VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

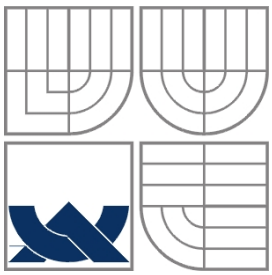
MULTIMEDIÁLNÍ PODPORA VÝVOJE APLIKACÍ PRO SENZOROVÉ SÍTĚ

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

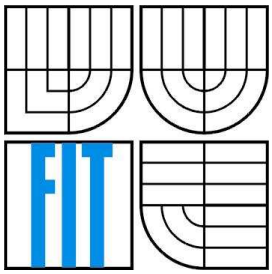
AUTOR PRÁCE
AUTHOR

RADIM MARTÍNEK

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MULTIMEDIÁLNÍ PODPORA VÝVOJE APLIKACÍ PRO SENZOROVÉ SÍTĚ

MULTIMEDIA SUPPORT FOR SENSOR NETWORK EDUCATION

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

RADIM MARTÍNEK

VEDOUCÍ PRÁCE
SUPERVISOR

MGR. ROMAN TRCHALÍK

BRNO 2008

Zadání bakalářské práce/6957/2007/xmarti42

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Martínek Radim**
Obor: Informační technologie
Téma: **Multimediální podpora vývoje aplikací pro senzorové sítě**
Kategorie: Počítačové sítě

Pokyny:
Cílem této práce je vytvořit materiály pro podporu výuky bezdrátových senzorových sítí.

1. Nastudujte problematiku senzorových sítí.
2. Danou problematiku roztřiďte podle obsahu do jednotlivých kapitol.
3. Dané kapitoly logicky setřiďte a navrhněte multimediální podporu, která bude sloužit pro podporu tvorby jednoduchých aplikací.
4. Navrženou multimediální podporu (WWW stránky) implementujte dle pokynů vedoucího práce.
5. Vytvořte jednoduché ukázkové aplikace pro senzorové sítě ZigBee. Jednotlivé aplikace dokumentujte krátkým manuálem.

Literatura:

- Ilyas, M., Mahgoub, I.: Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems, CRC Press LLC, Boca Raton, FL, USA, 2004.
- National Institute of Standards and Technology, NIST IEEE 1451, [online] < http://www.motion.aptd.nist.gov>
- ZigBee Alliance: ZigBee Specification v 1.1. ZigBee Alliance Board of Directors, 2006. [online] < http://www.zigbee.org>
- Pužmanová, R.: Routing and Switching, Time of convergence, ISBN: 9780201398618
- Furht, B., Ilyas, M.: Wireless Internet Handbook, ISBN:9780849315022
- Gutierrez, J. A.,Callaway, H. E., Barreett, R. L.: Enabling Wireless Sensors with IEEE 802.15.4, Low-Rate Wireless, IEEE Press, 2003
- Callaway, H. E.: Wireless sensor networks, 2004
- Varghese, G.: Network algorithmics, 2005
- Sohraby, K., Minoli, D.: Wireless sensor networks: technology, protocols, and applications; 2007

Při obhajobě semestrální části projektu je požadováno:
- Body 1 - 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Trchalík Roman, Mgr.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Radim Martínek**
Id studenta: 78870
Bytem: Mírová 230, 747 61 Raduň
Narozen: 09. 06. 1986, Opava
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Multimédiální podpora vývoje aplikací pro senzorové sítě
Vedoucí/školitel VŠKP: Trchalík Roman, Mgr.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Tato bakalářská práce byla psána s cílem podpořit vývoj aplikací vhodných pro použití v senzorových sítích. Poskytuje základní teoretické a praktické informace, které by studentovi mírně znalému oboru informačních technologií, měly pomoci seznámit se s oblastí bezdrátových senzorových sítí. Do větších podrobností je zde rozebrána technologie ZiGbee, která staví na standardu IEEE 802.15.4. Součástí této práce jsou webové stránky, zahrnující všechny důležité informace zde napsané.

Klíčová slova

senzorová síť, IEEE 802.15.4, ZigBee, PICDEM Z kit, PIC18F4620, MPLAB ICD 2, síťový analyzátor ZENA, MPLAB IDE, Microchip ZigBee stack

Abstract

The aim of this bachelor's thesis is to support development of applications, which are suitable for operations in a sector of a sensor network. It provides for theoretical and practical knowledge of a wireless sensor network, so that any student of information technology can be easily acquainted with its advantages and disadvantages. It uses technology ZiGbee, built on standard IEEE 802.15.4 as the main platform. This technology is thoroughly analysed in the text. An essential component of the thesis is also a website, which includes all the important information mentioned within the scope of this work.

Keywords

sensor network, IEEE 802.15.4, ZigBee, PICDEM Z kit, PIC18F4620, MPLAB ICD 2, ZENA network analyzer, MPLAB IDE, Microchip ZigBee stack

Citace

Martínek Radim: Multimediální podpora vývoje aplikací pro senzorové sítě, bakalářská práce, Brno, FIT VUT v Brně, 2008.

Multimediální podpora vývoje aplikací pro senzorové sítě

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Mgr. Romana Trchalíka.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Radim Martínek
9. 5. 2008

Poděkování

Děkuji vedoucímu mé bakalářské práce Mgr. Romanovi Trchalíkovi, za poskytnutí četných konzultací, rad a připomínek a dále bych rád poděkoval všem, kteří mně byli při tvorbě této práce nápomocní.

Obsah

Obsah 1
Úvod 3
1 Bezdrátové senzorové sítě 4
1.1 Rozdíly oproti klasické bezdrátové síti 4
1.2 Návrh bezdrátové senzorové sítě 4
1.3 Architektura 5
1.4 Distribuce dat 6
1.5 Sběr dat 7
1.6 MAC protokol 8
1.7 Lokalizace uzlů 9
1.8 Komunikace v reálném čase 9
1.9 Bezpečnost 10
1.10 Vlastnosti bezdrátových senzorových sítí 10
1.11 Požadavky v závislosti na komunikačních úrovních sítě 10
1.12 Dosažení bezpečného provozu v reálném čase 11
1.13 Aplikace 11
2 Standard IEEE 802.15.4 13
2.1 Fyzická vrstva 13
2.2 Linková vrstva 14
3 ZigBee technologie 16
3.1 Typy zařízení 16
3.2 Topologie sítě 17
3.3 Adresování 17
3.4 Profily 18
3.5 Formát datových rámců 19
3.6 Komunikační model standardu Zigbee 19
3.7 Optimalizace spotřeby elektrické energie 20
3.8 Bezpečnost 20
3.9 Srovnání s ostatními bezdrátovými standardy 22
4 Popis práce v laboratoři 24
4.1 Hardwarové vybavení 24
4.2 Softwarové vybavení 25
5 Ukázkové aplikace 29
5.1 První aplikace – mikrokontrolér 29

5.2 Druhá aplikace – světelné diody 29
5.3 Třetí aplikace – teplotní senzor 31
5.4 Možné komplikace 31
Závěr 33
Literatura 34
Seznam příloh 36
Přílohy 37

Úvod

Senzorové sítě se stávají čím dál častěji součástí našeho života. Už se s nimi nesetkáme jenom ve specializovaných odvětvích, kam má přístup jenom několik vyvolených, nýbrž nás začínají obklopovat v práci nebo v domácnostech a pomáhají nám ulehčit si námahu, zvyšují životní standard a zjednodušují mnoho činností, které byly dříve v oblasti scifi. Tyto moderní sítě jsou navrhovány tak, aby byly schopny již od sestavení fungovat samostatně a co nejdéle. Příkladem mohou být různé monitorovací a ovládací systémy, které samostatně mohou být doplňkem nějakých větších systémů, ať už bezdrátových nebo jiných.

Tato práce si klade za hlavní cíl seznámit širší okolí s existencí senzorových sítí, ulehčit proniknutí do základních principů a ukázat základy, jakým způsobem lze v praxi takovéto sítě vytvářet. Důležitost znalostí senzorových sítí je opodstatněná vzrůstajícími požadavky naší společnosti, přičemž je v této oblasti stále nedostatečné množství odborníků. Součástí této bakalářské práce bude multimediální podpora ve formě webových stránek, která bude obsahovat veškeré důležité informace zde uvedené, pro tvorbu aplikací bezdrátových senzorových sítí.

První kapitola se zabývá senzorovou sítí obecně. Je zde definován pojem bezdrátová senzorová síť, čím je odlišná od klasické sítě, jaké existují základní architektury a jakým způsobem může probíhat komunikace. Dále se tato část věnuje vlastnostem a požadavkům senzorových sítí a jejich uplatnění v praxi.

Druhá kapitola se zabývá standardem IEEE 802.15.4, který byl navržen, aby stanovil základní vrstvy (fyzickou a linkovou) komunikačního modelu senzorové sítě. Tento standard slouží jako základ pro kapitolu třetí, která se věnuje technologii ZigBee, což je standard definující ostatní vyšší vrstvy komunikačního modelu senzorové sítě. Budou zde zmíněny typy zařízení, topologie sítě, adresování, datové rámce, bezpečnost a bude zde provedeno srovnání s dalšími technologiemi, které jsou aktuálně k dispozici na trhu.

Ve čtvrté kapitole je proveden podrobný popis, jakým způsobem lze začít práci v laboratoři. Je zde uvedeno nutné hardwarové vybavení pro tvorbu senzorové sítě a jednotlivé prvky jsou v základech popsány. Dále je uvedeno nutné softwarové vybavení důležité pro vývoj aplikací společně s postupy, jak s nimi začít pracovat.

V páté kapitole jsou zdokumentovány tři ukázkové aplikace, kde dvě z nich jsou postaveny na ZigBee stacku od firmy Microchip a vychází z demonstračních příkladů, které byly upraveny pro potřeby této práce. Navíc jsou zde popsány možné komplikace, se kterými se lze při vývoji setkat. Poslední kapitola popisuje závěrečné zhodnocení práce, možné náměty na zdokonalení a přínos, který mi práce v oblasti senzorových sítí dala.

1 Bezdrátové senzorové sítě

Bezdrátová senzorová síť se skládá z prostorově distribuovaných autonomních zařízení, která obsahují senzory, navzájem spolu komunikují a monitorují určitý systém. Uzel senzorové sítě se skládá ze tří částí. Ze senzoru, což je snímač schopný změřit fyzikální, chemickou, nebo biologickou hodnotu v měřeném systému. Ze systému pro zpracování zaznamenaných dat a systému pro řízení komunikace. Zatímco senzor sám o sobě pracuje na velmi omezeném prostoru s co nejmenší výpočetní silou a nízkou spotřebou energie, sít poskládaná z velkého počtu těchto zařízení obvykle bývá robustní, spolehlivá, přesná a může pokrývat prostor o velkých vzdálenostech.

Bezdrátové senzorové sítě jsou v mnoha případech odolné vůči chybám, jelikož více uzlů mohou snímat stejnou informaci, jejíž správnost lze porovnat. Tyto sítě plní dva hlavní úkoly. Distribuce dat (propagace dat a dotazů skrze síť) a shromažďování údajů (sběr sesbíraných informaci z jednotlivých senzorů).

1.1 Rozdíly oproti klasické bezdrátové síti

Bezdrátové sítě se obecně skládají z uzlů, které mezi sebou komunikují. Počet těchto uzlů je v případě senzorické sítě několikanásobně větší než v případě „Ad Hoc“ sítí. Senzorové uzly nemusí mít unikátní adresu. Zdroj napájení v senzorových uzlech je většinou z baterie. Senzorové sítě jsou data-centrické, to znamená, že komunikace s uzly sítě může probíhat na základě senzorem naměřené hodnoty (např. dotaz může být poslán všem uzlům, které naměřili teplotu 40 °C). Z těchto důvodů požadují senzorové sítě jiné mechanismy pro směrování a následné odpovědi na dotazy.

Charakteristickým rysem senzorové sítě je spojení nahromaděných informací dohromady před odesláním, tím se docílí snížení potřeby šířky pásma, a zmenšení požadovaného příkonu pro komunikaci.

1.2 Návrh bezdrátové senzorové sítě

Je nutné počítat se situací, že senzory mohou být náhodně rozmístěny a z hlediska topologie nemusí mít žádný standardní tvar. Proto se požaduje, aby nastavení a údržba sítě probíhala zcela automaticky. Tato nestrukturovanost sítě vede k tomu, aby směrovací algoritmy a údržba byly zcela distribuovány.

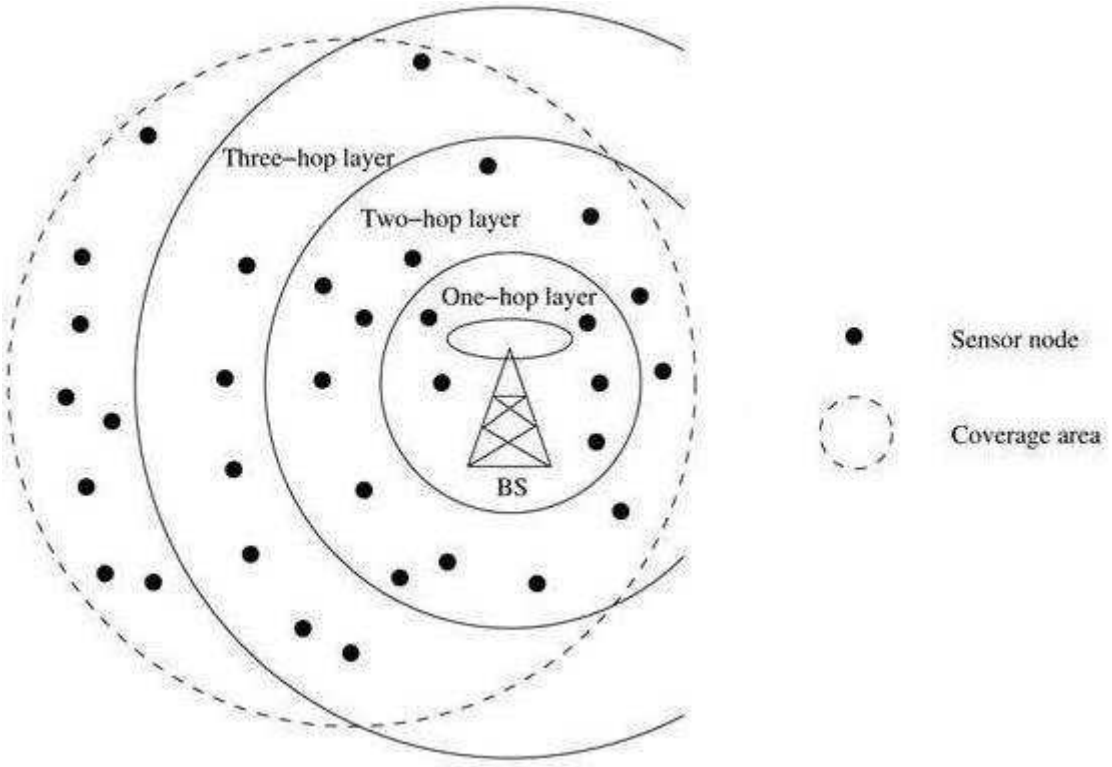
Senzory se obvykle spoléhají pouze na energii z baterie, což je považováno za nejvýznamnější omezení při navrhování protokolů. Uživatel by měl dostat možnost porovnat a analyzovat životnost, odolnost a přesnost výsledků jednotlivých zařízeni. Mikrokontrolér, operační systém a aplikace je nutné navrhovat s ohledem na nízkou spotřebu energetických zdrojů. Senzory by měly mít schopnost navzájem se v distribuovaném režimu synchronizovat, takže předepsané plánování a spánkové režimy musí být vykonávány bez dvojznačnosti. Síť musí být schopna reagovat na selhání nějakého zařízení a případně nahradit funkčnost zařízením jiným. Směrovací protokoly by měly umět dynamicky reagovat na změny v síti. Pro komunikaci v reálném čase musí být stanovena maximální doba odezvy a minimální šířka komunikačního pásma. Požaduje se, aby existovala opatření zajišťující bezpečný způsob komunikace a nedocházelo ke zneužití citlivých informací.

1.3 Architektura

Návrh senzorových sítí je ovlivněn faktory, jako jsou rozšiřitelnost, odolnost a příkon. Architekturu těchto sítí lze rozdělit na vrstvenou a seskupenou.

Vrstvená architektura se skládá ze základní stanice, (BS – base station) a senzorů (uzlů), které jsou soustředěny v okolí základní stanice. Jednotlivým uzlům je přiřazena náležitost do dané vrstvy podle vzdálenosti od základní stanice. Základní stanice je schopna přímo komunikovat se všemi uzly, kdežto uzly mohou poslat informace BS pouze prostřednictvím sousední vrstvy, která je blíže k BS. Zároveň platí, že vrstva vyskytující se nejblíže k BS už komunikuje přímo se základní stanicí.

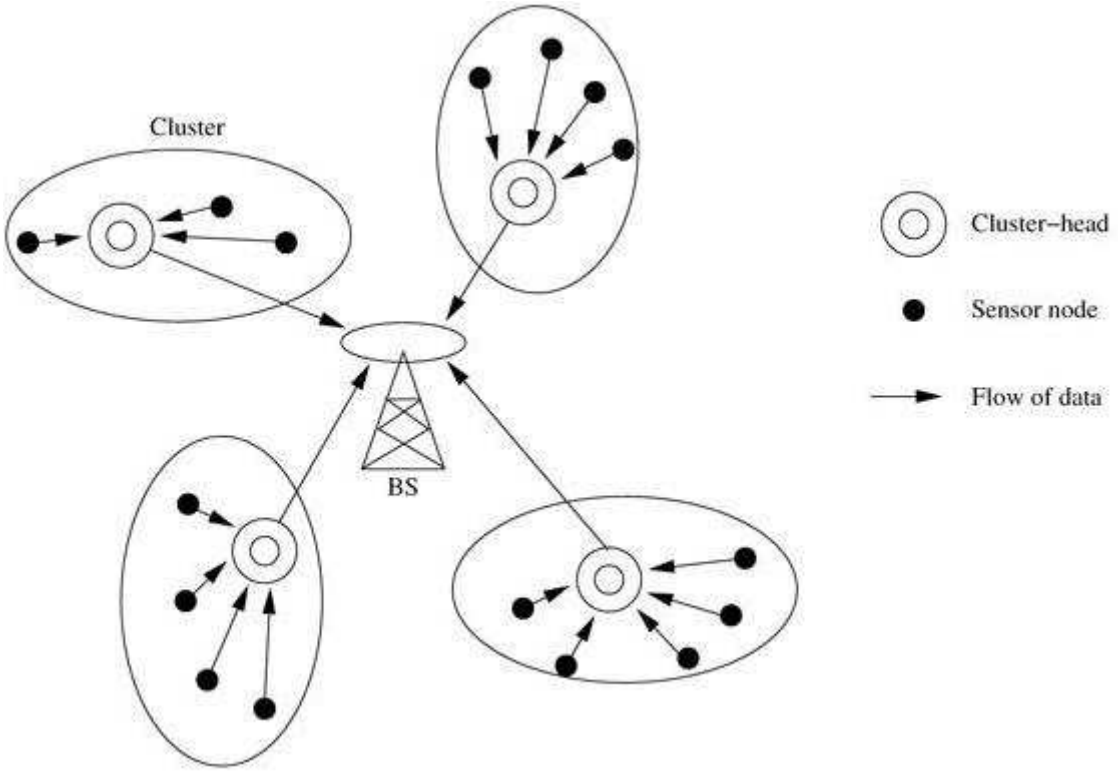
Obr. 1: *Vrstvená architektura.*



Základní stanice slouží jako spojení s klasickou sítí. Uživatelé této sítě mají zařízení, které jim umožňuje komunikovat pomocí jednotlivých uzlů bezdrátové sítě se základní stanicí, ta obvykle tyto informace shromažďuje a distribuuje je pomoci spojení dále do nadřazené sítě. Tento typ architektury je popsán skupinou protokolů UNPF (unified network protocol framework). UNPF definuje tři operace: inicializace a údržba sítě, MAC protokol a směrovací protokol. Výhody vrstvené architektury jsou v energetické nenáročnosti komunikace, v důsledku zapojení uzlů na krátký dosah.

Seskupená architektura organizuje uzly sítě do skupin, kde v každé skupině existuje jeden řídící uzel. Hlavu celé sítě tvoří opět základní stanice, která je jako v případě vrstvené architektury připojena k nadřazené síti. Koncové uzly komunikují s řídícím uzlem a ty dále komunikují se základní stanicí.

Obr. 2: *Seskupená architektura.*



Řídící uzly v jednotlivých skupinách shromažďují data od svých uzlů, s těmito daty provádějí různé výpočty a základní stanici posílají pouze výsledek. Tím se dosáhne velké úspory při komunikaci a spotřebě energie. Vytvoření skupin uzlů a zvolení řídících uzlů musí probíhat autonomně. Jak tyto autonomní a distribuované procesy probíhají je popsáno v protokolu LEACH (low-energy adaptive clustering hierarchy). Ten definuje dvě fáze, nastavení sítě a ustálení. Při fázi nastavování se uzly přiřadí do skupin a zvolí se řídící uzel. Poté následuje fáze ustálení, ta trvá mnohonásobně delší dobu, aby se minimalizovala režie vzniklá pří nastavování. Při této fázi plní síť svou funkci, sbírá informace, zpracovává je a dále posílá výsledky základní stanici. Tyto dvě fáze se pravidelně střídají. V rámci nastavení sítě jsou náhodně vybrány řídící uzly a k nim jsou přiřazeny uzly tvořící dané skupiny. Po určité době se tento proces opakuje, tím je zajištěna rovnoměrná spotřeba energie všech uzlů v senzorové síti.

1.4 Distribuce dat

Posílání dotazů a dat v rámci senzorové sítě nazýváme distribuce dat. Uzel generující data nazýváme zdroj a informace k odeslání nazýváme událost. Uzel vyhledávající informace o událostech nazveme přijímač. Komunikační modely vyvinuté pro senzorové sítě se skládají ze sběru a šíření dat. Sběrem dat se myslí poslání dat ze zdroje k zařízení, které je určeno pro sbírání informaci (například základní stanice), toto zařízení data vyhodnotí a rozhodne o další činnosti. Sběr dat se může konat periodicky, nebo na vyžádání příslušného zařízení. Šíření dat se skládá z dvou částí: zájmová propagace a propagace dat. Příjemce vysílá periodicky zájem (tj. identifikátor události) okolním uzlům pro každou událost, která ho zajímá. Tento zájem je zasílán skrze síť a každý uzel si udržuje ve své vyrovnávací paměti všechny ohlášené události, o které je projeven zájem. Pokud je detekována událost, ohlásí se to uzlům, které projevili zájem. Mezilehlé uzly udržují údaje ve vyrovnávací paměti, přičemž mohou

tato data seskupovat, nebo popřípadě měnit jejich poměr. Cesta určená k šíření dat může být dynamicky modifikována ku prospěchu nalezení nejkratší cesty. Mezi algoritmy implementující zájmovou propagaci a propagaci dat patří například Flooding, Gossiping, Rumor Routing, Sequential Assignment Routing.

Algoritmus **Flooding** je založen na principu, kde uzel, který přijme data, je vzápětí také vysílá v případě, že nebyl překročen maximální počet možných přeposlání a daný uzel není adresátem těchto dat. Flooding nevyžaduje znalost topologie, ale má několik nevýhod. Může se stát, že uzel přijme kopie stejného paketu vícekrát od sousedních uzlů. V případě, že více uzlů monitorují stejnou událost, bude do sítě vícekrát poslána informace o stejné události. Tato metoda nemonitoruje dostupnou energii sítě, proto je nutné zavést velké množství redundance, což snižuje celkovou životnost bezdrátové sítě.

Gossiping je modifikovaná varianta Floodingu, přičemž uzel přijímající paket jej odesílá pouze náhodně vybranému sousedovi, nikoliv všem. Tím odpadá řada problémů, avšak zvětší se doba, za kterou paket dorazí až k adresátovi, navíc není zaručeno, že všechny uzly senzorové sítě mohou přijmout vyslaný paket.

Rumor Routing je algoritmus založený na „agentech“. Agenti jsou vytvářeni náhodně jednotlivými uzly sítě a představují zvláštní typ paketu, který se pohybuje po síti a snaží se nalézt nejkratší cestu pro šíření událostí, s kterými se potkají. V případě, že agent nalezne uzel, který má nastavenou cestu k události delší než on sám, aktualizuje směrovací tabulku daného uzlu. Pokud je uzlem generován dotaz (tzn., že má v plánu přijmout událost), je poslán náhodným směrem a doufá se, že nalezne cestu (sestavenou agentem) vedoucí k požadované události. Jestliže přesto dotaz nenalezne adresáta, je použita metoda Flooding pro zaslání dotazu.

1.5 Sběr dat

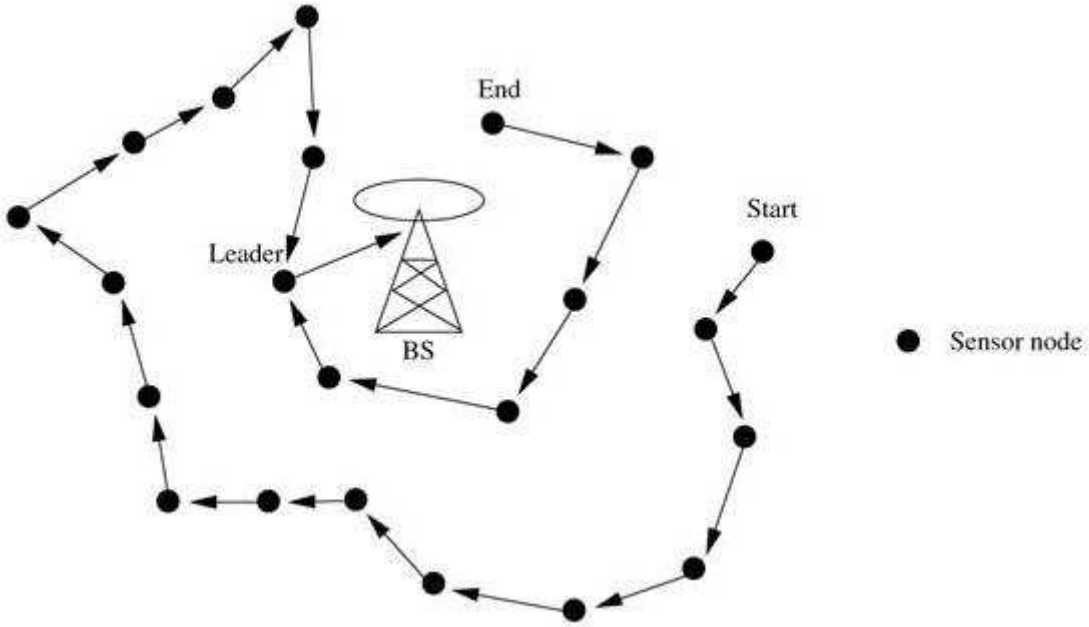
Problémem sběru dat je dopravit je z každého senzorového uzlu k základní stanici. Hlavním úkolem algoritmů implementujících sběr dat je maximalizovat počet přenesených dat v době životnosti sítě. To znamená, že se požaduje minimalizovat energii potřebnou pro komunikaci s minimální dobou odezvy. Tyto požadavky jsou protichůdné a hledá se proto vyhovující poměr mezi cenou energie a dobou odezvy. Algoritmy zabývající se sběrem dat jsou například Direct Transmission, Power-Efficient Gathering for Sensor Information Systems, Binary Scheme.

V algoritmu **Direct Transmission** senzorové uzly posílají data základní stanici přímo, což při velkých vzdálenostech od základní stanice klade velké nároky na potřebnou energii. Navíc jednotlivé uzly se musí v komunikaci střídat, aby zabránily kolizím, z čehož pramení delší doba odezvy. Tento algoritmus se příliš nepoužívá, protože poměr mezi cenou energie a dobou odezvy není vyhovující.

Power-Efficient Gathering for Sensor Information Systems (PEGASIS) předpokládá, že každý uzel v síti zná polohu všech ostatních uzlů a dokáže se spojit se základní stanicí přímo. Hlavní cíle algoritmu PEGASIS jsou minimalizovat vzdálenosti, přes které probíhá komunikace, minimalizovat režijní přenosy, minimalizovat počet zpráv nutných k zaslání základní stanici a distribuovat spotřebu energie rovnoměrně přes všechny uzly sítě. Z uzlů se zkonstruuje řetězec začínající nejvzdálenějším uzlem od základní stanice. Další člen řetězce se určí podle nejbližšího uzlu od právě připojeného uzlu, který ještě nebyl zapojen v řetězci. Řetězec se sestavuje před odesláním dat a nebo v případě výpadku nějakého senzoru. V každém uzlu může docházet k nahromadění dat k odeslání, protože pouze jedna zpráva muže být poslána z jednoho uzlu na druhý. Uzel určený jako

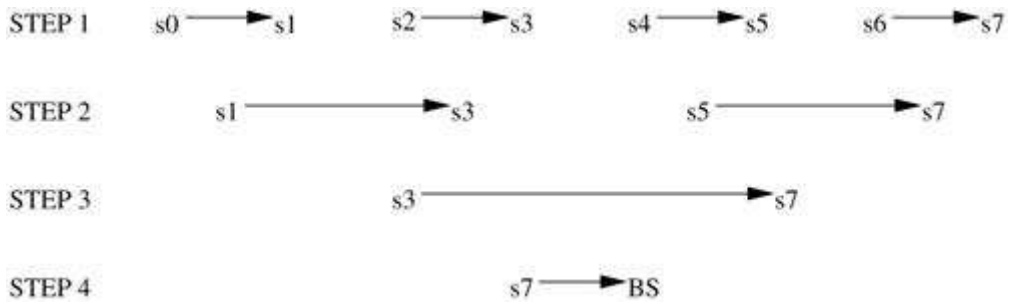
vedoucí, přepošle zprávu základní stanici. Uzly musí vědět, který uzel je v danou chvíli vedoucí, aby poslaly data správným směrem.

Obr. 3: *Algoritmus PEGASIS.*



Algoritmus **Binary Scheme** je založen opět na vytvoření řetězce (stejně jako PEGASIS), který klasifikuje uzly do několika úrovní. Požaduje se, aby komunikace používala CDMA, díky čemuž je možné komunikovat na všech vrstvách současně. Každý uzel, který příjme data na nějaké úrovní, je odesílá podle řetězce pro úroveň následující. Počet uzlů v následujícím řetězci je vždy poloviční než počet uzlů v předchozím řetězci.

Obr. 4: *Algoritmus Binary Schneme.*



1.6 MAC protokol

Úkolem protokolů MAC vrstvy v senzorových sítích je sestavit infrastrukturu sítě tak, aby existovalo spojení mezi náhodně rozmístěnými senzory, kterých může být na tisíce. Musí se zajistit rovnoměrné rozdělení komunikačních zdrojů mezi všemi uzly sítě pro maximální délku životnosti sítě. Navíc je nutné počítat s častou změnou topologie sítě v důsledku pohyblivosti uzlů nebo jejich výpadku. Existují tři základní typy MAC protokolů pro bezdrátové senzorové sítě. Pevně stanovené, založené na žádosti a založené na ohlášení.

U pevně stanovených MAC protokolů uzly sdílí společné komunikační prostředky, jejichž použití je předem přesně stanoveno. Poskytují pro každý uzel omezené zpoždění, což může v případě nárazové komunikace, kde se potřeba komunikačních prostředků v čase mění, způsobit neefektivní využití prostředků. Je vhodné jej použít pro senzorové sítě, kde se provádí průběžné sledování a generují se deterministická data. **Protokoly založené na žádosti** se používají v případech, kde jsou prostředky pro komunikaci přidělovány podle požadavku na komunikaci. Ačkoliv je tento přístup spojen se zvýšenou režií, je tento neduh kompenzován efektivním využitím komunikačního pásma. V případě **protokolů založených na ohlášení** se při potřebě poslat data generuje hlášení o komunikaci. Jsou vhodné pro nárazovou komunikaci, ale je zde možnost vzniku kolizí a tudíž nelze garantovat minimální dobu zpoždění. Nelze je tedy doporučit v sítích, kde se komunikuje v reálném čase. Příkladem MAC protokolů jsou SMACS (Self-Organizing MAC for Sensor Networks and Eavesdrop and Register), Hybrid TDMA/FDMA a CSMA-Based MAC Protocols.

1.7 Lokalizace uzlů

Informace o poloze uzlů je nutné zvažovat při agregaci měřených dat. Každý uzel musí znát svou polohu v rámci celé sítě a tuto informaci je nutné posílat v paketu současně s měřitelnými daty. Pro lokalizaci sítě se používají mechanizmy, po kterých se požaduje, aby byly dostatečně přesné a nespotřebovaly velké množství energie. Mezi tyto algoritmy patří Indoor Localization a Sensor Network Localization.

Při použití algoritmu **Indoor Localization** je nutné, aby byly v místě senzorové sítě napevno strategicky umístěny signalizační uzly. Poté náhodně rozmístěné uzly přijímají signál od signalizačních uzlů a podle intenzity jejich signálu, směru přijmu a časového rozdílu mezi příchodem různých signálů, dokážou odhadnout svou pozici. Základní stanice pak má obvykle v databázi uloženo aktuální rozmístění těchto uzlů.

Pro případy, kde se nevyskytuje pevná infrastruktura sítě, poskytuje řešení algoritmus **Sensor Network Localization**. Některé uzly jednají, jako by byly signalizačními uzly. Pomocí GPS zjistí svou polohu a periodicky posílají informační pakety svým sousedům.

1.8 Komunikace v reálném čase

Schopnost komunikace v reálném čase je základní požadavek senzorových sítí dohlížejících na nějaké kritické systémy, kde je potřeba vědět okamžitě změnu stavu nějaké citlivé události. Dobrým příkladem muže být sledování procesů v jaderné elektrárně. V těchto systémech je bezpodmínečně nutné mít garantovanou maximální dobu odezvy. Protokoly řešící otázku komunikace v reálném čase jsou SPEED a RAP.

SPEED je bezstavový protokol podporující komunikaci v reálném čase v bezdrátových senzorových sítích, představuje decentralizovaný algoritmus poskytující výběrové vysílání všem uzlům v specifickém regionu a podporu pro přenos paketů. Má minimální režii, nevyžaduje směrovací tabulku, ani žádnou speciální podporu fyzické vrstvy. Používá nedeterministické zasílání dat.

RAP protokol představuje API rozhrání pro aplikace adresující dotazy. Aplikace spuštěná na základní stanici specifikuje typ požadované události, oblast kam je dotaz adresován a maximální možnou respektovanou dobu příchodu odpovědi. Nižší vrstva RAP zajistí, že dotaz bude poslán všem

uzlům v předem specifikované oblasti a odpověď dorazí zpět základní stanici. Protokol RAP se skládá z protokolu LAP (lokalizuje adresy, pracuje na transportní vrstvě), protokolu VMS (geografický směrovací protokol) a MAC protokolu založeném na ohlášení s možností prioritní komunikace.

1.9 Bezpečnost

Bezdrátové sítě je nutné analyzovat i po bezpečnostní stránce, jelikož napadnout nezabezpečenou bezdrátovou síť je snazší než klasickou síť. Obvykle je z množiny uzlů v jedné oblasti poslána pouze jedna zpráva základní stanici, která obsahuje informace o stavu sledovaných událostí. Pokud by došlo k podvržení této zprávy, celá síť by mohla přestat plnit požadovanou funkčnost. Je tedy nezbytně nutné zajistit bezpečné komunikační kanály pro přenos citlivých dat.

Localized Encryption and Authentication Protocol (LEAP) je protokol zabývající se distribuci šifrovacích klíčů. Je založen na symetrickém šifrování, takže odesilatel i příjemce používají stejné klíče k šifrování. Rozmístění těchto klíčů se musí vykonat až po sestavení sítě.

1.10 Vlastnosti bezdrátových senzorových sítí

Každá bezdrátová síť má určité vlastnosti, které ji charakterizují a odlišují se tím od ostatních. Pro přehled jsou zde uvedeny základní z nich s popisem, co znamenají.

Významnou vlastností je **topologie sítě**, ta popisuje rozmístění uzlů v síti a jejich vzájemné propojení. Může být hvězdicová, stromová, nebo popřípadě různě kombinovaná. Další vlastností je **pokrytí sítě,** informuje o hustotě rozmístění senzorů v rámci jednoho území. Síť může nabývat pokrytí například hustého, řídkého, nebo nadbytečného. **Propojenost sítě** definuje, zda jsou uzly sítě navzájem v kontaktu po stálou dobu, a tedy je možné spolu komunikovat kdykoliv, nebo se některý z uzlů v čase odpojí a komunikaci vždy zprostředkovat nelze. Pak hovoříme, že propojenost je přerušovaná. Další vlastností je **velikost sítě,** zpravidla nám udává představu kolik navzájem propojených zařízení síť obsahuje. Senzorové sítě mohou mít velikost od jednotek až po tisíce navzájem propojených uzlů. Zajímavou vlastností je **rozmístění senzorů**. Na senzorové sítě je kladen velký důraz na samostatnost, proto mohou mít uzly sítě neplánovanou strukturu (například náhodnou, při rozmístění sesypem z letadla) a přesto budou fungovat. **Přemístitelnost senzorů** dále informuje, zda mají uzly schopnost se v průběhu času přemístit z místa na místo, nebo jsou případně připojené k pohybujícím se částem. Další významnou vlastností je **způsob komunikace**, která nemusí být jen radiová, ale také například světelná, nebo zvuková.

1.11 Požadavky v závislosti na komunikačních úrovních sítě

Architektura Senzorové sítě se dá rozdělit do tří úrovní. Nejnižší úroveň (actuator-senzor level), na které komunikují senzory mezi sebou. Provozní úroveň (field level), zde probíhá komunikace mezi centrální jednotkou a koncovými uzly. A nejvyšší úroveň (control level), podává informace o celé síti a kontroluje její funkcionalitu. Každá z těchto úrovní má trochu odlišné požadavky (komunikace

v reálném čase, objem přenášených dat, energetická spotřeba, bezpečnost a robustnost), které zde budou rozepsány.

Potřeba **komunikace v reálném čase** stoupá při komunikaci mezi koncovými zařízeními (zpoždění v tomto případě nabývá řádově několik milisekund), naopak tato potřeba klesá na vyšších úrovních. Doba zpoždění na provozní úrovní bývá obvykle v rozsahu mezi 100ms až 10s.

Objem přenášených dat je nejvyšší na kontrolní úrovni (přenáší se zde konfigurační informace a přenášená data), na nejnižší úrovni se přenáší povětšinou pouze několik bajtů (například, zda je něco zapnuto, nebo informace o teplotě), potřeba šířky pásma je proto minimální.

Samostatné senzory (koncové zařízení) bývají napájeny bateriově nebo pomocí alternativních zdrojů, což klade velký důraz na **nízkou spotřebu energie** pro dlouhou výdrž baterií (sto a více dní). A nakonec požadavky na **bezpečnost a robustnost** jsou největší na nejnižší úrovni „actuator-senzor level“.

1.12 Dosažení bezpečného provozu v reálném čase

V důsledku toho, že na nejnižší úrovni je požadováno zpoždění maximálně v rozpětí několika milisekund, musí docházet při zachování bezpečného provozu k vysoké míře redundance na různých úrovních. Dále jsou uvedeny redundantní techniky pro fyzickou, linkovou a síťovou vrstvu.

Redundanci na fyzické vrstvě lze docílit vícenásobným komunikačním spojením nebo popřípadě vícenásobně souběžně použitou frekvencí, což znamená, že systémy používají více než jednu komunikační frekvenci. Toho lze docílit dvěma způsoby. Pomocí vícenásobných frekvenčních pásem (například 433MHz a 868MHz) nebo pomocí vícenásobných frekvenci v jednom pásmu. Nejslibnější přístup je použití ortogonální frekvence (OFDM). Nicméně tento přístup je v současné době zřídka využívaný pro bezdrátové senzorové sítě, spíše se používá ve vyšších systémech, jako jsou IEEE 802.11g

Na linkové vrstvě téměř všechny protokoly využívají minimálně cyklickou kontrolu s ohledem na opravu chyb. Dále muže být dosaženo spolehlivosti pomocí systému potvrzování a opětovných přenosů v případech, kdy očekávané potvrzení nemůže být přijato originálním odesilatelem.

V síťové vrstvě lze využít možnost datových paketů putovat po vícenásobných cestách. Každý uzel může přijmout datový paket, pokud je v dosahu příjmu a proto může být aplikováno dynamické směrování, kde v případě husté topologie sítě lze aktivovat alternativní cestu, pokud předchozí selhala. Z důvodu odběru energie a složitosti sítě jsou směrovací algoritmy směrovány na požádání. Ve většině případů vede znovu přesměrování k zvýšení doby odezvy. Aby se dalo vyhnout zvýšení doby odezvy (zvláště pro „real-time“ aplikace), redundantní směrovací cesty jsou prováděny souběžně. Tato technika je dobře známa v telekomunikačních systémech. V současné době je tato metoda v bezdrátových senzorových sítích využívána pro armádní účely nebo speciální dopravu.

1.13 Aplikace

Aplikace bezdrátových senzorových sítí si najdou uplatnění zvláště v prostředí, kde vznikají požadavky na průběžné monitorování nějakého stavu nebo detekci specifických událostí. Například armádní systémy zahrnují pozorování bojiště, reakce na včasné události, navigaci řízených střel. Senzory se také používají pro **monitorování prostředí**, pro účely jako odhalení požáru nebo záplav, zkoumání výskytu živočichů v nějaké lokalizaci. Dále se hodí pro **pacientskou diagnostiku**,

monitorování životně důležitých funkcí pacienta v nemocnici. V neposlední řadě jsou významná odvětví pro použití senzorových sítí **správa a automatizace budov, ovládání topení, klimatizace, osvětlení a zabezpečení**.

[1], [2], [3]

2 Standard IEEE 802.15.4

IEEE 802.15.4 definuje fyzickou a linkovou vrstvu modelu ISO/OSI pro LR-WPAN (Low Rate Wireless Personal Area Network). Konkrétně popisuje komunikaci mezi zařízeními s malou přenosovou rychlostí a nízkou spotřebou energie, což přesně vyhovuje požadavkům senzorových sítí. Na fyzické a linkové vrstvě popsané v standardu IEEE 802.15.4 staví standard ZigBee popisující sítovou a aplikační vrstvu.

Standard je prakticky rozdělen do tří skupin. 802.15.4 – první edice byla vydána v roce 2003, ale v březnu 2004 po vzniku skupin 4a a 4b byl vývoj zastaven. 802.15.4a – sestaven v březnu 2005, zaměřen na možnosti rozšiřitelnosti sítě, rychlosti přenosu dat, delšího dosahu a nižšího příkonu sítě. 802.15.4b – zveřejněn v červnu roku 2006 jako IEEE 802.15.4-2006. Jak už z názvu napovídá, jde o specifikaci a objasnění standardu z roku 2003, řeší se zde hlavně otázky snižování zbytečné složitosti, zvýšení flexibility zabezpečení komunikace a možnosti nově dostupných frekvenčních pásem.

2.1 Fyzická vrstva

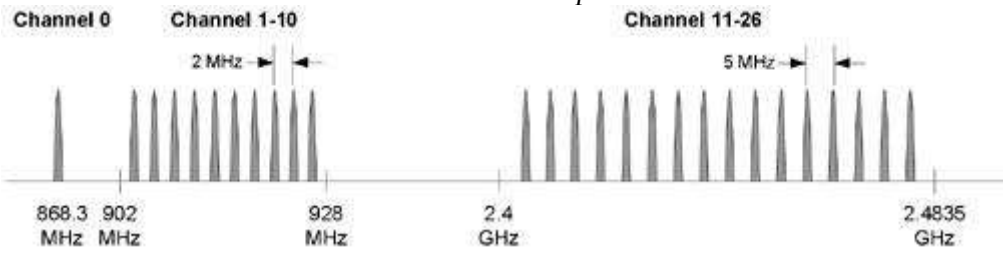
Fyzická vrstva má na starosti hlavně přenos a příjem dat, výběr vysílané frekvence, aktivaci a deaktivaci přijímání signálu, detekci minimálního množství energie nutného pro přenos signálu ED (Energy Detection), indikaci kvality komunikující linky LQI (Link Quality Indication) a detekci volného kanálu CCA (Clear Channel Assessment). Pro přenos dat je možno využít tři nelicencovaná frekvenční pásma.

První pásmo používá frekvenci 868 MHz, používá se v Evropě, rychlost přenosu dosahuje 20 b/s. Původně existoval jeden komunikační kanál, ale v roce 2006 byl rozšířen na tři. **Druhé pásmo** o frekvenci 902 – 928 MHz se používá v Americe a Austrálii, rychlost komunikace dosahuje 40 kb/s a počet kanálu byl v roce 2006 stanoven na třicet. **Třetí pásmo** má frekvenci 2400 – 2483.5 MHz, lze jej použít na celém světě s rychlostí až 250 kb/s a má 16 kanálů.

Šíření signálu v prostředí je založeno na technice přímého rozprostření spektra **DSSS** (Direct Sequence Spread Spectrum), kde každý bit pro přenos je nahrazen sekvencí bitů, tímto dochází k redundanci, díky které je signál rozprostřen do větší části radiového spektra. Signál je méně citlivý na rušení a to zvyšuje spolehlivost přenosu. Navíc se signál jeví cizím zařízením jako náhodný šum.

Pro přístup ke komunikačnímu kanálu se používá metoda **CSMA/CA** (Carrier Sense Multiple Access with Collision Avoidance). Její princip spočívá v naslouchání před vysíláním. Zařízení tím zjistí, zda je kanál volný a pokud ano, započne vysílání. V případě, že je kanál obsazen, zařízení vyčká, až bude moci vysílat.

Obr. 5: *Frekvenční pásma.*



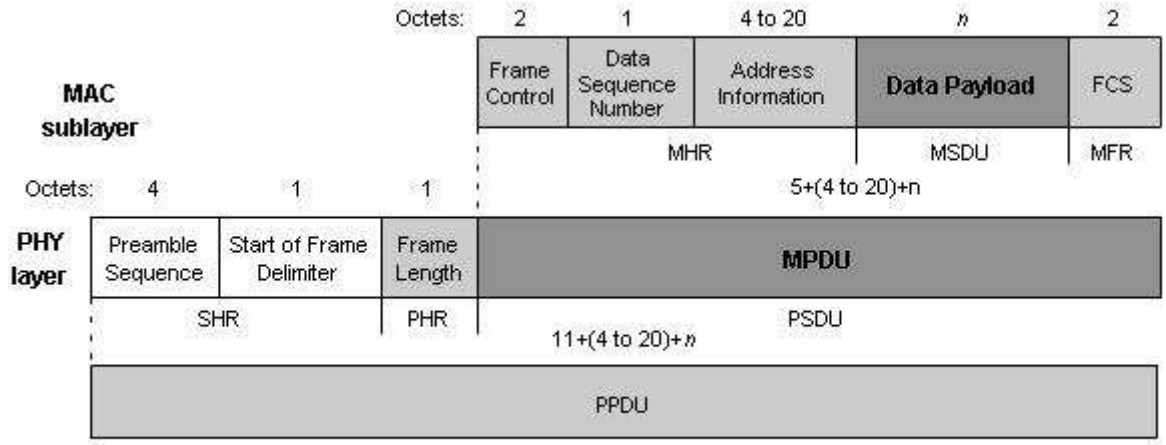
2.2 Linková vrstva

Linková vrstva nebo také MAC vrstva definuje komunikační protokol, typy datových rámců a typy zařízení.

Standard definuje dva základní typy zařízení. První je **úplně funkční zařízení** FFD (Full Function Device), umí přijímat a přeposílat zprávy. Může zastávat funkci koncového zařízení, ale také funkci koordinátora, popřípadě může koordinátora zastoupit. Koordinátor je zařízení, které řídí celou síť, sbírá od koncových zařízení data a poté s nimi dále pracuje, nebo je přeposílá do jiného typu sítě. Každá bezdrátová senzorová síť může mít současně pouze jednoho koordinátora. Druhý typ definovaný ve standardu je **redukované funkční zařízení** RFD (Reduced Function Device). Jedná se o koncové zařízení, jehož účel je změřit hodnotu fyzikální veličiny a v případě potřeby ji zaslat koordinátorovi.

Rozlišují se čtyři typy datových rámců. **MAC Command Frame** centrálně konfiguruje a řídí zařízení v senzorové sítí. **Beacon Frame** se používá k synchronizaci a konfiguraci zařízení v čase. Nastavuje kdy má být zařízení v režimu spánku a kdy nikoliv. Beacon Frame může například probudit koncový uzel, který pouze naslouchá na síti, zda mu přichází zpráva. **Acknowledgement Frame** nese informaci o potvrzení, například potvrzuje příjem datového rámce. Lze jej využít pouze na vrstvě MAC. **Data Frame** je určený k přenosu užitečných dat v rámci sítě, každý rámec je číslovaný a lze tak zjistit, zda dorazily k cíli všechny rámce. Navíc v sobě nese kontrolní součet, čímž se ověří bezchybnost rámce.

Obr. 6: *Formát Data Frame.*



Koordinátor sítě může posílat **super-rámce**, ty se skládají z 16 slotů, přičemž každý slot může být z každého ze čtyř typů rámců. Hlavní úlohou super-rámce je poskytování synchronizace (například údaje o aktivaci úsporného režimu) a konfigurační údaje ostatním zařízením.

Zařízení v síti jsou **adresována** pomocí 64 bitového binárního adresového kódu, nebo ve zkrácené podobě na 16 bitech, přičemž zkrácená podoba umožňuje adresovat až 65535 zařízení. Pro rozlišení více sítí v rámci jedné oblastí má každá síť založená na standardu IEEE 802.15.4 ještě 16-ti bitový identifikátor sítě, tzv. Personal Area Network ID (PAN ID).

Co se týče **zabezpečení**, tak to MAC vrstva přímo nedefinuje, pouze nabízí zařízením, která fungují na vyšších vrstvách, že specifikované úrovně bezpečnosti bude dosaženo. Zároveň se očekává, že vyšší vrstvy budou specifikovat, jak bude zabezpečení probíhat. Například pomocí

symetrické kryptografie. Otázka bezpečnosti bude více rozebrána v třetí kapitole (Zigbee technologie).

[4], [5], [6], [7], [8], [9]

3 ZigBee technologie

Zigbee je bezdrátový komunikační standard spravovaný Zigbee Aliancí. Jedná se o skupinu protokolů vyšších vrstev postavených na standardu IEEE 802.15.4. Od Zigbee se předpokládá, že bude jednodušší a levnější než ostatní WPAN technologie, jako například Bluetooth. Je předurčena pro aplikace s nízkými požadavky na přenos dat, nízkou energetickou spotřebou a relativně slušnou úrovní zabezpečení. Obecně spadá do kategorie bezdrátových senzorových sítí.

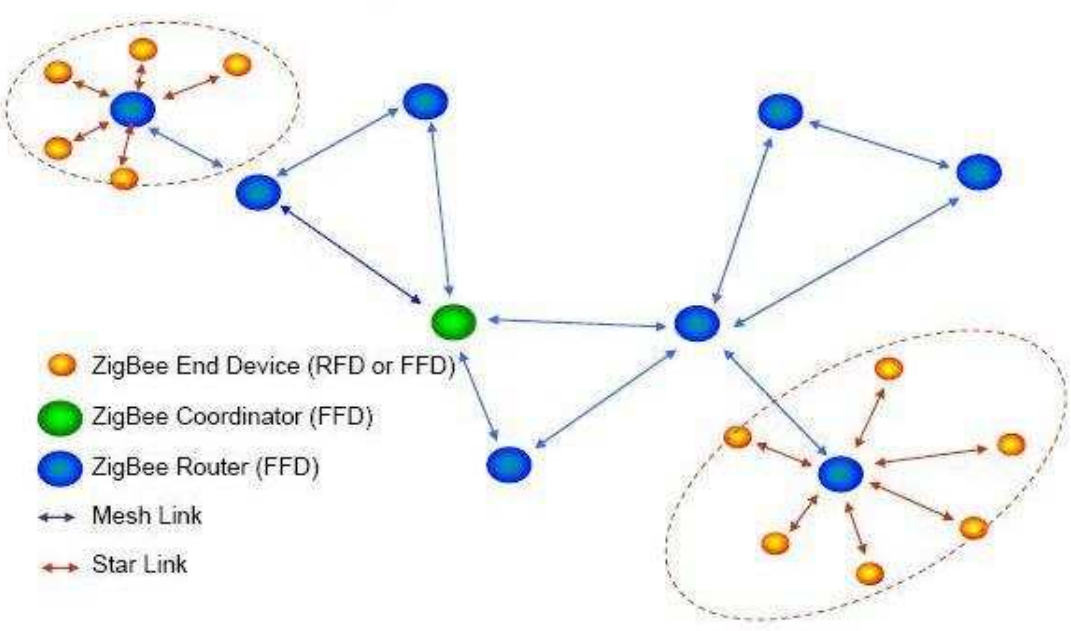
Zigbee Aliance je nadnárodní organizace sdružující průmyslové společnosti zabývající se vývojem v oblasti bezdrátových sítí. Mezi nejznámější z nich jsou například Freescale Semiconductor, Honeywell, Mitsubishi Electric, Motorola, Philips, Samsung, Invensys a další.

3.1 Typy zařízení

V ZigBee síti lze všechna zařízení z funkčního hlediska rozdělit na plně funkční zařízení (FFD) nebo redukované funkční zařízení (RFD), oba typy byly popsány v kapitole 2.2 (linková vrstva). Z logického hlediska se pak může jednat o koordinátora, směrovače nebo koncové zařízení.

Koordinátor (Coordinator) je plně funkční zařízení, spravuje a monitoruje celou síť. Sbírá od koncových zařízení data a poté s nimi dále pracuje, nebo je přeposílá do jiného typu sítě. Každá jedna síť může mít současně právě jedno zařízení, které plní funkci koordinátora. **Směrovač** (Router) je také plně funkční zařízení, jehož hlavní úlohou je přeposílání dat mezi koncovými zařízeními a koordinátorem sítě. Muže být také koncovým zařízením a nebo v případě potřeby zastávat roli koordinátora. Díky existenci směrovače se může bezdrátová síť skládat z tisíců zařízení a monitorovat oblasti o obrovských rozměrech. **Koncové zařízení** (End Device) muže být plně funkční zařízení nebo redukované funkční zařízení (častější případ), jehož úkolem je sběr a posílání dat koordinátorovi nebo, v případě rozsáhlejších sítí, směrovači. [6], [8], [10]

Obr. 7: *Příklad rozmístění zařízení v ZigBee síti topologie typu mesh.*

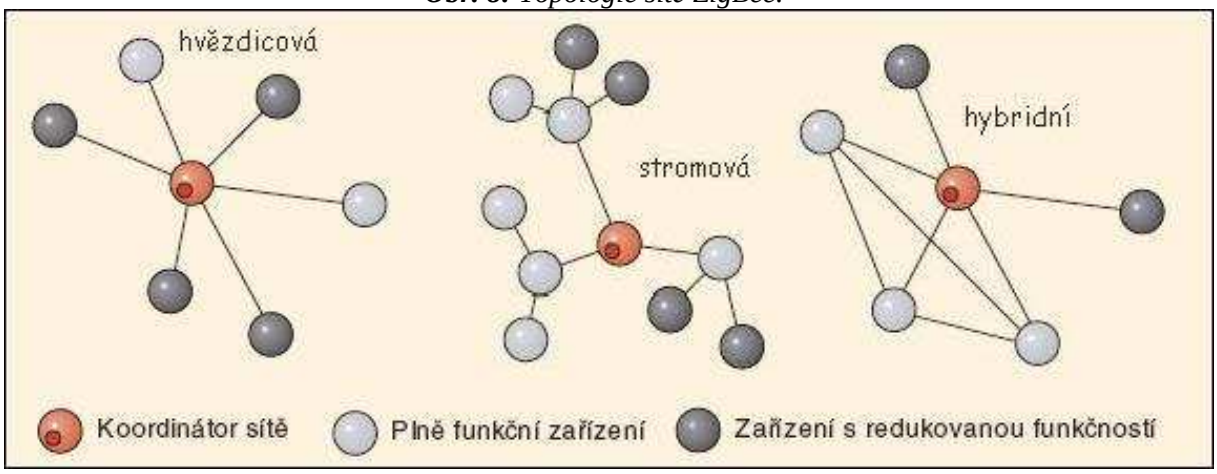


3.2 Topologie sítě

Síťová vrstva standardu ZigBee definuje hvězdicovou, stromovou a hybridní topologii sítě.

Hvězdicová topologie (star topology) je charakteristická přímou komunikací koncových zařízení s koordinátorem. Takto složená síť tvoří nejjednodušší strukturu. Příkladem může být jednoduchá síť s funkcí dálkového ovládání televize. **Stromová topologie** (cluster tree topology) narozdíl od hvězdicové je odlišná tím, že některé uzly mohou využívat při komunikaci směrovače jako prostředníka mezi koordinátorem a koncovými zařízeními. Vytvořená síť je složitější než hvězdicová a připomíná klasickou počítačovou síť. **Hybridní topologie** (mesh topology) kombinuje vlastnosti topologie stromové a hvězdicové. Tato struktura umožňuje sestavit síť libovolným způsobem. Muže zde existovat více komunikačních cest mezi dvěma zařízeními. Je nejsložitější a přináší největší funkčnost. [6], [10]

Obr. 8: *Topologie sítě ZigBee.*



3.3 Adresování

Každé zařízení v síti může mít dvě MAC adresy, IEEE Extended Unique Identifiers (EUI-64) a síťovou adresu. Pro zasílání zpráv lze využít přímé nebo nepřímé adresace.

EUI-64 je jednoznačná 64-bitová adresa identifikující zařízení v rámci celého světa. 24 bitů této adresy, tzv. Organizationally Unique Identifier (OUI), je definováno institucí IEEE (Institution of Electrical Engineers) a dalších 40 bitů se definuje ve výrobě. EUI adresa nalezne své uplatnění hlavně v rozsáhlých sítích, kde se mohou kombinovat zařízení pocházející od různých výrobců. **Síťová adresa** má délku 16 bitů, je odvozena z EUI adresy. Tuto adresu přiřazuje koordinátor v rámci připojení uzlu do sítě. Používá se v lokálních sítích.

Přímá adresace je klasické adresování jako například v počítačových sítích, zprávy jsou posílány na zařízení, které je identifikované svou jednoznačnou adresou v rámci sítě. Pro zjištění adresy se používá broadcast, tj. poslání zprávy všem uzlům s žádostí o získání adresy. Zařízení, které broadcast poslalo, dostane zpátky odpověď od všech zařízení s informací o jejich adrese. **Nepřímá adresace** je způsob, při kterém se využívá tabulky propojení (binding table). Je spravována koordinátorem a definuje, které uzly navzájem mezi sebou mohou komunikovat. Odeslaná zpráva je nejprve zaslána koordinátorovi, ten z tabulky propojení zjistí, kdo je příjemce zprávy a přepošle ji.

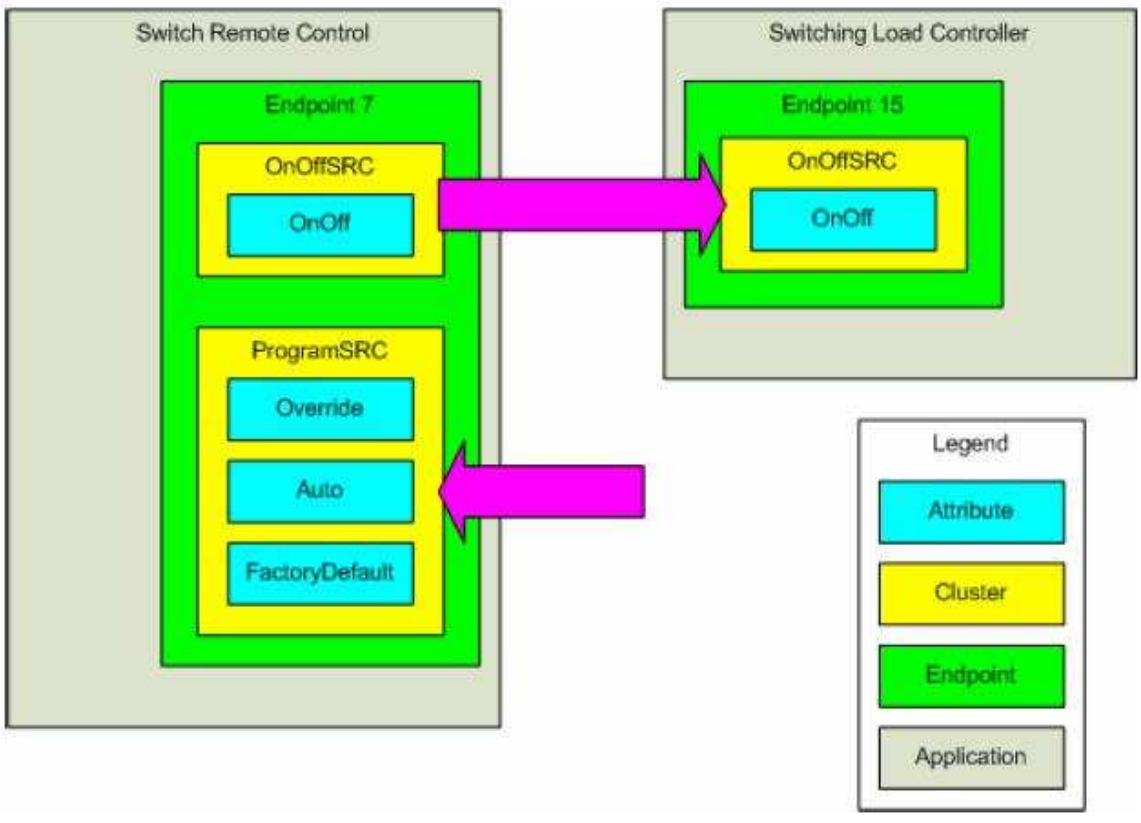
Záznam v tabulce propojení vzniká na požádání zařízení sítě (například odesláním požadavku z obou komunikujících zařízení současně) nebo popřípadě přímo příkazem na koordinátorovi. [8], [11], [12]

3.4 Profily

Standard ZigBee je navržený pro spolehlivé propojení, kontrolování a monitorování bezdrátových senzorových sítí. Aby byla zajištěna kompatibilita zařízení pocházející od různých výrobců, je nutné sestavit profily, které definují základní rozhraní a funkčnost pro každý typ aplikace. Schvalování a zveřejňování takovýchto profilů zajišťuje ZigBee Aliance. Profily zjednodušeně popisují logické komponenty, tj. zařízení a jejich rozhraní. Konkrétně definují formát a hodnoty jednotlivých atributů profilu a identifikátory clusterů. Nejsou sdruženy s žádným kódem.

Každý **atribut** představuje informaci, která může být zaslána mezi zařízeními, například přepnutí stavu periferie (zapnout/vypnout světlo) nebo zaslání informace z teplotního čidla. Jedná se o něco jako proměnnou, která má nadefinovaný vlastní datový typ (např. číslo, nebo řetězec) a identifikátor. Množina podobných atributů vytváří **cluster**. Každý cluster má jednoznačný identifikátor. Rozhranní zařízení je právě tvořeno na úrovni clusterů a nikoliv atributů jak by se mohlo na první pohled zdát. Clustery popisující podobné odvětví jsou umístěny do stejného profilu. **Endpoint** reprezentuje periferii koncového zařízení (například teplotní čidlo). Může implementovat i více clusterů. Například pro komparátor teploty bude první cluster nastavitelná teplota a druhý bude značit, zda je komparátor vypnutý nebo zapnutý. Rozdílná zařízení komunikují skrze svůj endpoint a cluster. [11], [13]

Obr. 9: *Obrázek zobrazuje spojitost atribut-cluster-endpoint.*



3.5 Formát datových rámců

ZigBee definuje dva typy datových rámců, KVP (Key Value Pair) a MSG (Message). Oba typy jsou spojeny s Clusterem. Aplikační profil specifikuje jaký typ rámce, pro jakou informaci se k přenosu použije. Vzhledem k rozdílu mezi KVP a MSG lze pro cluster definovat pouze jeden typ datového rámce pro přenos užitečných dat.

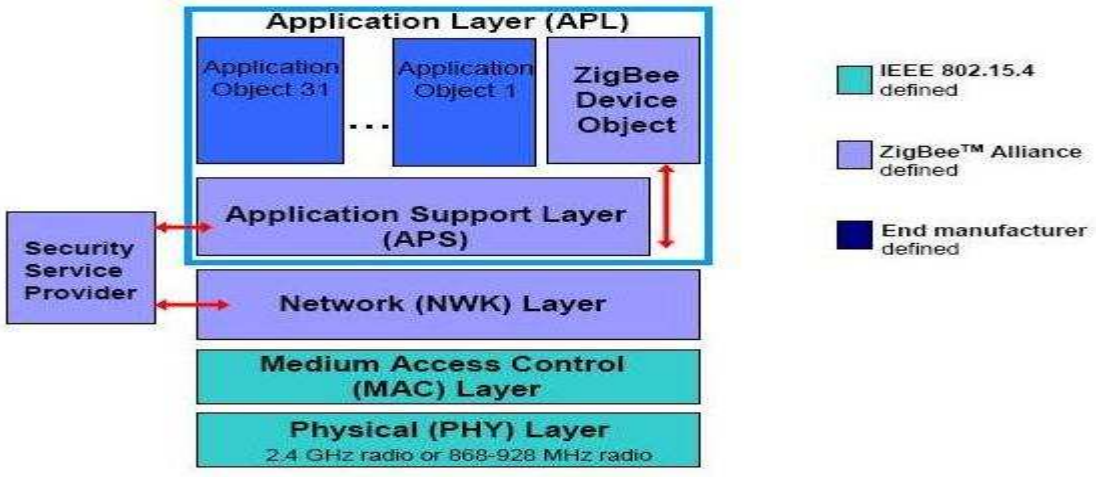
KVP je navržen k přenosu krátkých zpráv o pevné struktuře. Obsahuje sekvenční číslo transakce, typ příkazu, typ atributu, identifikátor atributu, hodnotu atributu a popřípadě chybový kód. **MSG** je volný formát zprávy o předem nedefinované struktuře. Obsahuje sekvenční číslo transakce, informaci o délce dat a daná data. [8], [13]

3.6 Komunikační model standardu Zigbee

Model pro komunikaci se skládá z vrstev podobně jako TCP/IP. Každá vrstva je charakterizována datovou jednotkou, poskytujícími službami a rozhraním pro přístup k službám vyšších vrstev, tzv. SAP (Service Access Point).

Fyzická a linková vrstva jsou definovány standardem IEEE 802.15.4 a byly popsány v předchozí kapitole (Standard IEEE 802.15.4). Vyšší vrstvy, jako je síťová a aplikační, jsou definovány standardem Zigbee. **Síťová vrstva** (NWK) se stará o připojování a odpojování ze sítě, zabezpečení rámců a směrování paketů. Zajišťuje nalezení přímého sousedního uzlu sítě. Pro koordinátora zajišťuje komunikaci a adresaci nových zařízení v síti. Poskytuje rozhraní službám aplikační vrstvě. Skládá se z dvou podvrstev NWK layer data entity (NLDE) a NWK layer management entity (NLME). NLDE umožňuje přenos dat přes SAP a NLME umožňuje služby pro řízení sítě přes SAP. NLME využívá služeb NLDE pro poskytnutí managementu a správu databáze spravovaných objektů, tzv. Network Information Base (NIB). **Aplikační vrstva** (APL) se skládá z application support sub-layer (APS), Zigbee device objekt (ZDO) a z aplikačních objektů definovaných výrobci daných zařízení. APS se stará a udržuje vazební tabulky, které obsahují informace o propojení dvou zařízení a tím je umožněna komunikace mezi těmito zařízeními dle poskytovaných služeb a požadavků. ZDO charakterizuje typ zařízení, spravuje poskytované služby, umožňuje navázání spojení, zřizuje zabezpečené spojení, umožňuje vyhledání zařízení v síti a poskytnutí informace o rozsahu poskytovaných služeb. [7], [8], [10]

Obr. 10: *Komunikační model standardu Zigbee.*



3.7 Optimalizace spotřeby elektrické energie

Protokoly technologie ZigBee jsou navrženy s důrazem na co nejmenší spotřebu energie výsledných aplikací. Očekává se, že aplikace fungující nad ZigBee budou přeposílat pouze malé objemy dat (pro větší objemy existují technologie jako Bluetooth). Navíc koncová zařízení mají především bateriové napájení s důrazem na co nejdelší výdrž. Nicméně koordinátor a směrovače by neměli být závislé pouze na bateriovém napájení, protože na nich zcela závisí funkčnost sítě.

Koncové zařízení je aktivováno (vzbuzeno z režimu spánku) synchronizačním rámcem beacon, který obdrží od koordinátora. Je to pro něj signál, aby poslalo svá data směrem koordinátorovi, ten data přijme a uloží si je do paměti. Koordinátor může také pomocí beacon rámce informovat koncové zařízení o tom, že má pro něj nějaká data. Předání těchto dat poté proběhne na žádost koncového zařízení. Pro koncové zařízení je tento způsob komunikace nejméně náročný na energetickou spotřebu a může být po většinu času přepnuto do úsporného režimu. Naopak na koordinátora jsou tímto kladeny zvýšené nároky. Navíc musí být schopen uložit data od všech ostatních uzlů sítě. [10], [13]

3.8 Bezpečnost

Bezpečností se myslí nejenom zabezpečení proti cíleně vedeným útokům, ale hlavně také vyrovnání se s kolizemi s jinou sítí a náhodnými signály. Senzorové sítě se proto musí umět s takovýmito možnými problémy vypořádat. Autentizací lze zabezpečit neautorizované vniknutí do sítě a šifrováním komunikace se dá bránit zcizení důležitých informaci. Vzhledem k nárokům a požadavkům na bezdrátové senzorové sítě musí být šifrování snadno realizovatelné a nesmí příliš zkomplikovat komunikaci. ZigBee využívá bezpečnostního modelu na úrovni linkové vrstvy, přičemž na této úrovni lze využít bezpečnostní služby jako **přístup na základě práv** (každé zařízení má seznam určující s kým může bezpečně komunikovat), **šifrování dat** (používá se symetrický 128-bitový klíč typu AES), **integrita sítě** (tj. ochrana rámce před modifikací útočníkem) a **odmítnutí opakujících se rámců**.

Implicitně není bezpečnost na základě standardu IEEE 802.15.4 povolena. Pro povolení je třeba specifikovat v rámci vyšších vrstev bezpečnostní úroveň. To se docílí nastavením stupně zabezpečení v příslušném záznamu v ACL (Access Control List). Doporučuje se, aby všechna zařízení v jedné síti používala stejnou úroveň zabezpečení. ACL je přístupový seznam, který má každé zařízení v síti. Může obsahovat 255 záznamů, přičemž jeden záznam uchovává informace právě pro jedno zařízení, s kterým může dané zařízení komunikovat. Každý záznam obsahuje adresu cíle a informaci o stupni zabezpečení. Vyšší vrstvy standardu ZigBee poskytují bezpečnostní služby typu generování klíče, přenos klíče, zabezpečení datového rámce a management zařízení.

3.8.1 Bezpečnostní režimy definované normou IEEE 802.15.4

Norma IEEE 802.15.4 definuje tři základní bezpečnostní režimy (nezabezpečený režim, režim s přístupovými seznamy a zabezpečený režim). Výchozím nastavením je **nezabezpečený režim**, neposkytuje žádné bezpečnostní služby. Při nastavení **režimu s přístupovými seznamy** se používá Access Control List (ACL), což je seznam zařízení, se kterými může určitý prvek sítě komunikovat. Nejedná se o žádný typ kryptografické ochrany. **Zabezpečený režim** se provádí pomocí algoritmu

AES a daného operačního režimu šifry (AES-CTR, AES-CCM, AES-CBC). Zajišťuje řízení přístupů, důvěrnost, integritu a aktuálnost přenosu dat.

3.8.2 Základní režimy šifry AES

AES-CTR zajišťuje důvěrnost a aktuálnost přenášených dat. Odesílaná zpráva je rozdělena na bloky o velikosti 16 bajtů. Šifrování je zajištěno pomocí operace XOR, která se provede na jednotlivé bloky nezašifrované zprávy a výstup algoritmu AES (generován na základě čítače a sdíleného klíče). Čítač je složen z adresy odesilatele, statické výplně, čítače rámce, čítače klíče a čítače bloku, přičemž plní roli inicializačního vektoru. Všechny dílčí čítače jsou průběžně aktualizovány. Dešifrování probíhá analogicky. Přijímací strana má možnost zkontrolovat hodnoty všech čítačů a tím detekovat případné útoky typu „opakovaný přenos“.

AES-CBC-MAC zajišťuje autenticitu a integritu přenosu dat. Zpráva se rozdělí opět do bloků. Při šifrování každého bloku se použije jako jeden ze vstupů blok předcházející. Pro vstup prvnímu bloku se použije inicializační vektor. Výsledkem tohoto mechanismu je kryptografický kontrolní součet, tzv. MIC (Message Integrity Code). Kontrolní součet se připojí ke zprávě přenášené v otevřené podobě. Příjemce provede autenticitu a integritu dat tím, že si vypočítá svůj MIC a porovná ho s kontrolním součtem připojeným ke zprávě.

AES-CCM zajišťuje autenticitu, integritu, důvěryhodnost dat a ochranu vůči útokům typu opakované přenosy. Kombinuje oba předchozí zmíněné režimy. Nejprve se z hlavičky a vlastních dat vypočítá hodnota MIC. Poté se provede zašifrování této hodnoty podle AES-CTR.

3.8.3 Bezpečnost na úrovni jednotlivých vrstev

Na úrovni fyzické vrstvy se využívá technika DSSS, regulace vysílacího výkonu a zamezení šíření radiového signálu. Nevýhodou je, že síť operuje v nelicencovaném pásmu, takže se proti potenciálním narušitelům nedá účinně bojovat. Útoky se zaměřují na vyčerpání omezených zdrojů, což má za následek nefunkčnost senzorové sítě. Mezi další útoky patří přímo kontakt se senzory a narušení fyzické integrity sítě. Cílem je ohrožení dostupnosti sítě, získání citlivých dat (např. kryptografických klíčů), modifikace funkce zařízení nebo zjištění polohy ostatních zařízení.

Na linkové vrstvě je použita univerzální symetrická šifra AES, která může mít několik stupňů zabezpečení v závislosti na režimu šifry AES. Šifra se vztahuje pouze na datové (Data Frame), řídící (MAC Command Frame) a synchronizační rámce (Beacon Frame), ale bohužel se už nedá použít na rámce potvrzující správné přijetí dat (Acknowledgement Frame). Další poskytované bezpečnostní služby jsou autentizace a aktuálnost dat. Management kryptografických klíčů norma IEEE 802.15.4 nespecifikuje, ponechává tuto roli vyšším vrstvám.

Síťová vrstva zajišťuje zabezpečení rámců, správu dynamických změn v topologii WPAN (například vložení, nebo odebrání uzlu) a zabezpečení směrování. Mezi nejnebezpečnější útoky na senzorové sítě patří právě napadení směrovacích protokolů.

Na úrovni aplikační vrstvy používá ZigBee algoritmus AES se 128-bitovým klíčem. Na rozdíl od IEEE 802.15.4 zahrnuje definici a management kryptografických klíčů. Architektura bezpečnostních prvků protokolu Zigbee se skládá z Trust Center, Trust Manager, Network Manager a Configuration Manager. **Trust Center** má význam důvěryhodného centra (většinou plní tuto roli koordinátor sítě), řídí přístup zařízení do sítě a provádí distribuci klíčů, může pracovat ve dvou režimech Residential mode, nebo Commercial mode. Rezidenční režim umožní zařízením přístup do sítě, ale neprovádí již ustavení a periodickou obnovu klíčů, používá se pro minimalizaci paměťových

nároků zvláště při možném rozšiřování sítě. Komerční režim zahrnuje centralizovanou distribuci a aktualizaci kryptografických klíčů, což zajišťuje mnohem větší bezpečnost. **Trust Manager** slouží pro autentizaci zařízení. **Network Manager** spravuje a distribuuje síťové klíče a **Configuration Manager** má hlavní úkol zabezpečit koncové body komunikace. Technologie ZigBee definuje tři typy klíčů, hlavní klíč (Master Key – určen pro dlouhodobé zabezpečení komunikace mezi dvěmi zařízeními, může být nastaven na zařízení již od výroby, nebo poslán skrze Trust Center), linkový klíč (Link Key – je odvozen od hlavního klíče, má kratší dobu platnosti a představuje aktuální zabezpečení mezi komunikujícími prvky sítě), síťový klíč (Network Key – určen pro globální zabezpečení sítě). Periodicky lze měnit klíče typu linkový a síťový.

3.8.4 Nedostatky v zabezpečení

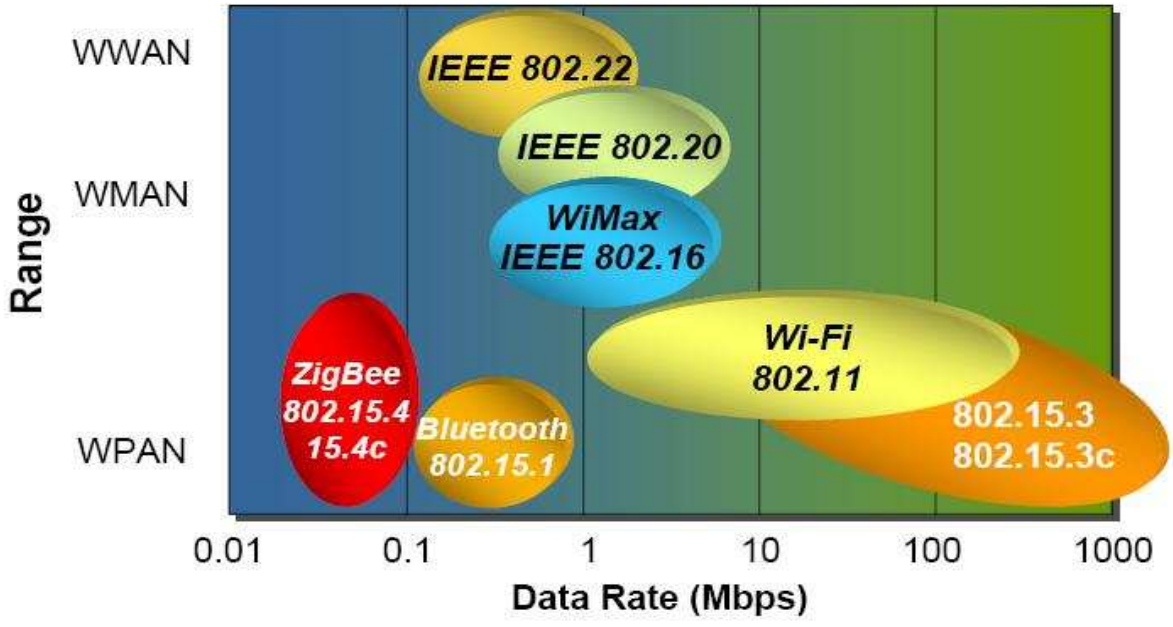
Výpadek napájení způsobí vymazání seznamu přístupových práv a nastavení čítačů. Chybí podpora pro skupinové klíče. Při sdílení jednoho klíče všemi stanicemi nelze zamezit útoku opakovaného přenosu. Slabá podpora sdílení klíčů mezi dvojicí uzlů. Šifrování přenášených dat se nevztahuje na potvrzovací rámce. Při použití režimu AES_CTR není žádným způsobem zajištěna integrita a autenticita dat.

[7], [14], [15]

3.9 Srovnání s ostatními bezdrátovými standardy

ZigBee představuje jednodušší a levnější podobu technologie Bluetooth, přičemž nejde o konkurenci, ale spíše o doplnění chybějícího prostoru. ZigBee je hlavně určeno pro aplikace, kde Wi-Fi ani Bluetooth neposkytují ideální řešení. To jest pro aplikace, kde není třeba přenášet velké množství dat, pouze nějaké nenáročné informace mezi zařízeními na relativně krátkou vzdálenost, ale zároveň může takto vytvořená síť obsahovat tisíce uzlů a pokrývat obrovskou plochu.

Obr. 11: *Srovnání s ostatními bezdrátovými standardy.*



Z tabulky 1 uvedené níže vyplývá, že technologie ZigBee v porovnání s ostatními standardy výrazně vyčnívá v požadavcích na systémové zdroje (pouhých 32 KB), délkou životnosti baterií (až několik roků) a počtem možných zařízení v jedné síti (téměř neomezeno). [16], [17]

Tab. 1: *Porovnání parametrů jednotlivých vlastností různých typu bezdrátových standardů.*

Market Name	ZigBee®	---	Wi-Fi™	Bluetooth™
Standard	802.15.4	GSM/GPRS CDMA/1xRTT	802.11b	802.15.1
Application Focus	Monitoring & Control	Wide Area Voice & Data	Web, Email, Video	Cable Replacement
System Resources	4KB - 32KB	16MB+	1MB+	250KB+
Battery Life (days)	100 - 1,000+	1-7	.5 - 5	1 - 7
Network Size	Unlimited (2^{64})	1	32	7
Bandwidth (KB/s)	20 - 250	64 - 128+	11,000+	720
Transmission Range (meters)	1 - 100+	1,000+	1 - 100	1 - 10+
Success Metrics	Reliability, Power, Cost	Reach, Quality	Speed, Flexibility	Cost, Convenience

4 Popis práce v laboratoři

Zde budou uvedeny základní postupy pro uvedení dosud popsaných informací do praxe. Čtenář by měl být schopen při dostupnosti příslušného vybavení sestavit bezdrátovou senzorovou síť definovanou standardem ZigBee. Bude zde popsáno hardwarové a softwarové vybavení laboratoře a postupy, jak s jednotlivými součástmi pracovat.

4.1 Hardwarové vybavení

Základní prvky senzorové sítě pro naše účely jsou tvořeny vývojovým kitem PICDEM Z 2.4 GHz, programátorem MPLAB ICD 2 a síťovým analyzátorem ZENA. Všechny prvky jsou k dostání od firmy Microchip.

4.1.1 Vývojový kit

Vývojový kit PICDEM Z 2.4 GHz je tvořen dvěma PICDEM Z deskami, kde každá deska obsahuje vysílač a příjímač CC2420, 8 bitový Mikrokontrolér typu PIC18LF4620, senzor teploty, 2 LED diody, 2 uživatelská tlačítka, reset tlačítko, konektor pro připojení antény k desce, konektor RJ-11 pro připojení desky k programátoru, konektor RS-232 pro připojení desky k počítači, obvody napájení (lze napájet 9V baterií nebo externím zdrojem elektrické energie) a plochu na desce, kde je možné vytvořit si vlastní obvody.

Obr. 12: *Vývojová deska PICDEM Z.*

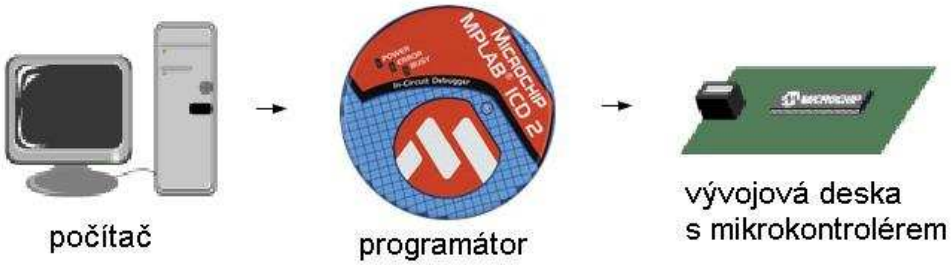
Anténa CC2420 pracuje na frekvenci 2,4 GHz a odpovídá standardu ZigBee. Obsahuje RF přijímač/vysílač (implementuje fyzickou a část linkové vrstvy), konektor pro připojení k desce a v případě potřeby konektor pro připojení externí antény.

Pro funkční aplikaci senzorové sítě je nutné mít alespoň dvě desky PICDEM Z, kde jedna bude naprogramovaná jako koordinátor a druhá nebo popřípadě další budou plnit roli koncových zařízení a směrovačů. [18]

4.1.2 Programátor

Programátor MPLAB ICD 2 je pro vývoj aplikací nepostradatelný. Slouží pro naprogramování mikrokontroléru, to znamená uložení přeloženého kódu z počítače do paměti mikrokontroléru. K počítači je programátor připojen přes rozhraní USB, nebo RS-232. K vývojové desce se poté připojuje přes konektor RJ-11. MPLAB ICD 2 lze využít jako debugger pro analýzu spuštěné aplikace v reálném čase. Umožňuje sledovat hodnoty vybraných proměnných a nastavovat breakpointy v zdrojových kódech psaných v jazyku C nebo asembleru. [19]

Obr. 13: Příklad zapojení zařízení při procesu programování mikrokontroléru.



4.1.3 Síťový analyzátor

Bezdrátový síťový analyzátor ZENA slouží pro zobrazení komunikace v síti, která je postavená na standardu IEEE 802.15.4 při frekvenci 2,4GHz. Podporuje protokol ZigBee a MiWi. Síťový analyzátor je zvláště nápomocný při tvorbě složitějších aplikací v rozsáhlejších sítích. Umí graficky zobrazit nejenom komunikaci, ale také například aktuální topologii dané sítě. Všechny tyto informace lze vyexportovat a použít pro další analýzu. K počítači je připojen přes rozhraní USB. [20]

Obr. 14: *ZENA analyzátor*.

4.2 Softwarové vybavení

Součástí zakoupeného hardwaru je většinou i softwarové vybavení, nicméně vždy je lepší stáhnout si z webu aktuální verzi dostupného softwaru. Jediný potřebný nástroj, který nelze v plné verzi přímo z webu stáhnout je překladač jazyka C (MPLAB C Compiler for PIC18 MCUs), ale existuje možnost stáhnout demo pro studentské účely, kterému se sníží funkčnost po vypršení 60 dnů. Ve škole je k dispozici zakoupená plná verze tohoto překladače.

Pro vytvoření aplikace fungující nad senzorovou sítí potřebujeme **vývojové prostředí** (MPLAB IDE), **stack** zajišťující softwarovou reprezentaci fyzické, linkové a síťové vrstvy komunikačního modelu ZigBee, **překladač jazyka C**, software potřebný pro **síťový analyzátor**, ovladače příslušející pro programátor MPLAB ICD 2 a nějaký **terminál** umožňující komunikaci počítače s vývojovou deskou (v prostředí Windows lze použít například HyperTerminal).

4.2.1 Postup instalace potřebného softwaru

Ze všeho nejdříve je vhodné nainstalovat vývojové prostředí MPLAB IDE. Aktuální verzi lze stáhnout z webových stránek firmy Microchip (www.microchip.com). Při výběru instalace "Custom" je třeba minimálně nainstalovat podporu pro 8-bitové mikrokontroléry (8 bit MCUs), podporu pro programátor (MPLAB ICD 2) a doporučuji nainstalovat také podporu pro softwarovou simulaci (MPLAB SIM).

Po úspěšném nainstalování vývojového prostředí je dobré připojit k počítači programátor, instalace ovladačů by měla proběhnout bez problémů. Dalším krokem je instalace překladače jazyka C, konkrétně MPLAB C Compiler for PIC18 MCUs. Při výběru seznamu komponent překladače je nejlepší nechat implicitní nastavení, tj. necháme vybrány všechny položky kromě „Preprocessor source code“. Poté ještě zbývá v následujícím dialogu zvolit konfigurační nastavení. Důležité je zaškrtnout možnost „Update MPLAB IDE to use this MPLAB C18“ a „Update MPLAB IDE to use

this MPLINK Linker“, čímž dojde k nakonfigurování vývojového prostředí MPLAB IDE tak, aby používalo překladač MPLAB C18.

Nyní je třeba nainstalovat ještě ZigBee stack, který je volně stažitelný na stránkách Microchipu (www.microchip.com), doporučuji stáhnout pokud možno verzi, která podporuje přijímač/vysílač typu CC2420, více viz kapitola *4.2.3 ZigBee stack*. Instalace stacku nevyžaduje žádné nastavování a po dokončení jsou soubory k nalezení v adresáři C:\MpZBee\. [21]

4.2.2 Vývojové prostředí MPLAB IDE

MPLAB IDE (Integrated Development Environment) je vývojové prostředí pro tvorbu aplikací podporující mikrokontroléry PIC (v laboratoři máme k dispozici 8-bitový mikrokontrolér PIC18LF4620). Pokud jsme úspěšně dodrželi postup instalace softwaru, máme nainstalované vývojové prostředí s překladačem jazyka C, programátorem a debuggerem. Pro vývoj aplikací je potřebné nastavit parametry programátoru MPLAB ICD 2.

Spustíme MPLAB IDE, klikneme na menu Programmer->Select Programmer->MPLAB ICD 2. Pokud to uděláme poprvé, automaticky se nám spustí průvodce nastavením programátoru. Vybereme typ konektoru pro komunikaci s PC (doporučuji USB), v dalším okně zvolíme položku „Target has own power supply“ (je nutné, aby měla každá vývojová deska vlastní napájení). V dalším dialogu necháme volbu „MPLAB IDE automatically connects to the MPLAB ICD2“ nezaškrtnutou a nakonec v dalším boxu zaškrtneme volbu „MPLAB ICD2 automatically downloads the required operating system".

Ovládání vývojového prostředí je intuitivní a standardní. Pro tvorbu vlastního projektu klikneme na menu Project->Project Wizard, vybereme typ zařízeni PIC18F4620, v dalším boxu zvolíme v políčku Active Toolsuite „Microchip C18 Toolsuite“ a můžeme zkontrolovat, zda jsou dobře zadány cesty k překladači, knihovnám a linkeru. Poté zadáme název a cestu k novému projektu a následně přidáme případné existující zdrojové soubory. Důležité je přidat skript pro linker, jehož cílem je informovat linker o paměťové organizaci vybraného zařízení. Skript se v našem případě jmenuje „18f4620i.lkr“ a nalezneme jej v adresáři \MCC18\lkr\18f4620i.lkr. V případě spuštění již existujícího projektu klikneme na menu Project->Open a vybereme soubor s koncovkou „mcp“, všechno ostatní už bude nastaveno v rámci projektu.

Nyní můžeme projekt přeložit kliknutím na menu Project->Build All, nebo Project->Make. Po zapojení vývojové desky k programátoru se připojíme pomocí menu Programmer->Connect a přeloženým kódem (který je uložený v souboru s koncovkou „hex“) naprogramujeme mikrokontrolér kliknutím na menu Programmer->Program. Po naprogramování je mikrokontrolér softwarově držen v resetu, tento stav zrušíme kliknutím na menu Programmer->Release from Reset a aplikace by se tím měla na vývojové desce spustit.

Při použití debuggeru a ladění aplikace klikneme na menu Debugger->Select Tool->MPLAB ICD 2. Následující procesy (připojení, naprogramování) jsou stejné jako v případě programátora, pouze se všechno nachází pod menu Debugger. Možnosti pro ladění jsou Run, Animate, Halt, Step Into, Step Over a Step Out. [21], [22]

4.2.3 ZigBee stack

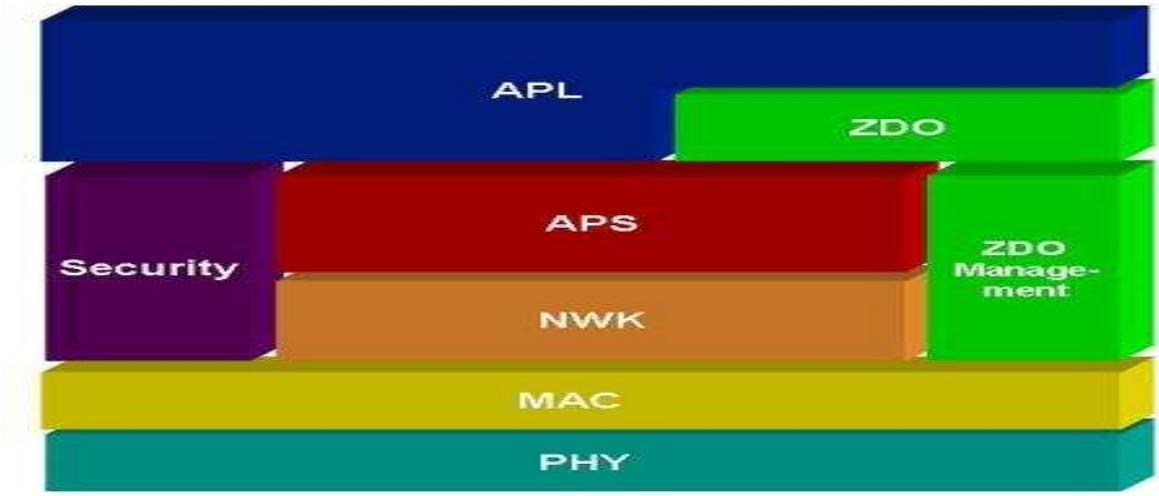
ZigBee stack si lze představit jako softwarovou implementaci vrstvené architektury protokolu ZigBee. Volně stažitelná verze pro nekomerční účely je zdarma na webu od Microchipu. Pro vývoj aplikací senzorové sítě je stack nepostradatelnou součástí. Zajišťuje funkci nižších vrstev a poskytuje

rozhraní pro tvorbu aplikační vrstvy. Od nás se požaduje nastudování základních mechanizmů, jakým způsobem stack pracuje. K tomu mohou dobře posloužit demonstrační aplikace, které jsou jeho součástí. Popřípadě jsou k dispozici šablony kódu pro tvorbu uzlů sítě (existuje zvlášť šablona pro koordinátora sítě a koncový uzel nebo směrovač). Navíc součástí této bakalářské práce jsou k dispozici ukázkové aplikace, z nichž dvě využívají ZigBee stack od Microchipu.

Aktuální verze ZigBee stacku od Microchipu je 3.8. Bohužel tato verze plně podporuje pouze vysílač/přijímač typu MRF24J40, přičemž ve školní laboratoři je k dispozici jako součást vývojového kitu starší typ CC2420. Tento vzniklý problém lze vyřešit použitím verze 3.5, která typ CC2420 podporuje.

Stack je napsán v jazyku C a je tvořen automatem, jehož funkci lze částečně ovládat v aplikační vrstvě. Rozhraní aplikační vrstvy s ostatními vrstvami je tvořeno proměnnou typu „ZIGBEE_PRIMITIVE“, ta určuje vždy aktuální stav, v jakém se automat nachází. Uživatel stacku může při definovaném stavu této proměnné nějakým způsobem reagovat a nastavit stav jiný. Tímto lze řídit funkčnost automatu a tím pádem celé aplikace. Podrobné informace, jak se ZigBee stackem pracovat, lze nalézt v dokumentaci po nainstalování v adresáři ..\MpZBee\Documents, nebo také na diskusním fóru provozovaném firmou Microchip (http://forum.microchip.com/). [13], [23], [24]

Obr. 15: *Vrstvy implementované ZigBee stackem.*



4.2.4 Zena analyzátor

Software pro Zena Analyzátor lze s výhodou použít nejenom pro ladění komunikace, ale také pro vygenerování aplikačně specifických konfiguračních souborů a skriptu pro linker k jednotlivým uzlům sítě ZigBee. Demo tohoto softwaru je součástí ZigBee stacku od firmy Microchip. Jeho omezení oproti plné verzi dodávané s hardwarem spočívá v nemožnosti sledovat síťovou komunikaci v reálném čase. Je tedy nutné komunikaci uložit a analyzovat až zpětně.

Pomocí tohoto softwaru lze nastavit pro každý uzel parametry jako například typ zařízení, MAC adresu, povolené komunikační kanály, profil, clustery a další. Následně jsou vygenerovány tři konfigurační soubory (zigbee.def, myZigBee.c, zLink.lkr). Soubory *ZigBee.def* a myZigBee.c obsahují konfigurační údaje důležité pro správnou funkci stacku. Soubor zLink.lkr je skript určený pro linker, obsahující informace o paměťové organizaci konkrétního typu zařízení. Je vysoce

doporučené upřednostnit používání softwaru ZENA namísto manuální úpravy těchto tří souborů. [13], [25]

4.2.5 Komunikace vývojové desky s počítačem

Ke komunikaci počítače s vývojovou deskou PICDEM Z slouží rozhraní RS-232. Tímto způsobem lze v závislosti na typu aplikace monitorovat a spravovat senzorovou síť. Nejvhodnější je mít takto připojený uzel plnící roli koordinátora, který podle potřeby může vypisovat informace o komunikaci, aktuálně připojených zařízeních nebo ladící výpisy. V prostředí Windows lze bez problému použít program HyperTerminal. Pokud počítač není vybaven sériovým portem, může se použít USB redukce (bohužel ne všechny typy redukcí fungují správně a je třeba si na to dát pozor).

V nastavení HyperTerminálu vybereme možnost připojit pomocí portu COM. Poté v nastavení daného portu nakonfigurujeme položky Bity za sekundu na hodnotu „19200“, počet datových bitů na „8“, paritu „žádnou“, počet stop bitů na „1“ a řízení toků na hodnotu „žádné“. Po připojení by se v okně HyperTerminálu měly zobrazovat zprávy pocházející od koordinátora. [13]

5 Ukázkové aplikace

Cílem bakalářské práce je kromě vysvětlení základních teoretických principu ukázat, jak může čtenář vyvíjet svou vlastní aplikaci pro senzorovou síť. Tato kapitola se věnuje přímo tvorbě aplikací. Předpokládá se, že čtenář si nejprve projde a v základech porozumí dříve zmíněným informacím. Bude mít k dispozici popsaný hardware (viz kapitola 4.1) a nainstalovaný a připravený software (viz kapitola 4.2). Postupně zde budou zdokumentovány tři ukázkové aplikace. První nejjednodušší je zaměřena na seznámení se s vývojovou deskou a mikrokontrolérem. Druhou a třetí aplikaci lze již považovat za laboratorní simulaci bezdrátové senzorové sítě, kde ke svému chodu využívají ZigBee stack. Aplikace lze otestovat pomocí vývojového prostředí MPLAB IDE. Stačí otevřít příslušný projekt a následovat instrukce popsané v kapitole 4.2.2. Po porozumění těmto základním příkladům by měl čtenář být schopen vytvořit své vlastní aplikace.

5.1 První aplikace – mikrokontrolér

Smyslem této aplikace je ukázat, jakým způsobem lze pracovat s hardwarem typu mikrokontrolér. U každého typu mikrokontroléru je práce s ním mírně odlišná, ale základní principy jsou shodné. Vždy je třeba seznámit se s architekturou konkrétního typu mikrokontroléru a mít k dispozici literaturu, kde se dají zjistit podrobné informace. Stěžejní pro vývoj těchto aplikací je znát základy. Například pro práci s LED diodami mikrokontroléru typu PIC18F4620 je nutné vědět, že jsou ovládány nastavením vstup/výstupního portu „A“. Řízení portu „A“ dále podléhá nastavením třech registrů (TRISA, PORTA a LATA).

Registr TRISA ovlivňuje datový směr portu „A“, což znamená, zda budou piny portu vstupní nebo výstupní. Registr PORTA je datový registr, obsahuje v sobě aktuální informaci o tom, jaká hodnota (úroveň napětí) se vyskytuje na příslušném pinu portu „A“. Registr LATA uchovává hodnotu registru PORTA při posledním čtení. Toho lze využít generováním přerušení při změně hodnoty, tzv. „interrupt-on-change“. Příznak přerušení se vygeneruje v případě, že se registr PORTB nerovná registru LATB.

Tato ukázková aplikace provádí v cyklu postupné zapínání a vypínání dvou LED diod. Přepínání je řízeno hodnotou registru PORTA. Nultý bit ovládá první LED diodu (RA0) a první bit řídí druhou LED diodu (RA1), kde hodnota 0 určeného bitu znamená, že příslušná dioda nesvítí a hodnota 1 znamená, že dioda svítí. Součástí zdrojového kódu je podrobný komentář téměř ke každému řádku. [21], [26]

5.2 Druhá aplikace – světelné diody

Druhá aplikace navazuje na předchozí příklad a rozšiřuje jeho funkčnost o použití ZigBee stacku. Celkem se skládá z dvou částí, jedna představuje implementaci role koordinátora (jedná se o samostatný projekt v rámci MPLAB IDE) a druhá část se týká koncového zařízení. To znamená, že je třeba naprogramovat jednu vývojovou desku kódem pro koordinátora a druhou kódem pro koncové zařízení. Aplikace demonstruje komunikaci mezi koordinátorem a koncovým zařízením, inicializaci hardwaru, obsluhu přerušení, komunikaci směrem na terminál při připojení k počítači přes rozhraní

RS-232. V rámci kódu pro koordinátora je zde vidět vytvoření sítě. Naopak koncové zařízení ukazuje vyhledání dostupných sítí, následné přihlášení se do ní a využití režimu spánku.

Aplikace vychází z demonstračních příkladů, které jsou součástí ZigBee stacku od Michrochipu. Tyto příklady jsou pro začátek zbytečně komplikované, což vedlo k jejich upravení. Pro posílání zpráv je použito přímé adresace, zprávy typu „přepni LED diodu“ nevyžadují odpověď. Pro názornost mají oba uzly (koordinátor i koncové zařízení) schopnost přepnout diodu druhého uzlu (funkce dálkového ovladače) a zároveň mohou na základě příchozí zprávy přepnout svou diodu (typické pro koncové zařízení). Režim spánku u koncového zařízení je upraven na delší dobu a je nastavena signalizace oznamující, že je zařízení v režimu spánku (dioda RA0 nesvítí).

Po zapnutí vývojové desky se nejprve musí inicializovat hardware, jedná se o inicializaci SPI jednotky a nastavení potřebných řídících registrů mikrokontroléru PIC18F4620. Poté dojde k inicializaci ZigBee stacku a následně bude v cyklu prováděn kód, který bude skrze proměnnou typu „ZIGBEE_PRIMITIVE“ řídit chod celé aplikace.

Nejprve je nutné zapnout vývojovou desku, kde je nahrán kód pro koordinátora, protože ten musí nejprve vytvořit síť. Aplikační vrstva koordinátora pošle žádost o vytvoření senzorové sítě síťové vrstvě. Ta, pokud jde vše bez problému, pošle zpět aplikační vrstvě potvrzení společně s identifikátorem sítě (PAN ID). Koordinátor se stane automaticky uzlem vytvořené sítě (vždy s MAC adresou 0x0000). Poté povolí připojení ostatním uzlům. V tuto chvíli je třeba zapnout vývojovou desku reprezentující koncové zařízení, to se pokusí po proběhnutí inicializačních procedur vyhledat dostupnou síť a následně se k ní připojit. Na straně koordinátora, v případě úspěšného připojení koncového zařízení, dojde k doručení zprávy s informací, že koncové zařízení s příslušnou MAC adresou bylo připojeno do sítě.

Nyní mohou začít uzly sítě mezi sebou komunikovat prostřednictvím zasílání zpráv. Naimplementovány jsou dvě zprávy. První možná zpráva je zaslána při stlačení tlačítka RB5 a vyšle se tím zpráva typu „broadcast“. Tuto zprávu automaticky přijímají všechny okolní uzly a smyslem zprávy je, aby uzel zjistil MAC adresy jeho sousedních uzlů. Příjemce umí odpovědět na zprávu typu „broadcast“ automaticky bez podpory aplikační vrstvy, posláním odesilateli svou MAC adresu. Jakmile dané uzly znají adresy uzlů svých sousedů, je možné použít přímou adresaci k zaslání druhé zprávy („přepni LED diodu“). Ta je poslána při zmáčknutí tlačítka RB4 a pro příjemce znamená oznámení o tom, aby přepnul svou LED diodu (RA1) do opačného stavu. Princip zasílání zpráv je takový, že při stlačení tlačítka se vyvolá přerušení a obslužná rutina přerušení podle toho, které tlačítko bylo stlačeno, nastaví žádost o zaslání příslušné zprávy („broadcast“, nebo „přepni LED diodu“).

Obzvláště názorná je demonstrace režimu spánku. Koncové zařízení, pokud nemá nic na práci, upadne vždy na nějakou dobu do úsporného režimu. Tento stav je indikován zhasnutou diodou (RA0) na vývojové desce. Po určité době je zařízení probuzeno, v první fázi po probuzení pošle dotaz koordinátorovi, zda pro něj má nějaké zprávy. Koordinátor má k dispozici paměť, kde jsou ukládány zprávy určené pro koncové zařízení. Pokud nějaká taková zpráva existuje, bude koncovému zařízení přímo poslána. Pokud žádné zprávy k dispozici koordinátor nemá, koncové zařízení se to příslušným způsobem dozví a pokud nemá nic jiného na prácí, přejde zpět do úsporného režimu. Na vývojové desce se tento proces projeví probliknutím diody RA0. Probuzení se provede automaticky vždy po vypršení stanoveného časového limitu nebo v případě stisknutí nějakého tlačítka a tím pádem následného vygenerování přerušení.

Je dobré si uvědomit a v praxi otestovat, že stlačením tlačítka RB4 na koordinátorovi se přepnutí LED diody (RA1) na koncovém zařízení v důsledku režimu spánku projeví až po nějakém čase. A naopak stisknutí tlačítka RB4 na koncovém zařízení okamžitě vyvolá probuzení koncového

zařízení a následně přepnutí LED diody (RA1) na koordinátorovi. Po připojení koordinátora nebo popřípadě koncové zařízení přes rozhraní RS-232 k počítači, lze v okně terminálu spatřit ladící výpisy, informující o procesech, které na patřičném uzlu probíhají. [13]

5.3 Třetí aplikace – teplotní senzor

Poslední aplikace stojí na stejném základě jako předchozí druhá aplikace. Je zde ukázáno, jakým způsobem lze pracovat s teplotním čidlem, které je součástí vývojové desky. Základní mechanismy, jako je princip posílání zprav nebo režim spánku, byl popsán v předchozím příkladě, a tudíž jsou zde uvedeny pouze funkční změny a odlišné ovládání. Je zde stejným způsobem implementováno zasílání zprávy typu „broadcast“, ale funkce přepínání diod při stlačení tlačítka RB4 již zde chybí.

Aplikace funguje na jednoduchém principu, nejprve je třeba zapnout koordinátora, ten opět vytvoří síť, poté se spustí koncové zařízení a připojí se do sítě (tyto procesy jsou stejné jako v druhé aplikaci). Koncové zařízení umístíme do prostoru, z kterého budeme chtít měřit teplotu okolního prostředí. Poté stlačíme tlačítko RB4 na koordinátorovi, čímž bude zaslána zpráva koncovému zařízení. To bude tuto příchozí zprávu interpretovat jako požadavek o změření teploty na svém teplotním senzoru a zasláním informace o změřené teplotě koordinátorovi. Tlačítko RB4 umístěné na koncovém zařízení nemá implementovanou žádnou funkčnost.

Nutnou součást pro chod aplikace je propojení koordinátora s počítačem přes rozhraní RS-232, aby koordinátor mohl příchozí zprávu od koncového zařízení (formátovanou informace o teplotě) přeposlat na terminál.

Relativně dlouhá odezva mezi stlačením tlačítka RB4 na koordinátorovi a výpisu informace o teplotě na terminálu může být způsobena úsporným režimem koncového zařízení. Pokud ovšem informace nepřichází ani po čase (řádově vteřiny), je možný problém dosah sítě, který v závislosti na terénu (stěny nebo různé překážky) bývá něco kolem desíti metrů. [13]

5.4 Možné komplikace

V průběhu návrhu, implementace a testování aplikací pro senzorové sítě mohou nastat jisté komplikace. Pokud jsou tyto možné problémy známy dopředu, jejich řešení je o mnoho jednodušší. Z tohoto důvodu budou některé z nich zde popsány.

Může se stát, že **koordinátor neumí sestavit síť** a to i v případě aplikací dodávaných jako součást ZigBee stacku. Jakoby se aplikace někde zacyklila a opakovaně prováděla kód, který má na starosti inicializaci sítě. Pokud nastanou tyto příznaky, je velice pravděpodobné, že je senzorová síť rušena nějakou WiFi sítí, která má v místě práce velký signál. Řešením je nějakým způsobem zmírnit rušení WiFi sítě.

Občas se stane, že se **uzel sítě přestane chovat stabilně** a na terminál začne posílat nesmyslné zprávy. Pokud se tento stav v čase nelepší, je dost možný problém v napájení. Baterii dochází energie a je třeba ji vyměnit.

Další komplikace je možná s **typem vysílače/přijímače (CC2420).** Nejnovější verze ZigBee stacku (v3.8) totiž plně nepodporuje ve škole dostupný typ vysílače/přijímače. Pro rozchození aplikace je třeba vygenerovat nějaké specifické konfigurační soubory a navíc nahradit zdrojové soubory reprezentující fyzickou a MAC vrstvu ZigBee protokolu, nebo použít k práci starší verzi

stacku (v3.5). Nevýhodou použití starší verze je absence podpory bezpečnosti, která je naimplementovaná až ve verzi 3.8.

Závěr

Cílem této bakalářské práce bylo vytvořit ucelené a souhrnné informace, poskytující základní porozumění v oblasti senzorových sítí. Daná problematika byla důkladně prostudována a byla snaha nalézt právě to nejdůležitější a nejpodstatnější, co člověk mírně znalý oboru informačních technologií k základnímu pochopení potřebuje. Proto byla práce vedena strukturovaně od teoretických základů senzorových sítí v kapitole první, přes konkrétní určení a popis standardu IEEE 802.15.4 v kapitole druhé, standard ZigBee v kapitole třetí a až po praktické návody a ukázky v kapitole čtvrté a páté. Dále pak byly na základě všech těchto informací vytvořeny webové stránky, jejichž formát je pro dokument podporující pochopení a tvorbu aplikací pro senzorové sítě daleko vhodnější, než pouze text bakalářská práce.

Cíl bakalářské práce se, dle mého názoru, podařilo splnit a rovněž pevně věřím, že případným zájemcům o tématiku senzorových sítí v počátečním seznamování pomůže. Pro mne byla tato práce velkým přínosem, protože mi dala slušný rozhled v této oblasti a pochopení širších souvislostí.

Případné rozšíření v návaznosti na tuto práci bych si představoval tvorbou pokročilejších a komplikovanějších ukázkových aplikací. Existuje zde velké pole působnosti, dala by se zohlednit a naimplementovat složitější topologie sítě (například stromová), což by mělo za následek tvorby kódu nejenom pro koordinátora a koncové zařízení, ale také pro směrovač. Dále by mohlo být dalším možným vylepšením použití novější verze ZigBee stacku od Microchipu a zakomponování podpory zabezpečení.

Literatura

[1] Murthy Ram Siva C., Manoj S. B., Ad Hoc Wireless Networks Architectures and Protocols, Prentice Hall, 2004, ISBN 978-0-13-147023-1.

[2] Wikipedia, the free encyclopedia, Wireless sensor network [online], Poslední modifikace: 2008-04-11, [cit. 2008-04-13], Dostupné na URL: <http://en.wikipedia.org/wiki/Wireless_sensor_network>.

[3] Sikora Axel, Wireless Sensor Networks, Munchen, 2007, technical tutorial, Wireless Congress 2007: Systems & Applications.

[4] IEEE Computer Society, IEEE Std 802.15.4™-2006, New York, 2006, ISBN 0-7381-4997-7.

[5] Ganz A., Ganz Z., Wongthavarawat K., Multimedia Wireless Networks: Technologies, Standards, and QoS, Prentice Hall PTR, 2003, ISBN 0-13-046099-0.

[6] Koval Miroslav, Technologie senzorových sítí, Brno, 2007, diplomová práce, FIT VUT v Brně

[7] Nagy Jan, Zabezpečení bezdrátových senzorových sítí, Brno, 2007, diplomová práce, FIT VUT v Brně.

[8] Freescale Semiconductor, ZigBee Overwiev [course online], Techonline learning course, [cit. 2007-12-15], Dostupné na URL: <http://www.techonline.com/learning/course/100324>.

[9] Wikipedie otevřená encyklopedie, ZigBee [online], Poslední modifikace: 2008-03-13, [cit. 2008-04-3], Dostupné na URL: <http://cs.wikipedia.org/wiki/ZigBee>.

[10] Bradáč Z., Bezdrátový komunikační standard ZigBee [online], Automatizace, ročník 48, číslo 4, Poslední modifikace: duben 2005, [cit. 2008-02-02], Dostupné na URL: <http://www.automatizace.cz/article.php?a=638>.

[11] Veselý Martin, Bezdrátové ovládání domácích spotřebičů s technologií ZigBee, Praha, 2006, bakalářská práce, FEL ČVUT v Praze.

[12] ZigBee Alliance, ZigBee Specification: Document 053474r13, December 2006, ZigBee-2006 Specification.

[13] Flowers D., Otten K., et al., AN965: Microchip Stack for the ZigBee Protocol [online], Poslední modifikace: 2007-01-07, [cit. 2008-02-12], Dostupné na URL :<http://ww1.microchip.com/downloads/en/AppNotes/00965c.pdf>.

[14] Vokál Martin, Principy zabezpečení bezdrátových standardů, Brno, 2007, diplomová práce, FIT VUT v Brně.

[15] Koton J., Číka P., Křivánek V., Standard nízkorychlostní bezdrátové komunikace ZigBee [online], Poslední modifikace: 2006-04-18, [cit. 2008-02-01], Dostupné na URL: <http://access.feld.cvut.cz/view.php?nazevclanku=standard-nizkorychlostni-bezdratove-komuni-kacezigbee-&cisloclanku=2006032001>.

[16] Heile B., Chairman, ZigBee Alliance, ZigBee Overview, 2007, Dokument ve formátu pdf dostupný na URL: <http://www.zigbee.org/en/resources/presentations.asp>.

[17] ZigBee Aliance, FAQ [online], [cit. 2008-03-12], Dostupné na URL :<http://www.zigbee.org/en/about/faq.asp>.

[18] Microchip, PICDEM™ Z ZigBee™ Technology Demonstration Kit Product Overview [online], Poslední modifikace: 2006-06-21 [cit. 2008-04-27], Dostupné na URL: <http://ww1.microchip.com/downloads/en/DeviceDoc/51504d.pdf>.

[19] Microchip, MPLAB ICD 2 [online], Poslední modifikace: 2008-03-26 [cit. 2008-04-20], Dostupné na URL: <http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE-&nodeId=1406&dDocName=en010046>.

[20] Microchip, ZENA Network Analyzer [online], Poslední modifikace: 2007-02-15 [cit. 2008-04-20], Dostupné na URL: <http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en520682>.

[21] Microchip, MPLAB C18 (v3.00) C Compiler Getting Started [online], Poslední modifikace: 2005-03-10 [cit. 2008-04-20], Dostupné na URL: <http://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB_C18_Getting_Started_51295f.pdf >.

[22] Microchip, MPLAB IDE Quick Start Guide [online], Poslední modifikace: 2007-07-06 [cit. 2008-04-10], Dostupné na URL: <http://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB_Getting_Started_51281g.pdf>.

[23] Microchip, Zigbee and wireless topics [online], Poslední modifikace: 2008-04-28 [cit. 2008-04-28], Dostupné na URL: <http://forum.microchip.com/tt.aspx?forumid=177>.

[24] Microchip, ZigBee™ Stack [online], Poslední modifikace: 2007-01-02 [cit. 2008-04-22], Dostupné na URL: <http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2113&param=en520422>.

[25] Microchip, ZENA Analyzer User's Guide [online], Poslední modifikace: 2007-02-16 [cit. 2008-04-15], Dostupné na URL: <http://ww1.microchip.com/downloads/en/DeviceDoc/ZENA%20Analyzer%20User's%20Guide%2051606b.pdf>.

[26] Microchip, PIC18F2525/2620/4525/4620 Data Sheet [online], Poslední modifikace: 2007-11-15 [cit. 2008-04-15], Dostupné na URL: <http://ww1.microchip.com/downloads/en/DeviceDoc/39626d.pdf>.

Seznam příloh

Příloha 1. Zdrojový kód ukázkové aplikace mikrokontrolér.

Příloha 2. Základní struktura aplikace stavějící na ZiGbee stacku od firmy Microchip.

Příloha 3. Ukázka kódu vytvoření sítě koordinátorem.

Příloha 4. Ukázka kódu připojení koncového zařízení do sítě.

Příloha 5. Ukázka kódu odeslání zprávy.

Příloha 6. Ukázka kódu přijetí zprávy.

Příloha 7. CD.

Přílohy

Příloha 1.

Zdrojový kód ukázkové aplikace mikrokontrolér.

```
#include <p18cxxx.h>
#pragma config WDT = OFF  //vypne watchdog

/**
 * Funkce delay provádí v cyklu prázdný příkaz, čímž je zaručeno zpoždění
 * po jehož dobu zůstane dioda ve stejném stavu.
 */
void delay (void)
{
  unsigned int i;
  for (i = 0; i < 50000 ; i++) ;
}

void main (void)
{
  TRISA = 0;  // Nastavením registru TRISA ovlivníme, zda je PORTA vstupní nebo
              // výstupní. Každý bit registru TRISA ovlivní nastavení
              // příslušného pinu portu A. Hodnota 0 registru TRISA nastaví
              // všechny piny portu A jako výstupní.

  while (1) {
    PORTA = 0x01; // 00000001 první LED dioda (RA0) svítí
    delay ();     // žádná změna
    PORTA = 0x02; // 00000010 druha LED dioda (RA1) svítí, RA0 nesvítí
    delay ();     // žádná změna
    PORTA = 0x03; // 00000011 první i druha LED dioda svítí
    delay ();     // žádná změna
    PORTA = 0x00; // 00000000 obě LED diody nesvítí
    delay ();     // žádná změna
  }
}
```

Příloha 2.

Základní struktura aplikace stavějící na ZiGbee stacku od firmy Microchip.

```
//definice proměnné currentPrimitive, určuje stav, ve kterém se zařízení nachází
ZIGBEE_PRIMITIVE currentPrimitive;

  currentPrimitive = NO_PRIMITIVE;

  // aplikace běží jako automat ve smyčce, což je implementované cyklem while(1)
  while (1)
  {
    CLRWDT(); // nastaví watchdog (pokud by k tomu nedošlo aplikace
              // by byla po čase vyresetována)

    ZigBeeTasks( &currentPrimitive ); //vyvolá se mechanismus pro vykonání
                                      //požadované operace komunikačního modelu
    switch (currentPrimitive) {
```

... pokračování na další straně

Základní struktura aplikace stavějící na ZiGbee stacku od firmy Microchip – pokračování.

...pokračování z předchozí strany

```
   switch (currentPrimitive) {

  //Zde jsou větve kódu pro příslušné stavy, ve kterých se aplikace nachází.
  //Podle proměnné currentPrimitive (určuje stav, ve kterém se aplikace nachází)
  //se provede příslušná větev programu. Na konci každé větve je nutné správně
  //nastavit hodnotu proměnné currentPrimitive.
    // Pro názornost jsou zde uvedeny některé typy použitých větví
    // z ukázkových příkladů.

      //odpověď na požadavek o vytvoření sítě ("NLME_NETWORK_FORMATION_request")
      case NLME_NETWORK_FORMATION_confirm:

        ... // konkrétní kód pro danou větev

        // Je nutné nastavit proměnnou currentPrimitive, díky čemuž budou nižší
        // vrstvy komunikačního modelu vědět co dělat.
        currentPrimitive = ...
        break;  // ukončení větve

      // Stavem APSDE_DATA_indication proměnné currentPrimitive Zigbee stack
      // oznamuje aplikační vrstvě, že byla přijata zpráva.
      case APSDE_DATA_indication:
        ...
        currentPrimitive = ...
        break;

      // stav NO_PRIMITIVE označuje, že nepřicházejí žádné informace, požadavky
      // nebo potvrzení z nižších vrstev.
      case NO_PRIMITIVE:
        if (...) {
        // Na tomto místě koordinátor testuje, zda je vytvořena síť. Pokud
        // není, tak jí vytvoří. Naopak koncové zařízení zde testuje, zda je
        // připojeno k síti, a pokud není, tak se o připojení pokusí.
        }

        else { // Síť už je vytvořena nebo je uzel připojen. Nejčastěji se zde
               // implementuje odesílání zpráv. Například po stisknutí tlačítka
               // se zašle nějaká zpráva sousednímu zařízení a podobně.

          // Nejdříve se testuje, zda je stack připraven pro odeslání zprávy.
          if (ZigBeeReady()) {

            if ( ... ) {  // zde přijde například test na stlačení tlačítka
              ...         // kód pro zaslání zprávy

              // Nastaví se proměnná currentPrimitive tak, aby byla v zápětí
              // poslána zpráva. To je zajištěno voláním funkce ZigBeeTasks.
              currentPrimitive = APSDE_DATA_request;
              // Zda proběhlo poslání zprávy úspěšně nebo ne, lze analyzovat
              // ve větvi APSDE_DATA_confirm.
            }
          }
        }
        break;

      // Aby nedošlo k zacyklení, je nutné mít v kódu implicitní větev.
      default:
        currentPrimitive = NO_PRIMITIVE;
        break;
    }
  }
```

Příloha 3. Ukázka kódu vytvoření sítě koordinátorem

Pro úspěšné vytvoření sítě koordinátorem se nejprve provede část kódu ve větvi `NO_PRIMITIVE`, kde se nastaví počáteční parametry sítě. Poté se provádění v aplikační vrstvě přemístí do větve `NLME_NETWORK_FORMATION_confirm`, kde se dozvíme, zda byla síť vytvořena úspěšně nebo ne. Dále se zažádá o zpřístupnění sítě pro ostatní uzly sítě. Jak zpřístupnění dopadlo, lze nakonec analyzovat ve větvi `NLME_PERMIT_JOINING_confirm`.

Ukázka kódu vytvoření sítě koordinátorem.

```
//odpověď na požadavek o vytvoření sítě ("NLME_NETWORK_FORMATION_request")
case NLME_NETWORK_FORMATION_confirm:

  // Status == 0 -> síť byla úspěšně vytvořena
  if (!params.NLME_NETWORK_FORMATION_confirm.Status) {
    ConsolePutROMString( (ROM char *)"PAN " );
    PrintChar( macPIB.macPANId.byte.MSB );
    PrintChar( macPIB.macPANId.byte.LSB );
    ConsolePutROMString( (ROM char *)" started successfully.\r\n" );
    params.NLME_PERMIT_JOINING_request.PermitDuration = 0xFF;// No Timeout

    // Nastaví se proměnná currentPrimitive tak, aby byla v zápětí
    // zpřístupněna síť ostatním uzlům.
    currentPrimitive = NLME_NETWORK_FORMATION_request;
    // Zda dopadlo zpřístupnění sítě úspěšně nebo ne, lze analyzovat
    // ve větvi NLME_PERMIT_JOINING_confirm.
    currentPrimitive = NLME_PERMIT_JOINING_request;
  }
  else { // Status == 1 -> nastal problém při vytváření sítě
    PrintChar( params.NLME_NETWORK_FORMATION_confirm.Status );
    ConsolePutROMString( (ROM char *)" Error forming network.  Trying
again...\r\n" );

    // Nastavením proměnné currentPrimitive na hodnotu NO_PRIMITIVE
    // se docílí opakovanému pokusu tvorby sítě.
    currentPrimitive = NO_PRIMITIVE;
  }
  break;

  // odpověď na požadavek o povolení připojení do sítě
  // ("NLME_PERMIT_JOINING_request")
  case NLME_PERMIT_JOINING_confirm:
  if (!params.NLME_PERMIT_JOINING_confirm.Status) {
    // vše v pořádku -> síť povolena
    ConsolePutROMString( (ROM char *)"Joining permitted.\r\n" );
    currentPrimitive = NO_PRIMITIVE;
  }
  else { // nastala chyba, síť nelze zpřístupnit
    PrintChar( params.NLME_PERMIT_JOINING_confirm.Status );
    ConsolePutROMString( (ROM char *)" Join permission unsuccessful. We cannot
allow joins.\r\n" );
    currentPrimitive = NO_PRIMITIVE;
  }
  break;

  // stav NO_PRIMITIVE označuje, že nepřichází žádné informace, požadavky
  // nebo potvrzení z nižších vrstev.
  case NO_PRIMITIVE:
    // test na to zda, už je síť vytvořena
    if (!ZigBeeStatus.flags.bits.bNetworkFormed) {

      // Síť vytvořena ještě není.
      // Testuje se, zda tvorba sítě právě probíhá.
```

... pokračování na další straně

Ukázka kódu vytvoření sítě koordinátorem – pokračování.

...pokračování z předchozí strany

```
      // Síť vytvořena ještě není.
      // Testuje se, zda tvorba sítě právě probíhá.
      if (!ZigBeeStatus.flags.bits.bTryingToFormNetwork) {
        // Vytváření sítě ještě nezačalo, nastaví se potřebné parametry.
        ConsolePutROMString( (ROM char *)"Trying to start network...\r\n" );
        params.NLME_NETWORK_FORMATION_request.ScanDuration = 8;
        params.NLME_NETWORK_FORMATION_request.ScanChannels.Val =
ALLOWED_CHANNELS;

        params.NLME_NETWORK_FORMATION_request.PANId.Val = 0xFFFF;
        params.NLME_NETWORK_FORMATION_request.BeaconOrder =
AC_PIB_macBeaconOrder;

        params.NLME_NETWORK_FORMATION_request.BeaconOrder =
AC_PIB_macBeaconOrder;

       params.NLME_NETWORK_FORMATION_request.SuperframeOrder =
MAC_PIB_macSuperframeOrder;

        params.NLME_NETWORK_FORMATION_request.BatteryLifeExtension  =
MAC_PIB_macBattLifeExt;

        // Nastaví se proměnná currentPrimitive tak, aby bylo v zápětí
        // zahájeno vytvoření sítě.
        currentPrimitive = NLME_NETWORK_FORMATION_request;
        // Zda dopadlo vytvoření sítě úspěšně nebo ne, lze analyzovat
        // ve větvi NLME_NETWORK_FORMATION_confirm.
      }
    }
    else { // Síť už je vytvořena. Nejčastěji se zde
           // implementuje odesílání zpráv. Například po stisknutí tlačítka
           // se zašle zpráva sousednímu zařízení a podobně.
```

Příloha 4. Ukázka kódu připojení koncového zařízení do sítě.

Pro úspěšné připojení zařízení k síti se musí nejprve provést část kódu ve větvi `NO_PRIMITIVE`. Tam se zjistí, zda se uzel má připojit k nové síti (to vede později k vyhledání dostupných sítí) nebo se má připojit rovnou k nějaké předem známé síti, v tom případě se nastaví parametry připojení. Pokud bylo nutné vyhledat dostupné sítě (zařízení se připojuje k nové síti), provádění aplikace přejde do větve NLME_NETWORK_DISCOVERY_confirm, kde se zjistí, zda byla nějaká vhodná síť nalezena. V případě že ano, nastaví se parametry připojení podobně jako u připojení do známé sítě ve větvi `NO_PRIMITIVE`. Poté se přejde do větve `NLME_JOIN_confirm`, kde se zjistí, zda připojení proběhlo v pořádku.

Ukázka kódu připojení koncového zařízení do sítě.

```
//odpověď na požadavek o vyhledání sítě ("NLME_NETWORK_DISCOVERY_request")
case NLME_NETWORK_DISCOVERY_confirm:
  currentPrimitive = NO_PRIMITIVE;

  // otestuje, zda proces hledání neskončil chybou
  if (!params.NLME_NETWORK_DISCOVERY_confirm.Status) {
    // otestuje, zda byla nějaká síť nalezena
    if (!params.NLME_NETWORK_DISCOVERY_confirm.NetworkCount) {
      // síť nenalezena
      ConsolePutROMString( (ROM char*)"No networks found. Trying again...\r\n");
    }
    else {  // síť nalezena
      // uloží si descriptor na nalezené sítě
      NetworkDescriptor=params.NLME_NETWORK_DISCOVERY_confirm.NetworkDescriptor;
      // vybere síť k připojení
      currentNetworkDescriptor = NetworkDescriptor;
      // mastaví se parametry pro připojení
      params.NLME_JOIN_request.PANId  = currentNetworkDescriptor->PanID;
      ConsolePutROMString( (ROM char *)"Network(s) found. Trying to join ");
      PrintChar( params.NLME_JOIN_request.PANId.byte.MSB );
      PrintChar( params.NLME_JOIN_request.PANId.byte.LSB );
      ConsolePutROMString( (ROM char *)".\r\n" );
      params.NLME_JOIN_request.JoinAsRouter   = FALSE;
      params.NLME_JOIN_request.RejoinNetwork  = FALSE;
      params.NLME_JOIN_request.PowerSource    = NOT_MAINS_POWERED;
      params.NLME_JOIN_request.RxOnWhenIdle   = FALSE;
      params.NLME_JOIN_request.MACSecurity    = FALSE;

      // Nastaví se proměnná currentPrimitive tak, aby bylo v zápětí
      // zahájeno připojení k síti.
      currentPrimitive = NLME_JOIN_request;
      // Zda dopadlo připojení k síti úspěšně nebo ne, lze analyzovat
      // ve větvi NLME_JOIN_confirm.
    }
  }
  else { // proces hledání sítě skončil chybou
    PrintChar( params.NLME_NETWORK_DISCOVERY_confirm.Status );
    ConsolePutROMString(ROM char*)"Error finding network. Trying again...\r\n");
  }
  break;

//odpověď na požadavek o připojení k síti ("NLME_JOIN_request")
case NLME_JOIN_confirm:
  // otestuje, zde byl uzel úspěšně připojen
  if (!params.NLME_JOIN_confirm.Status) {
    // připojení proběhlo úspěšně
    ConsolePutROMString( (ROM char *)"Join successful!\r\n" );

    if (NetworkDescriptor) {
      free( NetworkDescriptor );
    }
    currentPrimitive = NO_PRIMITIVE;
  }
  else { // připojení se nezdařilo
    PrintChar( params.NLME_JOIN_confirm.Status );
    ConsolePutROMString( (ROM char *)" Could not join.\r\n" );
    currentPrimitive = NO_PRIMITIVE;
  }
  break;

// stav NO_PRIMITIVE označuje, že nepřichází žádné informace, požadavky
// nebo potvrzení z nižších vrstev.
case NO_PRIMITIVE:
```

... pokračování na další straně

Ukázka kódu připojení koncového zařízení do sítě – pokračování.

```
...pokračování z předchozí strany

// stav NO_PRIMITIVE označuje, že nepřichází žádné informace, požadavky
// nebo potvrzení z nižších vrstev.
case NO_PRIMITIVE:
  // testuje, zda už je uzel připojen v síti
  if (!ZigBeeStatus.flags.bits.bNetworkJoined) {
    // testuje, zda připojování právě probíhá
    if (!ZigBeeStatus.flags.bits.bTryingToJoinNetwork) {
      // uzel ještě není připojen a ani neprobíhá připojování
      // testuje, zda se připojí k známé síti
      if (ZigBeeStatus.flags.bits.bTryOrphanJoin) {
        // připojuje se k známé síti, tzn., že už zde byl někdy připojen
        ConsolePutROMString( (ROM char *)"Trying to join network as an
orphan...\r\n" );

        // nastaví se parametry pro připojení
        params.NLME_JOIN_request.JoinAsRouter = FALSE;
        params.NLME_JOIN_request.RejoinNetwork = TRUE;
        params.NLME_JOIN_request.PowerSource = NOT_MAINS_POWERED;
        params.NLME_JOIN_request.RxOnWhenIdle = FALSE;
        params.NLME_JOIN_request.MACSecurity = FALSE;
        params.NLME_JOIN_request.ScanDuration = 8;
        params.NLME_JOIN_request.ScanChannels.Val = ALLOWED_CHANNELS;

        // Nastaví se proměnná currentPrimitive tak, aby bylo v zápětí
        // zahájeno připojení do známé sítě.
        currentPrimitive = NLME_JOIN_request;
        // Zda dopadlo připojení k síti úspěšně nebo ne, lze analyzovat
        // ve větvi NLME_JOIN_confirm.
      }
      else { // uzel se musí připojit do nové sítě.

        ConsolePutROMString( (ROM char *)"Trying to join network as a new
device...\r\n" );

        params.NLME_NETWORK_DISCOVERY_request.ScanDuration = 8;
        params.NLME_NETWORK_DISCOVERY_request.ScanChannels.Val=ALLOWED_CHANNELS;
        // Nastaví se proměnná currentPrimitive tak, aby bylo v zápětí
        // zahájeno vyhledání dostupné sítě.
        currentPrimitive = NLME_NETWORK_DISCOVERY_request;
        // Jak dopadlo vyhledání dostupných sítí, lze analyzovat
        // ve větvi NLME_NETWORK_DISCOVERY_confirm.
      }
    }
  }
  else { // Uzel je již připojen. Nejčastěji se zde
         // implementuje odesílání zpráv. Například po stisknutí tlačítka
         // se zašle zpráva sousednímu zařízení a podobně.
```

Příloha 5. Ukázka kódu odeslání zprávy.

Odesílání zpráv se nejčastěji provádí ve větvi `NO_PRIMITIVE` jako reakce na událost (například stisknutí tlačítka) nebo ve větvi `APSDE_DATA_indicationn` kde má pak zaslaná zpráva charakter potvrzení nějaké příchozí zprávy. Příklad kódu ukazuje odeslání zprávy ve větvi `NO_PRIMITIVE` jako reakci na zmáčknutí tlačítka. Zpráva informuje cílový uzel o přepnutí své světelné diody. Ve větvi APSDE_DATA_confirm lze zjistit, zda odeslání zprávy proběhlo úspěšně, či nikoliv.

Ukázka kódu odeslání zprávy.

```
// odpověď na požadavek o zaslání zprávy ("APSDE_DATA_request")
case APSDE_DATA_confirm:
  // Pro ošetření zákmitu při zmáčknutí tlačítka se povoluje přerušení až
  // na tomto místě.
  RBIE = 1;
  if (params.APSDE_DATA_confirm.Status) {
    // Status == 1 nastala chyba, zpráva nebyla odeslána
    ConsolePutROMString( (ROM char *)"Error " );
    PrintChar( params.APSDE_DATA_confirm.Status );
    ConsolePutROMString( (ROM char *)" sending message.\r\n" );
  }
  else { // Status == 0 vše v pořádku, zpráva byla odeslána úspěšně
    ConsolePutROMString( (ROM char *)" Message sent successfully.\r\n" );
  }
  currentPrimitive = NO_PRIMITIVE;
  break;

case NO_PRIMITIVE:
  if (...) {
  // Na tomto místě koordinátor testuje, zda je vytvořena síť. Pokud
  // není, tak jí vytvoří. Naopak koncové zařízení zde testuje, zda je
  // připojeno k síti, a pokud není, tak se o připojení pokusí.
  }
  else { // Síť už je vytvořena nebo je uzel připojen. Nejčastěji se zde
         // implementuje odesílání zpráv. Například po stisknutí tlačítka
         // se zašle       zpráva sousednímu zařízení a podobně.

    // otestuje, zda je ZigBee stack připraven pro odeslání zprávy
    if (ZigBeeReady()) {

      // otestuje, zda bylo stlačeno dané tlačítko
      if ( myStatusFlags.bits.bLightSwitchToggled ) {
        // Tlačítko bylo stlačeno, bude se zasílat zpráva nesoucí
        // informaci o přepnutí diody na cílovém zařízení.
        myStatusFlags.bits.bLightSwitchToggled = FALSE;
        // naplnění zprávy vhodnými parametry a daty
        TxBuffer[TxData++] = APL_FRAME_TYPE_KVP | 1;
        TxBuffer[TxData++] = APLGetTransId();
        TxBuffer[TxData++] = APL_FRAME_COMMAND_SET | (APL_FRAME_DATA_TYPE_UINT8
<< 4);

        TxBuffer[TxData++] = OnOffSRC_OnOff & 0xFF;
        TxBuffer[TxData++] = (OnOffSRC_OnOff >> 8) & 0xFF;
        TxBuffer[TxData++] = LIGHT_TOGGLE;
        ZigBeeBlockTx();
        params.APSDE_DATA_request.DstAddrMode = APS_ADDRESS_16_BIT;
        params.APSDE_DATA_request.DstEndpoint = EP_LIGHT;
        params.APSDE_DATA_request.DstAddress.ShortAddr = destinationAddress;
        params.APSDE_DATA_request.ProfileId.Val = MY_PROFILE_ID;
        params.APSDE_DATA_request.RadiusCounter = DEFAULT_RADIUS;
        params.APSDE_DATA_request.DiscoverRoute = ROUTE_DISCOVERY_ENABLE;
        params.APSDE_DATA_request.TxOptions.Val = 0;
        params.APSDE_DATA_request.SrcEndpoint = EP_LIGHT;
        params.APSDE_DATA_request.ClusterId = OnOffSRC_CLUSTER;
        ConsolePutROMString( (ROM char *)" Trying to send light switch
message.\r\n" );

        // Nastaví se proměnná currentPrimitive tak, aby bylo v zápětí
        // zahájeno poslání zprávy.
        currentPrimitive = APSDE_DATA_request;
        // Zda dopadlo poslání zprávy úspěšně nebo ne, lze analyzovat
        // ve větvi APSDE_DATA_confirm.
      }
    }
  }
  break;
```

Příloha 6. Ukázka kódu přijetí zprávy.

Stavem `APSDE_DATA_indication` proměnné `currentPrimitive` Zigbee stack oznamuje aplikační vrstvě, že byla přijata zpráva. To znamená, že se ve větvi `APSDE_DATA_indication` provádí kód pro zpracování příchozí zprávy. V tomto příkladě bude uvedeno zpracování zprávy pro světelnou diodu (endpoint – EP_LIGHT).

Ukázka kódu přijetí zprávy

```
// Stavem APSDE_DATA_indication proměnné currentPrimitive Zigbee stack
// oznamuje aplikační vrstvě, že byla přijata zpráva.
case APSDE_DATA_indication: {

  WORD_VAL    attributeId;
  BYTE        command;
  BYTE        data;
  BYTE        dataLength;
  //BYTE        dataType;
  BYTE        frameHeader;
  BYTE        sequenceNumber;
  BYTE        transaction;
  BYTE        transByte;

  currentPrimitive = NO_PRIMITIVE;
  //funkce "APLGet" vrací 1 byte přijaté zprávy, která je uložena v paměti
  frameHeader = APLGet();

  // DstEndpoint signalizuje pro jaký koncový bod je zpráva určena
  switch (params.APSDE_DATA_indication.DstEndpoint) {

    // koncový bod je dioda
    case EP_LIGHT:
      if ((frameHeader & APL_FRAME_TYPE_MASK) == APL_FRAME_TYPE_KVP) {
        // typ zprávy je KVP
        frameHeader &= APL_FRAME_COUNT_MASK;
        for (transaction=0; transaction<frameHeader; transaction++) {

          sequenceNumber          = APLGet();
          command                 = APLGet();
          attributeId.byte.LSB    = APLGet();
          attributeId.byte.MSB    = APLGet();

          command &= APL_FRAME_COMMAND_MASK;

          if ((params.APSDE_DATA_indication.ClusterId == OnOffSRC_CLUSTER) &&
            (attributeId.Val == OnOffSRC_OnOff)) {
            // jedná se o cluster "OnOffSRC_CLUSTER" a atribut "OnOffSRC_OnOff"
            if (command == APL_FRAME_COMMAND_SET) {

              // Data type for this attibute must be APL_FRAME_DATA_TYPE_UINT8
              data = APLGet();
              switch (data) {

                case LIGHT_TOGGLE:
                  // Zpráva obsahuje data oznamující, že se má
                  // světelná dioda přepnout do opačného stavu.
                  ConsolePutROMString( (ROM char *)"Toggling light.\r\n" );
                  MESSAGE_INDICATION ^= 1;
                  TxBuffer[TxData++] = SUCCESS;
                  break;
                default:
                  PrintChar( data );
                  ConsolePutROMString((ROM char *)"Invalid light message.\r\n");
```

... pokračování na další straně

Ukázka kódu přijetí zprávy-pokračování

...pokračování z předchozí strany

```
                default:
                  PrintChar( data );
                  ConsolePutROMString((ROM char *)"Invalid light message.\r\n");
                  TxBuffer[TxData++] = KVP_INVALID_ATTRIBUTE_DATA;
                  break;
              }
            }
            TxData = TX_DATA_START;
          }
        } // konec transakce
      } // konec APL_FRAME_TYPE_KVP
      break;
    default:
      break;
  }
  // po zpracování se musí aktuálně přijatá zpráva zrušit, to je důležité
  // aby mohly být přijímány další zprávy
  APLDiscardRx();
```

Příloha 7. CD

Přiložený nosič obsahuje veškeré výstupy vzniklé při tvorbě této práce. Tyto informace mají následující strukturu adresářů:

\dokument	Text bakalářské práce ve formátu pdf.
\webove_stranky	Zdrojové kódy multimediální podpory.
\aplikace	Ukázkové aplikace stavějící na ZigBee stacku.
\aplikace\1_aplikace	Zdrojový kód první aplikace – mikrokontrolér
\aplikace\2_aplikace	Zdrojové kódy (koordinátora, koncového zařízení, stacku) pro druhou aplikaci – světelné diody.
\aplikace\3_aplikace	Zdrojové kódy (koordinátora, koncového zařízení, stacku) pro třetí aplikaci – teplotní senzor.
\prilohy	Přílohy 1 – 6 v textové podobě.