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

In-Premises Display Device megvalósítása ZigBee Smart Energy Profile alapján

Diplomaterv

Belső konzulens	*Készítette*	*Külső konzulens*
dr. Sujbert László	Benyhe Tamás	Füredi Gábor

2013.

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

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

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

Budapest, 2013. május 24.

Benyhe Tamás
hallgató

Kivonat

Dolgozatom egy piaci igényeket kiszolgáló termék (a továbbiakban *StickBee*) beágyazott szoftverének megvalósításával kapcsolatos munkámat dolgozza fel. A StickBee a smart grid rendszerekben a fogyasztók tájékoztatását szolgáló *in-home display* funkciót tölti be. A StickBee-vel szemben elvárás, hogy más gyártók termékeivel is együttműködjön, ezért a *ZigBee* specifikáció egyik publikus profilja, a *Smart Energy Profile* (SEP) előírásainak megfelelően működik. Dolgozatom egyik célja a ZigBee szabványok és szakkifejezések áttekintése, és a ZigBee gyakorlatban történő alkalmazásának bemutatása.

A StickBee-nek képesnek kell lennie csatlakozni a ZigBee SEP hálózatokhoz, adatokat lekérdezni a hálózat többi eszközétől és az adatokat megjeleníteni. A StickBee nem rendelkezik saját kijelzővel, hanem USB porton keresztül csatlakozik egy gazda kijelzőhöz, és azt ruházza fel ZigBee SEP funkciókkal. A StickBee-ben két mikrovezérlő végzi el ezeket a feladatokat: a központi vezérlő kezeli az USB-t és gondoskodik az adatok megjelenítéséről; a ZigBee vezérlő a hálózati részt kezeli, és kiszolgálja a központi vezérlőt a képgeneráláshoz szükséges adatokkal. Ebből az én feladatom a ZigBee vezérlő szoftverének megtervezése, implementálása és tesztelése. A fejlesztéshez rendelkezésre áll a két vezérlő közti adatcsere alapjául szolgáló ICP szabvány.

A feladat megoldásához testreszabom az ICP-t és kidolgozom a megfeleltetést a ZigBee SEP adatok reprezentációja és a központi vezérlő számára szükséges adatok között. Az alkalmazás lekérdezi a hálózatban fellelhető mérőórák adatait és kezeli a szolgáltató felől érkező üzeneteket. Az alkalmazást a *Freescale MC13226V* ZigBee platformján valósítom meg, a fejlesztéshez a Freescale ZigBee stackjét és *IAR Embedded Workbench for ARM* C fordítót használok.

Az alkalmazás verifikálásához teszt hálózatot tervezek és üzemelek be. A tesztelést segítő program a *Perytons Protocol Analyzer*. A sikeres tesztek után a StickBee részt vehet bemutatókon és pilot projektekben.

A fejlesztés következő lépése a sorozatgyártás előkészítése lesz. Ennek keretében át kell térni a két mikrovezérlős megoldásról egy mikrovezérlőre, a végleges StickBee-t pedig fel kell készíteni a ZigBee tanúsítvány megszerzésére.

Abstract

The objective of this thesis is to present the embedded software development process of a product (*hereafter, StickBee*) which meets market demands. StickBee acts as an *in-home display* informing customers in smart grid systems. In order to provide manufacturer-independent operation, StickBee uses the public *Smart Energy Profile* (SEP) of the *ZigBee* specification. I aim to give an overview about the ZigBee standards and present how to apply them.

The StickBee has to be able to connect ZigBee SEP networks, request data from the network devices, and display this data on a user interface. The StickBee does not include own display but uses a host display instead. StickBee connects to the host display via USB port and extends its capabilities with ZigBee functionality. StickBee comprises two microcontrollers. The central controller handles the USB and displays the data, the ZigBee controller handles the network and supplies data to the central controller. My role in the project is to design, implement and test the embedded ZigBee software. The communication between the two controllers is based on the ICP standard.

To solve the problem I am customizing the ICP and designing the matching between ZigBee SEP data representation and the data needed by the central controller. The application polls the metering devices in the network and handles the messages coming from the utility provider. The application is implemented on the *Freescale MC13226V* ZigBee platform. For the development I am using the Freescale ZigBee stack and the *IAR Embedded Workbench for ARM* C compiler.

In order to verify the application I am designing and setting up a test network. The testing is performed with *Perytons Protocol Analyzer*. After successful testing, StickBee may take active part of demos and pilot projects.

The next development stage is going to be preparation for mass production. The application should be adapted to a single-chip solution and the final StickBee should be a ZigBee Certified product.

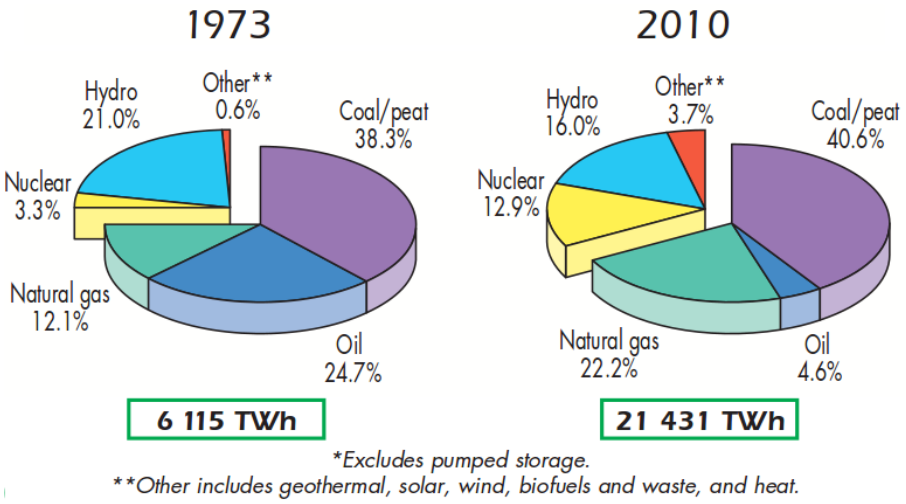
Bevezető

Smart grid

Dolgozatom motiválója az a *smart grid*, melyet a villamosenergia-iparban világszerte tapasztalható szemléletváltás szült. A smart grid - vagy ahogy itthon hívják, az *okos hálózat* - a villamosenergia-hálózat új, korszerű modellje, mely magába foglalja az energiaipar összes szereplőjét az elektromos energia előállításától egészen a fogyasztókig. De miért van szükség szemléletmódváltásra, miben különbözik a hagyományos hálózat az okos hálózattól?

Egy nemrég megjelent cikkben [14] több definíciót is olvashatunk a smart gridre, valamint átfogó képet kaphatunk az egyesült államokbeli és európai irányvonalak közti hasonlóságokról és eltérésekről. A különböző forrásokban megfogalmazott célokból, jellemzőkből és motiváló tényezőkből ugyanaz a smart grid koncepció bontakozik ki, eltérés legfeljebb csak a hangsúlyok helyéből és a prioritásokból adódhat. A smart griddel szemben támasztott igényeket illetve annak szolgáltatásait pontokba szedve próbálom meg összefoglalni.

Megújuló energiaforrások Az [9] tanúsága szerint a villamosenergia-termelés jelentős hányada fosszilis tüzelőanyagok elégetéséből és nukleáris energiából származik (1. ábra).



1. ábra. *Energiahordozók megoszlása a világ villamosenergia-termelésében.*

Közös jellemzőjük, hogy energiatermelésük kiszámítható, több napra előre tervezhető. Ugyanakkor mindkettő "fekete listán" van: a fosszilis tüzelésű erőműveket az üvegházhatást

okozó gázok kibocsátása miatt (lásd: Kiotói egyezmény), az atomerőműveket negatív társadalmi megítélésük miatt (lásd: csernobili, fukusimai katasztrófák) szorítják egyre inkább háttérbe. Az évről évre nagyobb arányban helyükbe kerülő megújuló energiát (pl. szél- vagy napenergiát) hasznosító erőművek azt eredményezik, hogy a megbízható, kiszámítható energiaellátó rendszerbe sztochasztikus viselkedés kerül, amit a jelenlegi merev struktúra csak szűk határok között tolerál.

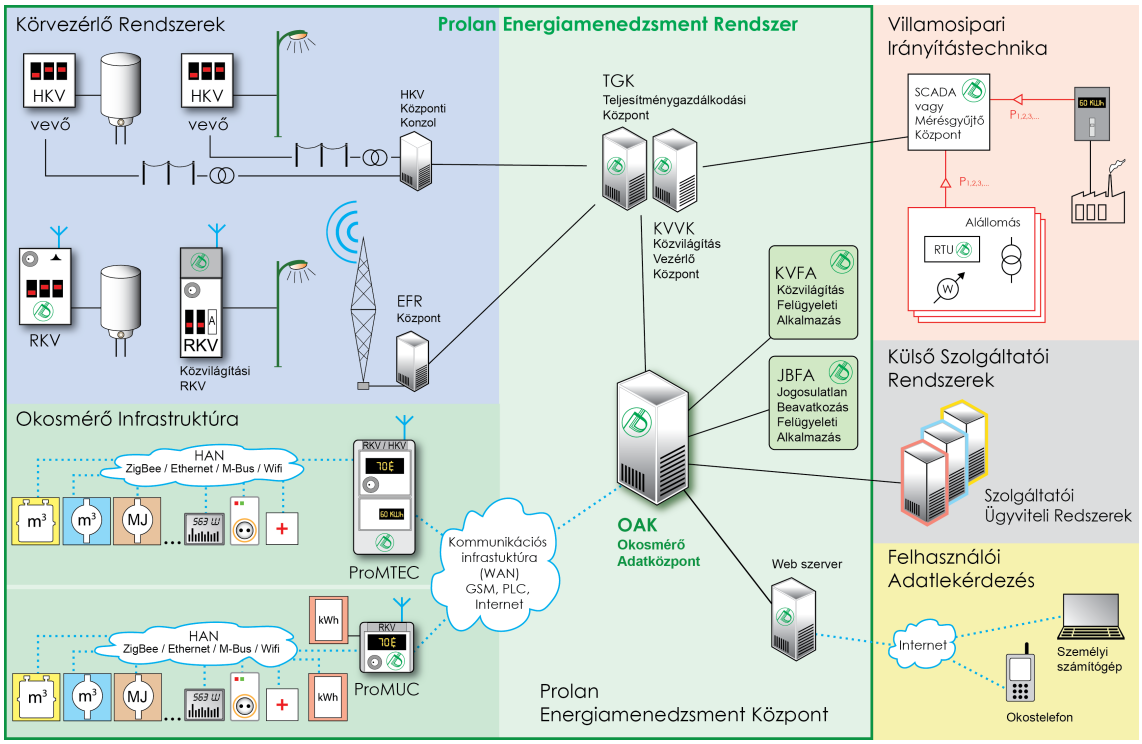
Új szolgáltatások Az informatikai szektor rohamléptékű fejlődésének köszönhetően számtalan innovatív szolgáltatás vár bevezetésre. Következzék néhány a teljesség igénye nélkül: okos mérés, szofisztikált tarifarendszer, közműhálózatok (víz, gáz, villamos energia, fűtés) integrációja, végpontra lebontott parancs- illetve üzenetküldés, valós idejű monitorozás.

Energiatudatosság Az 1. ábrából az is világosan látható, hogy az emberiség energiaigénye rohamosan növekszik. Az erőművek kapacitásának vég nélküli növelése helyett egyre inkább a fogyasztási szokások megváltoztatására tevődik át a hangsúly. Felismerve a tömegekben rejlő potenciált arra ösztönzik a lakosságot, hogy telepítsenek a hálózatra tápláló minierőműveket, használjanak nagy hatásfokú, kis fogyasztású berendezéseket, minimalizálják a felesleges fogyasztást. A fogyasztók befolyásolásának leghatékonyabb eszközei a hálózati terhelés ingadozásait követő tarifarendszer és a termelési csúcsok simítását célzó ad hoc tarifacsökkentések. Az energiatudatos magatartás azonban csak úgy várható el, ha a fogyasztó visszacsatolást kap valós idejű információk, statisztikák formájában, és nem csak az aktuális számla összesített adatain keresztül értesül az elmúlt időszakról. Az információk megjelenítése több módon is történhet: webes felületen vagy dedikált kijelzőn (In-Home Display).

Infokommunikációs technológiák (ICT) A smart gridben kulcsfontosságú szerepet kapnak az infokommunikációs technológiák(2. ábra), hogy a hálózat tagjai bármikor képesek legyenek egymással kommunikálni. Erre a célra részben a már meglévő infrastruktúrát (pl. celluláris hálózatok, internet) használják, részben újakat építenek ki (pl. Power Line Communication).

A smart gridben használt infokommunikációs technológiák földrajzi kiterjedésük alapján két nagy csoportra oszthatók. A WAN (Wide Area Network) hálózatokat nagy távolságok (több km) áthidalására használják, pl. a szolgáltató és a fogyasztó között. A smart gridben használt WAN technológiák közé tartoznak a GSM és más celluláris hálózatok, a PLC (Power Line Communication) CENELEC A sávja, az RKV (rádiós körvezérlő) és a HKV (hangfrekvenciás körvezérlő).

A HAN/PAN (Home/Personal Area Network) hálózatokat kisebb távolságok (jellemzően néhány 10 m) áthidalására használják, pl. az egy háztartáson belül elhelyezkedő készülékek (mérőórák, gateway-ek, termosztátok, kijelzők) között. A smart grid PAN technológiái közé sorolható az M-Bus vezetékes és vezeték nélküli változata, a PLC CENELEC B, C és D sávja, valamint a ZigBee.



2. ábra. *Prolan energiamenedzsment-rendszer [13].*

A ZigBee és a smart grid kapcsolata Új kommunikációs hálózatok kiépítésére előszeretettel alkalmaznak rádiós megoldásokat, mivel a kábelezés költsége sok esetben irreálisan nagy lenne, vagy rosszabb esetben nem is kivitelezhető. Gyakran hálózati feszültség sem áll rendelkezésre, ezért nagyon alacsony fogyasztású elemes berendezések jöhetnek csak szóba. Ilyen megkötések mellett optimális választás a ZigBee technológia használata.

A ZigBee Alliance által publikált szabványok nemcsak protokollrétegeket, hanem komplett alkalmazásprofilokat (Apllication Profile) specifikálnak. A profilok közt van a Smart Energy Profile (SEP) is, amit kifejezetten smart gridhez alkottak meg. Az alkalmazásprofil a hálózatot alkotó lehetséges eszközök profiljait (Device Profile) is definiálja, a SEP például a következőket tartalmazza: Energy Service Interface, Metering Device, In-Premises Display, Programmable Communicating Thermostat, Load Control Device, Smart Appliance, Prepayment Terminal. A készülékek egy átfogó tanúsítási folyamaton esnek át, így a különböző gyártók termékei kompatibilitási probléma nélkül alkothatnak hálózatokat. A ZigBee PAN hálózatot az Energy Service Interface (ESI) kapcsolja össze a WAN hálózattal.

A dolgozat felépítése

1. fejezet Az első fejezetben a Prolan SE Kft. tevékenységének rövid bemutatása után ismertetem termékfejlesztési projektjüket, hogy elhelyezhessem benne saját munkámat. A termék neve StickBee, a feladatom beágyazott szoftverfejlesztés volt.

2. fejezet A második fejezet irodalomkutatás, melyben a dolgozatommal kapcsolatos ZigBee szabványokat dolgozom fel, és bevezetem a dolgozat további részében használt fogalmakat.

3. fejezet A harmadik fejezetben a beágyazott szoftverfejlesztéshez használt fejlesztői környezetet mutatom be.

4. fejezet A negyedik fejezetben implementációs részleteket fejtek ki az általam fejlesztett beágyazott szoftverről.

5. fejezet Az ötödik fejezetben ellenőrzöm a program működésének helyességét, és bemutatom ennek eszközeit.

6. fejezet Az utolsó fejezetben összefoglalom a munkám során elért eredményeket és felvázolom a további teendőket, fejlesztési irányokat.

1. fejezet

A StickBee projekt

1.1. A Prolan SE Kft. tevékenysége

A magyar villamosenergia-ipari beszállítóként már hírnevet szerzett és az okosmérés területén jelentős hazai innovációs potenciállal rendelkező Prolan Irányítástechnikai Zrt. és az IPSOL Rendszerház Kft. stratégiai szövetségre lépett egymással, így alakult meg a Prolan Smart Energy Rendszerház Kft. A szoros együttműködés célja a két cég eredményeinek összehangolása a forradalmi léptékű növekedés és fejlődés előtt álló okos mérési piacon, amelynek eredményeként egységes fejlett megoldást kínálnak a meglévő és jövőbeli ügyfeleik számára [2].

A Prolan SE Kft. tevékenysége kiterjed energia audit és energia beszerzés támogatására, Smart Metering megoldások valamint intelligens eszközök közötti kommunikációs csatornák kiépítésére és működtetésére, továbbá tanácsadásra az említett és azokhoz kötődő területeken (Smart Office, GPS, stb.). Mindezek mellett a kutatás-fejlesztésre is nagy hangsúlyt helyeznek, hogy termékeik és szolgáltatásaik tökéletesítése révén partnereik új igényeit is magas színvonalon szolgálhassák ki [2].

A StickBee a Prolan Smart Energy Rendszerház Kft. egyik termékfejlesztési projektje, melynek én is részese voltam.

1.2. A projekt célja

A smart grid - a ZigBee-t is beleértve - elterjedés alatt álló technológia, a meglévő régi rendszer fokozatosan kerül leváltásra. A bevezetés korai szakaszában előnyös lehet, ha drága, komplex redszerek helyett egyszerűen használható, költséghatékony megoldások kerülnek a felhasználókhoz, biztosítva ezzel az átmenetet, folytonosságot. Egy népszerű angol kifejezéssel élve: *evolution, not revolution.*

Ezen elv alapján született meg a StickBee ötlete is. A StickBee egy olyan berendezés, amely a háztartásban általában megtalálható kijelzőket ruházza fel a ZigBee Smart Energy Profile előírásainak megfelelő otthoni kijelző (In-Premises Display) funkcióival, vagyis önmaga nem rendelkezik kijelzővel. A gazda kijelzők képességeiket tekintve széles skálán mozoghatnak, de USB csatlakoztatási lehetőség és diavetítés funkció már a legtöbb kijelzőben megtalálható. Ezt figyelembe véve a StickBee két üzemmódban képes működni:

1. Szerényebb képességekkel bíró eszközökhöz csatlakoztatva a StickBee USB meghajtóként érhető el. A meghajtón található képek periodikusan generálódnak sablonok alapján a ZigBee hálózatból nyert friss adatok felhasználásával. Ez az üzemmód pl. digitális képkeretek, okos TV-k esetén használható.

2. Személyi számítógéphez (PC) csatlakoztatva egy saját fejlesztésű alkalmazással kezelhető az eszköz. Ebben a módban grafikusan megjeleníthetőek a begyűjtött adatok, konfigurálhatjuk az eszközt, vagy frissíthetjük annak firmware-ét.

Az eszköz a kijelző hiánya miatt kis méretben elkészíthető. Az apró USB csatlakozású eszközök gyűjtőneve angolul az *USB stick*, amit a *ZigBee*-vel összevonva kapjuk a *StickBee* elnevezést. A StickBee segítségével a felhasználó releváns információkat kap saját fogyasztási szokásairól, melyeken szükség esetén változtathat.

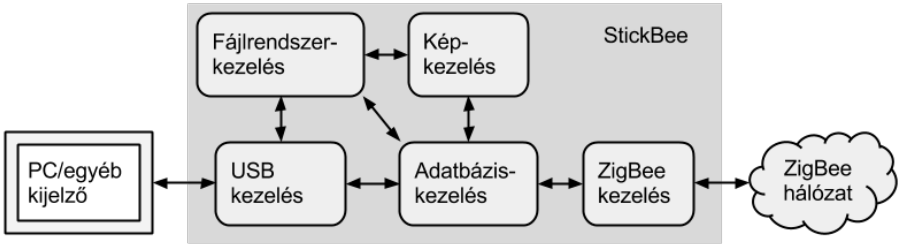
1.3. Funkcionális követelmények

A funkcionális követelmények megállapításához több tanulmány is készült. A belőlük eredeztetett követelmények a követelmények a projekthez való csatlakozásomkor már rendelkezésre álltak, azok kidolgozásában nem vettem részt. A tanulmányok részletezésének kihagyásával jöjjenek a belőlük levont következtetések funkcionális blokkokra bontva.

1. **USB kezelés**
   - *Mass Storage Device* a generált képek és a konfigurációs állományok eléréséhez
   - *Communication Device Class* (CDC) a PC-vel való kommunikációhoz, COM port emulálás
   - a két szerepkör megvalósítása *USB 2.0 Full Speed composit device*-on keresztül
2. **Képkezekés**
   - *JPEG baseline* képformátum
   - 480×272 pixel felbontás
   - 256 színű *BMP* sablonképek
3. **ZigBee kezelés**
   - *ZigBee 2007 PRO* feature set
   - *Smart Energy 1.x* application profile
   - *In-Premise Display* device profile
4. **Fájlrendszerkezelés**
   - generált képek, konfigurációs állományok, ZigBee adatok tárolása
   - fájlrendszer: *FAT*
5. **PC szoftver**

- OS független *JAVA* alkalmazás
- képsablonok szerkesztése, konfigurációs állományok generálása
- szoftverfrissítés kezelése
- ZigBee adatok megjelenítése

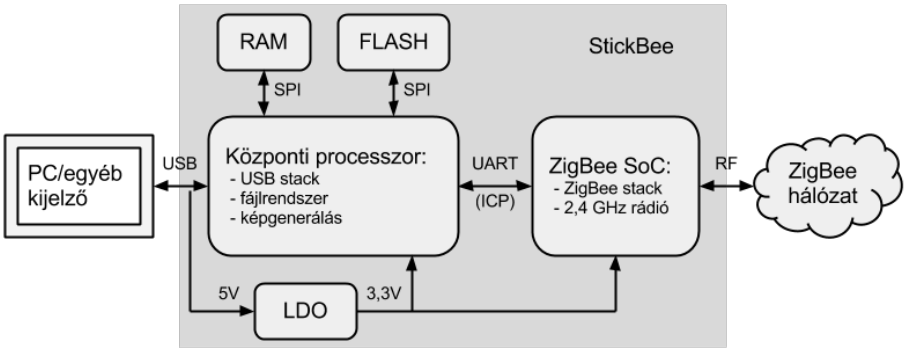
A funkcionális blokkok közötti kapcsolatokat szemlélteti az 1.1. ábra.



1.1. ábra. *A StickBee funkcionális blokkvázlata.*

1.4. Hardverfelépítés

A funkcionális modulok hardverben történő realizálására több koncepció is született a tervezés során. A dolgozatban is tárgyalt megvalósítás blokkvázlata az 1.2. ábrán látható.



1.2. ábra. *A StickBee hardverfelépítése.*

Ez a megoldás nem optimális, mivel két mikrokontrollert használ. A hosszútávú cél áttérni egy olyan System-on-Chip (SoC) megoldásra, ami a két mikrokontroller feladatát egyszerre el tudja látni. Ezért olyan ZigBee platform került kiválasztásra, ami USB kezelésre is alkalmas.

A döntés meghozatalakor egyetlen ZigBee platform sem volt a piacon, ami egyben képes lett volna USB kezelésre is, és csak 2 gyártó, a Texas Instruments[1] és a Freescale[2] hozta nyilvánosságra ezirányú szándékát. Mindkét platform megjelenésének apropója a legújabb ZigBee specifikáció, a 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) alapú ZigBee IP kiadása. Az 1.1. táblázat a két gyártó USB-s ZigBee platformját hasonlítja össze.

[1] `http://www.ti.com/product/cc2538/`
[2] `http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=KW20`

1.1. táblázat. *A Texas Instruments és a Freescale SoC összehasonlítása.*

Gyártó	**Texas Instruments**	**Freescale**
SoC típus	CC2538	KW20
processzormag	ARM Cortex M3	ARM Cortex M4
CPU órajel	32 MHz	50 MHz
RAM	32 kB	64 kB
FLASH	512 kB	512 kB
USB	1	1
SPI / UART / I^2C	2 / 2 / 1	1 / 2 / 2
tokozás	QFN56	LGA56
RF be-/kimenet	differenciális	differenciális
rádió vételi érzékenység	-97 dBm	-102 dBm

Az alapvető paramétereket összehasonlítva látható, hogy a két SoC hasonló tulajdonságokkal bír. A Freescale előnye az erősebb, gazdagabb utasításkészletű processzormag, a nagyobb RAM és az érzékenyebb rádió. A Texas Instruments mellett szól az, hogy a Windows-os IAR EWARM fordító mellett a Linux rendszereken is használható Code Composer Studio-t is támogatja, illetve a teljes stack forráskód szinten elérhető.

A döntéshez vállalatstratégiai szempontokat is szem előtt kellett tartani. A választott ZigBee platform a cég más termékeiben is felhasználásra fog kerülni, ezért a skálázhatóság és időtállóság még nagyobb hangsúlyt kapnak. A KW20-ban helyet foglaló rádió később önálló IC-ként is kapható lesz, ami azt eredményezi, hogy más Kinetis szériás processzorokkal igen jól skálázható hardvert lehet kialakítani az alkalmazás igényeinek megfelelően, emiatt a döntés végül a Freescale platformjára esett.

A KW20 megjelenéséig a szintén Freescale gyártmányú MC13226V SoC került kiválasztásra, hogy az áttérés a lehető legzökkenőmentesebb legyen. A központi processzor egy 32 bites PIC.

1.5. Fejlesztési státuszok

A cég minőségirányítási rendszere az ISO9001 szabványra épül, amely egyebek mellett a termékfejlesztési folyamatokat is szabályozza. A rendszer egy fejlesztési projekt esetén több státuszt is megkülönböztet az alapján, hogy milyen célokat szolgál, az ötlet megszületésétől egészen a sorozatgyártásig.

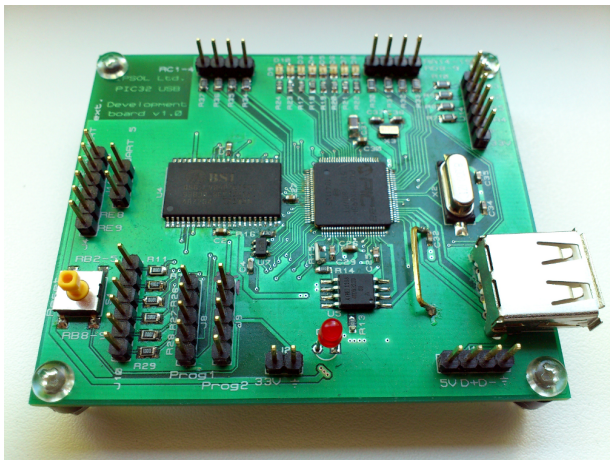
A kezdeti szakasz feladatai a koncepció műszaki megvalósíthatóságának és piacképességének vizsgálata. Ekkor történik az első prototípusok megalkotása és vele párhuzamosan elkezdődik a piac megismertetése a termékkel. A StickBee-t több kiállításon is nagy érdeklődés övezte (1.3. ábra) és a prototípstesztek is sikeresnek bizonyultak (1.4. ábra), ezért a következő fejlesztési státuszba léphetett (1.5. ábra).

A második státusz feladata a valós működés megközelítése, hogy a termék kiadható legyen üzleti partnerek számára pilot projektek céljából. Én itt kapcsolódtam a projekthez, munkám ennek elérését segítette.

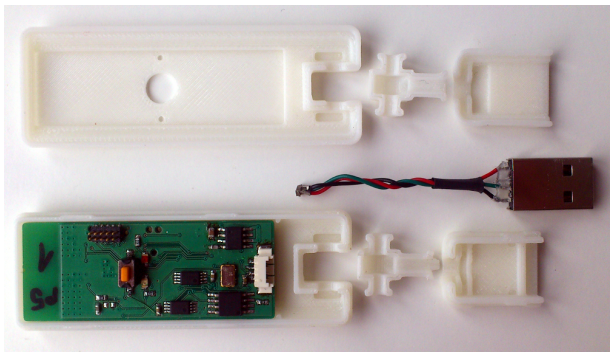
Az utolsó státusz a költségoptimalizálást és a sorozatgyártást célozza meg, kritériumfaktora a profit termelése. Ezekről a szempontokról a 6.2 fejezetben írok bővebben.

A minőségirányítási rendszer előírja a státuszokon belüli verziókövető rendszer

1.3. ábra. *A StickBee 3D látványterve.*



1.4. ábra. *A StickBee első prototípusa.*



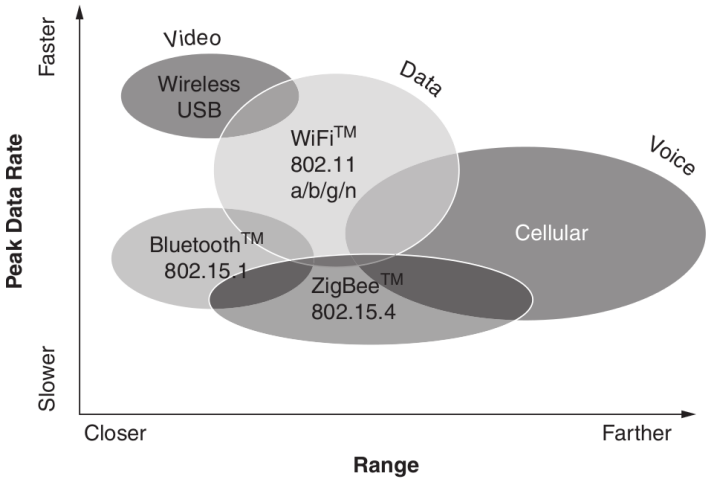
1.5. ábra. *A StickBee legújabb prototípusa.*

használatát is, ami a nagyobb szoftverfejlesztési projekteknél egyébként is nélkülözhetetlen. A cég verziókövető rendszere a széles körben elterjedt Subversion (röviden SVN), ezt a *Tortoise SVN* nevű kliens programmal használtam.

2. fejezet

A ZigBee bemutatása

A ZigBee egy vezeték nélküli technológia, amely számos szabványt magában foglal. Más rádiós szabványokhoz hasonlóan a ZigBee-nek is megvannak az erősségei és korlátai (2.1. ábra).



2.1. ábra. *A ZigBee viszonya más vezeték nélküli technológiákkal [12].*

A szabványok feldolgozását a primer források mellett két szakkönyv segítette. A [10] a rádiós kommunikáció oldaláról közelíti meg a témát, a [12] gyakorlati példákon keresztül mutatja be a ZigBee-t.

2.1. ZigBee szabványok áttekintése

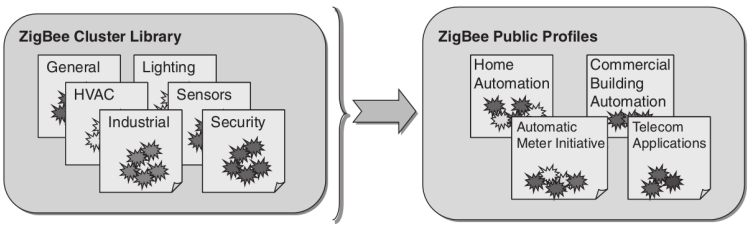
Az IEEE 802.15.4 szabvány több LR-WPAN (Low-Rate Wireless Personal Area Network) protokoll alapját képezi, melyek közül a ZigBee a legelterjedtebb. Az LR-WPAN hálózatok főbb jellemvonásai:

- viszonylag rövid hatótávolságú, kis sebességű információátvitel;
- minimális infrastrukturális háttér;
- kis méretű, energiahatékony, olcsó eszközök

A hálózatot alkotó eszközök beállítása (komissiózás) után egy jól megtervezett hálózat nem igényel karbantartást. Kis erőforrásigényű ad hoc szervezésű vezeték nélküli hálózatok számtalan területen alkalmazhatók hatékonyan. Az LR-WPAN hálózatok alkalmazási körébe tartozik az épületautomatizálás, egészségügy, szórakoztató elektronika, telekommunikáció, stb.

Ahhoz, hogy a különböző területek igényeit kielégíthesse, a ZigBee technológia több szintre oszlik. A *ZigBee specifikációk* azt írják le, hogy az egyes készülékek hogyan kommunikálnak illetve alkotnak hálózatot egymással. Eddig három ZigBee specifikáció jelent meg, mindhárom alapja az IEEE 802.15.4 szabvány. A következő szintet a *ZigBee szabványok* alkotják, melyek már a hálózatot alkotó eszközök lehetséges típusait is definiálják, illetve meghatározzák a specifikációk által nem vagy csak részben szabályozott részeket, esetleg kiegészítik azt. Eddig kb. egy tucat szabvány jelent meg, mindegyik egy-egy alkalmazási területet fed le.

A szabványok által testreszabott, egy adott alkalmazási területet felölelő specifikációkra *Application Profile*-ként hivatkozik a ZigBee, a profilhoz tartozó eszköztípusokat pedig *Device Profile* írja le. A Device Profile megszabja, hogy egy eszköztípus milyen funkcionális blokkokból, ún. *cluster*ekből állhat. A clusterek nagy része a specifikációktól és szabványoktól független ZigBee Cluster Library-ben (ZCL) található, egyes alkalmazási területhez kapcsolódó clustereket pedig az adott ZigBee szabvány írja le (2.2. ábra).



2.2. ábra. *Kapcsolat a ZCL és az Application Profile között [12].*

ZigBee specifikáció Ez a legáltalánosabb specifikáció, a legtöbb szabvány ezen alapszik. A specifikáció legelső kiadása a ZigBee 2004 volt, ezt váltotta le a ZigBee 2006. A ZigBee 2007 legfőbb újítása az volt, hogy bevezette Feature Set fogalmát. A 2006-os változat ZigBee Feature Set néven továbbra is része maradt a 2007-nek, mely kisebb ($\sim$100 eszköz) mesh hálózatok kialakításához ideális. Ehhez jöttek hozzá a továbbfejlesztett funkciók ZigBee PRO Feature Set néven, melyek kiterjesztik a technológiát nagyobb (1000+ eszköz), komplexebb hálózatokra. A két Feature Set egymással kompatibilis. Aki a ZigBee-t emlegeti, az általában a ZigBee 2007 PRO-ra gondol, mely a ZigBee 2012-ben az innovatív Green Power opcióval bővült.

A ZigBee specifikációt az alábbi szabványok használják:

- Building Automation
- Health Care
- Home Automation

- Light Link
- Smart Energy
- Telecom Services
- Retail Services (fejlesztés alatt)

ZigBee RF4CE specifikáció Ez a ZigBee specifikáció egyszerű, robusztus hálózatok kialakítására alkalmas a távvezérlés területén. Az RF4CE az otthoni környezetben felmerülő vezérlési feladatokat (pl. kapunyitás, szórakoztatóelektronika) célozza meg, hogy egységesítse a különböző gyártók távvezérlőit (vö. Bluetooth HID).

A ZigBee RF4CE specifikációra jelenleg két szabvány épül, a Remote Control és az Input Device.

ZigBee IP specifikáció A ZigBee IP specifikáció IPv6 hálózatok létrehozását teszi lehetővé LR-WPAN hálózatokon, ezáltal natív módon támogatja az *Internet of Things* koncepciót köztes gateway használata nélkül. A ZigBee IP rendelkezik az energiatakarékos rádiós mesh hálózatok előnyeivel, emellett olyan sztenderd internetes protokollokat támogat, mint a 6LoWPAN, IPv6, PANA, RPL, TCP, TLS és UDP.

A ZigBee IP specifikációt egyedül a Smart Energy 2.0 szabvány használja.

2.2. ZigBee Alliance

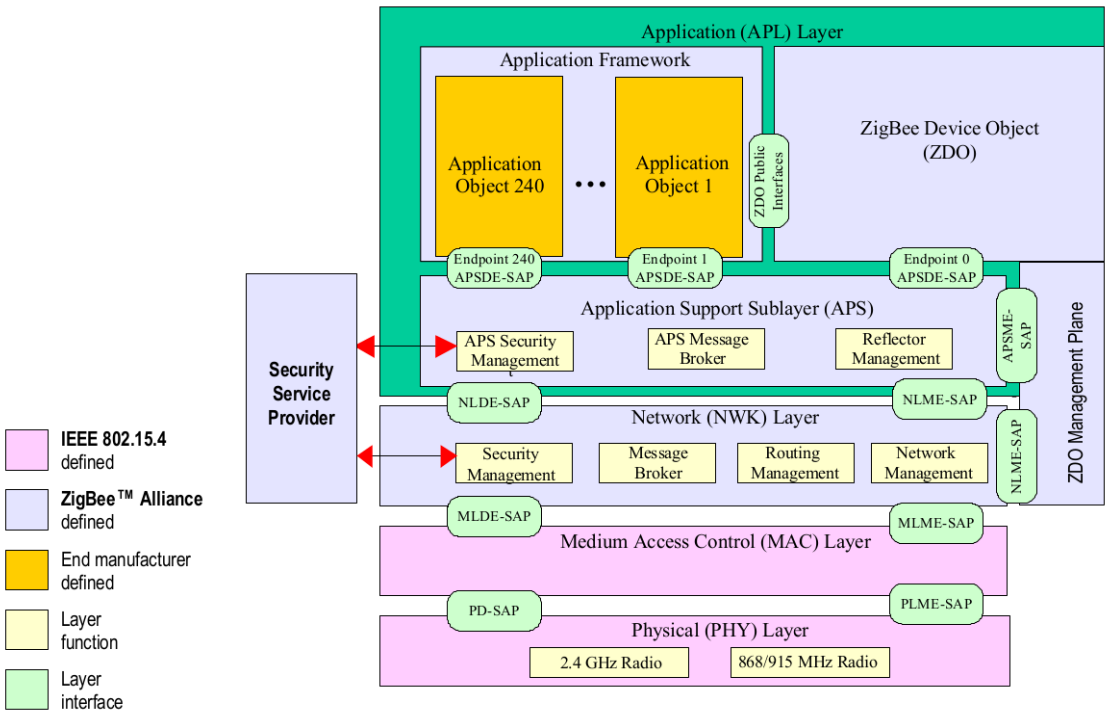
2002-ben az IEEE 802.15.4 szabvány fejlesztői vezető ipari cégekkel közösen létrehozták a ZigBee Alliance-t, amely mára már világméretű szervezetté nőtte ki magát, tagjai száma meghaladja a 400-at. A nonprofit szervezet megalakulásának célja egy nyilvános, széles körben támogatott és széles fejlesztői bázissal működő szabvány létrehozása, legfőbb feladata tehát, hogy koordinálja és ipari szereplők bevonásával segítse a szabványalkotási folyamatokat.

A szervezet másik fontos feladata annak biztosítása, hogy csak a szabványi előírásoknak minden tekintetben eleget tevő ZigBee termékek kerülhessenek piacra. Ezért minden ZigBee termék egy szigorú tesztelési procedúrán esik át, mielőtt megkapja a jogot a ZigBee logó használatára. A teszet független laboratóriumok végzik (NTS, TÜV), pozitív eredmény esetén pedig a ZigBee Alliance kiállítja a tanúsítványt. Nemcsak termékek kapnak tanúsítást, hanem ZigBee platformok is. A ZigBee platformot rádió és beágyazott ZigBee firmware-t futtató mikrokontroller alkotja, ezekre a platformokra épülnek a termékek.

ZigBee eszközt gyártó vállalatoknak legalább *Adopter* szintű tagsággal kell rendelkezniük a ZigBee Alliance-ben (a vállalatok többsége ebbe a csoprtba tartozik). A tagság éves tagdíj fizetésével jár, amiért cserébe kedvezményeket és különböző kiváltságokat kapnak a vállalatok. Engedélyt kapnak például a ZigBee logó használatára (szigorú feltételek betartása mellett), termékeik felkerülnek a ZigBee hivatalos honlapjára [3]. A *Promoter* és *Participant* tagsággal rendelkező vállalatok a szabványok fejlesztésében is aktívan részt vesznek.

2.3. A ZigBee 2007 protokoll felépítése

A ZigBee az OSI modell szerint épül fel, ennek megfelelően a kommunikáció absztrakciós szintjei önálló rétegekbe szerveződnek. Az együttműködő rétegek halmazaként megvalósuló protokollt szokás stacknek nevezni, ezt a terminológiát használja a ZigBee is. A rétegek viszonyát a 2.3. ábra szemlélteti.



2.3. ábra. *ZigBee protokollrétegek [5].*

A stack alsó két rétegét (PHY és MAC) az IEEE 802.15.4 szabvány definiálja, ezekre épül a ZigBee Alliance által definiált NWK és APL réteg. Egy réteg által nyújtott funkciók, szolgáltatások SAP (Service Access Point) segítségével érhetők el a felette elhelyezkedő rétegek számára. Egy réteg funkcióinak vezérlését a Management Entity (ME) látja el, az üzenetek értelmezését és keretbe ágyazását a Data Entity (DE) végzi. a A PHY, MAC, NWK és ASL rétegek egyes jellemzőikről konstans- és attribútumlisták formájában vezetnek nyilvántartást, ezekre a szabvány *Information Base* (IB) néven hivatkozik.

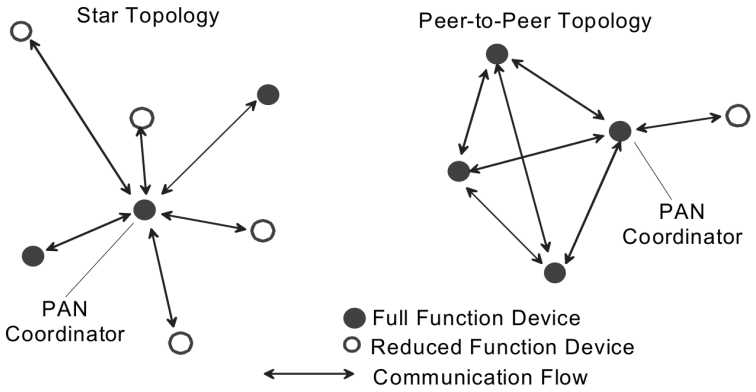
Az eszköz viselkedését és működését lényegében a 0-tól 240-ig sorszámozott végpontokon (endpoint) elhelyezkedő objektumok határozzák meg. A 0-s végponton minden esetben a ZDO (ZigBee Device Object) helyezkedik el, ez valósítja meg minden eszközben a nélkülözhetetlen funkciókat, állapotgépeket, valamint felületet biztosít az alkalmazásoknak a protokoll alsóbb rétegei által megvalósított funkciók eléréséhez. A ZDO mellett találhatók az alkalmazásspecifikus funkciókat megvalósító Application Object-ek, melyek 1-től 240-ig tetszőleges végponton helyet kaphatnak.

2.3.1. IEEE 802.15.4

A [4] szabvány által tárgyalt WPAN legalapvetőbb komponensei az eszközök, melyek két típusa a Full Function Device (FFD) és a Reduced Function Device (RFD). Az FFD három működési módja a PAN koordinátor, a koordinátor és az eszköz. Az FFD képes kommunikálni RFD-vel és másik FFD-vel, az RFD viszont csak FFD-vel. Minden eszköznek rendelkeznie kell egy globálisan egyedi 8 bájtos azonosítóval, amivel egyértelműen azonosítható. Ezt a 8 bájtos azonosítót többféleképpen is nevezik, munkám során az alábbiakkal találkoztam: IEEE, EUI64, MAC vagy hosszú cím. Én ezek közül az IEEE cím elnevezést fogom használni.

Hálózat (PAN) alkotásához legalább 2 eszköz kell, melyek ugyanazt a fizikai csatornát használják és egymás rádióinak hatótávolságában vannak. Minden PAN tartalmaz pontosan egy FFD-t, ami PAN koordinátor funkciót lát el. Hálózat létrehozásakor a PAN koordinátor választ egy 2 bájtos azonosítót (PAN ID), ami a fizikai csatornán belül egy logikai hálózatot jelöl ki. Így akár több logikai hálózat is képes ugyanazon a fizikai csatornán működni anélkül, hogy zavarnák egymás kommunikációját. A többi eszköz a PAN-hoz csatlakozva szintén kap egy 2 bájtos azonosítót a PAN koordinátortól, a PAN-on belül ennek segítségével történik a címzés. Az IEEE címet csak indokolt esetben kell használni, mivel ez növeli a forgalmat.

Ezen a szinten két topológiát különböztet meg a szabvány: a csillag és a peer-to-peer topológiát. A két topológia közti különbségeket szemlélteti a 2.4. ábra.



2.4. ábra. *A csillag és a peer-to-peer topológia [4].*

A csillag topológia esetén minden eszköz csak a PAN koordinátorral válthat közvetlenül üzenetet. A csillag topológia előnye, hogy a végletekig leegyszerűsíthető vezeték nélküli hálózatokat alkothatunk vele. Ezzel szemben a peer-to-peer topológia lehetővé teszi, hogy bármelyik eszköz kommunikálhat bármelyik hatótávolságában lévő eszközzel. A peer-to-peer topológiát használja a ZigBee is mesh hálózatokhoz, kiegészítve a magasabb absztrakciós rétegek nyújtotta lehetőségekkel.

Fizikai réteg (PHY)

Rádió aktiválás és deaktiválás A szabványt alacsony fogyasztású alkalmazásokhoz optimalizálták, hogy a berendezés elemes táplálás esetén akár éveken keresztül tudjon

működni karbantartási feladatok nélkül. A rádió fogyasztása vétel közben a 10 mA nagyságrendbe eshet, ezért bizonyos alkalmazások megkövetelik, hogy a rádió csak alkalmanként kapcsoljon be, az idő nagy részében pedig kikapcsolt állapotban legyen. A rádió be- illetve kikapcsolása a PHY réteg feladata.

Csatornaválasztás A [4] szabványnak megfelelő berendezéseknek legalább egy frekvenciasávot kell támogatnia a 2.5. ábrán felsoroltak közül a feltüntetett modulációs paraméterekkel.

PHY (MHz)	Frequency band (MHz)	Spreading parameters		Data parameters		
		Chip rate (kchip/s)	Modulation	Bit rate (kb/s)	Symbol rate (ksymbol/s)	Symbols
868/915	868–868.6	300	BPSK	20	20	Binary
	902–928	600	BPSK	40	40	Binary
868/915 (optional)	868–868.6	400	ASK	250	12.5	20-bit PSSS
	902–928	1600	ASK	250	50	5-bit PSSS
868/915 (optional)	868–868.6	400	O-QPSK	100	25	16-ary Orthogonal
	902–928	1000	O-QPSK	250	62.5	16-ary Orthogonal
2450	2400–2483.5	2000	O-QPSK	250	62.5	16-ary Orthogonal

2.5. ábra. *Megengedett frekvenciasávok és modulációs módszerek [4].*

A ZigBee eszközök szinte kizárólag a 2450 MHz-es sávközéppel rendelkező ISM (Industrial, Scientific, Medical) sávot használják, mivel ez nemzetközi viszonylatban engedély nélkül használható. Ugyanakkor a spektrum korlátos erőforrás, aminek hatékony, zavarásoktól mentes használatát nemzetközi frekvenciakoordináció keretében ellenőrzik. Hazánk, csatlakozva az Európai Unió szabályozásaihoz rögzítette az ISM sávban működő rövid hatótávolságú rádiós eszközök maximális kimeneti teljesítményeit a 35/2004. (XII. 28.) IHM rendelet 4. számú mellékletében, miszerint a 2400-2454 MHz tartományban 100 mW EIRP[1], a 2400-2483,5 MHz tartományban 10 mW EIRP a megengedett legnagyobb teljesítmény.

A [4] szabványban a 2450 MHz-es sávban 11-től 26-ig terjed a csatornák számozása, sávszélességük egyenként 2 MHz, a csatornák frekvenciaközepét az

$$f_c = 2405 + 5(k - 11)MHz, \tag{2.1}$$

formula határozza meg, ahol $k = 11, 12, ..., 26$.

Az ISM sáv a szabad használat miatt meglehetősen szennyezett, a ZigBee szempontjából a Wi-Fi jelenti a legnagyobb zavarforrást. Az interferencia elleni egyik védekezési módszert a ZigBee PRO Feature Set-ben vezették be, amely lehetővé teszi, hogy az egyébként fix csatornán működő PAN hálózat működés közben csatornát váltson. A módszer neve *Frequency Agility*.

[1] EIRP = Equivalent Isotropically Radiated Power

Energy detection (ED) PAN létrehozásakor a megfelelő csatorna kiválasztásához ellenőrizni kell az adott csatornán mérhető energiaszintet. A felsőbb protokollrétegek számára a PHY végzi el ezt a mérést, legalább $\pm 6dB$ pontossággal. A mérés során csak az energiaszint számít, a jel egyéb jellemzői nem.

Link Quality Indicator (LQI) A PHY minden beérkezett csomaghoz kiszámolja a csomag minőségét jellemző mennyiséget, az LQI-t. Ennek implementációja történhet ED, SNR becslés, esetleg ezek kombinációja alapján. Az LQI a csomagok útvonalának megtervezésénél fontos mesh hálózatokban.

Clear Channel Assessment (CCA) A rádiós csomagok ütközésének elkerüléséhez a [2] szabvány CSMA-CA (Carrier Sense Multiple Access with Collision Avoidence) módszert használ. Ehhez meg kell tudni állapítani, hogy van-e aktív forgalom a csatornán, vagy szabad a csatorna. Erre szolgál a CCA, mely három módon történhet:

1. Az ED eredményének összehasonlítása a küszöbértékkel.
2. A [4] szabvány szerinti modulációt használó jel detektálása.
3. Az 1. és 2. módszer kombinációja.

Adat küldése és fogadása Mind a bit-, mind a bájtsorrend little-endian (legkisebb helyiértékű bit/bájt először).

2.1. táblázat. *A fizikai réteg keretformátuma [4].*

Octets: 4	**1**	**1**		**variable**
Preamble	SFD	Frame length (7 bits)	Reserved (1 bit)	PSDU
SHR		PHR		PHY payload

Az üzenet keretformátuma (2.1. táblázat) egy szinkronizációs fejléccel kezdődik, mely a vevő oldali szinkronizációt szolgálja. Erre a 32 db nullás bitből álló preamble (bevezető) és a szinkronizációs fejléc végét jelző SFD (start-of-frame delimiter) bájt szolgál. A PHY fejléc által hordozott egyetlen információ a hasznos üzenet (payload) hossza 7 biten ábrázolva. A legrövidebb üzenet a MAC réteg nyugtázó üzenete, ami 5 bájtos, a leghosszabb üzenet 127 byte lehet. Az SHR-rel és PHR-rel együtt ez összesen 133 bájt, ami 250 kb/s sebességnél 532 μs, vagyis a leghosszabb üzenet is alig több mint fél ms ideig veszi igénybe a csatornát.

Adatkapcsolati réteg (MAC)

Beacon, superframe A beacon egy kitüntetett csomagkeret, melyet csak FFD küldhet. A beacon-vezérelt hálózatokban ezek a csomagok határolják a superframe-nek nevezett egységeket. A PAN koordinátor által periodikusan küldött beaconök egyik feladata, hogy leírja a superframe-ek struktúráját. A superframe 16 egyenlő nagyságú időszeletre oszlik, melyen belül a PAN koordinátor garantált időszeleteket (guaranteed time slot, GTS)

biztosíthat egyes eszközöknek. A beacon inaktív periódust is meghatározhat két superframe között, amikor nincs forgalom a csatornán. A beacon-vezérelt hálózatoknak csak olyan alkalmazások esetén van létjogosultságuk, ahol pontos szinkronizáció szükséges az eszközök között, vagy ha kicsi a megengedett késleltetés.

A nem beacon-vezérelt hálózatok (ilyen pl. a ZigBee mesh is) szintén használnak beacont, de csak a hálózat felderítésekor.

CSMA-CA csatornahozzáférés Ahogy a PHY rétegnél is említettem, az üzenetek ütközésének elkerülésére használt módszer a CSMA-CA, ami majdnem minden üzenet küldésénél kötelező. Ha a csatorna foglaltnak bizonyul, a küldési kísérletet véletlenszerű idő elteltével megismétli az eszköz. Nem kell viszont CSMA-CA módszert alkalmazni beacon küldésekor illetve GTS alatt.

PAN kezelés Bár a PAN létrehozása a PAN kiválasztása a MAC réteg feladata, arról a felsőbb rétegekben születik meg a döntés, hogy mi alapján választ ki a PAN koordinátor egy csatornát PAN létrehozásához, és arról is, hogy csatlakozás során melyik PAN-t választja ki egy eszköz. A PAN karbantartásával kapcsolatosan még jó pár teendője lehet egy eszköznek: csatorna szkennelés, PAN ID konfliktus feloldása, PAN elhagyása.

Biztonsági szolgáltatások Az LR-WPAN hálózatokat ugyanazok a veszélyek fenyegetik, mint bármilyen vezeték nélküli kommunikációt: a lehallgatás és az illetéktelen beavatkozás. A korlátozott erőforrások (számítási teljesítmény, tárhely, tápellátás) miatt azonban talán ezek a hálózatok a legvédtelenebbek a támadásokkal szemben. A MAC réteg az alábbi biztonsági szolgáltatásokat nyújtja a felsőbb protokollrétegeknek:

- *Adatok bizalmassága (data confidentiality):* biztosítja, hogy az elküldött üzenet csak a címzettek számára legyen felfedhető. Ehhez szimmetrikus kulcsú AES-128 titkosítást ír elő a szabvány.
- *Adatok hitelessége (data authenticity):* biztosítja, hogy az üzenetet tartalma nem változott az átvitel során.
- *Visszajátszás elleni védelem (replay protection):* biztosítja, hogy ne kerülhessen feldolgozásra kétszer ugyanaz az üzenet.

Adatátvitel két eszköz között A MAC réteg üzenete a PHY réteg payload-ja. A MAC üzenetkeretek (2.2. táblázat) fejlécének elején található Frame Control mező az üzenet többi részének értelmezésében segít, megadja többek közt az üzenet típusát.

A MAC üzeneteknek 4 típusa van, ezek közül az egyik a már említett *Beacon frame*. Általános adatküldésre a *Data frame* típusú üzenet szolgál, a küldő és a fogadó eszköz azonosítóit a címzésre szolgáló mezőkben (Addressing fields) kell megadni. Az *Acknowledgement frame* az üzenet sikeres megérkezésének nyugtázására szolgáló üzenettípus. A nyugtázás nem kötelező, de ha egy nyugtázandó üzenet megérkezéséről egy adott időn belül nem érkezik meg az értesítés, a küldő eszköz újra próbálkozik. A

2.2. táblázat. *Általános MAC keretformátum [4].*

Octets: 2	**1**	**0/2**	**0/2/8**	**0/2**	**0/2/8**	**0/5/6/10/14**	**variable**	**2**
Frame Control	Sequence Number	Destination PAN Identifier	Destination Address	Source PAN Identifier	Source Address	Auxiliary Security Header	Frame Payload	FCS
		Addressing fields						
MHR							MAC Payload	MFR

nyugtázandó és nyugtázó üzenetek párosítása a fejlécben található Sequence Number mező alapján történik. A negyedik üzenettípus a *MAC command frame* a MAC réteg funkcióinak vezérlésére szolgál.

A biztonsági szolgáltatások a Frame control mezőben engedélyezhetők, ekkor a MAC fejléc kiegészül az Auxiliary Security Header-rel. A beérkezett üzenetek bithibáinak detektálására egy 16 bites CRC kód, a keret láblécében található FCS (frame check sequence) szolgál.

2.3.2. ZigBee specifikáció

A ZigBee más terminológiát használ az az eszközökre, mint az IEEE 802.15.4 szabvány (2.3. táblázat).

2.3. táblázat. *A ZigBee hálózatot alkotó eszközök.*

IEEE 802.15.4 eszköz	ZigBee szerep	ZigBee rövidítés	eszköz típusa
PAN koordinátor	ZigBee coordinator	ZC	FFD
koordinátor	ZigBee router	ZR	FFD
eszköz	ZigBee end device	ZED	RFD/FFD

Az eszközök háromféle topológia szerint alkothatnak hálózatot (2.4. táblázat).

2.4. táblázat. *A ZigBee hálózati topológiái.*

ZigBee topológia	IEEE 802.15.4 topológia	beacon-vezérelt	megkötés a komm. irányára
csillag	csillag	opcionális	csak a ZC-vel
fa	peer-to-peer	opcionális	hierarchikus
mesh	peer-to-peer	nem	nincs

Hierarchikus fa topológiájú hálózatok a *parent-children* (szülő-gyerek) kapcsolaton alapszanak. Minden parent csak korlátozott számú children-t tud kezelni, ezért egy hierarchiaszint meghatározott véges számú eszközt fog tartalmazni, amit hálózati cím allokálásakor ki is használhatunk. A ZigBee PRO feature set csak mesh topológiát használ, ebben nincs semmilyen megkötés a hálózat felépítésére vonatkozóan.

Hálózati réteg (NWK)

PAN kezelése A NWK réteg irányítja a hálózathoz csatlakozást (*joining*) és leválást (*leaving*), ZC esetén a PAN létrehozását (*starting*). A döntések meghozatalához az alsóbb rétegek szolgáltatásait használja.

Adatátvitel eszközök között útvonalválasztással A MAC réteghez hasonlóan a NWK kéreg kerete (2.5. táblázat) is tartalmaz egy kezdő és egy végcímet, de itt már nem elvárás, hogy az üzenet feladója és címzettje egymás rádiójának hatótávolságában legyen. Az üzenet ilyenkor csomópontról csomópontra halad a hálózat topológiájának megfelelő útvonalválasztási szabálynak megfelelően. A köztes csomópontok listáját a keret Source route subframe része tartalmazza. Az ugrások maximálás számát a Radius paraméter szabja meg, mely minden ugrás alkalmával dekrementálódik. Ha a célcím egy konkrét csomópontot jelöl, akkor *unicast* kommunikációról beszélünk. Az üzenetküldés másik módja a *broadcast*, ilyenkor a célcím a 0xffff értéket veszi fel. Broadcast üzenet érkezésekor minden ZC és ZR továbbítja azt a hálózatban, amíg a Radius 0 nem lesz. Az üzenetküldés harmadik módja a *multicast*, amivel csomópontok meghatározott csoportja címezhető. Minden eszköz nyilvántartja magáról, hogy melyik csoportnak tagja és ez alapján értelmezi a multicast üzenetet.

2.5. táblázat. *Általános NWK keretformátum [5].*

Octets: 2	**2**	**2**	**1**	**1**	**0/8**	**0/8**	**0/1**	**Variable**	**Variable**
Frame control	Destination address	Source address	Radius	Sequence number	Destination IEEE Address	Source IEEE Address	Multicast control	Source route subframe	Frame payload
NWK Header									Payload

Összeségében a NWK réteg feladata üzenetküldésekor az útvonal kiválasztása, ami a ZigBee specifikáció egyik erőssége.

Alkalmazástámogató alréteg (APS)

Binding A binding segítségével két eszköz endpointja köthető össze és érhető el egyszerűen, az eszközön belüli címzés egyik hatékony eszköze. Az APS réteg a Binding Table-ben tartja nyilván az összerendelt endpointokat.

Address Map A ZigBee általában a 16 bites rövid címet használja címzéshez, de vannak üzenetek, amik ennél pontosabb címzést, IEEE címet követelnek meg. Az Address Map Table-ben összerendelt IEEE cím-NWK cím párok találhatók, a táblázat kezelése az APS feladata.

Group Addressing A NWK rétegnél említett multicast üzenetküldéshez az APL réteg tartja nyilván a csoportokat.

Adatátvitel clusterek között Az APS üzenetkeretével (2.6. táblázat) elérkeztünk a legkisebb címezhető egységig, a clusterig. A protokollon átívelő címzési lánc tehát: PAN ID (MAC) $\rightarrow$ hálózati cím (NWK) $\rightarrow$ endpoint (APS) $\rightarrow$ cluster(APS).

2.6. táblázat. *Általános APS keretformátum [5].*

Octets: 1	**0/1**	**0/2**	**0/2**	**0/2**	**0/1**	**1**	**0/ Variable**	**Variable**
Frame control	Destination endpoint	Group address	Cluster identifier	Profile identifier	Source endpoint	APS counter	Extended header	Frame payload
	Addressing fields							
APS header								APS payload

2.4. A Smart Energy Profile sajátosságai

Az eddig bemutatott protokollból látható, hogy a ZigBee specifikáció alapján nagyon eltérő hálózatokat lehet alkotni. A stack tele van opciókkal és paraméterekkel, melyek megválasztása az alkalmazási területtől függ. Létrehozhatunk néhány node-ból álló beacon-vezérelt csillag hálózatot, de szükség lehet akár több ezer node-ot magába foglaló mesh hálózatra is. A Smart Energy Profile célja, hogy a ZigBee specifikációt a smart gridhez adaptálja. Az opciókat olyan halmazra szűkíti le, amivel garantálható a készülékek hálózaton belüli megbízható együttműködése. A StickBee-ben alkalmazott paraméterekre és más SEP specifikus részletekre a 4. fejezetben térek ki részletesebben.

Biztonság A SEP egyik vonzó tulajdonsága a közműszolgáltatók szempontjából, hogy számlázás alapjául szolgáló adatokat tudnak vele távolról lekérdezni. Az adatok meghamisításának elkerülése érdekében a szolgáltató szeretné garantálni, hogy csak az általa megbízhatónak ítélt készülékek csatlakozhassanak a hálózathoz. Minden Smart Energy Profile-lal rendelkező eszközt ezért kötelező ellátni egyedi hitelesítő adattal. A ZigBee hálózatot és a szolgáltatót az ESI köti össze, ez az eszköz minden SEP hálózatban megtalálható. Kitüntetett szerepe miatt az ESI felel a csatlakozni kívánó eszközök hitelességének ellenőrzéséért is, amit a ZigBee *Trust Center*-ek (TC) nevez. Csak az az eszköz kaphatja meg a megfelelő titkosító kulcsot a hálózatban, amelyik megfelelő hitelesítő adattal rendelkezik. A hitelesítő adat ellenőrzésének folyamatát a szabvány *Certification Based Key Establisment*-nek (CBKE) nevezi.

Device Profile-ok Egy SEP hálózatot az alábbi eszközök alkothatják:

- *Energy Service Interface (ESI):* A SEP hálózatban központi betöltő készülék. Ő a ZC, a TC és általában a villanyóra is egy eszközben, emellett gateway szerepét is betölti.
- *Metering Device:* Általános mérőeszköz, feladata a mérés elvégzése, az adatok naplózása és kiszolgálása.
- *In-Premise Display (IPD):* Aktuális és hisztorius fogyasztási adatok, tarifák, fontos szolgáltatói üzenetek megjelenítésére szolgáló eszköztípus, célja a fogyasztó tájékoztatása. Ezt az eszköztípust valósítja meg a StickBee is.

- *Programmable Communicating Thermostat (PCT):* Fűtés illetve légkondícionáló vezérlésére alkalmas eszköz, melynek érdekessége, hogy a szolgáltató indokolt esetben beavatkozhat a működésébe. A kezelőfelületek jobb kihasználása miatt gyakran egy fizikai készülékbe kerül az IPD-vel.

- *Load Control Device:* Nagy fogyasztók szolgáltató általi vezérlésére szolgáló eszköz, segítségével pl. vízmelegítőket lehet kikapcsolni a fogyasztási csúcsok ideje alatt.

- *Smart Appliance:* A háztartási készülékek egy része felruházható a SEP hálózatok adatait kihasználó intelligenciával. Tipikus példája ennek, hogy az előre betöltött mosógép egy megadott intervallumon belül akkor mos, mikor legkedvezőbb a tarifa.

- *Prepayment Terminal:* Olyan helyen használják, ahol a szolgáltatónak előre kell fizetni a közműszolgáltatásért. Különböző fizetési módokat támogathat, kijelzi a hátralévő fogyasztási keretet és figyelmeztető üzeneteket.

Clusterek A SEP által használt clusterek egy része a ZCL-ből származó általános cluster, másik részét maga a SEP definiálja. A SEP clusterek listájából az alábbiakat szeretném külön kiemelni.

- *Time:* Ez a ZCL cluster szolgál a SEP hálózatban az ESI, mint referenciaóra RTC-jének lekérdezésére. Az aktuális idő a 2000. január 1. 0:00 óta eltelt másodpercek formájában érhető el, emellett vannak még időzónaadatot és nyári időszámítást támogató attribútumai.

- *Simple Metering:* A mérőóra attribútumait tartalmazó cluster, kötelezően előír olyan attribútumokat mint a fogyasztás aktuális állása, a mérő típusa, vagy a mértékegység.

- *Messaging:* Ez a cluster 4 parancsot tartalmaz, kettőt szerver oldalon, kettőt kliens oldalon. A szerver által küldött parancsok az üzenet küldése és üzenet visszavonása, a kliens parancsai az aktuális üzenet nyugtázása és a legutóbbi üzenet lekérdezése.

- *Key Establishment:* A SEP hálózat szigorú biztonsági követelményei előírják, hogy miden egymással kommunikálni kívánó eszközöknek páronként közös kulccsal kell rendelkezniük. A Key Establishment cluster ennek kezelésére hivatott.

3. fejezet

A fejlesztői környezet bemutatása

Minden ZigBee termék alapja egy tanúsítással rendelkező ZigBee platform. A ZigBee platformot rádió és beágyazott ZigBee firmware-t futtató mikrokontroller alkotja. Egy termék minőségét alapjaiban határozza meg a választott ZigBee platform minősége, ezért a termékfejlesztők jogos elvárása az, hogy megbízható, jól dokumentált, szoftver- és hardvereszközökkel teljes körűen támogatott platformot kapjanak. Ebben a fejezetben azt mutatom be, hogy a Freescale milyen környezetet biztosít ZigBee termékek fejlesztéséhez.

3.1. Hardver

A Freescale platformja már a kezdetektől jelen van a ZigBee piacon. Kínálatában először csak 8 bites HCS08 mikrokontroller alapú megoldások voltak. Ahogy a ZigBee szabványok egyre összetettebbek lettek, erőforrásigényük megnőtt. Ezzel párhuzamosan a 32 bites kontrollerek is fejlődtek, és ár/fogyasztás szempontból is összehasonlíthatóvá váltak a 8 bites társaikkal. A Freescale 2008-ban dobta piacra a 32 bites ARM7 magra épülő SoC-jét, az MC1322x családot.

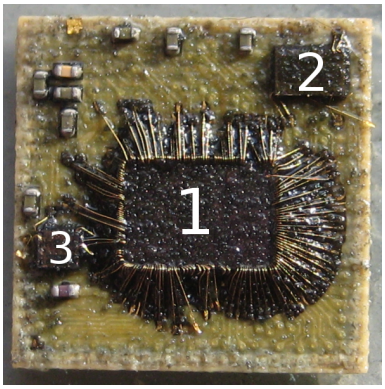
3.1.1. MC1322x SoC

Az MC1322x családnak két tagja van, az MC13224V és az MC13226V. Hardveres szempontból a két SoC megegyezik: 32 bites ARM7TDMI-S mag, 96 kB RAM, 80 kB ROM, 128 kB soros FLASH, egyedi *99 pin LGA Platform-in-Package* tokozás. A ROM gyárilag programozott bootloader-t, drivereket és más szoftverkomponenseket tartalmaz, a két SoC egyedül ebben különbözik egymástól (3.1. táblázat). Az MC13226V esetén az új ROM kiosztásra azért volt szükség, hogy a ZigBee PRO alkalmazások (tipikusan a Smart Energy alkalmazások) nagyobb RAM igényét kielégíthessék.

3.1. táblázat. *Az MC1322x különbségek összehasonlítása.*

Szoftver komponens	MC13224V	MC13226V
BeeStack funkciók	RAM	ROM
ADC driver	ROM	RAM
LCD font	ROM	RAM
SSI driver	ROM	RAM

Perifériák Az MC1322x SoC-be mindent belezsúfoltak, amit lehetett, ezért egy 50 ohmos antennával és egy 24 MHz-es kvarccal már működőképes áramkör hozható létre. A tokozásától megfosztott chipen (3.1. ábra[1]) jól látszik, hogy a hordozó nyomtatott huzalozású lapon 3 félvezető lapka és passzív alkatrészek is helyet kaptak.



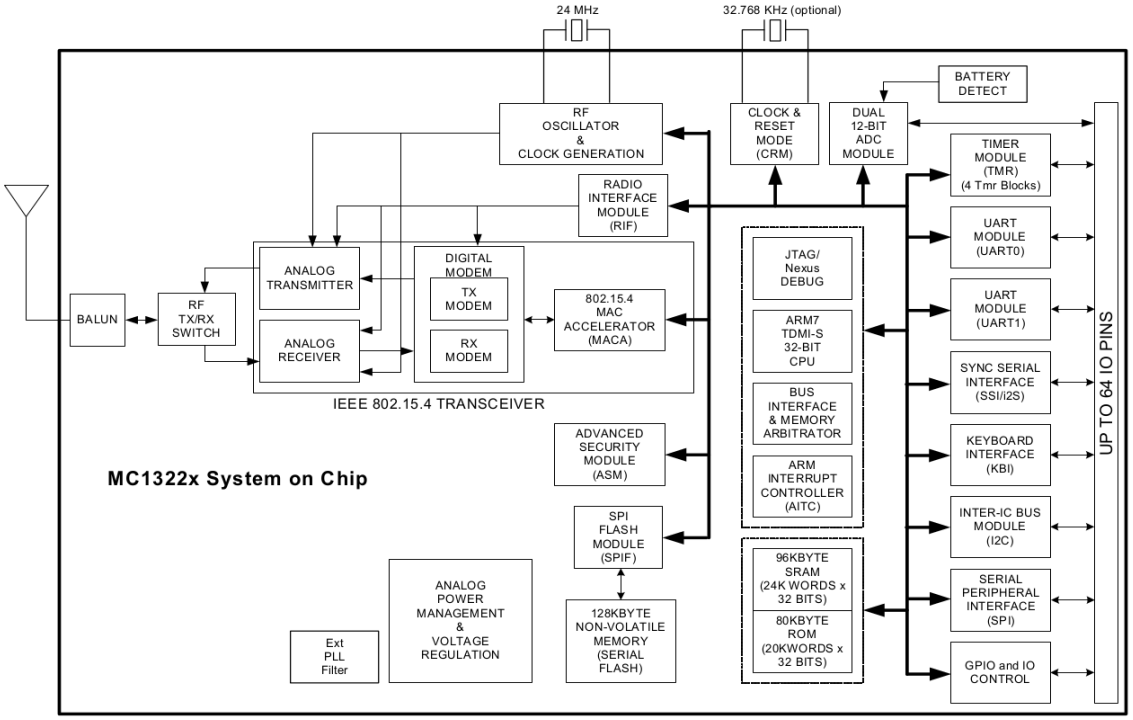
3.1. ábra. *Az MC1322x SoC tokozás nélkül.*
1: MCU+rádió, 2: soros elérésű FLASH, 3: balun transzformátor

Az MC1322x blokkvázlatán (3.2. ábra) látható, hogy gazdag perifériakészlettel rendelkezik. Az általános perifériák mellett olyan extrák is belekerültek, amik kifejezetten az elemes tápellátást támogatják:

- low power processzor módok: reset, hybernate, doze
- opcionális RTC kvarc
- 4 RAM blokk, amik külön ki/be kapcsolhatók
- opcionáis buck feszültségszabályzó
- alacsony tápfeszültség detektálás beépített referenciafeszültséggel
- IEEE 802.15.4 MAC protokollt és titkosítást támogató hardvermodulok (MACA és ASM)

Debug portok A SoC két debug porttal rendelkezik: az egyik a kevésbé népszerű Nexus, másik a széles körben alkalmazott JTAG. A JTAG szabványt eredetileg azért fejlesztették ki, mert az egyre kisebb tokozású IC-k és a többrétegű nyomtatott huzalozású áramköri panelek elterjedése miatt egyre körülményesebbé vált a gyártás során az alkatrész-összeköttetések vizsgálata. A JTAG a *boundary scan* technológiával, 4 jelvezetéken keresztül, a kompatibilis eszközöket láncba fűzve képes alapvető hardver tesztek elvégzésére. Később a hardver tesztek mellett már debug szolgáltatásokkal is bővült a JTAG, és eredeti funkciója háttérbe szorult. Napjainkban a JTAG az egyik legalapvetőbb debug port, a PC-s fejlesztőkörnyezet és a cél hardver között in-circuit debuggerek (más néven JTAG adapterek) teremtenek kapcsolatot. Az MC1322x-ben található JTAG modul

[1] http://www.flickr.com/photos/travisgoodspeed/5540047315



3.2. ábra. *Az MC1322x blokkvázlata [11].*

sajnos nem támogatja a boundary scan technológiát, csupán debugolásra és a firmware letöltésére szolgál. Az MC1322x-t a SEGGER J-Link adapter támogatja.

3.1.2. MC1322x fejlesztő kit

A Freescale által tervez(tet)ett fejlesztői panelek (development board) lerövidítik a fejlesztési időt és referenciaként szolgálnak saját hardver megtervezéséhez. A paneleket 2 rétegű NYÁK-ra tervezték, műanyag burkolatuk van, és mindegyik kapott egyedi IEEE címet, ami egy matricáról leolvasható. Perifériakészletület tekintve széles skálán mozognak, de az UART1 interface majdnem mindegyiken egy FTDI USB átalakítón keresztül érhető el.

USB Dongle (sniffer) A legkevesebb perifériával ellátott panel, csak az FTDI UART1 interface, egy reset nyomógomb, egy tápfeszültségjelző LED és egy GPIO LED található rajta. Kerámia chip antennával tervezték, mert ez a leghelytakarékosabb. Feladata, hogy belehallgasson a rádiós kommunikációba, erről részletesebben az 5.2.1 alfejezetben írok.

Network Node Ez a legtöbb perifériával ellátott panel, legszembetűnőbb sajátossága a monokromatikus, 128×64 pixel felbontású LCD. A 4 nyomógomb, 4 GPIO LED, az LCD és a JTAG csatlakozó miatt ideális bármilyen ZigBee alkalmazás fejlesztéséhez. Antennája egy SMA koaxiális csatlakozóhoz illeszkedik, de a panelen megtalálható egy invertált F antenna (IFA) is. Ezen a panelen fejlesztettem a StickBee alkalmazást (3.3. ábra).

3.3. ábra. *MC1322x Network Node.*

Sensor Node, Low Power Node A két véglet közt helyezkedik el ez a két panel. A Low Power Node RTC kristállyal is rendelkezik, ezért ez a legalkalmasabb elemes alkalmazások fejlesztéséhez. Mindkét panel IFA-val rendelkezik.

3.2. Beágyazott szoftver

A ZigBee platformmal rendelkező gyártókra általában jellemző, hogy a ZigBee mellett más, az IEEE 802.15.4 szabványra épülő, különböző erőforrás-igényű és komplexitási szinttel bíró protokollt is támogatnak. A Freescale-nél a támogatott protokollok közt található a SynkroRF, SimpleMAC, MAC, BeeStack Consumer és a BeeStack. A Freescale a beágyazott szoftvercsomagokra codebase-ként hivatkozik. Minden processzortípushoz, rádióhoz és protokollhoz külön codebase tartozik, jelenleg összesen 12 codebase érhető el 3 különböző chiphez. Mindegyik codebase-t C nyelven írták.

A *BeeKit ARM7 BeeStack Codebase 3.0.12* codebase-t használtam, melyet 2012-ben adtak ki. A BeeStack a [5] valamint a [8] szabványoknak felel meg. A függvények, adattípusok, változók, makrók elnevezésénél ügyeltek arra, hogy - ahol csak lehet - kövessék a szabványokban használatos terminológiát, azonban a szabványok változásai miatt előfordulnak eltérések. A BeeStack jelentős részét csak előre fordított állományok, ún. library-k formájában publikálták, emiatt csak egy meghatározott fordító használata esetén garantálják a helyes működést. A BeeStack az alábbi komponenseket tartalmazza:

- ZigBee 2007 protokoll stack, Stack Profile 1 (ZigBee Feature Set) és Stack Profile 2 (ZigBee PRO Feature Set) támogatással
- 802.15.4 MAC/PHY
- platform API (PLM)
- szoftver támogató modulok (SSM)
- ZigBee Cluster Library (ZCL) infrastruktúra
- Home Automation 1.x (HA) Application Profile példa alkalmazásokkal

- Smart Energy 1.x (SE) Application Profile példa alkalmazásokkal
- Health Care (HC) alkalmazások
- általános (szabványos profilba nem sorolható) ZigBee alkalmazások
- BlackBox alkalmazások

Ezek közül az SSM szorul bővebb magyarázatra, melynek részei a task ütemező (TS) és a ZigBee Test Client (ZTC).

Task ütemező A BeeStack nem használ operációs rendszert, a taskokat egy egyszerű prioritásos, nem-preemptív ütemező kezeli. A taskok száma fordítási időben dől el, alapértelmezésben az alábbiakat használja a BeeStack:

- *idle task:* ez az üres task fut ha épp semmi más nem fut
- *MAC task:* a MAC réteg rutinjait szolgálja ki
- *NWK task:* a network réteg rutinjait szolgálja ki
- *ZDO task:* a ZigBee Device Object rutinjait szolgálja ki
- *APS task:* az Application Support Sub-layer rutinjait szolgálja ki
- *AF task:* az application framework rutinjait szolgálja ki
- *PLM task:* a platform management rutinjait szolgálja ki
- *Application Task:* a felhasználói alkalmazást szolgálja ki

Az ütemező nem preemptív, ezért a programozó felelőssége, hogy az alkalmazás task ne éheztesse ki a többi taskot. A Freescale ajánlása szerint a taskok 2 ms-nál rövidebb idő alatt kell, hogy lefussanak. Ezt figyelembe véve kellett megtervezni és állapotgéppel több szakaszra osztani a hosszabb futási idejű függvényeket.

A taskok közti kommunikáció események küldésén keresztül lehetséges. Minden taskhoz összesen 16 esemény tartozhat, amiket egy flag beállításával aktiválunk. Amikor egy task futási jogot kap, sorra ellenőrzi a flageket, és szükség esetén meghívja a hozzájuk tartozó kiszolgáló rutint.

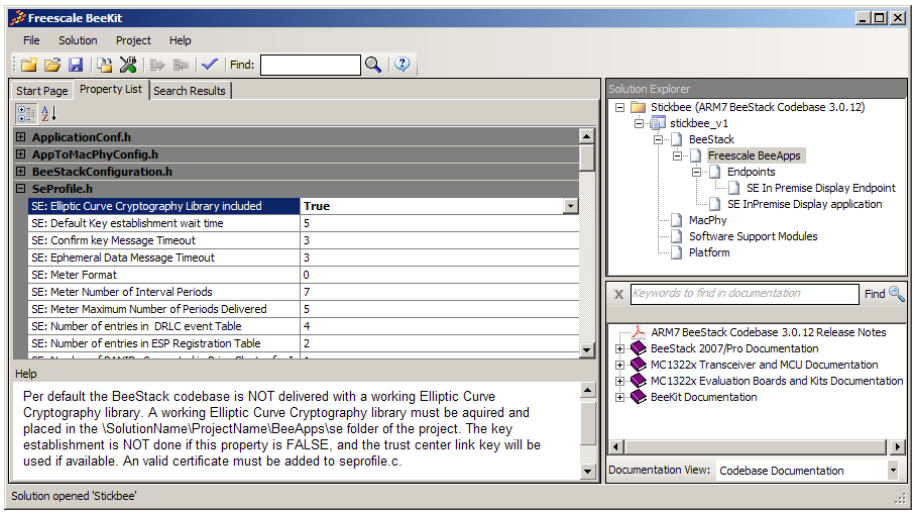
ZigBee Test Client A BeeStack-ben minden protokollréteghez tartozik egy ZTC modul, amivel az adott réteg SAP-jaihoz férhetünk hozzá. A ZTC lehetővé teszi, hogy kívülről vezéreljük a BeeStacket az IC valamelyik soros portján érkező hívásokkal. A host eszköz lehet a PC vagy egy mikrokontroller is. A ZigBee SoC ilyenkor *fekete dobozként* viselkedik, a ZTC API-n keresztül kiszolgálja a hívásokat, elvégzi a megfelelő ZigBee-specifikus feladatot, annak eredményét pedig a host eszközre bízza. A Freescale BlackBox alkalmazásnak hívja, ha az eszközön nem fut semmilyen felhasználói alkalmazás, csak a ZTC kliens. Ezt használja több PC-s teszt program is (lásd: 3.3.2 alfejezet).

3.3. PC szoftver

A Freescale Windws alapú toolchaint biztosít a ZigBee projektek fejlesztéséhez. Az legfrissebb PC-s szoftverek, beágyazott szoftverek és a dokumentációk mind 1 telepítőben letölthetők a Freescale honlapjáról.

3.3.1. BeeKit

A BeeKit új projektek létrehozására szolgál. Valójában nem csinál mást, mint testreszabja a kiválasztott codebase-t. A BeeStack codebase-t választva először a projekt típusát kell megadni: Home Automation, Smart Energy, Health Care, Generic, és különböző BlackBox projektek közül választhatunk. Ez után egy sablot kell választani, pl. Smart Energy projekt típus esetén az Se InPremiseDisplay sablont. Projektünk kiindulópontja tehát minden esetben egy korlátozott funkcionakitású demó alkalmazás lesz.



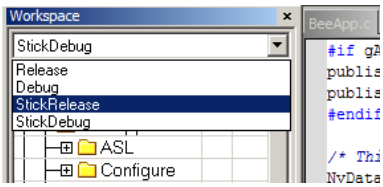
3.4. ábra. *BeeStack projekt konfigurálása a Beekit-ben.*

A sablon kiválasztása után egy néhány lépésből álló varázsló indul el, ami a projekt legfontosabb beállításain vezet végig. Ennek során konfigurálhatjuk a hardvert és a ZigBee stack legalapvetőbb paramétereit. A varázsló akár át is ugorható, hiszen egy valós alkalmazás ennél jóval több beállítást igényel. A varázsló után a főablak nagy részét kitevő `Property list` fülön kategóriákra bontva találjuk a többi beállítási lehetőséget (3.4. ábra). A legtöbb paraméter header állományokban állít meghatározott paramétereket, ez alól két kategória jelent kivételt:

1. A BeeStack kategóriában beállítható paraméterek (a hálózati kulcsoktól eltekintve) közvetlenül a fordító preprocesszorának definiálnak különböző értékeket (pl. az IC konkrét típusát, a fejlesztői panelt, az alkalmazott biztonsági szintet, stb.). Ez lehetővé teszi, hogy a fordítóban egy projekten belül több konfigurációt hozzunk létre (3.5. ábra).

2. Az endpoints kategóriában a sablon alkalmazáshoz tartozó alapértelmezett endpoint mellé adhatunk még más endpointokat. A ZigBee specifikáció ugyan 240 különböző

endpointot is megenged egy fizikai eszközön, a gyakorlatban mégis az eszközönkénti egy endpoint jellemző.



3.5. ábra. *Projektkonfiguráció kiválasztása a fordítóban.*

A korlátozott RAM méret miatt fontos, hogy ki legyen kapcsolva minden olyan opció, ami nem kell a projektbe. A beállítások végeztével exportálni kell a projektet. Ilyenkor a program a projekt mappájába másolja a codebase szükséges részeit, alkalmazza a beállítás során elvégzett módosításokat a header állományokban, és generál egy projekt leíró állományt a fordítónak.

3.3.2. Test Tool

A Test Tool a Freescale segédalkalmazásainak gyűjtőhelye, legtöbbjük valóban tesztelési célokat szolgál.

Command Console ZTC parancsok manuális küldésére illetve fogadására szolgál soros porton keresztül. Segítségével kisebb hálózatokat lehet konfigurálni, tesztelni. Az vezérelni kívánt eszközöknek tartalmazniuk kell a ZTC modult.

Script Server Az előző program automatizált változata, a ZTC üzeneteket Python scriptek küldik ki illetve dolgozzák fel.

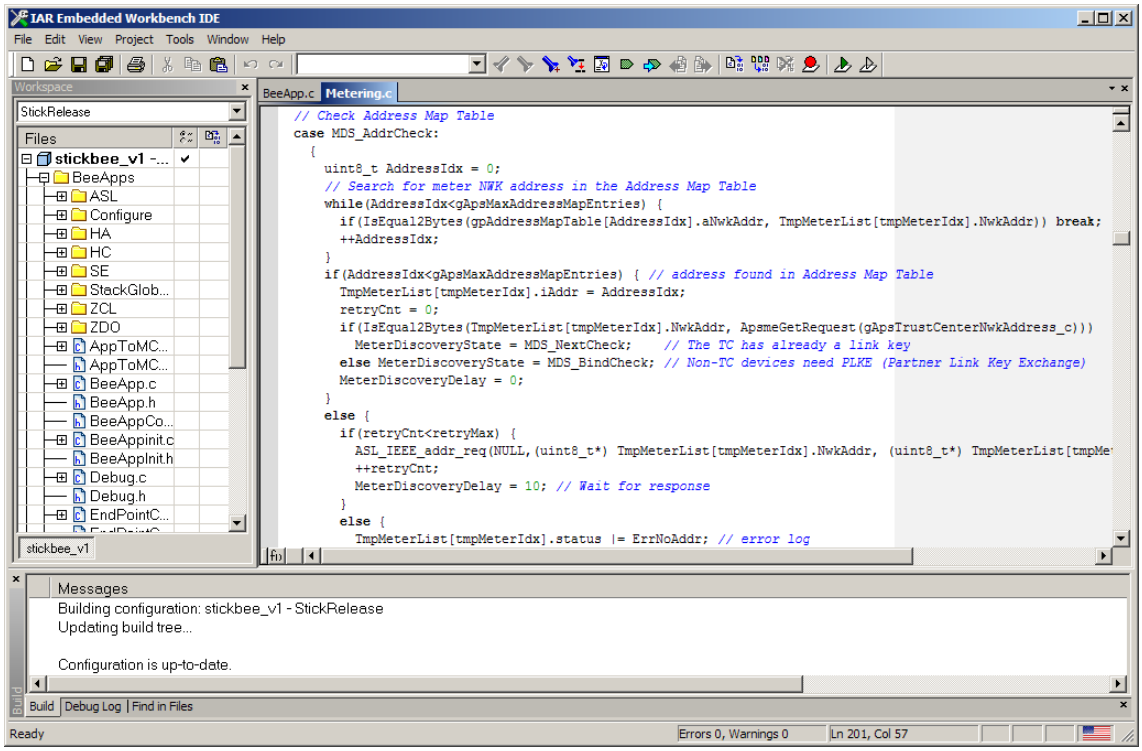
Protocol Analyzer A rádiós forgalom lehallgatását teszi lehetővé a sniffer segítségével. A program egy kiválasztott rádiócsatornán dekódolja a rádiós csomagokat és megjeleníti grafikus formában. A lehallgató hardver ilyenkor nem része a logikai PAN hálózatnak, csupán passzív megfigyelő, a hálózati kulcsokat manuálisan kell megadni, ha a kommunikáció titkosított. A Freescale Protocol Analyzer programja sajnos nem alkalmas ZigBee kommunikáció megfigyelésére, mert csak MAC rétegig végzi el a csomagok értelmezését. Helyette a Peryton Protocol Analyzer programot használtam (lásd: 5.2.1 fejezet).

Radio Test Ez a segédprogram alacsony szintű rádiós tesztek futtatására szolgál, ZTC ehhez is szükséges. A Packet Error Rate (PER) tesztben 2 node vesz részt, egyik a ZC, másik a ZED. A sikeres csatlakozás után a ZED a beállított paraméterek szerinti csomagokat küld a ZC-nek. A beállítható paraméterek a következők: payload mérete (1-102 bájt), payload tartalma, csomagok száma, rádió teljesítménye (0x0-0xF), fizikai csatorna (11-26). Ezt a programot használtuk a StickBee hardveres tesztekhez.

Firmware Loader A bináris állományokat tölthetjük vele a FLASH memóriába. Az alapértelmezésként felajánlott bináris a fent bemutatott ZTC-s tesztekhez használó. A firmware frissítés két módon is megtehető: J-Link debuggerrel vagy soros porton. A bootloaderekről bővebben a 4.5 fejezetben írok.

3.3.3. IAR Embedded Workbench for ARM

Az összes nagyobb ZigBee stack-hez az IAR Embedded Workbench fordítót használják, nincs ez másképp a BeeStack esetén sem. A kiváló beágyazott C/C++ fordítóhoz sajnos egy szegényes kezelőfelület társul (3.6. ábra), ami egy több tízezer kódsoros projektnél (pl. BeeStack) már jelentősen hátráltatja a fejlesztést. Ezt a hiányosságot pótolja az Eclipse CDT-hez kiadott plugin, ami az EWARM fordító 6.10 verziójától érhető el. A plugin segítségével egyszerűen importálhatjuk az IAR projektet és dolgozhatunk a felhasználóbarát Eclipse-es környezetben.



3.6. ábra. *Az IAR EWARM fordító felülete.*

4. fejezet

A ZigBee alkalmazás implementációja

A ZigBee egy komplex szabványrendszer, részletekbe menő megismerését nehezíti, hogy folyamatos fejlesztés alatt áll. Korábbi ZigBee-s tapasztalatok hiányában a kód írása párhuzamosan haladt a szabvány előírásainak és a rendszer működésének elsajátításával. A tanulási folyamat jellege miatt a kód fejlesztése is iteratív jelleget öltött. A tervezés során végig követtem azt az alapelvet, hogy a rádióforgalmat minimális szinten kell tartani. A teljes beágyazott szoftver projekt a melléklet részét képezi.

4.1. Projekt konfigurálás

A projekt első lépése a BeeStack konfigurálása. A konfigurálás a BeeKit nevű szoftverrel történik a 3.3.1 fejezetben ismertetett módon. Ebben az alpontban a StickBee alkalmazás konfigurálásának sarkalatos pontjait ismertetem. Az összes beállítást tartalmazó `.bksln` (BeeKit SoLutioN) kiterjesztésű XML állomány a projekt mappájában található.

NVM Az NVM a Non-Volatile Memory (nem-felejtő memória) rövidítése, és elengedhetetlen része a ZigBee eszközöknek. Egy készülék tápellátása több okból is megszűnhet: elemcsere, áramszünet, vagy üzemszerű kikapcsolás (pl. StickBee esetén az USB csatlakozóból kihúzás) miatt. A ZigBee szabvány előírja azokat az információkat[1], amiket ilyen esetekben a készüléknek helyre kell tudni állítani a legközelebbi bekapcsoláskor. Ilyen információk tipikusan a komissiózás során begyűjtött adatok vagy hálózati titkosító kulcsok. Ezek az adatok a BeeStack-ben automatikusan be vannak állítva az NVM-ben tároláshoz. Emellett a felhasználói program is tárolhat bizonyos információkat a fennmaradó helyen.

Az MC1322x-nél a belső FLASH memória az elsődleges NVM. Az adatok tárolására az a hely áll rendelkezésre, ami a bináris RAM image (vagyis a programunk) mellett megmarad a FLASH-ben. Mivel a RAM 96 kB, a FLASH pedig 128 kB méretű, legkedvezőtlenebb esetben is ∼32 kB NVM-mel lehet számolni.

[1] [5] Chapter 2.2.8.1 Persistent Data, [5] Chapter 3.6.8 Persistent Data

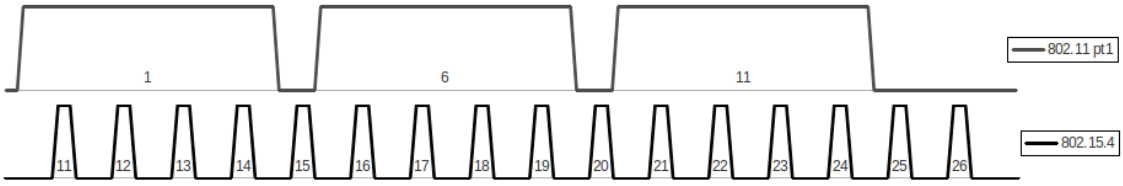
UART Az MC1322x és a PIC közötti soros kommunikáció fizikai rétege az UART (Universal Asynchronous Receiver/Transmitter), ezért ezt a modult mindenképp engedélyezni kell. A két lehetséges UART közül az UART1 került kiválasztásra, mert ez használható boot forrásként is (lásd: 4.5 alfejezet).

LED A szabvány előírja, hogy bizonyos eseményekről az eszköznek egyértelműen tájékoztatnia kell a felhasználót. Ilyen esemény lehet például, ha az eszköz sikeresen csatlakozott a hálózathoz[2], vagy ha nyugtázandó üzenetet kap[3]. A leggyakrabban használt "felhasználói felület" ilyen célra a LED. A StickBee-ben a Freescale alapértelmezett LED jelzéseit használtam fel 1 LED-re redukálva.

ZigBee End Device beállítások A StickBee RAM spórolás miatt ZED szerepet kapott. Alapértelmezésben a ZED elemes tápellátás miatt ideje nagy részét alvó módban tölti, ilyenkor a rádiója kikapcsolt állapotban van (`gRxOnWhenIdle_d = FALSE`). A ZED-nek címzett üzeneteket az alvás ideje alatt a szülő eszköz kapja meg. A ZED fix időközönként (`mDefaultValueOfIndirectPollRate_c`) lekérdezi a szülő eszközt, hogy kapott-e üzenetet. A StickBee nem telepes tápellátású, ezért a rádiót nem kell kikapcsolva tartania, így nem generál felesleges forgalmat a polling, és a szülő eszközt is kevésbé veszi igénybe.

Fizikai csatornák A SEP előírja[4] a 2,4 GHz-es sávban preferált csatornák listáját. Ez azt jelenti, hogy hálózat keresésekor vagy új hálózat létrehozásakor először ezeket a csatornákat szkenneli végig az eszköz, és csak utána a fennmaradó csatornákat. Ez a leggyakrabban használt Wi-Fi csatornák (1, 6, 11) elkerülése miatt lényeges (4.1. ábra). A Wi-Fi 22 MHz szélességű, átlapolódó csatornákat használ, a 802.15.4 csatornái 2 MHz sávszélességet használnak, a sávközepek mindkettőnél 5 MHz-re vannak egymástól.

A preferált ZigBee SEP csarornák a 2,4 GHz-es sávban: 11, 14, 15, 19, 20, 24, 25.



4.1. ábra. *A ZigBee csatornák és a 3 leggyakrabban használt Wi-Fi csatorna viszonya a 2450 MHz-es ISM sávban.*

4.2. Módosítások a BeeStack-ben

Az alkalmazás fejlesztése közben egy jól konfigurált stack minimális módosítást igényel. Nálam is alapelv volt a változtatások számának minimális szinten tartása. A BeeStack-ben végzett módosítások pontos dokumentációja megtalálható a mellékletben.

[2] [8] Chapter 5.5.1 Forming the Network (Start-up Sequence)
[3] [8] Chapter D.5.2.3.1 Display Message
[4] [8] Chapter 5.8.1 Preferred Channel Usage

4.2.1. ECC library

A SEP hálózatok kommunikációja több szinten titkosított. A titkosításhoz szükséges kulcsokat egy kitüntetett eszköz, a Trust Center (TC) kezeli. A SEP hálózatokban a TC szerepét minden esetben az Enery Service Interface, röviden ESI (régebbi nevén Energy Service Portal) látja el, mivel ez áll közvetlen kapcsolatban a szolgáltatóval. Ha egy eszköz (ebben a példában legyen IPD) csatlakozni szeretne a hálózathoz, akkor először a hálózati kulcsot (Network Key) kell beszereznie a TC-től. A Networ Key nem küldhető kódolatlanul rádión keresztül, ezért titkosítani kell egy előre konfigurált kulccsal (Preconfigured Link Key). Amennyiben a Preconfigured Link Key nem megfelelő (nem egyezik a TC nyilvántartásában az IPD IEEE címéhez rendelt kulccsal), a TC elutasítja a csatlakozást. Ha a Network Key átvitele sikeres volt, az IPD korlátozott hozzáférést kap a hálózaton[5]. A teljes hozzáféréshez Certification Based Key Establishment, röviden CBKE hitelesítési folyamaton kell átesnie. Ezt a hitelesítési módszert a ZigBee profilok közül egyedül a SEP tartalmazza. A hitelesítő adatokat minden eszközhöz egyedi módon egy harmadik fél, a Certification Authority adja ki. Jelenleg az egyetlen ilyen hatóság a Certicom, az általuk kiadott *implicit tanúsítványok* (implicit certificate) Elliptic Curve Qu Vanstone (ECQV) titkosítást használnak [6]. Ha a CBKE sikeresen lezajlott, az IPD Preconfigured Link Key helyére új Application Link Key kerül, a TC pedig egy flag segítségével jelzi saját nyilvántartásában, hogy az IPD-nél megtörtént a CBKE, így teljes jogú tagja lett a hálózatnak. Sikertelen CBKE esetén marad a régi Preconfigured Link Key és a korlátozott jogosultságok.

Az ECQV titkosító algoritmus (vagy más néven Ecliptic Curve Cryptography, ECC) helyett a BeeStack egy csonkot tartalmaz, amire a projekt konfigurálsakor a BeeKit is felhívja a figyelmet. A Certicom előzetes regisztrációt követően minden elterjedtebb platformhoz díjmentesen biztosítja az optimalizált ECC library-t, ezt manuálisan kell a projekthez hozzáadni.

4.2.2. Link Key Request

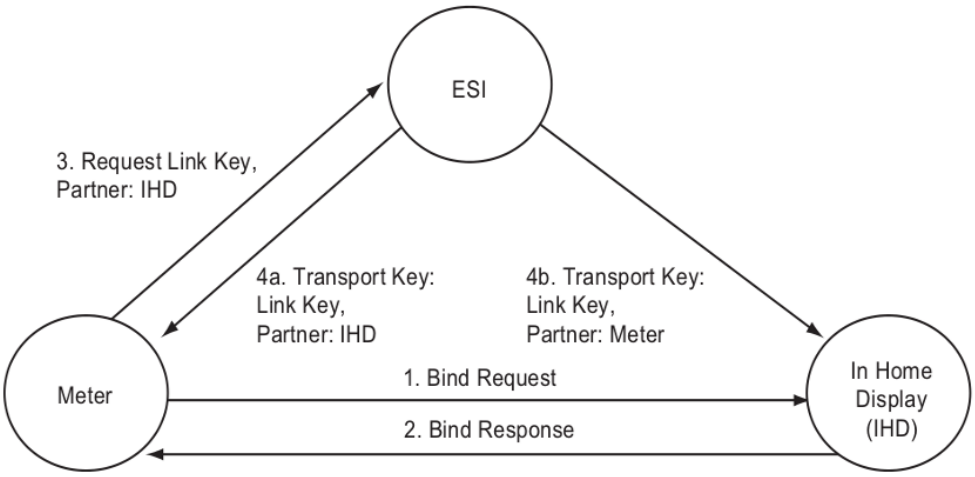
Sikeres CBKE után előfordulhat, hogy nemcsak a TC-vel szeretnénk Application Link Key-t igénylő kommunikációt folytatni, hanem a hálózat többi eszközével is, például az IPD szeretné lekérdezni egy mérőóra állását. Ekkor az IPD-nek és a mérőnek is rendelkeznie kell egy közös kulccsal. Általánosítva: ahány eszközzel szeretne az IPD közvetlenül kommunikálni, annyi különböző Application Link Key-t kell használnia. A Link Key-ek cseréje a TC közreműködésével zajlik le (4.2. ábra[6]).

Érdekes módon a BeeStack-ből kimaradt a 3. lépés, azaz a codebase legújabb verziójából is hiányzik a

```
void APP_ZDP_RequestKeyRequest(zbCounter_t  *pSequenceNumber, zbNwkAddr_t
aDestAddress, zbApsmeRequestKeyReq_t  *pApsmeRequestKey);
```

[5] [8] Chapter 5.4.6 Cluster Usage of Security Keys

[6] az ábrán a mérőóra kezdeményezi a kulcs átvitelt, de ez a végeredmény szempontjából lényegtelen



4.2. ábra. *Link Key Exchange menete [8].*

prototípusú rutin. A Freescale kérésre elküldte az ehhez hiányzó kódrészleteket.

4.2.3. Request-response párosítás

Az elküldött és beérkezett rádiós csomagokat azonosítása a protokollban sorszámok alapján lehetséges. Egy adott csomaghoz a protokoll minden rétegében tartozik egymástól független sorszám, az Application Support Sublayer-ben `APS counter`, a ZCL üzenetekben `Transaction Sequence Number` a neve. Mindegyik sorszám 8 bites számláló, ezért rétegenként 256 üzenet megkülönböztetésére szolgálnak. Ez általában elég is, jellemzően nincs egyszerre ennyi megkülönböztetendő üzenet egy ZigBee hálózatban.

A BeeStack az alkalmazás fejlesztőjére bízza, hogy hogyan kezeli az elküldött üzenet (*request*) és a rá érkező válasz(ok) (*response*) párosítását. A probléma szemléltetésére vegyük például a ZDP rutinjai közt található `Match_Desc_req` request-et. A visszaérkező `Match_Desc_rsp` response másképp kezelendő, ha a request-tel az ESI endpointot keressük, és másképp akkor, ha a hálózatban található mérőórákat szeretnénk felderíteni. Utóbbi esetben ráadásul azt sem tudhatjuk, hogy hány response-ra számítsunk. Vagyis ha egy request típust (pl. `Match_Desc_req`) több célból is küldhetünk, akkor még küldésekor meg kell jegyeznünk, hogy miért küldtük és ennek megfelelően kell kezelni a visszaérkező választ.

A párosítás ötletét az Ember Z-Stack-jéből merítettem, ahol egy request hívásnál elég egy callback függvényre mutató pointert megadni, ami a response érkezésekor meghívódik. A megoldásomban az APS és ZCL sorszámhoz is tartozik egy-egy 256 elemű tömb. Egy request meghívásakor eltárolom a megfelelő callback azonosítóját a tömb sorszám indexű elemébe. Response érkezésekor a megfelelő tömb sorszám indexű eleméhez tartozó callback-et hívom meg. Ez a megoldás a következő feltételezésekkel él:

- Minden, az alkalmazás által kezelni kívánt request típust callback függvény hozzárendelésével hívunk meg (lehet üres callback függvény is).
- A válasz hamarabb megérkezik egy request-re (vagy sose érkezik válasz), mint hogy az adott sorszám körbeérne.

Így nem fordulhat elő olyan hiba, hogy az adott tömb sorszám indexű eleme érvénytelen callback függvényre hivatkozik.

A tömbökben nem a callback függvényre mutató pointert tárolom, hanem csak egy 1 bájtos callback azonosítót, amivel extra nyilvántartási munka és hibalehetőség került a rendszerbe. Cserébe viszont értékes RAM terület spórolható meg mindaddig, amíg a callback függvények száma el nem éri a 768-at. Jelenleg mindkét sorszámhoz kevesebb, mint 10 callback tartozik, így a helymegtakarítás jelentős (a szükséges RAM terület 2 kB helyett ∼0,5 kB).

4.3. BeeApp

A BeeStack-ben a `BeeApp.c` alkotja a tényleges alkalmazást, ami a projekt exportálásakor a választott demó program. A `BeeApp.c` több publikus függvényt is tartalmaz, amiket a stack többi része használ, először ezeket veszem sorra.

BeeAppInit Regisztrálja az endpointo(ka)t, allokálja a timereket, inicializálja az alkalmazással kapcsolatos komponenseket. Ide kerül minden, amit a program indulásakor el kell végezni. A `BeeAppInit.c`-ből hívódik meg.

BeeAppTask Az alkalmazás task eseményeit kiszolgáló függvény (a taskokról a 3.2 fejezetben írok bővebben). A kötelező események mellett 3 saját eseményt hoztam létre:

- `gAppEvtUARTRx_c`: akkor keletkezik, ha megszakítás érkezik az UART1-től. Feladata ICP parancsok értelmezése és kiszolgálása. Bővebben a 4.4.1 alfejezetben foglalkozok vele.
- `gAppEvtMetering_c`: a ZigBee hálózat mérőóráinak kezelését segítő esemény. Bővebben a 4.4.2 alfejezetben foglalkozok vele.
- `gAppEvtAppCmd_c`: ez az esemény akkor keletkezik, ha egyedi ICP parancs érkezik, amire az ICP modul nincs felkészítve. Jelenleg gombnyomások szimulálására szolgál, mivel a StickBee egyetlen nyomógombját a központi mikrovezérlő kezeli.

BeeAppHandleKeys A Freescale a felhasználói felülethez (LED-ek, nyomógombok, LCD) két üzemmódot definiál, ezzel próbálja kibővíteni a szűkös lehetőségeket. Bekapcsoláskor az eszköz `gConfigurationMode_c` üzemmódban van, ilyenkor a gombokat az `ASL_UserInterface.c` kezeli. `gApplicationMode_c` üzemmódban a `BeeAppHandleKeys` kezeli a gombokat, pl. ha egy üzenetet nyugtázni kell a felhasználónak.

BeeAppUpdateDevice A stack bizonyos eseményeinek alkalmazásspecifikus kezelésére szolgál ez a függvény. Például alkalmazás által kezelendő esemény érkezik a ZDO felől, ha a hálózathoz csatlakozás sikeresen megtörtént (`gZDOToAppMgmtZEDRunning_c`), vagy ha egy `Match_Desc_req`-re válasz érkezik (`gMatchDescriptorSuccess_c`).

A "kötelező" részek után tekintsük át az alkalmazás többi részét!

gaNvAppDataSet Az alkalmazás elmenthet olyan adatokat, amiket egy esetleges újraindítás után vissza kell állítani (lásd: 4.1 alfejezet). Az elmentett adatok közt van az ESI endpoint-ja és egy flag, ami azt jelzi, hogy megtörtént-e már a CBKE.

BeeAppFindESP Ez a rutin indítja el a CBKE folyamatát, ha a csatlakozás megtörtént.

BeeAppUpdateTimeFromESP A szabvány előírja[7], hogy a hálózati eszközök idejét az ESI idejéhez kell szinkronizálni, az eltérésnek ± 1 percen belül kell maradni. Az időszinkron miatti rádiós forgalmat minimum szinten kell tartani, ezért a szinkronizációs időköz a StickBee-ben 24 óra. Értelemszerűen bekapcsolást követően szintén szinkronizál a StickBee.

Request-response párosítás A 4.2.3 alfejezetben leírtaknak megfelelően a `BeeApp.c`-ben definiáltam az APS és ZCL számlálókhoz tartozó tömböket (`aslCallbackIdList`, `zclCallbackIdList`) és a párosításhoz szükséges segédfüggvényeket.

4.4. Modulok

Az alkalmazás többi részét az átláthatóság miatt modulokra osztottam úgy, hogy a modulok között a lehető legkisebb függőség legyen.

4.4.1. ICP

Az ICP modul feladata, hogy kiszolgálja a host processzort a képek generálásához szükséges adatokkal. A cél egy olyan felület létrehozása volt, amin keresztül a konkrét hálózat előzetes ismerete nélkül, előre konfigurálható módon elérhetjük a ZigBee hálózat eszközeinek releváns adatait illetve a hálózatra jellemző információkat. A cégnél specifikáltak egy egyszerű, univerzálisan használható protokollt, mely az Internal Communication Protocol (ICP) névre hallgat [7]. Ezt a protokollt használtam fel és szabtam testre a ZigBee SoC és a host processzor közötti adatcseréhez.

Az ICP pont-pont kapcsolatot tesz lehetővé egy szerver és egy kliens között, UART fizikai rétegen keresztül. Az ICP által definiált parancsok elsősorban a szerver adatainak lekérdezésére irányulnak, de saját parancsok is definiálhatók. Az adatok attribútumok formájában kérdezhetők le, melyeket 6 bájtos ID azonosít, értékük dekódolását pedig előre definiált típusok segítik. Minden attribútum csoportokba van rendelve, ezeket a csoportokat 8 bájtos MAC cím azonosítja.

Az ICP üzenetek felépítése

A kommunikációt mindig a kliens eszköz (StickBee esetén a host processzor) kezdeményezi.

Az ICP üzenet (4.1. táblázat) az alábbi mezőkből áll:

[7] [8] Chapter 5.12.1.1 ZCL Time Cluster and Time Synchronization

4.1. táblázat. *Az ICP általános üzenetformátuma.*

1 bájt	1 bájt	0...252 bájt	1 bájt
üzenet hossza	üzenet típusa	hasznos adat	ellenőrzőösszeg

- **Üzenet hossza:** a teljes üzenet hosszát adja meg bájtban, beleértve a magát a hosszt jelölő bájtot is. Maximális értéke 255, így a hasznos adat maximális mérete 252 bájt lehet.
- **Ellenőrző összeg:** hibadetektálásra szolgáló mező, az üzenet többi bájtjának összegeként áll elő, 8 biten előjel nélkül ábrázolva.
- **Üzenet típusa:** megadja, hogy hogyan kell értelmezni a hasznos adatot. Az MSB azt adja meg, hogy az üzenet kérés (query) vagy válasz (answer) volt-e, a maradék 7 bit jelentését a 4.2. táblázat foglalja össze.

4.2. táblázat. *ICP üzenetazonosítók.*

ID	Üzenettípus neve
1	Get MAC Addresses
2	Get Attribute List
3	Get Value
4	Get Time

Látható, hogy a rendelkezésre álló 128 üzenettípusból csak 4-et használ a protokoll. Ha a 0-s típust nem számoljuk bele, akkor is 123 üzenettípus marad a testreszabásra.

Get MAC Addreses Ezzel az üzenettel tudja lekérdezni a kliens a szerver attribútumcsoportjainak azonosítóit, más néven a MAC címeket (4.3. táblázat[8]).

4.3. táblázat. *A Get MAC Addresses típusú üzenetek felépítése.*

	üzenet hossza	üzenet típusa	hasznos adat	ellenőrzőösszeg
kérés	3	129	-	132
válasz	$3 + m \times 8$	1	MAC1; MAC2; ... MACm	XX

Get Attribute List Ezzel az üzenettel kérdezheti le a kliens az egy MAC címhez tartozó attribútumazonosítók listáját (4.4. táblázat[9]).

Get Value Ez az üzenet 1 db attribútum lekérdezésére szolgál (4.5. táblázat[10]).

A ZigBee hálózat és az ICP közti megfeleltetés

Az ICP nem szabja meg a MAC címek jelentését, ezért itt lehetőségem nyílt olyan attribútumcsoportok létrehozására, amik a legjobban illenek az alkalmazáshoz. Állandó attribútumkészlettel rendelkező kategóriákat definiáltam, amikbe besorolhatók az egyes

[8] m = MAC címek száma
[9] a = attribútumok száma
[10] v = attribútum értékét tároló tömb hossza bájtban

4.4. táblázat. *A Get Attribute List típusú üzenetek felépítése.*

	üzenet hossza	üzenet típusa	hasznos adat	ellenőrzőösszeg
kérés	11	130	MAC	XX
válasz	$11 + a \times 6$	2	MAC; Attr1; Attr2; ... Attra	XX

4.5. táblázat. *A Get Value típusú üzenetek dekódolása.*

	üzenet hossza	üzenet típusa	hasznos adat	ellenőrzőösszeg
kérés	17	131	MAC; Attr	XX
válasz	25 + v	3	MAC; Attr; Időbélyeg; Típus; Érték	XX

logikai eszközök. A kategóriát a MAC cím 1-es bájtja jelöli, az eszköz sorszámát pedig a 0-s bájt (4.6. táblázat). Így összesen 256 kategória és ezeken belül 256 eszköz lehet. Pazarlásnak tűnhet, hogy a rendelkezésre álló 8 bájtból csak kettőt használok, viszont a kompatibilitás miatt célszerű volt megtartani a 8 bájtos MAC címet, a felhasznált 2 bájt pedig bőven kielégíti a ZigBee hálózat igényeit.

4.6. táblázat. *ICP MAC address újradefiniálása.*

7 (MSB)	6	5	4	3	2	1	0 (LSB)
0x00	0x00	0x00	0x00	0x00	0x00	kategória	eszköz sorszáma

Az attribútumazonosítók szintén redukálva lettek, csak az LSB hordoz hasznos információt (4.7. táblázat).

További korlátozásként bevezettem, hogy az eszközök és az attribútumok számozásának is nullától kell indulnia és folytonosnak kell lennie. Így elég azt nyilvántartani, hogy egy kategóriához hány eszköz és hány attribútum tartozik, nem kell pontos listát vezetni.

A ZigBee hálózat adatainak lekérdezése és az ICP-n keresztül történő kiszolgálás aszinkron módon történik. Ez azt jelenti, hogy ha ICP-n keresztül lekérdezünk egy attribútumot, akkor a választ már meglévő adatokból fogja az érintett modul összeállítani, az ICP lekérdezés nem generál extra ZigBee kommunikációt. Az attribútum lekérdezéséhez minden kategória rendelkezik egy függvénnyel, melynek prototípusa a következő:

```
uint8_t ReportAttr(uint8_t iDev,    // device index of the query
                   uint8_t iAttr,   // attribute index of the query
                   uint8_t *Val,    // value of the attribute returns here
                   uint8_t *Size,   // size of the attribute value returns here
                   uint8_t *Type,   // attribute type returns here
                   uint32_t *Time); // timestamp returns here (ZB format)
```

Visszatérési értéke 0, ha a lekérdezés sikeres volt, egyéb esetben a hiba típusának megfelelő egész értéket adja vissza (pl. érvénytelen `iAttr` vagy `iDev`). Bemeneti paraméterei az eszköz sorszáma (`iDev`) és az attribútum sorszáma (`iAttr`). Kimeneti paraméterei az attribútum értéke (`Val`), az attribútum mérete bájtban (`Size`), az attribútum ICP típusa (`Type`) és egy ZigBee formátumú időbélyeg (`Time`).

Az ICP attribútum értékét a lekérdezés időpontjában generálja a függvény a meglévő adatok felhasználásával, a származtatás módja szabadon választható. Sok esetben a származtatás egyszerű másolást jelent, de vannak összetettebb attribútumok is.

Az eddig leírtakat a `General` kategórián mutatom be, ami a `BeeApp.c`-ben található. A

4.7. táblázat. *ICP attribute ID újradefiniálása.*

5 (MSB)	4	3	2	1	0 (LSB)
0x00	0x00	0x00	0x00	0x00	kategória

kategória StickBee-re vonatkozó általános attribútumokat fog egybe, ezért egyetlen logikai eszköz tartozik csak ide. A kategória azonosítója 0x00, vagyis ezeket az attribútumokat a csupa 0 MAC címen érjük el. A `General` kategóriához tartozó attribútumok listája a 4.8. táblázatban látható.

4.8. táblázat. *ICP General kategória attribútumai.*

ID	Leírás
0	StickBee IEEE címe
1	StickBee NWK címe
2	aktuális PAN ID
3	aktuális fizikai csatorna
4	szülő eszköz IEEE címe
5	szülő eszköz NWK címe
6	Trust Center IEEE címe
7	Preconfigured Link Key
8	aktuális idő UTC formátumban
9	CBKE flag

4.4.2. Metering

Ez a modul a hálózatban található mérőórák felderítéséért és lekérdezéséért felel. A mérőórák felderítése a CBKE után automatikusan elkezdődik, ennek folyamata leegyszerűsítve a következő:

1. *Előzetes lista készítése a mérőórákról:* Mérőórának számít minden eszköz, amely rendelkezik metering cluster szerverrel. A `Match_Descriptor_req`-t ezzel a kritériummal broadcast üzenetben elküldve a keresett eszközök fognak válaszolni. A válaszban a mérőóra NWK címe és az endpoint érkezik vissza, ezek alkotják az előzetes listát.

2. *Mérőórák felvétele az Addres Map-be:* A továbbiakban szükség lesz a NWK cím mellett az IEEE címre is. Ha a StickBee Address Map-jében még nincs bejegyzés az adott NWK címmel, a StickBee kiad egy `IEEE_addr_req`-t. A ZDO gondoskodik róla, hogy a válasz automatikusan bekerüljön az Address Map-be. Ha a mérőóra NWK címe valamiért megváltozik, a változásról értesítést küld, és az Address Map bejegyzés is automatikusan frissül.

3. *Link Key ellenőrzése:* Ahhoz, hogy a StickBee le tudja kérdezni egy mérőóra attribútumait, szüksége van Link Key-re (lásd: 4.2.2 fejezet). A Link Key meglétének ellenőrzése két lépcsőben történik. Először a Binding Table-ben kell ellenőrizni, hogy a Binding létrejött-e már, ha még nem jött létre, ki kell adni a `Bind_req`-t. Ha a Binding létrejött, a Trust Center-től kérni kell egy Link Key-t. A 2. és a 3. pont kihagyható, ha a mérő maga az ESI.

4. *Végleges lista készítése a mérőórákról:* A végleges lista csak azokat a mérőórákat tartalmazza, amiknél a 2. és 3. lépés is rendben lezajlott.

5. *Összes releváns attribútum lekérdezése:* A releváns attribútumokat tartalmazó lista azokat az attribútumokat tartalmazza, amikre a egy kép generálásakor szüksége lehet a StickBee-nek. A listán szereplő összes attribútumot egyszer lekérdezi a StickBee minden mérőórától.

6. *Dinamikusan frissülő attribútumok lekérdezése:* A releváns attribútumok közt vannak dinamikusan frissülő attribútumok (pl. a mérőóra aktuális állása). Ezeket fix időközönként lekérdezi a StickBee minden mérőórától.

A modulhoz tartozó ICP kategória azonosítója 0x01, az eszközök száma a felderített mérőórák száma szerint változik. A kategória attribútumlistáját a 4.9. táblázat foglalja össze.

4.9. táblázat. *ICP Metering kategória attribútumai.*

ID	Leírás
0	formázott fogyasztás (string)
1	mértékegység (string)
2	mérő típusa
3	mérő IEEE címe
4	mérő endpoint
5	nyers fogyasztás

A fogyasztási adatok nyers formában, és a szabványban leírt formázás szerint[11] stringként is elérhetők.

4.4.3. Messaging

Ez a modul a ZCL Messaging cluster kliens funkcionalitását valósítja meg. A modulhoz tartozó ICP kategória azonosítója 0x02, és csak egy logikai eszközt tartalmaz. Attribútumlistája a 4.10. táblázatban látható.

4.10. táblázat. *ICP Messaging kategória attribútumai.*

ID	Leírás
0	Üzenet szövege
1	Üzenet státusz (0: nincs üzenet, 1: üzenet érkezett, 2: üzenet aktív)
2	Üzenet prioritása

Ez a modul felel a jóváhagyást igénylő üzeneteknél a nyomógomb kezeléséért és az üzenet érvényességi idejének figyeléséért is.

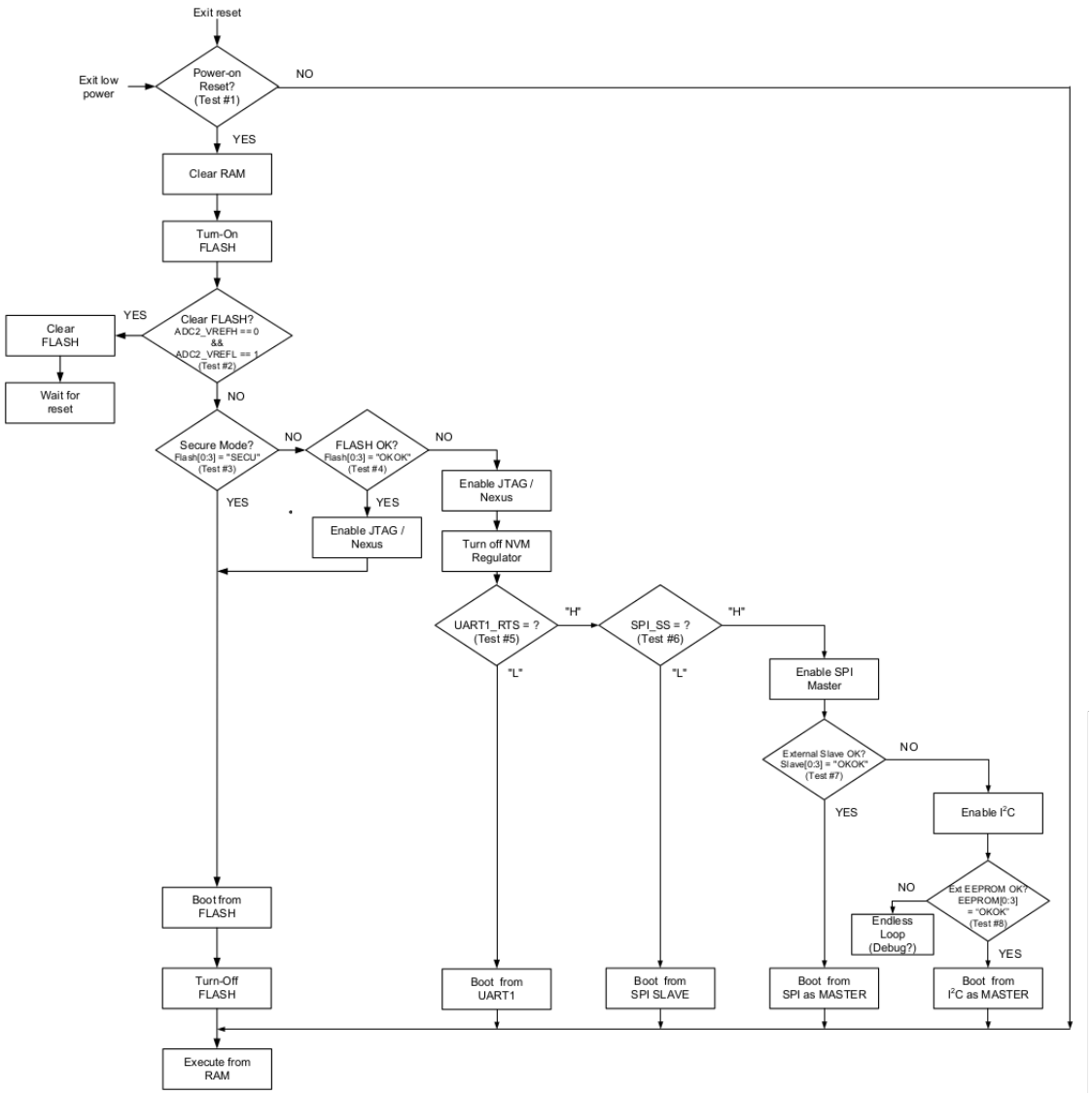
4.5. Bináris állomány letöltése FLASH-be

Az MC1322x chip-ek a hardveres reset állapotból kilépve a ROM-ban található bootloader kódot kezdik futtatni. A legáltalánosabb eset az, mikor a belső FLASH memória érvényes

[11] [8] Chapter D.3.2.2 Attributes

kódot tartalmaz. Ilyenkor a bootloader bemásolja a FLASH tartalmát a RAM-ba, majd elkezdődik a program futása a RAM kezdő címétől (0x00400000). Ha a resetet egy alvó üzemmódból való felébredés okozta, akkor a bootloader folyamatot kihagyva rögtön a RAM kezdő címénél kezdődik a futás.

Bonyolultabb a helyzet, ha a FLASH tartalmát törölni kell, vagy új firmware-t szeretnénk letölteni. A 4.3. folyamatábra a bootloder működését szemlélteti.



4.3. ábra. *A bootolás folyamata.*

A FLASH törlésének legegyszerűbb módja, ha az `ADC2_VREFH` lábat (64) logikai alacsony, `ADC2_VREFL` lábat (61) pedig logikai magas állapotba húzzuk, majd reset-eljük az IC-t. Néhány másodperc múlva - amikor a FLASH tartalma már biztosan törlődött - vissza kell állítani a lábak eredeti állapotát, és egy újabb reset után folytatódhat a bootolás.

Az érvényes firmware-t az jelzi a FLASH-ben, ha a legelső 4 bájt vagy a `SECU` vagy az `OKOK` ASCII karaktersorozatot tartalmazza. `SECU` mód esetén a bootloader nem engedélyezi a debug portokat (JTAG/Nexus), ezért ilyenkor csak a fent leírt módszerrel törölhető a

FLASH.

Üres FLASH esetén a bootloader sorra megvizsgálja az alábbi perifériákat:

1. UART1 (pl. host IC)
2. SPI slave (pl. host IC)
3. SPI master (pl. külső FLASH)
4. I^2C (pl. külső EEPROM)

Ha egyik perifériáról sem tud bootlolni, akkor végtelen ciklusba kerül, ilyenkor a debuggerrel lehet vezérelni.

Akármelyik forrásból is történjen a bootolás, először mindig a RAM-ba betölteni kívánt adat hosszát kell megadni 32 bit little-endian formátumban. Ezt követően a bootloader betölti a megadott számú bájtot a RAM-ba, majd futtatja azt a 0x00400000 kezdőcímtől.

A "belső" FLASH memória nem része a címtérnek (lásd: 3.1.1 fejezet), ezért debuggerrel nem tölthető bele közvetlenül adat. A FLASH-be íráshoz először egy futtatható kódot kell a RAM-ba írni, ami a dedikált SPIF porton keresztül bemásolja a firmware-t a FLASH-be. A Freescale Firmware Loader programja is először egy ∼1 kB méretű kódot (`flashloader.bin`) tölt a RAM-ba, majd a megadott firmware-t 4 kB méretű darabokban felmásoltatja vele a FLASH-be.

A `flashloader.bin` kódot felhasználva én is írtam egy egyszerű parancssoros programot C nyelven, amivel írható a FLASH. Ennek előnye a Freescale programjával szemben, hogy szabadon testreszabható, ami pl. sorozatgyártásnál lehet előnyös, mikor egyedi adatokat kell a készülékekbe automatizált módon letölteni (pl. IEEE címet, Certicom tanúsítványt, vagy Preconfigured Link Key-t). A program `binflasher` néven megtalálható a mellékletben.

5. fejezet

A ZigBee alkalmazás verifikációja

5.1. Teszt hálózat felépítése

A StickBee teszteléséhez létre kellett hoznom egy teszt hálózatot SEP eszközökből. Célom a valósághoz minél közelebb álló környezet megteremtése volt, lehetőleg ZigBee tanúsítással rendelkező eszközökből. A ZigBee Alliance honlapján[1] minden eszköz megtalálható, amelyik ZigBee tanúsítást kapott.

Az eszközök kiválasztásánál azt is figyelembe kellett vennem, hogy a Certicom két fajta tanúsítványt bocsát ki: *Test Certificate* és *Production Certificate*. Egy valós SEP hálózatban minden eszköz Production Certificate-tel rendelkezik, Test Certificate-es eszköz CBKE próbálkozása sikertelen lenne. Ugyanez fordítva is igaz, ha a Trust Center Test Certificate-tel rendelkezik, a CBKE csak Test Certificate-es eszköz esetén lehet sikeres. A Certicom az ingyenes regisztrációval Test Certificate generálására alkalmas webes felületet biztosít, a Production Certificate éves díja 10000 USD [6]. Ebből kifolyólag csak olyan teszt eszközök jöhettek szóba, amiknél a Test Certificate elérhető. A következő alpontokban a teszteléshez kiválasztott készülékeket mutatom be.

5.1.1. Digi ConnectPort X2

A ConnectPort X2 a Digi XBee moduljára épülő ZigBee-Ethernet átjáró (5.1. ábra). ZigBee hálózati szerepét tekintve koordinátor, és ESI Device Profile-lal rendelkezik. A Digi egy Device Cloud szolgáltatáson[2] keresztül egységes adminisztrációs felületet biztosít internetképes eszközeihez, így a ConnectPort X2-höz is. A szolgáltatás célja újabb és újabb készülékek bevonása az *Internet-of-Things*-be. A ConnectPort X2 egy másik webes felületen is elérhető, itt kifejezetten a ZigBee SEP-hez kapcsolódó fejlesztői lehetőségeket érhetjük el [1].

A ConnectPort X2 gyárilag Production Certificate-tel érkezik, ezt a [1] felületen keresztül lecserélhetjük Test Certificate-re. Ugyanitt érhető el a Debug Console is, ahol az EmberZNetPro stackből megszokott CLI (Command Line Interface) parancsok adhatók ki az eszköznek XML formátumba ágyazva. A párhuzam nem véletlen, az XBee modulok

[1] `http://www.zigbee.org/Products/ByStandard/ZigBeeSmartEnergy.aspx`
[2] `http://www.etherios.com`

5.1. ábra. *Szabványos ESI (Digi ConnectPort X2).*

a Silicon Labs (korábban Ember) ZigBee platformjára épülnek.

5.1.2. Silicon Labs EM357 developent kit breakout board

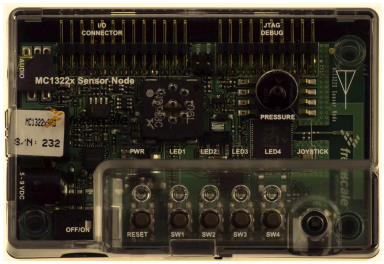
Tapasztalataim szerint az egyik legpopulárisabb ZigBee PRO platformja a Silicon Labs-nek van (5.2. ábra), a stack funkciógazdagsága, strukturáltsága és dokumentáltsága példaértékű.

5.2. ábra. *Silicon Labs EM357 developent kit breakout board.*

A rajta futó, teszteléshez használt program egy volt kollégám munkája. Hálózati szerepét tekintve lehet router és koordinátor is. ESI Device Profile-lal rendelkezik, de van rajta metering cluster server is, amivel mérőórát tud szimulálni. Kezelésére soros porti CLI szolgál.

5.1.3. Freescale MC13226 Sensor Node

A Freescale demó mérőórája is részt vett a tesztekben (5.3. ábra), a demó alkalmazáson csak a hálózathoz csatlakozás miatt végeztem változtatásokat. Hálózati szerepe: router, Device Profile: Metering.



5.3. ábra. *Freescale MC13226 Sensor Node.*

5.2. Teszt eszközök

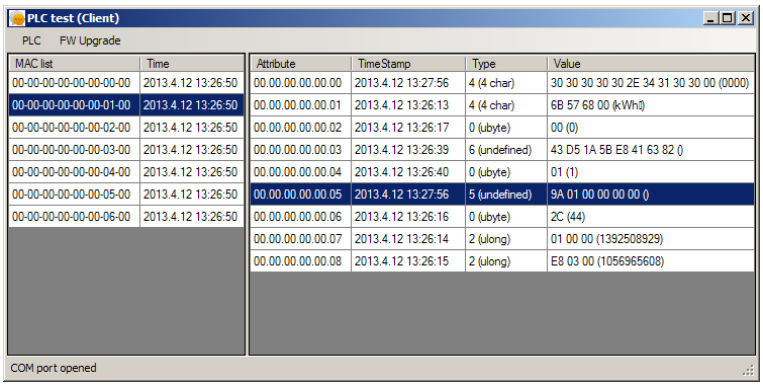
Ebben a fejezetben a teszteléshez használt szoftvereszközöket mutatom be.

5.2.1. Peryton Protocol Analyzer

ZigBee rádiós forgalom monitorozására és dekódolására alkalmas program (5.7. ábra). A 12 támogatott sniffer hardver között megtalálható az általam használt Freescale USB Dongle is. Használható valós idejű lehallgatásra és elmentett felvételek utólagos analizálására is. A program részletes működését a 5.3 alfejezetben mutatom be.

5.2.2. PLC (soros porti ICP tesztelő)

Az ICP protokoll teszteléséhez készült program, ICP kliens és ICP szerver üzemmódot is támogat. ICP kliens üzemmódban a ZigBee kontroller részére tudja szimulálni a StickBee központi mikrokontrollerét. Az 5.4. ábrán az látható, hogyan jeleníti meg a PLC a StickBee adatait.



5.4. ábra. *A PLC működés közben.*

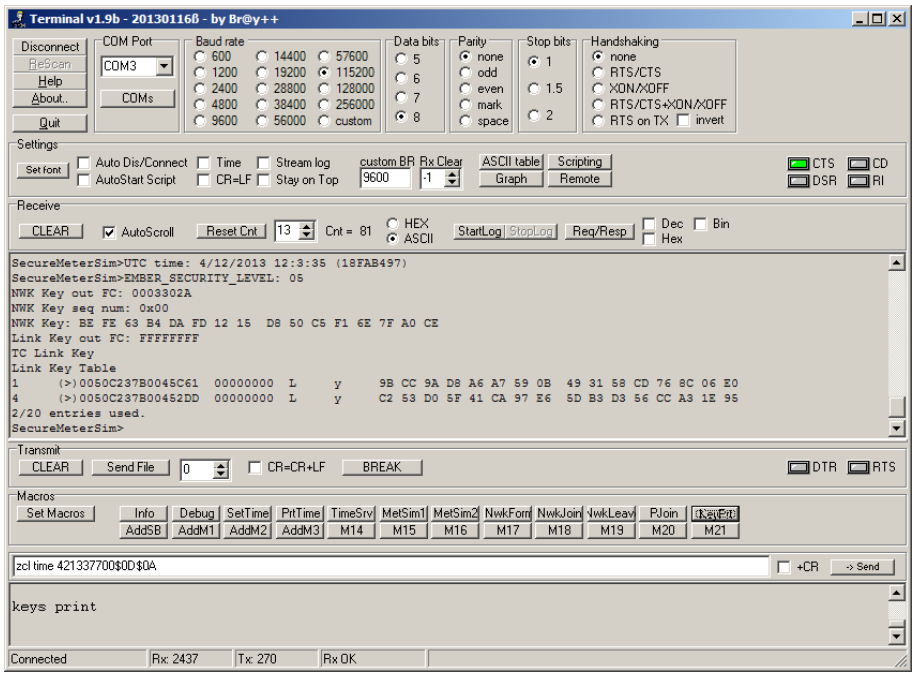
5.2.3. Terminal

A Terminal[3] egy ingyenes terminál emuláló program soros porthoz, ami kifejezetten beágyazott rendszerek soros kommunikációjának tesztelésére készült. Számtalan praktikus funkciója közül kiemelnék kettőt, amiért a StickBee tesztelése közben nélkülözhetetlennek bizonyult:

[3] `https://sites.google.com/site/terminalbpp/`

- Szabad átjárást biztosít az elküldendő és fogadott adatok ASCII, hexadecimális és deximális módban történő reprezentációjához.
- 24 db makrót használhatunk egyszerre, amik egyedileg címkézhető gyorsgombbal és billentyűkombinációval is elérhetők.

A 5.5. ábrán a Silicon Labs breakout board vezérlése közben látható a program.



5.5. ábra. *A Terminal működés közben.*

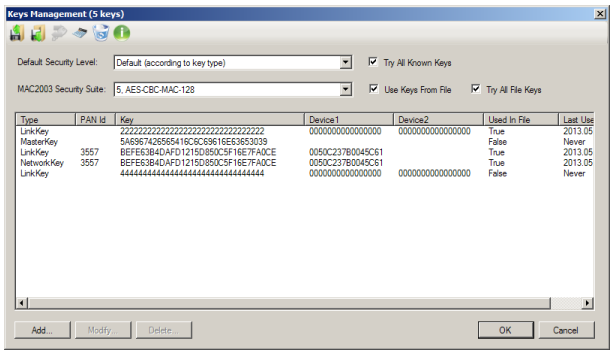
5.3. A StickBee működésének tesztelése

A StickBee ZigBee-s részének verifikálásához a rádiós forgalom ellenőrzése volt a célom. A rádiós forgalmat, tehát az eszköz OTA (Over-The-Air) viselkedését vizsgálják a tanúsító laborokban is, vagyis tekintet nélkül a megvalósítás részleteire fekete dobozként kezelik a tesztelendő eszközt, és ún. *test harness*-ek[4] segítségével tesztsorozatokat végeznek rajta. Test harness hiányában a rendelkezésemre álló eszközökből állítottam össze hálózatot és a Perytons szoftverével vizsgáltam az OTA csomagokat (ennek eredménye a mellékletben). Két tesztösszeállítást mutatok be, az egyikben a mérők kezelését mutatom be, a másikban az üzenetküldést.

A teszt előtt be kell állítani a Perytons-ban a Preconfigured Link Key-eket (5.6. ábra), ezek ismeretében a program automatikusan észleli és beállítja a Network Key-t. Ugyanígy lehetőség volna visszafejteni az Application Link Key-eket az OTA üzenetekből, ezt azonban nem támogatja a program, az Application Link Key megadása a felhasználó feladata.

Miután az ESI-vel létrehoztuk a hálózatot a Perytons segít megkeresni, hogy melyik csatornát választotta az ESI. A csatornák szkennelése történhet aktív vagy passzív módon.

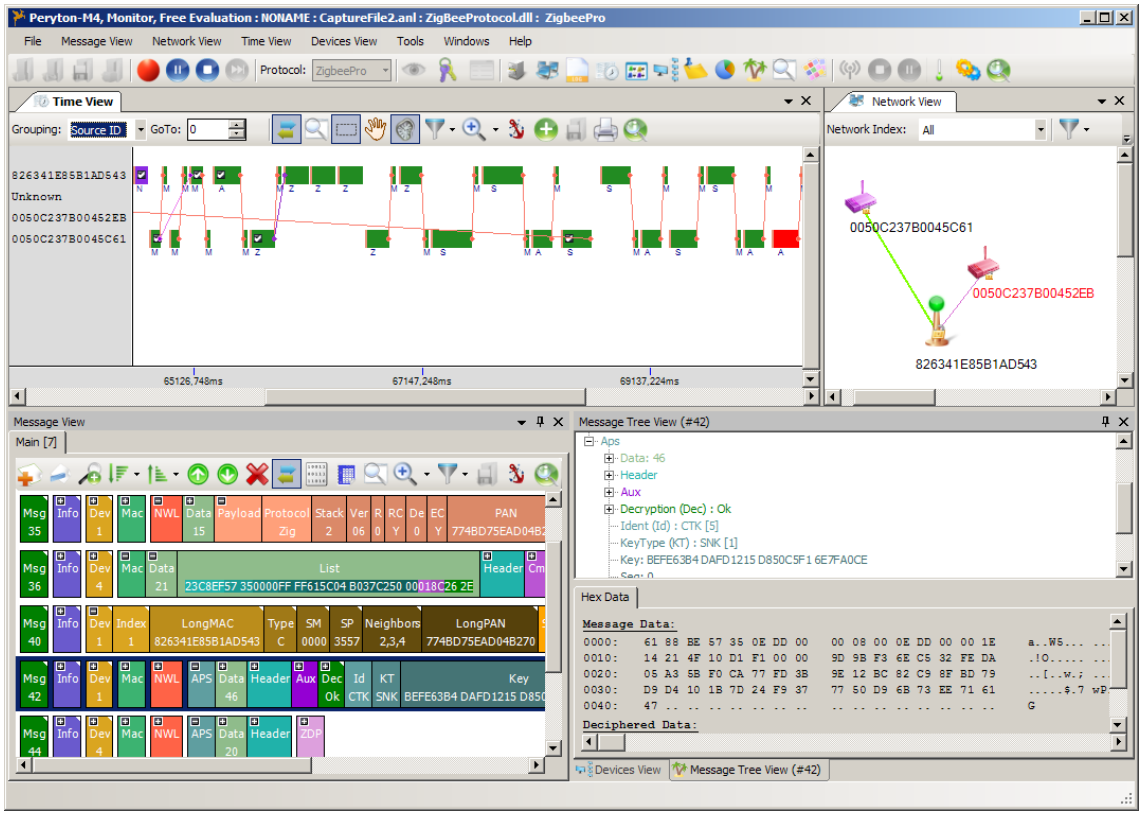
[4] `http://www.nts-at.com/products/setest_tool`



5.6. ábra. *Kulcsok beállítása a Peryton-ban.*

Passzív szkennelés esetén a sniffer nem küld *beacon request*-et, ezért ekkor a keresés tovább tarthat.

A lehallgatni kívánt csatorna kiválasztása után a főképernyőn több nézetben is nyomon követhetjük a hálózat forgalmát (5.7. ábra). A bal felső ablakban az üzenetek kronológia szerint követik egymást. Minden eszköz üzenetei külön sorban jelennek meg, emellett színezéssel és a request-response párok összekötésével segítik átlátni az üzenetfolyamot. A jobb felső ablakban a hálózat topológiája figyelhető meg, ahol a koordinátort jeladó szimbólummal emelik ki a többi eszköz közül. A képernyő alján az üzenetfolyamból kiválasztott üzenetek tartalmát elemezhetjük, bal oldalon grafikus, jobb oldalon karakteres nézetben. A 5.7. ábrán a StickBee és az ESI közötti üzenetváltás figyelhető meg a komissiózás közben.



5.7. ábra. *A Peryton főképernyője.*

Teszt 1 Az első tesztkonfiguráció eszközeit a 5.1. táblázat foglalja össze.

5.1. táblázat. *A teszt hálózat eszközei.*

Eszköz megnevezése	ZigBee szerep	Device profile	IEEE cím
StickBee (Freescale Network Node)	ZED	IPD	00 50 C2 37 B0 04 5C 61
Silicon Labs EM357 breakout board	ZC	ESI	82 63 41 E8 5B 1A D5 43
Freescale Sensor Node	ZR	MD	00 50 C2 37 B0 04 52 EB

A teszt során a következőket vizsgáltam:

1. IPD csatlakozása az ESI-hez, CBKE, időszinkronizáció
2. ESI Metering cluster szerver felderítése, mérőóra adatok kiolvasása (közvetlen)
3. MD hozzáadása a hálózathoz, IPD újracsatlakozás (silent rejoin)
4. Új Metering cluster szerver észlelése a hálózaton
5. Link Key létesítése az IPD és az MD között
6. MD mérőóra adatok kiolvasása (közvetett, közvetítő: ESI)
7. MD és ESI adatainak periodikus lekérdezése

Teszt 2 A második tesztkonfiguráció eszközeit a 5.2. táblázat foglalja össze.

5.2. táblázat. *A teszt hálózat eszközei.*

Eszköz megnevezése	ZigBee szerep	Device profile	IEEE cím
StickBee (Freescale Network Node)	ZED	IPD	00 50 C2 37 B0 04 5C 61
Digi ConnectPort X2	ZC	ESI	00 40 9D FF FF 4C 02 E2

A teszt során a következőket vizsgáltam:

1. IPD csatlakozása az ESI-hez, CBKE, időszinkronizáció
2. üzenet küldése az IPD-nek (nyugtázás nélkül)
3. IPD újracsatlakozás (silent rejoin), legutolsó üzenet lekérdezése
4. üzenet visszavonása
5. üzenet küldése az IPD-nek (nyugtázással)

A tesztek során a kommunikáció hibamentesen zajlott, csak néhány megismételt üzenet volt, aminek az oka valószínűleg a környezetben található Wi-Fi hálózatok okozta interferencia lehetett. Az elkészült szoftvert ez alapján alkalmasnak találtam arra, hogy részt vegyen a cég pilot projektjeiben.

6. fejezet

Összefoglalás

6.1. Eredmények

A StickBee költségoptimalizált változatához a Freescale KW20 került kiválasztásra, noha egy megjelenés előtt álló processzorra alapozni mindig kockázatos döntés. Eredeti feladatom a KW20 alapú StickBee szoftverének megtervezése lett volna, azonban az IC megjelenése azóta is folyamatos késésben van. A termékfejlesztés ettől függetlenül nem állhatott meg, ezért a fejlesztés Freescale másik ZigBee platformján, az MC13226V SoC-n folyt.

Munkám során beállítottam a stack paramétereket, pótoltam a stack hiányosságait, megterveztem és megvalósítottam a StickBee központi processzorát kiszolgáló alkalmazást. Az elkészült alkalmazás képes csatlakozni ZigBee Smart Energy hálózatokhoz, onnan adatokat begyűjteni és azokat a megjelenítés számára kiszolgálni. A StickBee bekapcsolás után automatikusan megkeresi a hálózatban található összes mérőórát és állandó időközönként lekérdezi őket. A StickBee szolgáltatói üzeneteket is képes fogadni, amik szükség esetén nyugtázhatók. Az alkalmazás moduláris felépítése miatt könnyen bővíthető további funkciókkal.

A StickBee kommunikációjának tesztelésére teszt hálózatot üzemeltem be, a teszteket a Perytons Protocol Analyzerrel végeztem. A tesztek alapján a StickBee részt vehet pilot projektekben.

Munkám során feldolgoztam és alkalmaztam a ZigBee szabványokat, tapasztalatokat szereztem a kis hatótávolságú rádiós adatátvitelben. Emellett gyakorlatot szereztem beágyazott C programozásban, és a JTAG debuggerek működésében, melynek eredményeként a `binflasher` is megszületett.

6.2. Kitekintés

A StickBee elérte azt a fejlesztési állapotot, mikor bemutatható és tesztelési célzattal kiadható a cég ügyfeleinek. A termék funkcionalitása azonban még nem teljes, és a költségoptimalizáláson is át kell esnie a sorozatgyártás megkezdése előtt. ZigBee szempontból hiányzó funkció a tarifák kezelése. Ennek oka, hogy a Price cluster használatára még nincs egységesen bevett módszer, attribútumai a [8] szabvány figyelmeztetése szerint ideiglenesek. Az implementáció függhet a piactól is, a jelenlegi

moduláris felépítés flexibilis az ilyen jellegű bővítés szempontjából.

Fontos lépés lesz az egy processzorra való áttérés, mikor össze kell hangolni a jelenleg szeparáltan, két processzorban futó kódrészeket. A processzorváltás a komplett ZigBee stack kicserélését is magával vonhatja, ami tovább gyarapítaná a ZigBee területén megszerzett tapasztalataimat.

A StickBee piaci megjelenésnek előfeltétele a tanúsítási folyamat is, aminek a végén a StickBee *ZigBee Certified Product* lesz. A tanúsítást átfogó belső teszteknek kell megelőzni, ezek megtervezése és lebonyolítása szintén az én feladatom lesz.

Inspirációt jelent a fejlesztésre a ZigBee IP specifikáció és a Smart Energy Profile 2.0-s változatának közelmúltban történt megjelenése, a platformváltáskor meg fogom vizsgálni az áttérés lehetőségét.

Irodalomjegyzék

[1] iDigi-SE. `https://idigi-se.appspot.com`.

[2] Prolan Smart Energy Rendszerház Kft. `http://www.prolan.hu/a-prolan-csoport/ipsol-rendszerhaz-kft-2/`.

[3] ZigBee Alliance. `http://www.zigbee.org/`.

[4] IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). *IEEE Std.*, 2003.

[5] ZigBee Specification, Zigbee document 053474r19. *ZigBee Alliance*, 2008.

[6] FAQ: Certicom ZigBee Smart Energy. `http://www.certicom.com/images/pdfs/ZigBee/faq-zigbee-security.pdf`, 2011.

[7] Internal Communication Protocol Specification. *IPSOL Kft.*, 2011.

[8] ZigBee Smart Energy Profile Specification, ZigBee document 075356r17. *ZigBee Alliance*, 2011.

[9] International Energy Agency. Key World Energy Statistics 2012. `http://www.iea.org/publications/freepublications/publication/kwes.pdf`.

[10] Sahin Farahani. *ZigBee Wireless Networks and Transcievers.* Newnes Publications, 2008.

[11] Freescale Semiconductor. *MC1322x, Advanced ZigBee-Compliant SoC Platform for the 2.4 GHz IEEE 802.15.4 Standard, Reference Manual*, 2012. rev. 1.6.

[12] Drew Gislason. *ZigBee Wireless Networking.* Newnes Publications, 2008.

[13] Roland Menyes, Salacz Ágoston, and Ervin Szabó. *Prolan Energiamenedzsment Rendszer.* PROLAN Irányítástechnikai Zrt., 2011.

[14] György Varjú. Az okos hálózati technológiák jelenlegi helyzete és jövőbeni irányai. *MEEE Elektrotechnika*, 106(2):5–8, February 2013.