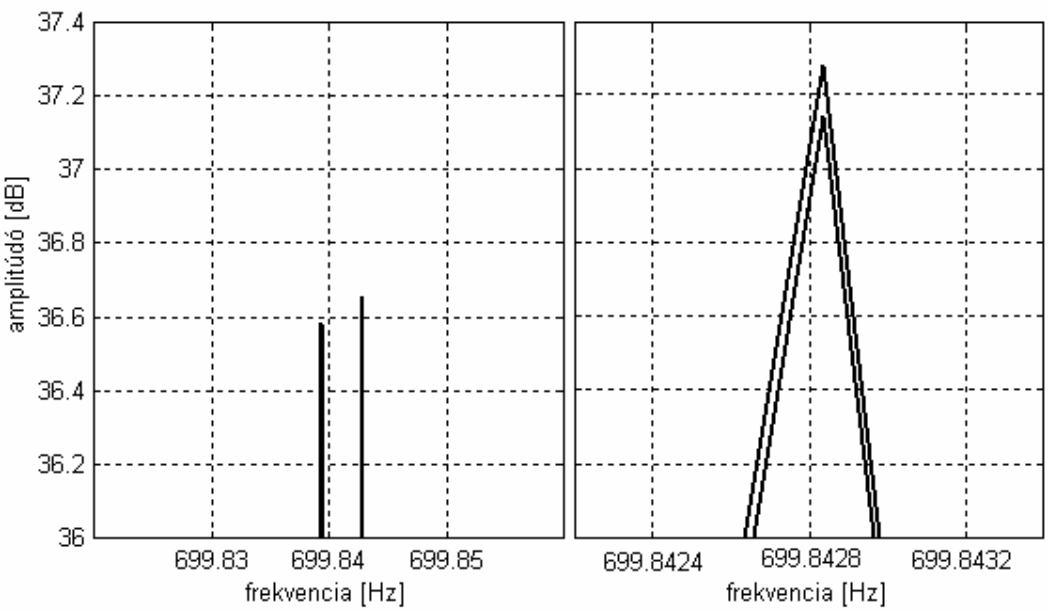
9.6. ábra. Csomag érkezésének javított technikájú érzékelése

Látható, hogy az új módszerrel igen pontosan lehet a csomagok érkezésének idejét érzékelni. Annak az oka, hogy nem próbáltuk ily módon megoldani a csomagok érkezésének észlelését az egyszerű mintagyűjtő rendszerben is az, hogy ez az operációs rendszer változtatásával jár, mely hosszadalmas munka.

Az elosztott rezonátoros rendszeren végzett, szinkronizált esetre vonatkozó mérések szinte teljesen hasonlítanak a mintagyűjtő rendszerben végzett mérésekre – hiszen a szinkronizációs algoritmus megegyezik – ezért azokról nem közlünk újabb ábrákat.

Az eddigi mérési eredmények arról tanúskodnak, hogy a szinkronizáció megvalósítása sikeres volt. Az 5.1.1. alfejezetben leírtak alapján, amennyiben szinkronizálatlan forrásokból érkeznek egy periodikus jel mintái, akkor a két forrás adatait feldolgozva ugyanazon jel frekvenciáját a két esetben eltérőnek érzékeljük, szinkronizált esetben viszont a két forrás mintáinak feldolgozásakor megegyezik a jel frekvenciája. Ennek bizonyítására elvégzett mérés eredményeit láthatjuk a 9.7. a) és b) ábrákon, melyek közvetett bizonyítékként szolgálnak a szinkronizáció sikeres végrehajtásáról. A mérés során egy 700 Hz frekvenciájú szinuszos hangot adtunk ki egy jelgenerátor segítségével, és első esetben két szinkronizálatlan mote, második esetben ugyanazon két mote szinkronizált állapotban küldte mikrofonjaik kimenetéről származó mintákat a bázisállomás felé, mely soros porton továbbította azt egy PC-hez, ahol az adatok feldolgozása történt. Feldolgozás során a jelek spektrumát állítottuk elő FFT segítségével 2^{20} db mintából. Azért szükséges ekkora mennyiségű minta, mivel a két mote mintavételi frekvenciájának eltérése csekély, az FFT által szolgáltatott spektrum felbontása pedig egyenesen arányos a feldolgozott minták számával. Ahhoz tehát, hogy

a kis mértékű frekvenciabecslési hibát érzékelni tudjuk, nagy mennyiségű mintát kell feldolgozni.



a) ábra. Szinkronizálatlan eset b) ábra. Szinkronizált eset

9.7. ábra. Szinkronizáció hatása a frekvencia becslésre

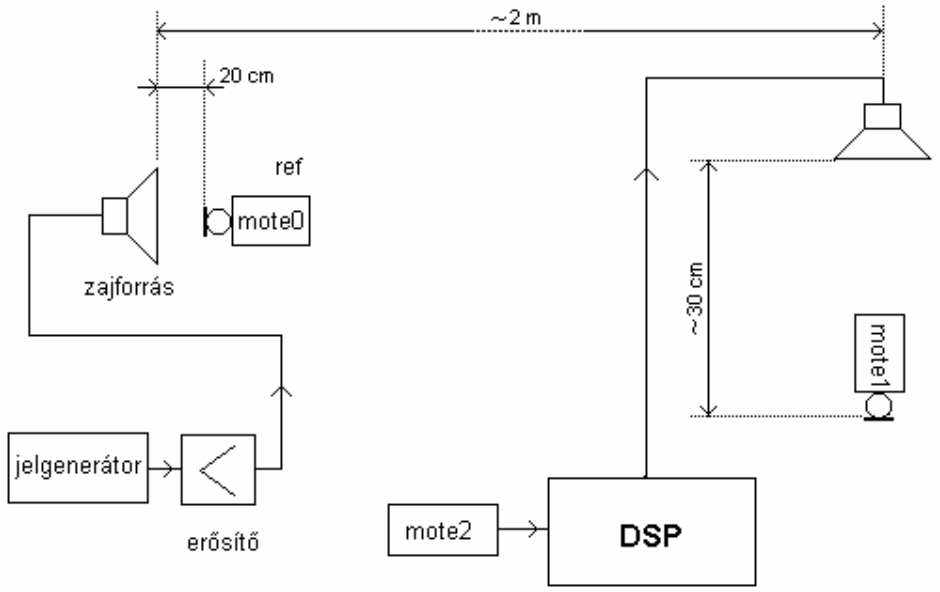
A fenti ábrákon látható, hogy szinkronizálatlan esetben ugyanazon jel frekvenciáját a két különböző csatornán érkező adatok alapján megmérve két különböző értéket kapunk. Ezzel szemben szinkronizált esetben a két csatornából érkező adatok alapján elvégzett frekvenciamérés mindkét csatorna esetén megegyező frekvenciát ad eredményül, tehát a szinkronizáció sikeres volt. A mért frekvenciaértékek azért nem pontosan 700 Hz-nek adódnak, mint ahogyan várnánk, ugyanis a jelet előállító jelgenerátor, és a mintavételezésért felelős időzítők órajel-generátorai sem teljesen pontosak, és a számítások elvégzése azok névleges értékei alapján történt.

9.2. Zajcsökkentés adatgyűjtő hálózat felhasználásával

Az adatgyűjtő szenzorhálózat felhasználásával egycsatornás LMS alapú, valamint kétcsatornás rezonátoros zajcsökkentő algoritmus megvalósítására került sor.

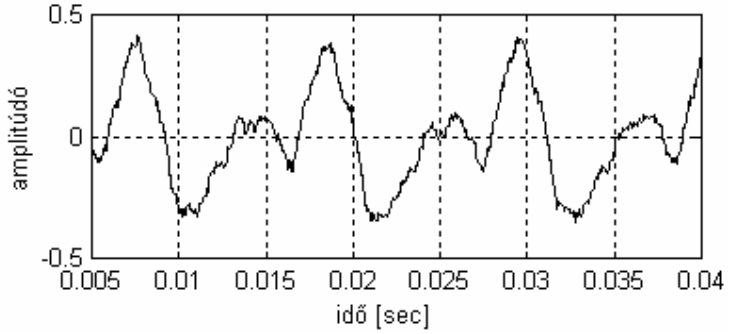
9.2.1. Zajcsökkentés LMS algoritmussal

Az LMS alapú zajcsökkentő rendszer tesztelése a 9.8. ábrán látható elrendezésben történt.



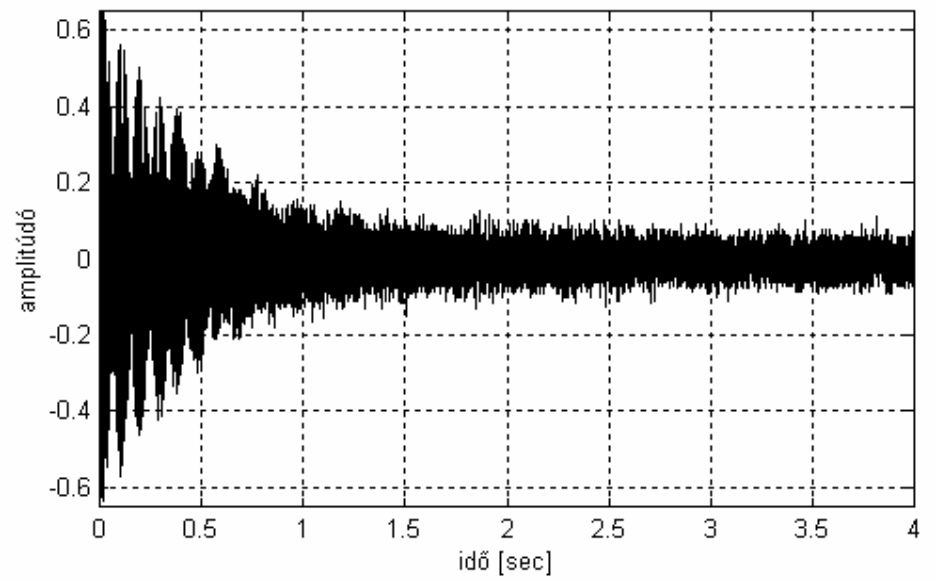
9.8. ábra. LMS alapú zajcsökkentő algoritmus tesztkörnyezete

A nullás azonosítójú $mote_0$ szolgáltatja a referenciajelet az algoritmus számára. Referenciajelként a zajforrásból érkező zaj szolgál. A $mote_1$ a zajérzékelő szerepet látja el, $mote_2$ pedig a bázisállomás. A rendszert többféle jelalakú periodikus zaj elnyomására teszteltük. Az itt bemutatásra kerülő mérési eredmények felvételekor a zajjel előállítása, melynek időfüggvénye az 9.9. ábrán látható, egy zajgenerátor segítségével történt. A beállítások alapján 511 hosszúságú szekvenciánként ismétlődnek az álvéletlen számok. Ennek eredményképpen az ábrán látható periodikus jel alakult ki, melynek hangja eléggé hasonlít egy forgógép által keltett zajhoz. Frekvenciája 91 Hz, mely a zajgenerátor mintavételi frekvenciájának és az álvéletlen szekvencia hosszának eredményeként alakult ki. A mote-ok 1.8 kHz-es mintavételi frekvenciával üzemelnek.



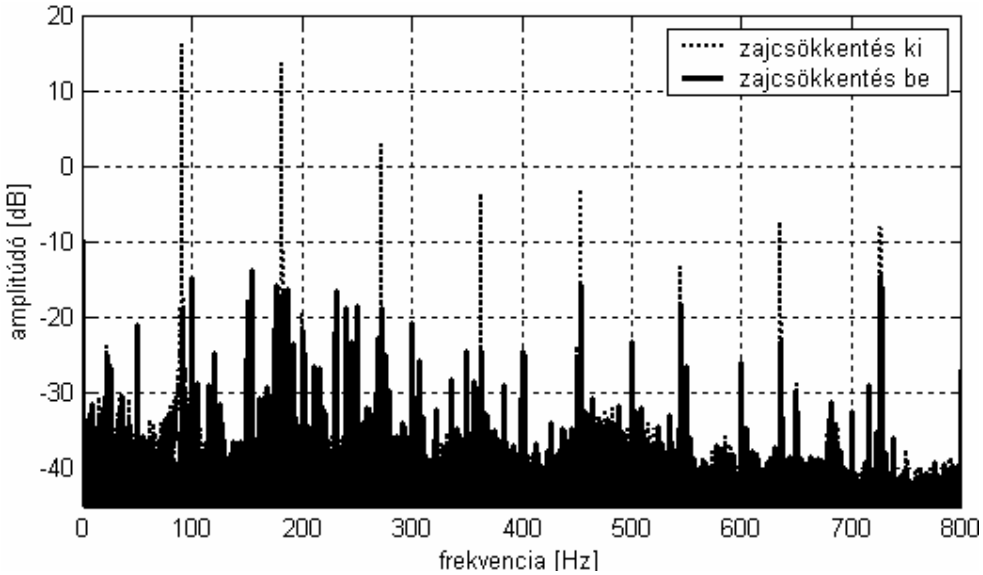
9.9. ábra. Elnyomandó zaj jelalakja

A 9.10. ábrán látható tranziens beállási szakasz alapján megállapítható, hogy a periodikus jelre történő beállási idő körülbelül 1,5 másodperc. A mérések elvégzése a zajérzékelő mikrofonhoz igen közel helyezett mikrofon segítségével történt.



9.10. ábra. Beállási szakasz

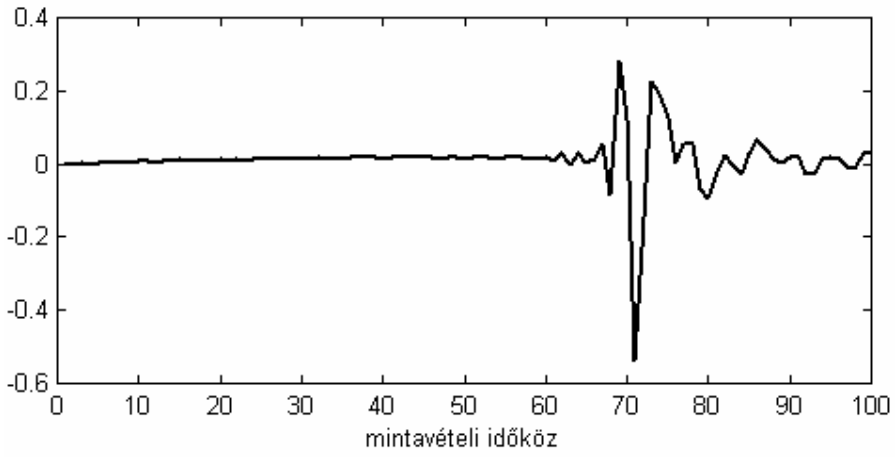
A rendszert állandósult állapotban jellemző mérés a 9.11. ábrán látható.



9.11. ábra. LMS alapú zajcsökkentő rendszer elnyomása állandósult állapotban

Az ábrán pontvonallal kikapcsolt zajcsökkentő rendszerrel mért zaj spektruma, míg folytonos vonallal a bekapcsolt zajcsökkentő rendszer esetén mért spektrum látható. Megállapíthatjuk, hogy a 91 Hz-es alapharmonikusra nézve körülbelül 35 dB-es, és a negyedik harmonikusig több mint 20 dB-es elnyomást értünk el harmonikusonként.

Habár az LMS alapú zajcsökkentő rendszerek képesek sztochasztikus jelek elnyomására is, ennek tesztelésére a 9.8. ábrán látható zajcsökkentő rendszerben nem volt lehetőségünk. Ennek oka, hogy a referenciajelek gyűjtésére szolgáló mote mikrofonja és a DSP közötti jelút késleltetése igen nagy. A mote_0 jeleit használjuk referenciaként az FxLMS algoritmusban található $W(z)$ szűrő bemeneteként – lsd. 2.4. ábra –, ennek alapján állítjuk elő a beavatkozó jelet. Ez azt jelenti, hogy ezek alapján becsüljük a zaj értékét. Amennyiben a zajforrás és a zajérzékelő mote közötti késleltetés kisebb, mint a referenciajel továbbítása miatt fellépő, valamint a beavatkozó hangszórók által reprezentált késleltetések összege, akkor a rendszer fehérzaj elnyomására alkalmatlan. Ez a következőképpen magyarázható. A zajforrás és a zajérzékelő mikrofon közötti út átviteli függvényét jelölje $P(z)$. Ez jellemezhető egy frekvenciafüggő amplitúdóval és késleltetéssel. Amennyiben a referenciajelnek a zajforrástól a jelfeldolgozó algoritmusig vett késleltetése nagyobb, mint a $P(z)$ késleltetése, akkor úgy kellene becsülni a zajt, hogy még nem érkezett róla referencia információ, akauzális rendszer alakul ki. Ráadásul az előállított beavatkozójel is csupán a $W(z)$ szűrő, a DA. átalakító, valamint a beavatkozó hangszóró és mikrofon közötti akusztikus út késleltetésével jelenik meg. Periodikus jelek esetén ez nem probléma, hiszen mintái korreláltak, fehérzaj esetén viszont nem lehet becsülni a mintákat más minták segítségével. Ennek következtében a zajforrásként szolgáló hangszórót legalább olyan messzire kell tenni a mikrofontól, hogy legalább akkora késleltetés legyen közöttük, mint amekkora a referenciaforrás mote és a $W(z)$ szűrő miatt megjelenő késleltetés, valamint a DSP digitális kimenete, és a zajérzékelő mikrofon között létrejövő késleltetés összege. Ezen késleltetés meghatározására azt használjuk ki, hogy ez a késleltetés gyakorlatilag megegyezik egy olyan másodlagos út átviteli függvényével, melyben a mikrofon és a hangszóró olyan távolságra egymástól, mint a referencia mote a zajforrástól, hiszen ebben az elrendezésben ugyanúgy megjelenik a referencia jel továbbításának késleltetése és a beavatkozó jel által elszenvedett késleltetés. Ez identifikációval meghatározható a 2.14. ábrán látható elrendezéssel. Egy mérési eredmény látható a 9.12. ábrán, ahol a 2.14. ábrán $W(z)$-vel jelölt FIR szűrő együtthatóit ábrázoltuk, mely megegyezik a szűrő imlulzusválaszával.



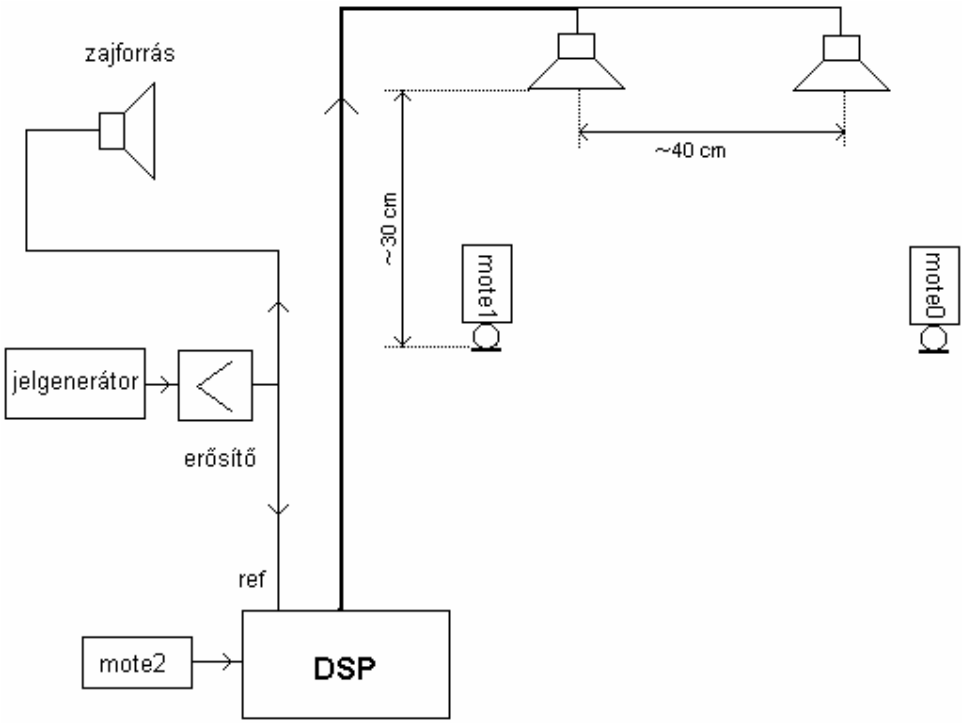
9.12. ábra. Identifikált átviteli függvény impulzusválasza

A késleltetés mértékére jó becslést ad az impulzusválaszban látható domináns csúcs nulladik mintavételi időponttól mért távolsága. Ez esetünkben 1.8 kHz-es mintavételi frekvencián 71 mintavételi időköz, ami körülbelül 40 ms. Amennyiben a hang terjedési sebessége 340 $^{m}/_{s}$, akkor ehhez a késleltetéshez a zajforrást legalább 13.5 m-re kell tenni a zajforrást a zajérzékelő mikrofontól. Ezt a laboratórium méretei nem tették lehetővé. A 13.5 m-es távolság számításakor ráadásul még nem is vettük figyelembe a $W(z)$ adaptív szűrő késleltetését.

Habár a referenciajel szerzése tötrénhetett volna közvetlenül a zajforrásként szolgáló jelgenerátor kimenetéről AD-átalakítóval, melynek késleltetése körülbelül 10 minta, ezt nem alkalmaztuk, hiszen gyakorlatban a sztochasztikus zajok forrásai általában nem adnak ki elektromos jelet (pl. szél, beszéd ...). Az LMS algoritmus működőképességét zaj referenciajel esetén pedig az identifikációs folyamat bizonyítja, hiszen az identifikációhoz zaj gerjesztést alkalmazunk, mely megegyezik a referenciajellel (lásd: 2.14. ábra, $\mathbf{x}_n$).

9.2.2. Zajcsökkentés rezonátoros struktúrával

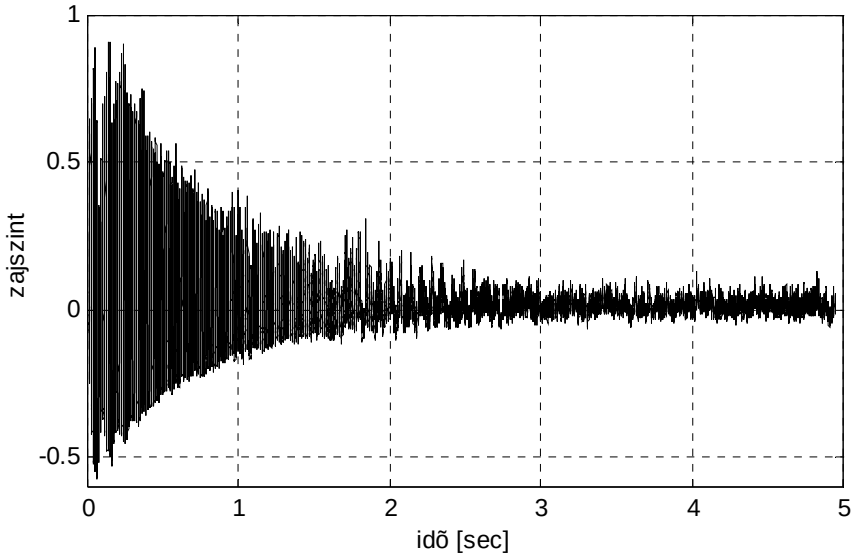
Az LMS alapú zajcsökkentő rendszerhez hasonlóan ebben a felépítésben is két mintagyűjtő mote szerepel, melyek 1.8 kHz-es mintavételi frekvenciával üzemelnek. A mérési elrendezés a 9.13. ábrán látható. Ebben az esetben egy kétcsatornás rendszert építettünk ki, ahol a zajérzékelést a mote-ok valósítják meg.



9.13. ábra. Rezonátor alapú zajcsökkentő rendszer

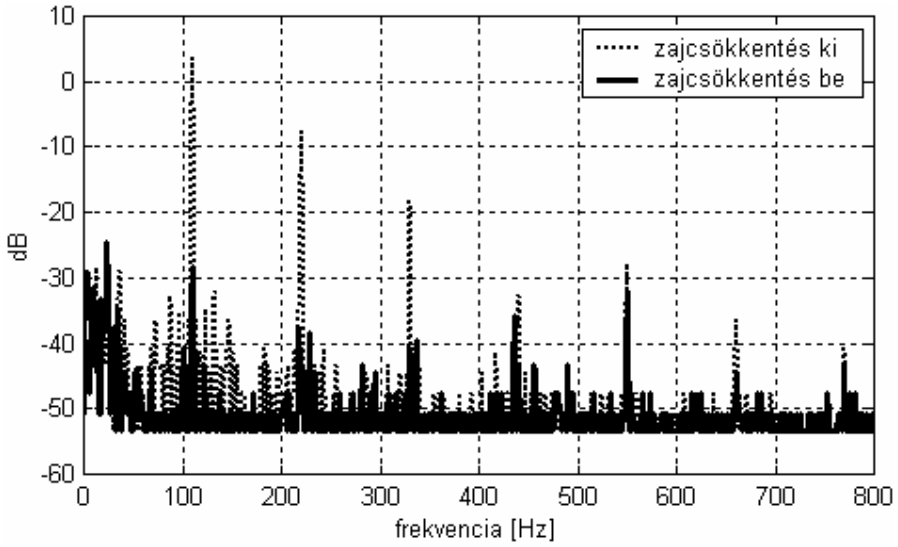
A zajérzékelő mote-ok jeleit a DSP a $mote_2$ bázisállomás segítségével fogadja, a referenciajelet pedig a hozzá kapcsolódó kodek segítségével mintavételezi. A referenciajel periodikus jelek esetén egyébként sokszor elérhető ilyen formában, tehát úgy, hogy nem magát az akusztikus zajt tekintjük referenciának. Ilyen például, ha egy forgó gép zajának csökkentésekor a tengelyére szerelt jeladó kimenetét, vagy transzformátor zajának csökkentésekor a hálózati feszültséget használjuk referenciaként. Elvileg olyan megoldás is lehetséges, melyben a zaj alapharmonikus frekvenciájának numerikus értékét ismerjük.

A 9.13. ábrán látható elrendezés egy olyan gyakorlati esetet szimulál, melyben a hangszórók egy szék fejtámláján hátul, a zajérzékelő mote-ok pedig fülmagasságban helyezkednek el. A 9.14. ábrán látható módon a beállási idő körülbelül 2 másodperc. Ez kissé több, mint az egycsatornás LMS algoritmus esetén, de ebben az elrendezésben két pozícióban kell zajelnyomást elérni.



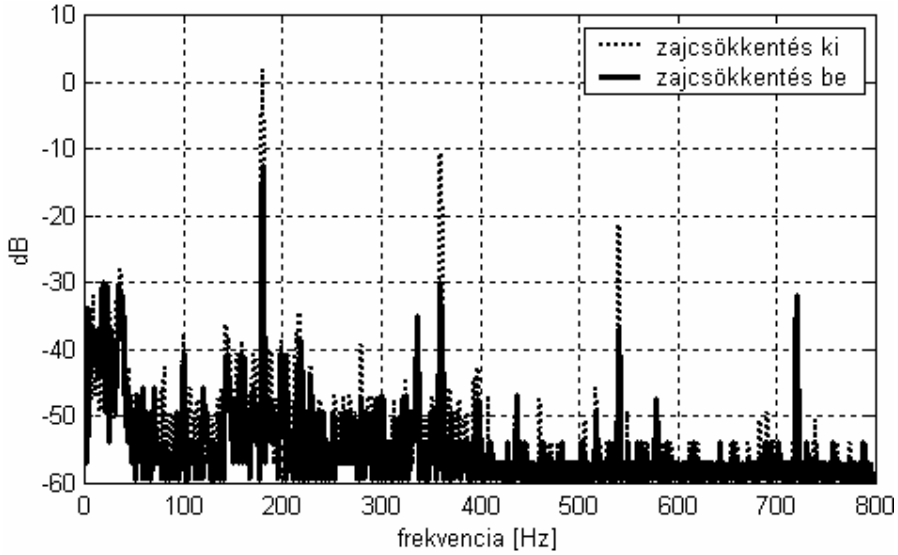
9.14. ábra. Beállási szakasz

A 9.15. ábrán az egyik zajérzékelő mote-nál mért spektrum látható. Az elnyomás alapharmonikusra és második harmonikusra nézve valamivel több mint 30 dB.



9.15. ábra. Elnyomás a zajérzékelő mikrofonnál

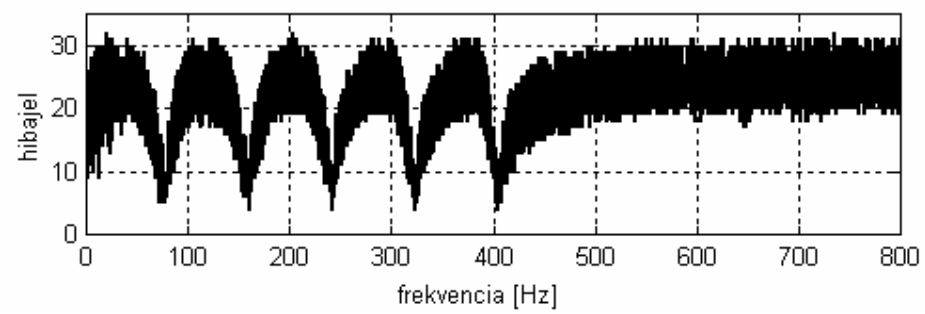
A zajelnyomó rendszer a mikrofonok helyén igyekszik a zajszintet minimalizálni. A 9.16. ábrán egy olyan mérés eredménye látható, melyben a zajszint mérése nem a zajcsökkentő rendszer valamely zajérzékelő mikrofonjánál történik, hanem ebben az esetben például a két zajérzékelő mikrofon között. Láthatóan így csökken a zajelnyomás mértéke, de az első három harmonikus esetén így is megközelítően 15-20 dB-es elnyomás tapasztalható.



9.16. ábra. Elnyomás a zajérzékelő mikrofonok között

9.3. Zajcsökkentés elosztott rezonátoros struktúrával

A rezonátoros rendszer ismertetésénél láthattuk, hogy a rezonátoroknak mote-okon történő implementálása igen sok nehézséget rejt, hiszen ez a platform nem kifejezetten alkalmas bonyolult jelfeldolgozói eljárások megvalósítására. Ennek következtében takarékoskodni kell az ábrázolási pontossággal és műveletekkel, amely sokszor csak az algoritmus minőségi jellemzőinek rovására lehetséges. Kérdéses ezért tehát, hogy a megvalósított algoritmus mennyire elégíti ki a vele szemben támasztott követelményeket. Ennek meghatározására az FA blokkok átviteli függvényét vizsgáltuk. A vizsgálat során léptetett frekvenciájú szinuszos gerjesztésre adott választ mértük, mely az átviteli függvényt határozza meg. A 9.17. ábrán az FA blokk hibajelére vonatkoztatott átvitele látható.



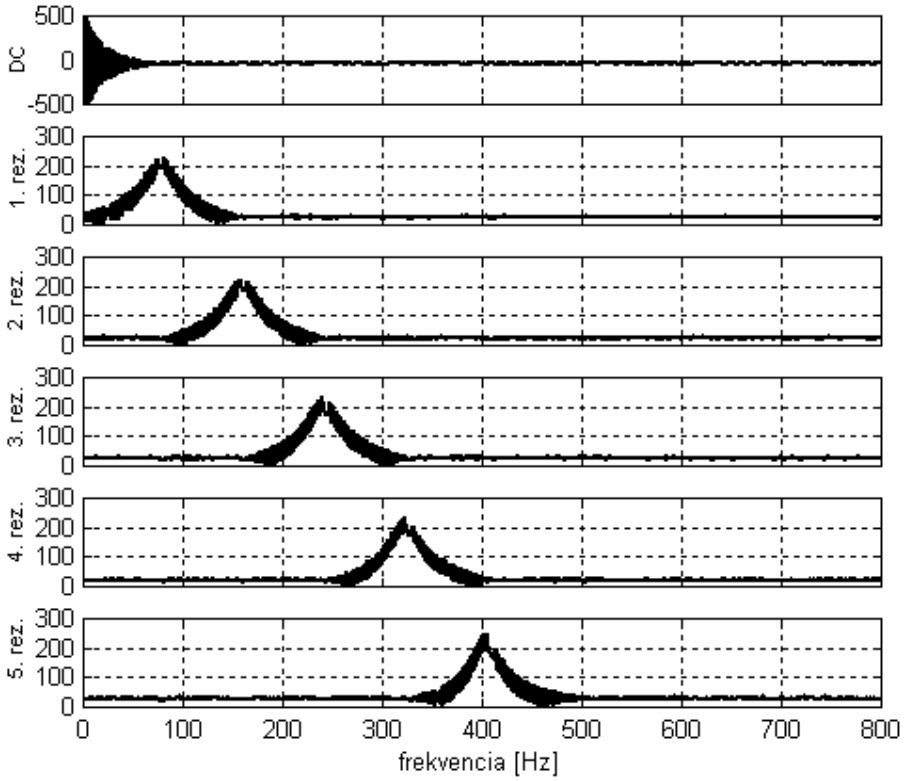
9.17. ábra. Hibajelre vonatkoztatott átvitel

Az FA-ban 5 rezonátor működött 80 Hz-es alapharmonikus frekvencián. Az ábra tulajdonképpen nem a valódi átviteli függvényt adja meg, de jól közelíti azt. A mérése a következő módszerrel történt: a léptetett frekvenciájú szinuszos gerjesztés hatására adott hosszúságú intervallumokban mértük a hibajel maximumát. Látható, hogy a rezonátor frekvenciákon a hibajel csökken, bár nem éri el a nullát.

Másik jellemző átviteli függvény a bemenetnek az egyes rezonátor állapotváltozókra vonatkoztatott átvitele. Ez a 9.18. ábrán látható.

Megjegyezzük, hogy az átviteli függvény mérés itt sem teljesen korrekt elnevezés. A fenti karakterisztikák mérése a következő módszerrel történt: a léptetett frekvenciájú szinuszos gerjesztés hatására adott hosszúságú intervallumokban mértük az egyes rezonátorok állapotváltozóinak átlagértékét. A grafikonok az állapotváltozók átlagértékének abszolút értékét ábrázolják.

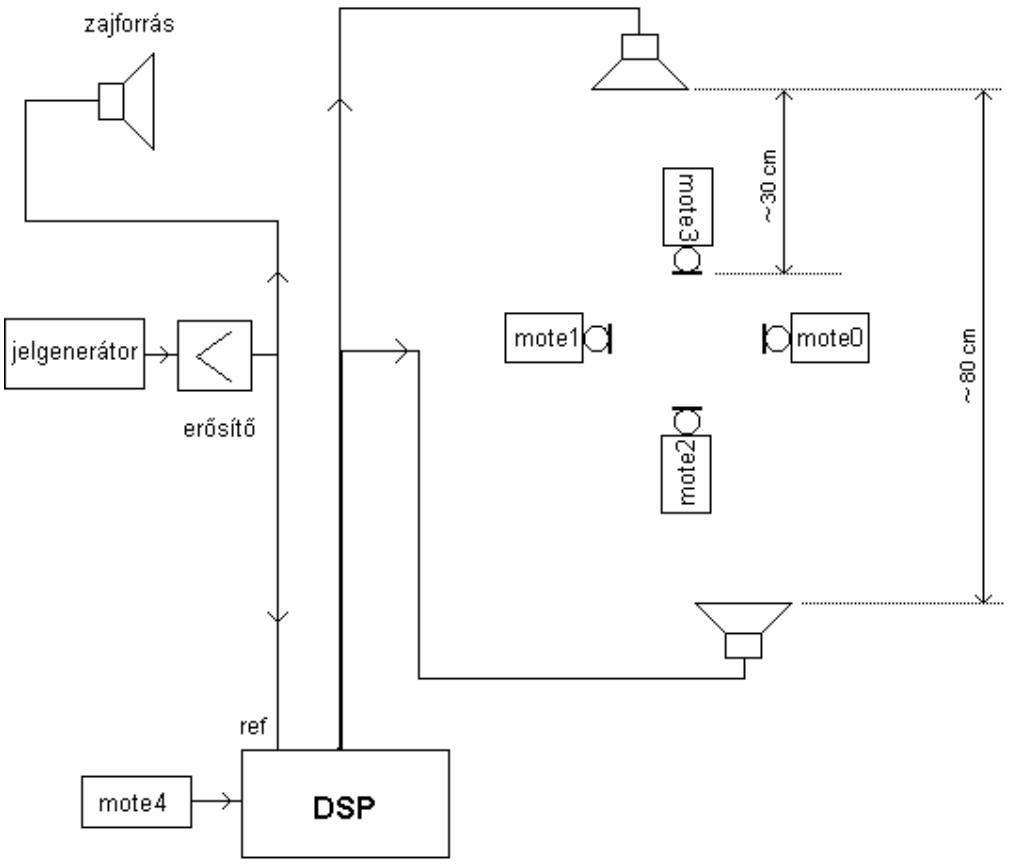
Láthatjuk, hogy amint a frekvencia közeledik a rezonátorfrekvenciákhoz, a rezonátor állapotváltozók értéke nő, és rezonátorfrekvenciákon elérik maximumaikat, mint ahogyan az várható, habár csupán jellegre egyezik a rezonátorok elméleti sinc függvényű átviteli karakterisztikájával.



9.18. ábra. Rezonátorokra vonatkoztatott átviteli függvény

Összességében megállapítható, hogy az FA blokk mote-okon történő implementálása ugyan nem teljesen adja meg az elméletileg elvárt eredményeket, de jól közelíti azokat. Ennek oka az is, hogy az átviteli függvények mérése nem teljesen precíz módon zajlott le, ugyanis a mote-okon található információk nehezen hozzáfárhetőek, hiszen a mote-ok nem tartalmaznak pl. DA-átalakítót, ami a vizsgált jelek hálózatanalizátorral való feldolgozását lehetővé tenné. A jeleket csupán rádiós kommunikációval lehetséges továbbítani, mely külön tesztkörnyezet kiépítését teszi szükségessé. A tesztkörnyezet bonyolultsága, és a mérés szükséges pontossága között jó kompromisszumnak bizonyultak a fenti megoldások. A mérések elvégzéséhez tehát bizonyos egyszerűsítéseket alkalmaztunk, melyek a mérés pontosságát rontják, de a rendzser működőképességét jól mutatják.

Az FA-t futtató mote-ok segítségével felépített zajcsökkentő rendszerrel végzett kísérlet teszt elrendezését a 9.19. ábra mutatja.



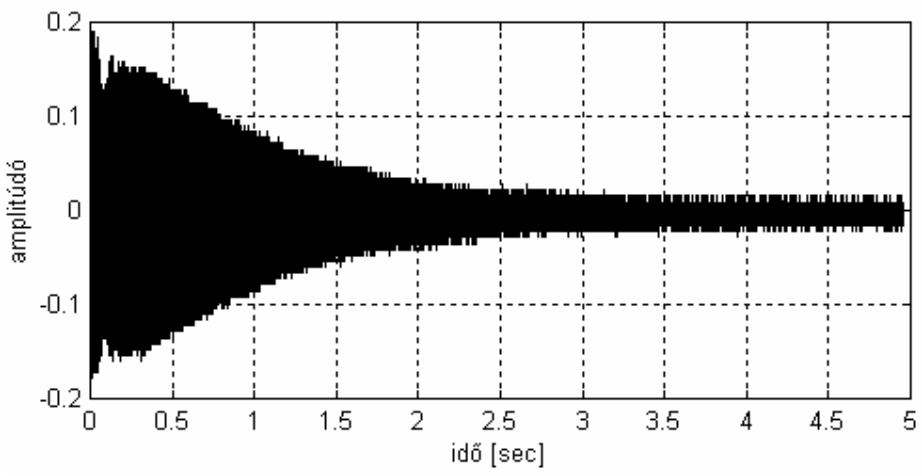
9.19. ábra. Tesztkörnyezet elosztott rezonátoros zajcsökkentő rendszerhez

Az, hogy a mote-ok nem a mintákat továbbítják a hálózatban, lehetővé teszi a mote-ok számának növelését adott sávszélesség mellett. Ebben a rendszerben négy mote segítségével érzékeljük a zajt változatlanul 1.8 kHz-es mintavételi frekvencia mellett. Az elvi lehetőség azonban fennáll a mote-ok számának növelésére. Maximális korlátot az szabhat, hogy a mote-ok számának növelésével a 8.3. ábrán T_p-vel jelölt hálózati periódusidő nő, mivel csak így lehet a megnövekedett mennyiségű adatot továbbítani. T_p növekedése két problémát okoz. Egyrészt a DSP ritkábban kap információt a zajszintről, ami a zajcsökkentő rendszer lassulását okozza. Másik probléma, hogy amennyiben a frekvencia T_p periódusidőhöz képest gyorsan változik, akkor a rezonátorok szinkronizálása nem lesz elegendően pontos. A rendszerben felhasználható zajérzékelő mote-ok számát tehát maga a zaj jellege is meghatározza.

Mivel a DSP csupán sztereo DA kimenettel rendelkezik, így a rendszerben csupán két hangszórót lehet felhasználni. Amennyiben a hangszórók száma nem éri el a mikrofonok számát, úgy általános esetben nem biztosítható a mikrofonoknál a teljes elnyomás. A rendszer ilyen esetben az érzékelőkből származó eredő zajszint minimalizálására törekszik. A zajérzékelést ebben az elrendezésben $mote_0$…$mote_3$ látja el, $mote_4$ szolgál bázisállomásként. A referenciajelet a DSP a hozzá kapcsolódó kodek segítségével mintavételezi. Ebben az esetben nem is lenne lehetőség a mote-ok

segítségével gyűjteni a referenciajelet, ugyanis azok működéséhez már eleve szükséges egy referenciafrekvencia megadása.

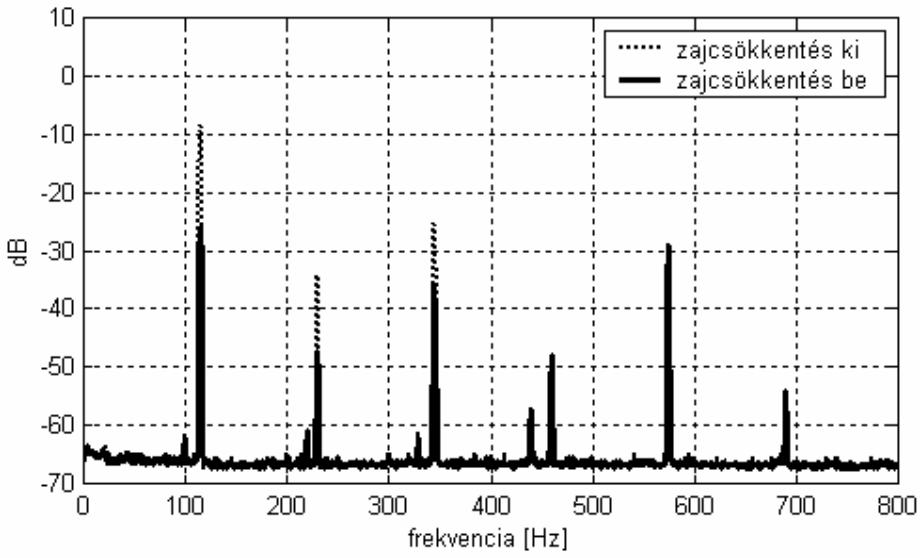
A 9.20. ábrán a rendszer beállási tranziense látható. A beállási idő ebben az esetben körülbelül 2 másodperc. Ez nem marad el lényegesen az előzőekben bemutatott rendszerek gyorsaságától.



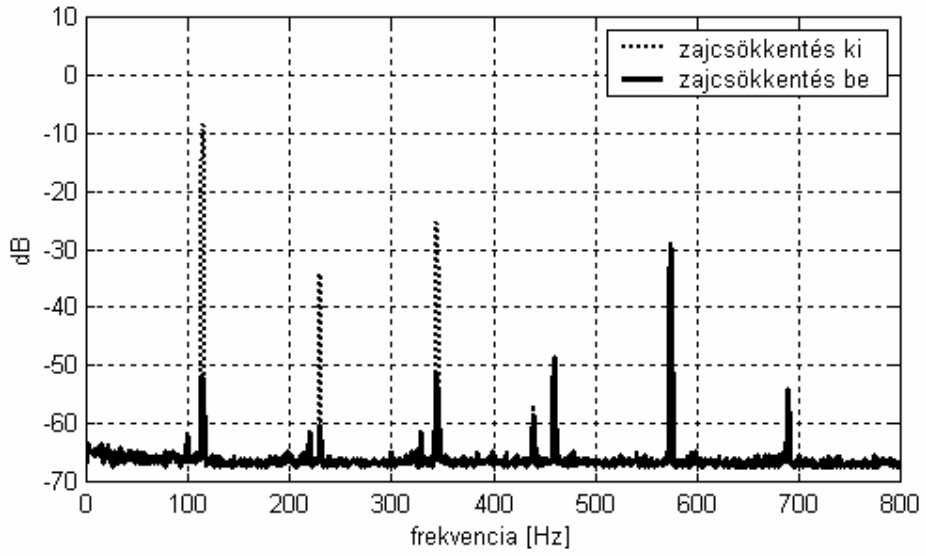
9.20. ábra. Beállási szakasz

A rendszer állandósult állapotban tapasztalható hatékonyságát ebben az esetben is két pozícióban mértük. Első esetben közvetlenül az egyik zajérzékelő mote mellett, második esetben pedig a mote-ok által meghatározott négyszög középpontjában. A mérési eredmények a 9.21. és a 9.22. ábrán láthatók. Zajjelként egy 115 Hz-es háromszögjelet alkalmaztunk.

Megállapíthatjuk, hogy közvetlenül a mote-nál mért zajcsökkentés hatásfoka rosszabb, mint a mote-ok között. Ennek oka, hogy amennyiben kevesebb hangszóró található a rendszerben, mint ahány mikrofon, akkor nem biztos, hogy a zajérzékelő szenzorok pontjában minimális a zajszint, ugyanis az eredő zajszint minimalizálása a cél.



9.21. ábra. Zajérzékelő mote-nál mért zajszint



9.22. ábra. Zajérzékelő mote ok között mért zajszint

A négycsatornás rendszer esetén az eddig bemutatott rendszerekhez képest jóval nagyobb stabilitást tapasztaltunk abban az esetben, ha a mote-ok pozícióját kisebb-nagyobb mértékben megváltoztattuk. Egy- és kétcsatornás rendszereknél a mote-ok helyzetét körülbelül 15 cm-rel megváltoztatva a rendszer biztosan instabillá vált, míg a négycsatornás esetben egy mote pozíciójának akár 40 cm-es megváltoztatása sem okozott instabilitást.

10. Összefoglalás, kitekintés

Diplomamunkám keretében sikeresen megvalósítottam olyan zajcsökkentő rendszereket, melyekben a zajszint érzékelése szenzorhálózat segítségével történt. A feladat megoldása során megismerkedtem a különböző zajcsökkentő algoritmusok elméleti alapjaival, és azok gyakorlati alkalmazásának módszerével. A zajszint érzékelését megvalósító szenzorhálózattal kapcsolatban felmerülő szinkronizációs problémák kapcsán szükséges volt a szinkronizációval kapcsolatos alapismeretek elsajátítása, és azok gyakorlati alkalmazása. Mivel a szenzorhálózat elemeinek programozása egy TinyOS nevű operációs rendszerben történt, ezért alkalmam volt gyakorlatban is megismerni a beágyazott operációs rendszerek használatának előnyös és hátrányos szempontjait.

A dolgozatomban szerepel az LMS és rezonátor alapú zajcsökkentő algoritmusok elméleti összefoglalása, és az alapstruktúrák zajcsökkentő rendszerekben történő felhasználásának bemutatása. Mivel a zajcsökkentő algoritmusokban fontos szerepet játszik a rendszer visszacsatoló ágának minél pontosabb meghatározása, ezért sor került különféle identifikációs eljárások ismertetésére is.

A szenzorhálózat alkalmazása során felmerülő szinkronizációs problémák analizálásához szükséges elméleti alapok ismertetése után bemutattam négy elterjedt szinkronizációs algoritmust. Ezek kapcsán megismerhettünk néhány alapvető szinkronizációs eljárást és alapelvet.

A zajérzékelő hálózatban felmerülő szinkronizációs problémák elemzése, és a lehetséges megoldási módszerek vizsgálata után bemutattam a zajérzékelő hálózatra kifejlesztett szinkronizációs algoritmust, és annak implementálását egy valós rendszerben. Ezen rendszerben az állomások a zajmintákat továbbítják a jelfeldolgozást végző egység felé. Gondot jelent, hogy a hálózat viszonylag alacsony adatátviteli sebessége miatt a mote-ok száma korlátot szab a mintavételi frekvencia nagyságának, mivel a hálózatban keletkező mintákat adott sávszélességű csatornán kell továbbítani.

Bemutatásra került egy olyan zajérzékelő szenzorhálózat is, melyben az egyes elemek előfeldolgozást végeznek a zajmintákon, és a feldolgozás eredményeként előálló Fourier-együtthatókat továbbítják. Mivel az együtthatók általában lassabban változnak, mint maga a jel, így továbbításuk ritkábban is lehetséges. Ezzel lehetővé válik a hálózatban található érzékelők számának növelése a mintavételi frekvencia csökkentése nélkül.

A diplomaterv zárásaként bemutattam a szinkronizációval és a megvalósított zajcsökkentő rendszerekkel kapcsolatos teszteredményeket, melyek az algoritmusok működőképességét bizonyítják. A szinkronizáció jellemzéseként elmondható, hogy az állomások $\pm 5\ \mu$sec pontossággal képesek szinkronizálódni a referenciaidőhöz. A

megvalósított zajcsökkentő rendszerek mindegyikében alapharmonikusra nézve elértünk legalább 30 dB-es elnyomást, tehát a rendszer működőképes, és megfelelő elnyomást biztosít.

Jövőbeni tervek között szerepel a már meglévő rendszerek továbbfejlesztése. Ezen fejlesztés vonatkozik egyrészt egy olyan rendszer kidolgozására, mely az itt bemutatottaknál több mikrofont és hangszórót tartalmaz, így nagyobb térben képes a zajszint csökkentésére. Mivel a zajérzékelést megvalósító szenzorok telepes táplálásúak, így egy másik továbbfejlesztési lehetőségként kínálkozik a hálózat energiatakarékosságának javítása, mely segítségével növelhető a hálózat maximális működési ideje. Mindezen fejlesztések közelebb visznek egy gyakorlatban is jól működő és felhasználható zajcsökkentő rendszer kiépítéséhez.

Irodalomjegyzék

[1] Elliott., S. J.: „Distributed Control of Sound and Vibration”, in 2004 *Int. Symp. On Active Noise Control of Sound and Vibration*, ACTIVE ’04, pp 1-25, Williamsburg, Virginia, Sept. 2004

[2] Elliott., S. J., P. A. Nelson: „Active noise control”, *IEEE Signal Processing Magazine*, Vol. 10, No. 4. October, 1993, pp. 12-35.

[3] Kuo, S. M., D. R. Morgan: „Active Noise Control: A Tutorial Review”, *Proceedings of the IEEE*, vol. 87. No. 6., pp. 943-973, June. 1999.

[4] Péceli, G.: „A common structure for recursive discrete transforms”, *IEEE Trans. Circuits Syst.* Vol. CAS-33, pp.1035-1036, Oct. 1986.

[5] Nagy, F.: „Measurement of signal parameters using nonlinear observers”, in *IEEE Trans. on Instrumentation and Measurement*, vol. IM-41 No. 1. pp. 152-155, Feb.1992

[6] Morgan, D. R: „An analysys of multiple multiple correlation cancelation loops with a filter in the auxiliary path”, *IEEE Trans. Acoustic and Speech, Signal Pocessing*, Vol. ASSA-28, pp. 454-467, Aug. 1980.

[7] Sujbert, L., G. Péceli, Gy. Simon, „Resonator based non-parametric identification of linear systems”, in *IEEE Trans. on Instrumentation and Measurement*, vol. 54. no. 1., Feb. 2005, pp. 386-390.

[8] Akyildiz, I. F., W. Su, Y. Sankarasubramaniam, and E. Cayirci: „*Wireless sensor networks: A survey*”, Comput. Netw., vol. 38, no. 4, pp. 393-422, 2002

[9] Mathiesen, M., G. Thonet, N. Aakwaag: „Wireless ad-hoc networks for industrial automation: a current trends and future prospects”, in *Proceedings of the IFAC World Congress*, Prague, Czech Republic, July 4-8, 2005

[10] Kömer, K., P. Blum, L. Meier: „*Time Synchronization and Calibration in Wireless Sensor Networks*”, in Ivan Stojmenovic (ed.) : „*Handbook of Sensor Networks: Algorithms and Architectures*”, Wiley, 2005

[11] Raffaelle Noro: „Synchronisation over packet-switching networks: Theory and applications”, PhD Thesis, École Polytechnique Fédérale de Lausanne, 142 p.

[12] Kay Römer: „Time synchronization in ad hoc networks” in *ACM Symposium on Mobile Ad-Hoc Networking and Computing*, October 2001.

[13] Ganeriwal, S., R. Kumar, M. B. Srivastava: „Timing-sync protocol for sensor networks” in *First ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Los Angeles, California, USA, November 2003.

[14] Elson, J., L. Girod, and D. Estrin: „Fine-grained network time synchronization using reference broadcasts” in *Fifth Symposium on Operating Systems Design and Implementation* (OSDI 2002), Boston, MA. December 2002, December 2002.

[15] Maroti, M., B. Kusy, Gy. Simon, and A. Ledeczi: „The flooding time synchronization protocol” Technical Report ISIS-04-501, Institute for

Software Integrated Systems, Vanderbilt University, Nashville Tennessee, 2004.

[16] Mills, D. L: „Internet Time Synchronization: The network time protocol" *IEEE Transactions on Communications* COM 39 no. 10, p. 1482–1493, October 1991.

[17] Molnár, K., L. Sujbert, G. Péceli: „Synchronisation of sampling in distributed signal processing systems", in *Int Symp. On Intelligent Signal Processing, WISP 2003.*, Budapest, Hungary, sept. 2003.

[18] „*ADSP-2106x SHARC EZ-KIT Lite Reference Manual*", Analog devices Inc. P.O Box 9106 Norwood, 1997

[19] „*ADSP-2106x SHARC User's Manual (second edition)*", Analog devices Inc. P.O Box 9106 Norwood, 1997.

[20] http://www.xbow.com/Support/Support_pdf_files/ XBOW_Smart_Dust_ProductInfoGuide.pdf

[21] IEEE std. 802.15.4 - 2003: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Networks (LR-WPANs) http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf

[22] http://www.tinyos.net/nest/doc/tutorial/

[23] http://www.analog.com/UploadedFiles/Application_Notes/ 399447663EE191.pdf

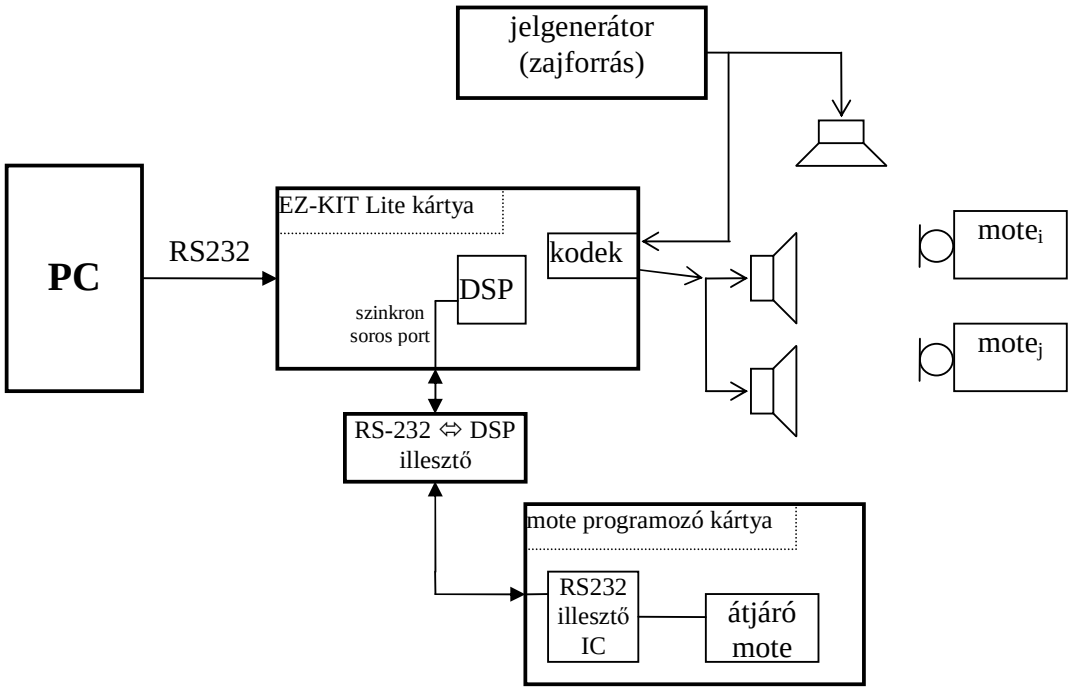
[24] Widrow, B., S.D. Stearns, „*Adaptive Signal Processing*", Prentice Hall, Inc. 1985.

[25] Sujbert, L., G. Péceli, „Periodic noise cancellation using resonator based controller", in *1997 Int. Symp. on Active Control of Sound and Vibration, ACTIVE '97*, pp. 905-916, Budapest, Hungary, Aug. 1997

Függelék

I. Zajcsökkentő rendszer fizikai kiépítése

A zajcsökkentő rendszerek gyakorlatban történő megvalósítása az I.1. ábrán látható. Ezen elrendezés alapján elvégezhető a rendszer összeállítása. Az elrendezés mind a mintatovábbító, mind a rezonátorokat futtató mote-okat alkalmazó zajcsökkentő rendszerek esetén megegyezik, csupán az eszközökön futó szoftverek különböznek.



I.1. ábra. Rendszerfelépítés

A rendszer indításakor mind a mote programozói kártyájának, mind a DSP fejlesztői kártyának biztosítjuk a tápfeszültséget tápegység segítségével. A mote-ok programozói kártyáját soros porton keresztül a PC-hez csatlakoztatva a mote-ok felprogramozhatóak a feladatuknak megfelelően. Ezt a programozást csupán egyszer kell elvégezni, mivel a mote-ok mikrokontrollerének programmemóriája nem törlődik kikapcsolásukkor. A programozás után a programozói kártyát az eddig programozásra használt soros porton keresztül csatlakoztatjuk a mote és DSP közötti illesztést megvalósító áramkörhöz, melyet előzőleg az EZ-KIT Lite kártyához, a DSP SPORT1 szinkron soros portjának kivezetéseihez csatlakoztattunk.

A mote-ok programozásakor meg kell adnunk azok hálózatban használt címeit. Ezt a mintatovábbító és a rezonátorokat futtató mote-ok esetén is a következő képlettel számítjuk:

$$moteCim = 8 \cdot (hálózatMéret - 1) + azonosító \qquad \text{I.1}$$

a jelölések magyarázata a következő:

- *azonosító*: nullától kezdjük kiosztani, minden mote-hoz saját azonosítót rendelünk
- *hálózatMéret*: a hálózatban lévő mote-ok száma

A mote-ok a *moteCim* információból bekapcsoláskor kiszámítják a saját azonosítójukat, valamint a hálózat méretét. Ez a módszer ebben a formában 8 mote-ból álló hálózat kialakítását teszi lehetővé, de ez könnyedén, a mote-ok programjában található konstrans módosításával növelhető. Ennél nagyobb méretű hálózat kiépítésével egyelőre nem próbálkoztunk.

Az átjáró mote hálózati címét a következő képlettel számítjuk:

$$moteCim = hálózatMéret \qquad \text{I.2}$$

Ebből a bázisállomás képes a hálózat méretét meghatározni.

A DSP-n futó zajcsökkentő program letöltése a PC segítségével lehetséges. Ehhez a PC-t soros porton keresztül csatlakoztatni kell a fejlesztői kártyán található RS-232-es csatlakozóhoz. A programozás elvégzése minden bekapcsolás esetén szükséges, mivel a DSP programmemóriájának tartalma minden kikapcsolás alkalmával törlődik, így a rendszer indításakor a PC-n futó VisualDSP++ fejlesztői környezet segítségével le kell tölteni a zajcsökkentő programot.

A DSP-hez a beavatkozó hangszórókat a kodek segítségével illesztjük. A fejlesztői kártyán a kodek ki- és bemeneteihez jack dugó segítségével férünk hozzá. Látható módon zajforrásként egy jelgenerátor használható, melynek kimeneti jelét egy hangszóróra, valamint a fejlesztői kártya kodekjére vezetjük, így felhasználva azt referenciajelként, amennyiben ez szükséges.

Az zajérzékelő hálózat élesztése a mote-ok bekapcsolásával történik. Utoljára a nullás azonosítójú mote-ot kell bekapcsolni, ugyanis ez indítja a mintavételezést, és az adatok átjáró mote felé történő továbbítását. A programozó kártyán lévő átjáró mote-ot nem kell bekapcsolni, az folyamatos tápfeszültséget kap, de ezt is a nullás mote indítja, és folyamatosan továbbítja soros porton a hálózat mote-jaitól érkező adatokat a DSP felé.

A VisualDSP++ fejlesztői környezetben a DSP-re töltött programot elindítva lép működésbe a zajcsökkentő rendszer. Ezt a szenzorhálózat beindítása után kell elvégeznünk, ugyanis amennyiben nem működik a hálózat, a DSP nem kap bemenetet, ami bizonyos esetekben instabilitáshoz vezethet.