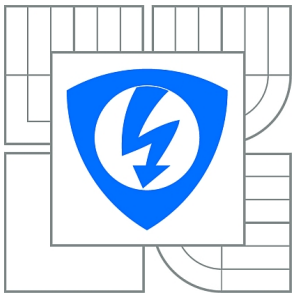
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

STANDARD IEEE 802.15.4 A JEHO PRAKTICKÁ REALIZACE

IEEE 802.15.4 STANDARD AND ITS PRACTICAL REALIZATION

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

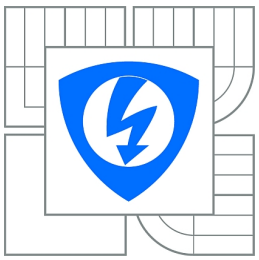
AUTOR PRÁCE — JAN KOLAŘÍK

AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN KOUTNÝ

SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Jan Kolařík	***ID:***	109675
Ročník:	3	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Standard IEEE 802.15.4 a jeho praktická realizace

POKYNY PRO VYPRACOVÁNÍ:

Podrobně se seznamte s bezdrátovým komunikačním standardem IEEE 802.15.4 ZigBee. Prostudujte strukturu tohoto standardu, princip komunikace, topologie sítě, zabezpečení a zaměřte se na praktické využití v různých podmínkách provozu. Na základě poznatků navrhněte a realizujte dvě laboratorní úlohy. První úloha bude prakticky a jasně seznamovat s daným standardem a seznamovat s odlišnostmi reálně měřených paramatrů s parametry teoretickými. Druhá úloha pak bude podrobně seznamovat se samostatnými moduly a možnostmi jejich programování.

DOPORUČENÁ LITERATURA:

[1] LABIOD, H., et al. Wi-Fi, Bluetooth, Zigbee and WiMax. 2007. 316 s. ISBN 978-1402053962.
[2] AXELSON, Jan. Embedded Ethernet and Internet Complete. 2003. 482 s. ISBN 978-1931448000.

Termín zadání:	29.1.2010	***Termín odevzdání:***	2.6.2010

Vedoucí práce: Ing. Martin Koutný

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Bakalářská práce se věnuje standardu IEEE 802.15.4, bezdrátové komunikační technologii ZigBee a zaměřuje se na její praktickou realizaci. Hlavní částí této práce je vytvoření dvou laboratorních úloh za pomoci vývojového kitu od firmy Jennic a analyzátoru sítě ZENA. Za pomoci těchto laboratorní úloh se studenti seznámí s vlastnostmi a strukturou standardu ZigBee, jako je dosah zařízení, rušení jinými technologiemi a složením datových rámců a naučí se vytvářet jednoduché aplikace pro moduly ZigBee. Laboratorní úlohy jsou také na závěr proměřeny a v jednotlivých kapitolách jsou popsány a uvedeny výsledky těchto úloh.

KLÍČOVÁ SLOVA

IEEE 802.15.4, ZigBee, praktická realizace, laboratorní úloha, JN5139, ZENA

ABSTRACT

The bachalor thesis deals with IEEE 802.15.4 standard, ZigBee wireless communication technology and describes its practical implementation. The main part of this work is the creation of two laboratory tasks using the development kit from Jennic and the ZENA network analyzer. In the laboratory assignments, students become familiar with the properties and structure of the ZigBee standard, as is the range of equipment, interference caused other technologies, and passing data frames and learn how to create simple applications for ZigBee module. Finally, laboratory assignments are measured and in each chapter provides descriptions and results of these tasks.

KEYWORDS

IEEE 802.15.4, ZigBee, practical implementation, laboratory assignment, JN5139, ZENA

KOLAŘÍK, Jan *STANDARD IEEE 802.15.4 A JEHO PRAKTICKÁ REALIZACE*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2010. 58s. Vedoucí práce byl Ing.Martin Koutný

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „STANDARD IEEE 802.15.4 A JEHO PRAKTICKÁ REALIZACE“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

Brno ....

(podpis autora)

Děkuji vedoucímu bakalářské práce Ing. Martinu Koutnému za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

Brno ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

ZigBee je bezdrátová komunikační technologie postavená na standardu IEEE 802.15.4. Jedná se o poměrně novou technologii platnou od roku 2004. Tato technologie má doplnit stávající rozšířené technologie, jako Wifi a BlueTooth. Primárně je určena pro spojení zařízení s malým výkonem na malé vzdálenosti do 75 m, při využití multiskokového ad-hoc směrování je komunikace umožněna i bez přímé rádiové viditelnosti. Díky své nižší přenosové rychlosti, oproti BlueTooth, není určena pro přenos velkého množství dat, ale jelikož ZigBee zařízení vynikají zejména svojí jednoduchostí, malou spotřebou energie a také vysokou odolností proti rušení, jeho využití spadá spíše do odvětví průmyslové automatizace a do senzorových sítí, kde se dá využít například pro automatizaci budov a dálkové ovládání spotřebičů.

V první části této práce bude představen standard IEEE 802.15.4, na kterém je technologie ZigBee postavena. Podrobněji budou rozebrány síťové vrstvy specifikované tímto standardem. Pozornost bude věnována použitým frekvenčním pásmům, síťovým topologiím a způsobu komunikace.

Ve druhé části bude rozebrán komunikační standard ZigBee. Zde budou popsány síťové vrstvy vytvářené ZigBee aliancí, zejména adresování, způsob zabezpečení a také typy uživatelských aplikací.

Třetí, čtvrtá a pátá část bude podrobněji popisovat zařízení, které budou sloužit k praktické realizaci ZigBee sítě a k vytvořená laboratorních úloh. Jedná se o analyzátor ZENA a vývojový kit JN5139–EK010 od firmy Jennic. V těchto částech bude také probrán základ vývoje ZigBee aplikací.

V následujících částech budou již popisovány způsoby realizace ZigBee aplikací a samotné laboratorní úlohy, které budou hlavním výsledkem této bakalářské práce.

1 STANDARD IEEE 802.15.4

Standard IEEE 802.15.4 [2] je vytvářen pracovní skupinou IEEE 802.15. Vyznačuje se zejména nízkou spotřebou elektrické energie, vysokou mírou zabezpečení a spolehlivostí přenosu dat, malou přenosovou rychlostí a také malou složitostí. Díky těmto vlastnostem mají zařízení postavená na tomto standardu nízké pořizovací a provozní náklady a proto jsou vhodná k použití i na místech, kde není rozvinutá infrastruktura. IEEE 802.15.4 specifikuje fyzickou a linkovou vrstvu bezdrátových osobních sítí WPAN.

1.1 Fyzická vrstva (PHY)

Fyzická vrstva je nejnižší vrstvou komunikačního systému OSI. Zajišťuje komunikaci mezi linkovou vrstvou a fyzickým rozhraním sítě. Tato vrstva definuje frekvenční pásma a typy modulace signálu.

1.1.1 Frekvenční pásma

Dle standardu IEEE 802.15.4 můžou být tato zařízení použita v 3 různých frekvenčních pásmech (shrnutí parametrů těchto pásem v tab. 1.1):

- 868 MHz – pásmo pro použití v Evropě, s jedním kanálem a přenosovou rychlostí, která se dle typu použité modulace pohybuje od 40 kb/s do 250 kb/s;
- 915 MHz – pásmo používané v Americe, s deseti kanály a přenosovou rychlostí, které se dle typu použité modulace pohybuje od 40 kb/s do 250 kb/s;
- 2 450 MHz – pásmo, které je možné používat celosvětově, obsahuje 16 kanálů a přenosová rychlost jde zde až 250 kb/s. Toto pásmo je ze všech tří nejpoužívanější [2].

1.1.2 Modulace signálu

Pro přenos signálu se používají 3 různé typy modulace [1], [2], [7]:

- BPSK (Binary Phase Shift Keying) – je to dvoustavový typ modulace používaný na frekvenčních pásmech 868 MHz a 915 MHz. Fyzické vrstvy používající tuto metodu modulování byly popsány již v roce 2003 a přenosové rychlosti činili 20 kb/s a 40 kb/s. Od té doby byly vyvinuty další fyzické vrstvy s většími přenosovými rychlostmi používající modulace typu ASK a O–QPSK;
- ASK (Amplitude Shift Keying) – tento typ modulace je používaný na frekvenčních pásmech 868 MHz a 915 MHz. Přenosové rychlosti se při použití tohoto typu modulace vyrovnaly rychlostem v nejpoužívanějším pásmu 2 450 MHz;

- O–QPSK (Offset–Quadrature Phase Shift Keying) – principielně vychází z modulace typu QPSK. Rozdíl je v tom, že u ní nedochází ke změnám stavu z 11 na 00 a z 01 na 10, což v konstelačním diagramu znamená změnu fáze o 180°. To zavádí parazitní amplitudovou modulaci s hloubkou 100 % a zvyšuje nežádoucí frekvenční postranní spektra a chybovost přenosu. U O–QPSK může dojít pouze ke změně stavu o 90°, čímž se sníží hloubka parazitní amplitudové modulace na 30 %. Oproti QPSK lze symboly posílat dvojnásobnou rychlostí. Tím se zvýší přenosová rychlost, ale také se rozšíří přenosová šířka pásma.

Tab. 1.1: Frekvenční pásma a typy modulace používané v ZigBee

Pásmo [MHz]	**Frekvenční rozsah [MHz]**	**Lokalita**	**Čísla kanálů**	**Přenosová rychlost [kb/s]**	**Typ modulace**
868	868.0–868.6	Evropa	0	20	BPSK
				250	ASK
				100	O–QPSK
915 (ISM)	902.0–928.0	Amerika Austrálie	1–10	40	BPSK
				250	ASK
				250	O–QPSK
2 450 (ISM)	2 400.0–2 483.5	Celosvětově	11–26	250	O–QPSK

1.2 Linková (MAC) vrstva

Další vrstvou komunikačního systému OSI je linková vrstva. Tato vrstva zajišťuje přístup ke kanálům, stará se o nastavení přenosových parametrů linky a vytváří fyzické rámce.

1.2.1 Typy uzlů

Standard definuje dva typy uzlů, a sice plně funkční zařízení (FFD – Full Function Device) a zařízení s omezenými funkcemi (RFD – Reduced Function Device).

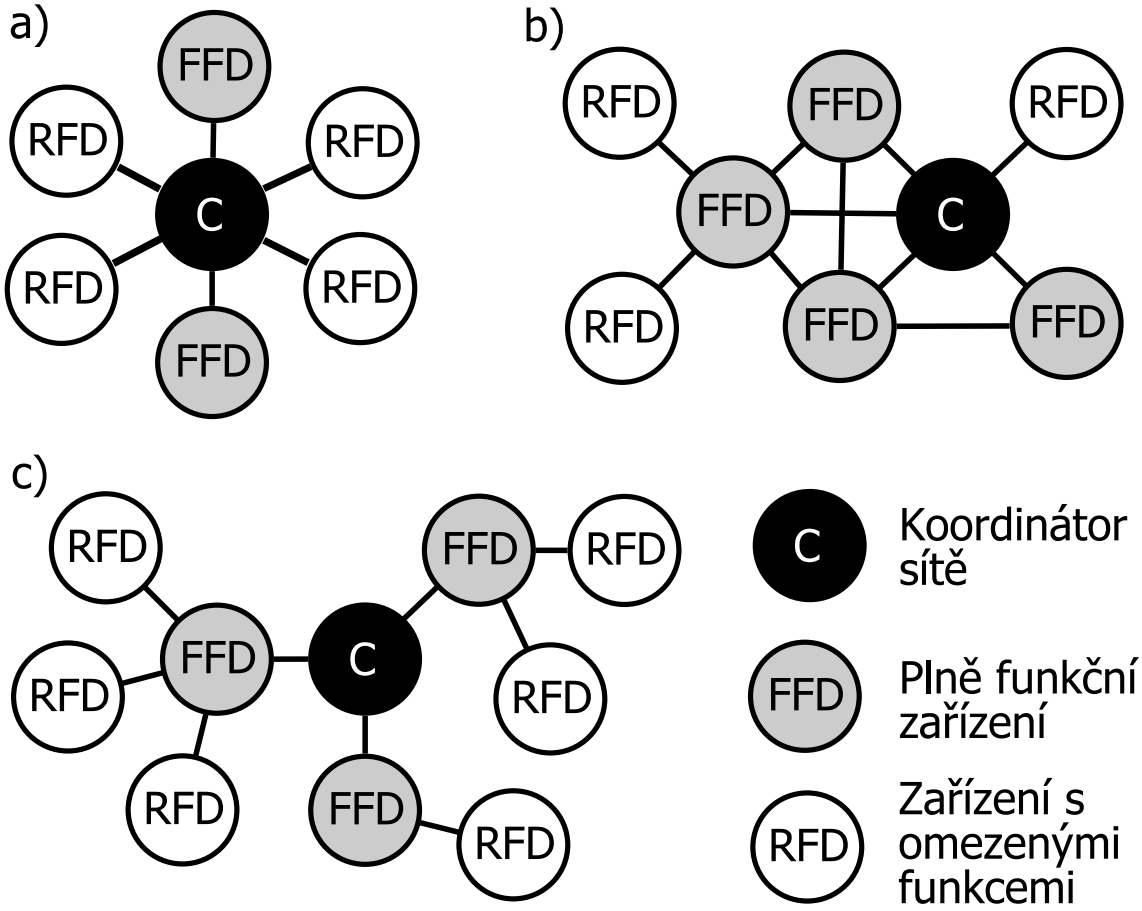
FFD je schopno zajistit veškeré služby, které definuje standard. Může sloužit jako koordinátor sítě, stejně tak jako směrovač.

RFD je realizované co nejjednodušším způsobem za účelem snížení hardwarové náročnosti. Může pracovat jen jako koncové zařízení.

1.2.2 Topologie sítě

IEEE 802.15.4 definuje 3 různé druhy uspořádání sítí (Obr. 1.1):

- hvězda – jedná se o základní typ uspořádání. Centrálním prvkem sítě je koordinátor, se kterým přímo komunikují všechny ostatní zařízení;
- strom – topologie typu strom umožňuje zvětšit dosah sítě tak, že koncová zařízení nemusejí být v přímém dosahu koordinátora, ale můžou být připojeni (větvit se) přes směrovače;
- síť (mesh) – tento typ je podobný topologii typu strom. Umožňuje však navíc vytvoření redundantních spojení, kdy nemusí být koncové zařízení připojeno jen přes jeden směrovač, ale může k němu vést více cest [7].



Obr. 1.1: Topologie sítí a) hvězda, b) síť, c) strom

1.2.3 Přístupové metody

Pro přístup k fyzickému médiu se používá metoda CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance and optional time slotting). Pro tuto metodu je charakteristický vícenásobný přístup a naslouchání nosné. Princip spočívá v tom, že stanice před započetím vysílání poslouchá, zda je přenosové médium volné. Jestliže

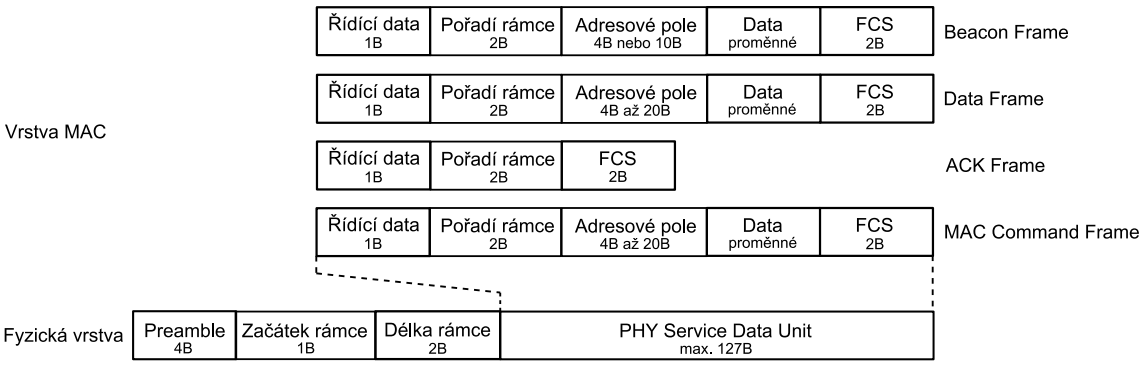
zjistí, že je volné, tak začne vysílat datový rámec. V opačném případě počká na dokončení právě probíhajícího vysílání [2] a [8].

1.2.4 Datové rámce

Pro přenos dat nebo pro řídící účely se v sítích podle standardu IEEE 802.15.4 používají 4 typy datových rámců [1] (Obr. 1.2):

- Beacon Frame – využívá se pro synchronizace zařízení při použití módu beacon enable, ve kterém jsou klientská zařízení uvedena do stavu nízké spotřeby;
- Data Frame – rámec využívaný pro všechny typy přenosu užitečných dat;
- Acknowledgement (ACK) Frame – je vysílán ihned po přenosu paketu a využívá se pro přenos potvrzovací informace;
- MAC Command Frame – rámec sloužící k nastavování a řízení klientských zařízení v síti.

Na obrázku jsou uvedeny struktury jednotlivých typů rámců používaných v linkové (MAC) vrstvě. Maximální délka těchto rámců je 127 byte [2].



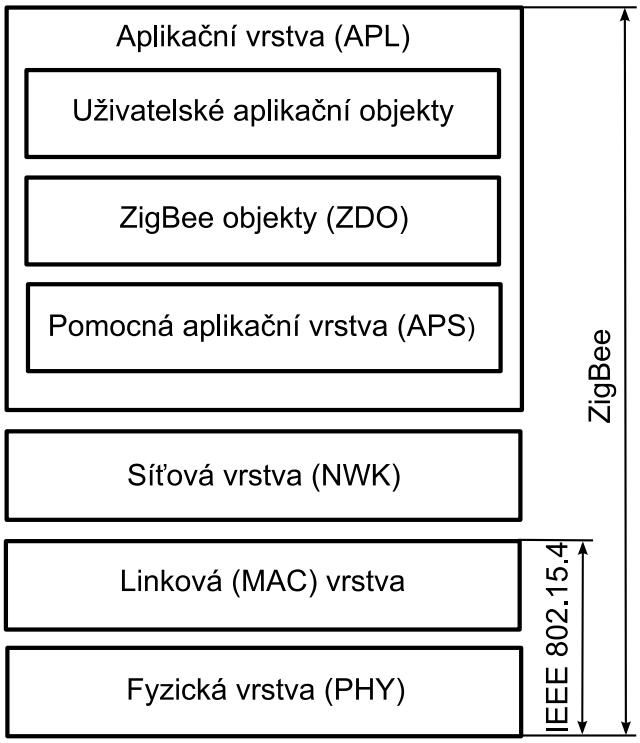
Obr. 1.2: Datové rámce

2 ZIGBEE

Protokol ZigBee je postavený na standardu IEEE 802.15.4. Byl vyvinut pro bezdrátové propojení elektronických zařízení s nízkou spotřebou (provoz na baterie) a nízkými náklady zejména do oblasti průmyslové automatizace a senzorových sítí. Může však nalézt uplatnění i v jiných aplikacích.

Kvůli nutnosti implementovat standard ZigBee i do málo výkonných 8bitových mikrokontrolérů bylo dbáno na maximální jednoduchost implementace protokolů. Díky tomu struktura protokolů nezabere více než 30 kB programové paměti.

ZigBee aliance [11] vytváří podobu síťové a aplikační vrstvy síťového modelu OSI. Na Obr. 2.1 si můžete prohlédnout referenční model ZigBee.



Obr. 2.1: Referenční model ZigBee

2.1 Síťová vrstva (NWK)

Třetí vrstvou komunikačního systému OSI je síťová vrstva. Tato vrstva poskytuje spojení mezi standardem IEEE 802.15.4 a uživatelskými aplikacemi ZigBee. Stará se o adresování, směrování a zabezpečení v síti.

2.1.1 Typy zařízení

V ZigBee síti můžeme mít 3 různé typy zařízení [7]:

- koordinátor – jedná se o FFD zařízení a jak už sám název vypovídá, koordinátor monitoruje a spravuje chod celé sítě. Takovéto zařízení může být v síti jen jedno;
- směrovač – jedná se taktéž o zařízení FFD. Směrovač může sloužit jako koncové zařízení, ale také může přeposílat data od koordinátora ke koncovému zařízení a naopak. Tím umožňuje vytvoření rozsáhlejší sítě, kdy koncové zařízení nemusí být v přímém dosahu koordinátora;
- koncové zařízení – je to RFD zařízení a v síti stojí vždy na posledním místě. Toto zařízení většinou plní funkci měřící nebo senzorové jednotky.

2.1.2 Adresování a sestavení sítě

Vzhledem k tomu, že je teoreticky možné vytvořit v jednom prostoru až 216 ZigBee sítí, musí se nějak identifikovat. Tato identifikace je zajištěná pomocí 16bitového identifikátoru PAN ID.

Jelikož v jedné ZigBee síti může být také až 216 zařízení, byla jedinečná identifikace zařízení vyřešena pomocí dvou adres [7] a [8]:

- IEEE (MAC) – 64bitová adresa. Tuto adresu přiděluje institut IEEE a je jedinečná. To znamená, že žádné dvě zařízení nemůžou mít stejnou IEEE adresu;
- síťová adresa – 16bitová adresa. Je používána pouze pro adresování v místní síti. Přiděluje ji koordinátor, při připojování do sítě. Tato adresa je jedinečná pouze v rámci jedné sítě. V dalších sítích můžou být zařízení se stejnou síťovou adresou.

Síť sestavuje koordinátor. Nejprve je nutné najít vhodný frekvenční kanál (obvykle nejméně aktivní). Dále koordinátor přiřadí síti PAN ID a následně už čeká na připojení nových uzlů do sítě. Připojování nového uzlu probíhá v následujících krocích.

1. Nový uzel prohledá všechny dostupné kanály, čímž najde všechny dostupné sítě a vybere si tu, do které se chce připojit.
2. Dále hledá uzel sítě (koordinátora nebo směrovač) s nejsilnějším signálem, ke kterému by se mohl připojit.
3. Následně pošle zprávu s žádostí o připojení k síti.
4. Nejbližší uzel sítě zprávu příjme, a pokud se nejedná o zakázané zařízení, dovolí novému uzlu připojení a přidělí mu síťovou adresu.

2.2 Zabezpečení

Pro nejzákladnější zabezpečení ZigBee sítě se používá AES (Advanced Encryption Standard) s klíčem o délce 128 bitů. Toto zabezpečení je řešeno na úrovni síťové vrstvy. Pokud je potřeba zabezpečit i Acknowledgement Frame, Beacon Frame a MAC Command Frame, řeší se toto na úrovni linkové vrstvy pomocí AES. Díky tomu je možné ověřit autenticitu a integritu MAC rámce a zajistit jeho důvěrnost. Při požadavku na ověření integrity je vytvořen MIC (Message Integrity Code) o délce 4, 8 či 16 oktetů a je vyslán společně s MAC rámcem. V tomto případě je použit AES algoritmus v CTR (Counter) módu. Pokud je nutné zajistit důvěrnost MAC rámce je k němu přidána informace o pořadí rámce a klíče (Frame Count, Key Sequence Count). Na vysílací a přijímací straně je udržována aktuální informace o čísle rámce. Pokud obdrží přijímací zařízení rámec s neplatným číslem je detekováno narušení bezpečnosti. AES je použit v CBC-MAC (Cipher Block Chaining) módu. Při implementaci jak ověřování integrity, tak šifrování je použit AES v módu, jež je nazýván CCM.

Síťová vrstva používá k zabezpečení SSP (Security Services Provider). Tato vrstva zajišťuje zabezpečení ochozích rámců, dekódování a ověřování pravosti příchozích rámců. Jako zabezpečovací algoritmus je použit AES v mírně modifikovaném módu CCM, jež nese označení CCM. Síťová vrstva je zodpovědná za realizaci zabezpečení. Vyšší vrstvy se starají o nastavení SSP (nastavení klíčů a udávají, jakým způsobem bude použit CCM pro jednotlivé rámce) [2] a [8].

2.3 Aplikační vrstva (APL)

Aplikační vrstva je druhou a poslední vrstvou komunikačního systému OSI určovanou standardem ZigBee. Tato vrstva zajišťuje všechny potřebné služby a skládá se z pomocné aplikační vrstvy, ZigBee objektů a uživatelských aplikačních objektů [1].

2.3.1 Pomocná aplikační vrstva

Pomocná aplikační podvrstva je zodpovědná za párování zařízení podle poskytovaných služeb a požadavků. To je realizováno pomocí tzv. párovací (binding) tabulky.

2.3.2 ZigBee objekty

ZigBee objekt definuje roli jednotlivých zařízení v rámci sítě (koordinátor, směrovač, koncové zařízení). Dále zajišťuje vyhledávání nových zařízení a jimi poskytovaných služeb. V neposlední řadě zodpovídá za zabezpečení (volí jeho způsob, jako např. veřejné klíče, symetrické klíče).

2.3.3 Uživatelské aplikační objekty

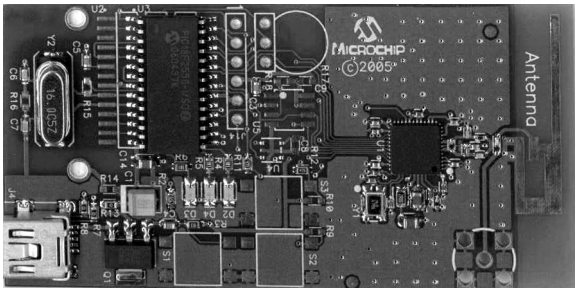
Uživatelské aplikační objekty implementují konkrétní požadavky aplikace dle definovaného ZigBee profilu. ZigBee profil zastřešuje definice možných zařízení, formátů a typů zpráv. Každý profil je určen unikátním 16bitovým identifikátorem podle specifikace ZigBee Alliance.

3 ZENA WIRELESS NETWORK ANALYZER

Analyzátor ZENA [10] je jedno ze zařízení, které umožňuje analyzovat provoz sítě postavené na standardu IEEE 802.15.4. Součástí balíčku analyzátoru ZENA je:

- modul analyzátoru ZENA (Obr. 3.1);
- USB mini-B kabel;
- základní software dodávaný na CD-ROM;
- rozšiřující software dodávaný na CD-ROM.

Modul analyzátoru ZENA je postavený na mikrokontroléru PIC18LF2550, dále obsahuje IEEE 802.15.4 transceiver. V současné době podporuje frekvenční pásmo 2,4 GHz. K počítači je možné ho připojit pomocí vestavěného USB portu a dodávaného USB mini-B kabelu.



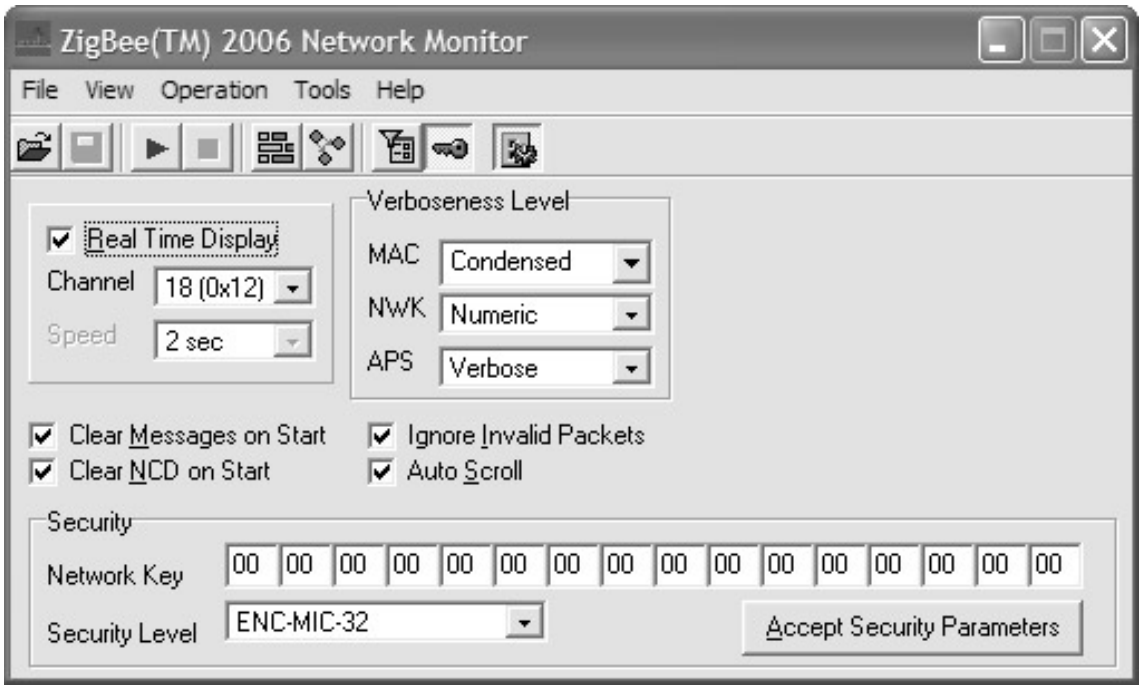
Obr. 3.1: ZENA Wireless Network Analyzer

Základní software (Obr. 3.2) umožňuje rychlý vývoj ZigBee stacku pro moduly firmy Microchip [9]. Dále je schopen dekódovat pakety protokolu ZigBee a také protokolu MiWi. Pomocí analyzátoru ZENA je také možné analyzovat topologii sítě a graficky si ji zobrazit. Pomocí základního softwaru není možné dekódovat data zabezpečených sítí. Tyto data lze ale dekódovat pomocí rozšiřujícího softwaru, do kterého lze vložit síťový šifrovací klíč.

3.1 Dekódování paketů protokolu ZigBee

Dekódování paketů může probíhat jak v reálném čase, ale je také možné dekódovat již dříve zachycené pakety, které jsme si uložili do souboru typu `.zna`, což je formát souboru, který v sobě uchovává právě všechny informace o zachycených paketech.

Při dekódování v reálném čase je nejprve potřeba nastavit několik parametrů. Prvním, a tím nejdůležitějším, je zaškrtnout `Real Time Display`, dále je potřeba nastavit kanál, na kterém vysílají zařízení, od kterých chceme data zachytávat. Další



Obr. 3.2: Software analyzátoru ZENA

z parametrů je úroveň dekódování. Nastavuje se na třech různých vrstvách, a to na linkové vrstvě, síťové vrstvě a na pomocné aplikační vrstvě. Možné jsou 3 úrovně dekódování. První z nich je možnost `Condensed`, které zobrazí jen hlavičky jednotlivých vrstev. Druhá je možnost `Numeric`, která již jde do větší hloubky a zobrazuje nám numericky obsah jednotlivých hlaviček rozdělených na typy dat. Poslední z nich je možnost `Verbose`, které jde do největší hloubky a rozebírá nám na jednotlivé části i kontrolní rámce jednotlivých hlaviček. Úroveň dekódování lze v průběhu zachytávání i po jeho ukončení libovolně měnit. Poslední parametr je hodnota síťového šifrovací klíče. Výchozí hodnota jsou samé nuly, které značí, že není použito žádné šifrování.

Porovnání 3 úrovní dekódování v jednotlivých vrstvách si je možné prohlédnout na obrázcích (Obr. 3.3), (Obr. 3.4) a (Obr. 3.5)

3.2 Grafické zobrazení topologie sítě

Topologii sítě si můžeme opět, stejně jako dekódování paketů, zobrazit z uloženého souboru nebo přímo při zachytávání v reálném čase.

Při zachytávání v reálném čase je možné zobrazit společně, jak dekódování paketů, tak topologii sítě. Když zachytí analyzátor ZENA zprávu z jednoho zařízení do druhého, zobrazí obě zařízení na displeji (Obr. 3.6). Zařízení jsou rozlišovány 64bi-

Úroveň dekódování: Condensed

MAC Header
0x61 0x88 0xED 0x15 0xC4
0x00 0x00 0x6F 0x79

Úroveň dekódování: Numeric

MAC Frame Control	Seq Num	Dest PAN	Dest Addr	Source Addr
0x8861	0xED	0xC415	0x0000	0x796F

Úroveň dekódování: Verbose

MAC Frame Control					Seq Num	Dest PAN	Dest Addr	Source Addr
Type	Sec	Pend	ACK	IPAN				
DATA	N	N	Y	Y	0xED	0xC415	0x0000	0x796F

Obr. 3.3: Úrovně dekódování na vrstvě MAC

Úroveň dekódování: Condensed

NWK Header
0x44 0x00 0x00 0x00
0x79 0x04 0xC5

Úroveň dekódování: Numeric

NWK Frame Control		Dest Addr	Source Addr	Radius	Seq Num
0x00	0x44	0x0000	0x796F	0x04	0xC5

Úroveň dekódování: Verbose

NWK Frame Control				Dest Addr	Source Addr	Radius	Seq Num
Type	Ver	Route	Sec				
DAT	0x1	EN	N	0x0000	0x796F	0x04	0xC5

Obr. 3.4: Úrovně dekódování na vrstvě NWK

Úroveň dekódování: Condensed

APS Header
0x00 0x40 0x12 0x23
0x41 0x21 0x01

Úroveň dekódování: Numeric

APS Frame Control	Dest EP	Cluster ID	Profile ID	Source EP	APS Counter
0x00	0x40	0x2312	0x4101	0x21	0x01

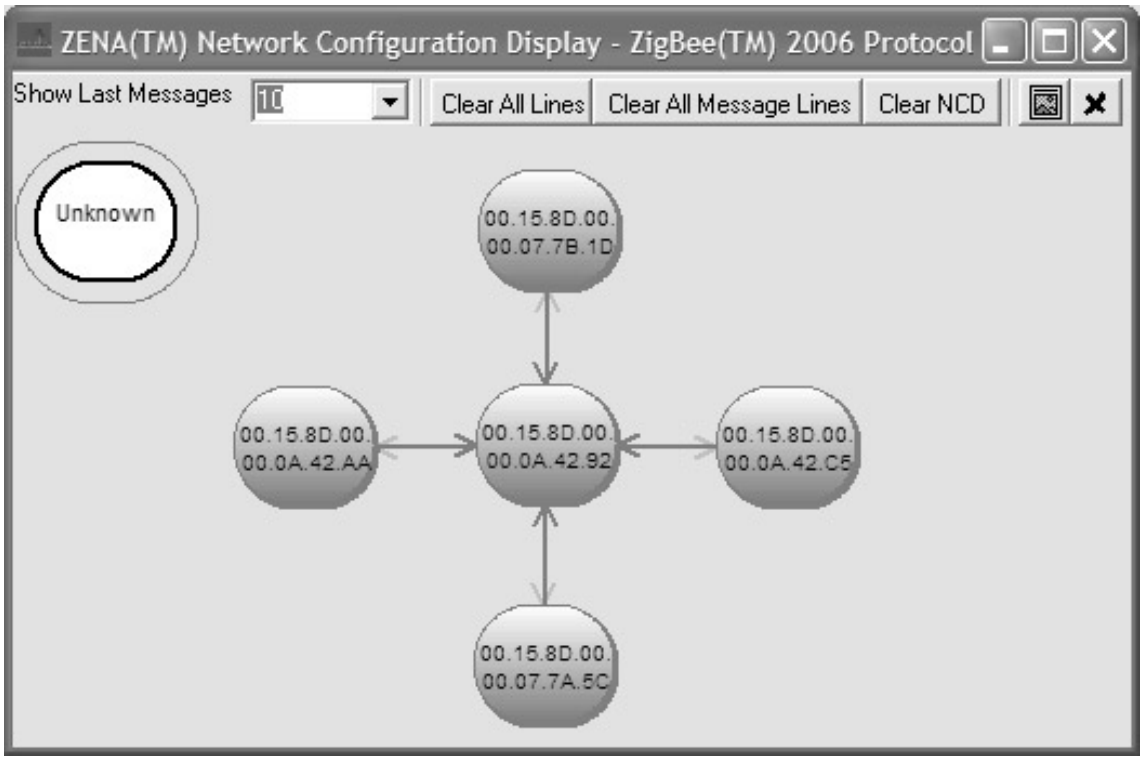
Úroveň dekódování: Verbose

APS Frame Control					Dest EP	Cluster ID	Profile ID	Source EP	APS Counter
Type	Deliv	Mode	Sec	ACK					
DAT	UNI	N/A	N	N	0x40	0x2312	0x4101	0x21	0x01

Obr. 3.5: Úrovně dekódování na vrstvě APS

tovou MAC adresou a barevně podle typu zařízení (koordinátor, směrovač, koncové zařízení). Při najetí myší nad jedno ze zařízení se nám dále zobrazí ještě PAN ID a 16bitová síťová adresa. Analyzátor ZENA dále graficky, pomocí šipek, zobrazuje cestu a směr zpráv v síti. Pokud je z některého ze zařízení odeslána broadcastová zpráva, zobrazí se kolem zdrojového zařízení kruh.

Software také umožňuje vložit do pracovního okna obrázek, na kterém může být zobrazen plán prostoru, ve kterém jsou zařízení umístěna. Pomocí této funkce lze symbolicky pomocí myši přesunout zařízení do místností na obrázku, kde jsou ve skutečnosti umístěna. To umožňuje lepší orientaci v síti.



Obr. 3.6: Grafické zobrazení topologie sítě

4 VÝVOJOVÝ KIT JN5139–EK010 ZIGBEE

JN5139–EK010 ZigBee od společnosti JENNIC poskytuje kompletní prostředí pro rychlý vývoj Zigbee aplikace na bezdrátovém JN5139 mikrokontroléru. Umožňuje vytvořit rozsáhlé sítě typu hvězda strom nebo síť až s 1 000 uzly. Obsahuje jednu řídící jednotku s displejem, čtyři senzorové jednotky a dva High power moduly pro větší dosah [3].

4.1 Řídící jednotka (1x)

Řídící jednotka poskytuje následující funkce umožňující rychlý vývoj ZigBee aplikací [4]:

- ZigBee modul JN5139:
  - citlivost přijímače -96 dBm;
  - vysílací výkon +2,5 dBm;
  - vysílací / přijímací proud 37 mA / 37 mA;
- 128×64 pixel LCD displej;
- senzory pro měření teploty, vlhkosti a osvětlení;
- 5x LED diody (1 indikuje stav on/off, 4 jsou volitelné);
- 6 mikrospínačů (1 pro reset, 1 pro přepnutí do programovacího módu, 4 jsou volitelné);
- sériovou paměť EEPROM 128 kB;
- 2x UART rozhraní pro komunikaci a zápis uživatelských aplikací do paměti EEPROM;
- expansion port pro rozšíření o další senzory;
- bateriové nebo externí napájení.

4.2 Senzorová jednotka (4x)

Senzorová jednotka poskytuje následující funkce umožňující rychlý vývoj Zigbee aplikací [5]:

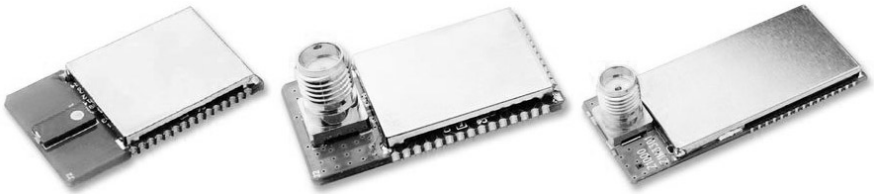
- ZigBee modul JN5139:
  - citlivost přijímače -96 dBm;
  - vysílací výkon +2,5 dBm;
  - vysílací / přijímací proud 37 mA / 37 mA;

- senzory pro měření teploty, vlhkosti a osvětlení;
- 3x LED diody (1 indikuje stav on/off, 2 jsou volitelné);
- 4 mikrospínačů (1 pro reset, 1 pro přepnutí do programovacího módu, 2 jsou volitelné);
- sériovou paměť EEPROM 128 kB;
- 2x UART rozhraní pro komunikaci a zápis uživatelských aplikací do paměti EEPROM;
- expansion port pro rozšíření o další senzory;
- bateriové nebo externí napájení.

4.3 High power modul (2x)

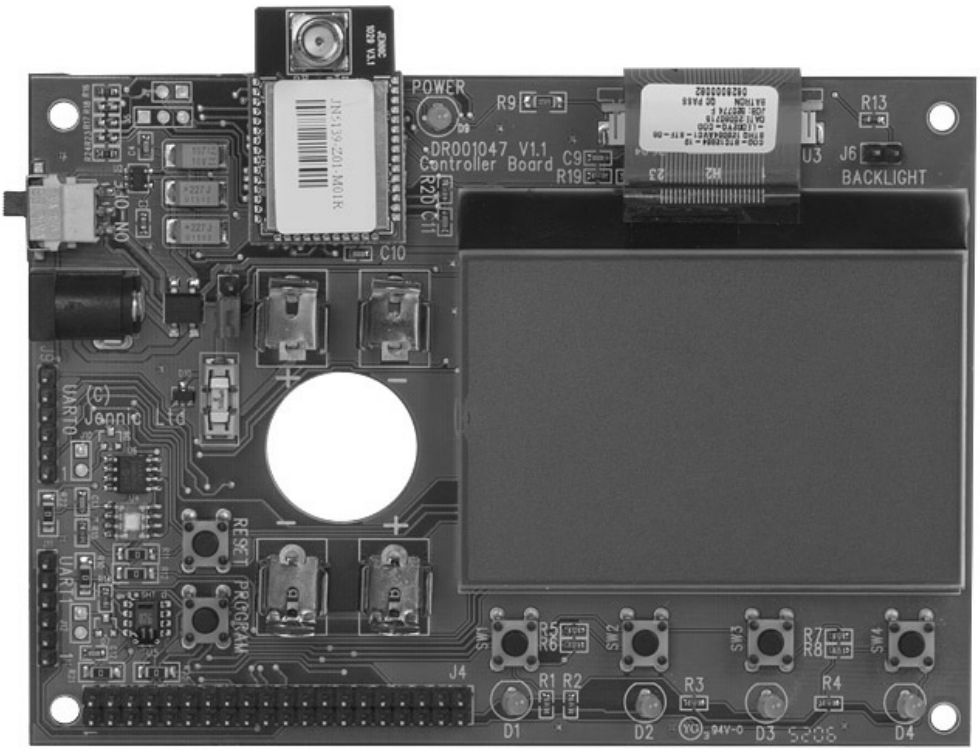
- citlivost přijímače -102 dBm;
- vysílací výkon +17,5 dBm;
- vysílací / přijímací proud 125 mA / 45 mA.

Na obrázku (4.1) jsou vidět jednotlivé typy modulů, včetně High power modulu.

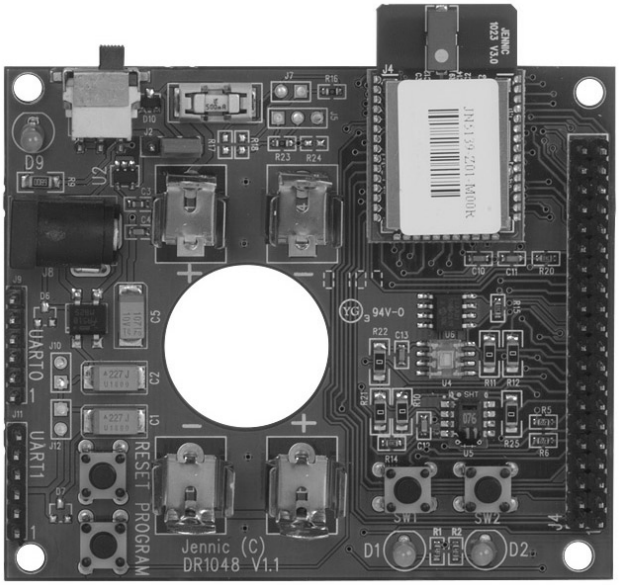


Obr. 4.1: Moduly – M00R1, M01R1 a M02R1 (High power modul)

Na následujících obrázcích jsou pro představu nákresy řídící (Obr. 4.2) a senzorové (Obr. 4.3) jednotky. Bližší informace o nich můžeme nalézt na stránkách výrobce [6]. Volba napájení mezi bateriovým (2x AAA baterie) nebo externím napájením (5 až 7 V DC; 4,5 až 6 V AC) se provádí přepojením jumperu J2 [4] a [5].



Obr. 4.2: Řídící jednotka



Obr. 4.3: Senzorová jednotka

5 VÝVOJ APLIKACÍ ZIGBEE

Vývoj aplikací probíhá pomocí vývojářského balíčku Jennic SDK. Existují dvě verze, jedna je určena pro vývoj aplikací v CLI (Command Line Interface) a druhá v IDE (Integrated Development Environment). V této práci budeme vyvíjet aplikace pouze za pomocí IDE a z tohoto důvodu se budu dále věnovat jen tomuto vývojovému prostředí. Balíček dále obsahuje ještě JENNIC JN51xx kompilátor a Flash programátor.

5.1 Postup vytvoření projektu

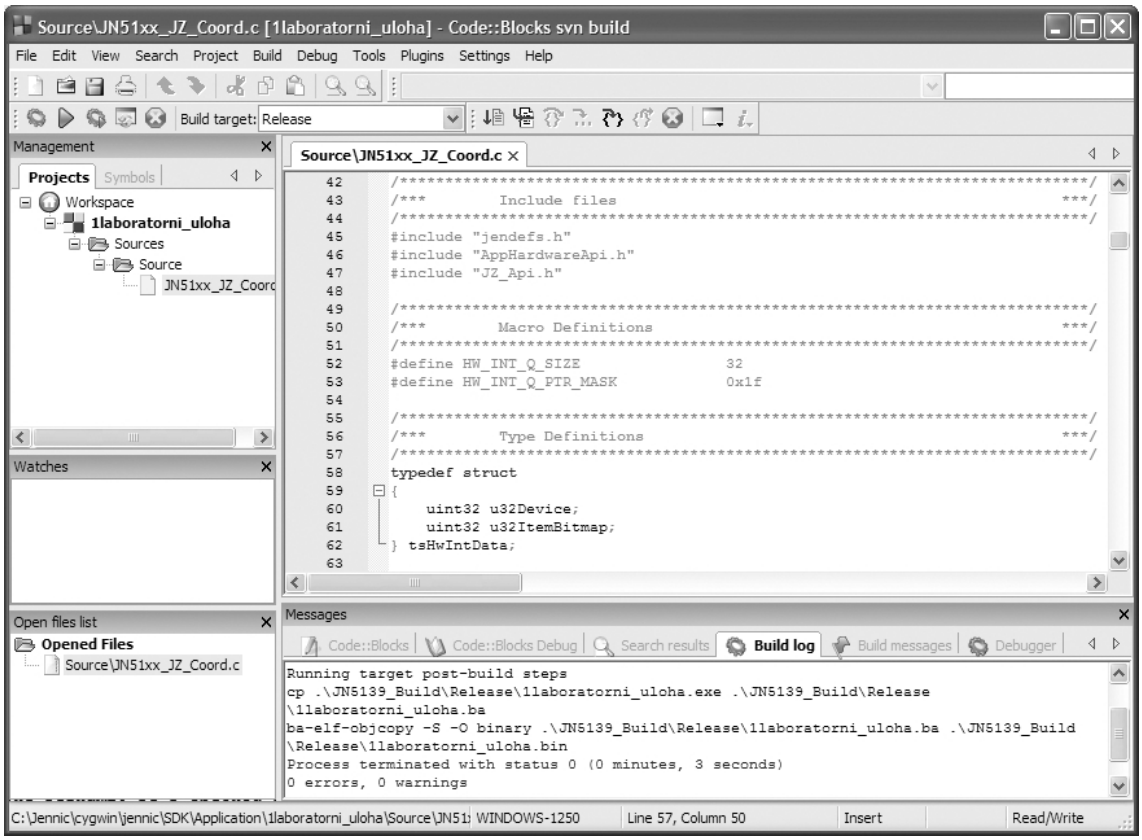
Program budeme psát v jazyku ANSI C za pomoci Code::Blocks IDE (Obr. 5.1). Postup vytvoření nového projektu je následující. V záložce `File` vybereme `New` → `Projekt`, tímto se nám zobrazí okno `New from template`, ve které vybereme šablonu `Jennic`. Opět se nám otevře nové okno, ve kterém zadáme název projektu a jeho umístění. Pro správnou funkci kompilátoru je vhodné ponechat původní nastavení cesty k souboru, a sice `Jennic\cygwin\jennic\SDK\Application\`. Klikneme na `Next` a zvolíme typ kompilátoru, který je v našem případě `JN51xx Compiler`. V dalším okně máme na výběr typ aplikace. Zde vybereme jednu ze tří možností ZigBee podle toho, pro které zařízení chceme aplikaci vyvíjet. Následně zvolíme typ desky `DK2 (JN513x--EK000 or JN513x--EK010)` a mikrokontolér `JN513x`. Tím jsme si vytvořili šablonu, které již obsahuje některé základní knihovny a funkce.

5.2 Popis základních funkcí

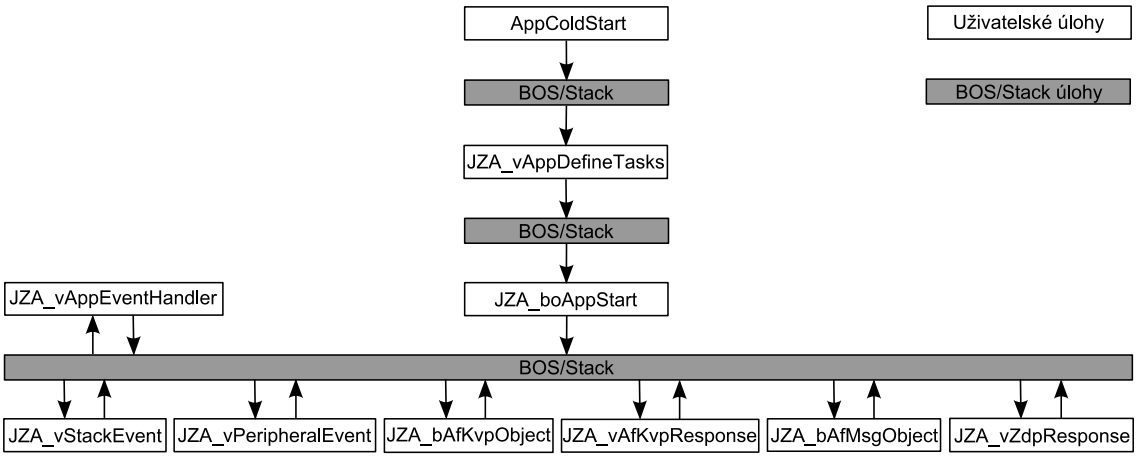
Uživatelské aplikace musí sdílet kontrolu nad procesorem se ZigBee protokol stackem. To je umožněno díky základnímu operačnímu systému (BOS), což je jednoduchý plánovač úloh. Za spuštění BOS i ZigBee protokol stacku je odpovědná uživatelská aplikace. Před samotným programováním je ještě nutné seznámit se s obecnou strukturou základní ZigBee aplikace, kterou si můžete prohlédnout na obrázku (Obr. 5.2), a popsat si jednotlivé funkce.

5.2.1 Inicializace

Po zapnutí zařízení je jako první volána funkce `AppColdStart`. Zde se inicializuje celé zařízení. Pomocí uživatelských aplikací je také možné zavolat další inicializační funkce, například pro inicializaci proměnných nebo systémových periferií jako je časovač a UART. Probíhá zde také nastavení čísla rádiového kanálu a PAN ID sítě. Pro pásmo 2,4 GHz je možné nastavit číslo kanálu na 11–26. V případě že je zadána hodnota 0, automaticky se vybere nejvhodnější kanál.



Obr. 5.1: Aplikace Code::Blocks IDE



Obr. 5.2: Vývojový diagram obecné ZigBee aplikace

Poté je spuštěn BOS, který přebere kontrolu nad systémem. Ten vykoná některé interní funkce a poté zavolá funkci `JZA_vAppDefineTasks`. Je to uživatelská aplikace, ve které je možné doplnit další úlohy, které nejsou součástí BOS. Není to ale nezbytné.

Opět je kontrola vrácena BOS, kde proběhnou potřebné vnitřní funkce a je zavolána poslední inicializační funkce `JZA_boAppStart`. Za pomoci této funkce je možné přiřadit volitelný ZigBee deskriptor k uzlu sítě. Po ukončení je nutné zavolat ZigBee protokol stack a vrátit kontrolu základnímu operačnímu systému.

V základní šabloně vytvořené pomocí Code::Blocks IDE můžeme najít ještě jednu funkci, které zde nebyla probrána. Jedná se o uživatelskou funkci a je volána ve funkci `AppColdStart` hned po nastavení čísla kanálu a přidělení PAN ID a jmenuje se vInit. V této funkci jsou právě inicializované systémové periferie popsané u `AppColdStart` a spuštěn ZigBee protokol stack a BOS [7].

5.2.2 Hlavní aplikace

Jakmile je spuštěn BOS a ZigBee protokol stack, kontrola nad uživatelskou aplikací je předána několika funkcím. Některé z nich v pravidelných intervalech zavolá BOS, jiné jsou spuštěny, jen pokud dojde k určité události. Po ukončení aplikace je kontrola předána vždy základnímu operačnímu systému. Ten také pravidelně předává kontrolu ZigBee protokol stacku, který vykoná veškeré potřebné kroky.

Uživatelské aplikace nejsou blokovány, ale pokud dojde k dlouhodobé nečinnosti způsobené smyčkou nebo čekáním na podmínku, je nutné pro správnou funkci pravidelně volat BOS a ZigBee protokol stack.

Hlavní aplikační funkce jsou [7]:

- `JZA_vAppEventHandler` – v pravidelných intervalech ji volá BOS, můžou v ní být umístěny jakékoliv uživatelské aplikace, které je nutné pravidelně spouštět;
- `JZA_vStackEvent` – zpracovává různé události z nižších vrstev stacku, jako například potvrzení přenosu dat;
- `JZA_vPeripheralEvent` – je volána pouze když dojde k hardwarovému přerušení z periferie, jako je například časovač;
- `JZA_bAfMsgObject` – je spuštěna v případě přijetí MSG rámce (MSG rámec obsahuje jen data a informace o délce datového pole) od jiného uzlu v síti. Může zde být funkce pro zpracování příchozích zpráv;
- `JZA_bAfKvpObject` – je spuštěna, pokud byl přijat KVP příkazový rámec (KVP je podobný rámec jako MSG, jen uvádí další pole např. typ dat) od jiného uzlu. Může zde být funkce pro zpracování příchozího příkazu a následné odpovědi;

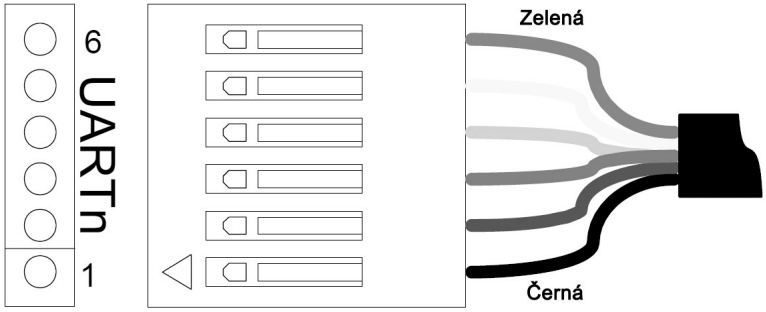
- `JZA_vAfKvpResponse` – je zavolána v případě, že byl přijat rámec odpovědi KVP od jiného uzlu. To může být buď potvrzením předcházejícího příkazového rámce, nebo odpověď na žádaná data;
- `JZA_vZdpResponse` – je volána pouze tehdy, když je přijata odpověď od ZigBee objektu. Například zde může být přijata odezva na žádost o adresu.

5.3 Nahrání programu do paměti zařízení

Po dokončení aplikace je nejprve potřeba v prostředí Code::Blocks IDE zkompilovat projekt na výsledné zdrojové soubory. To provedeme tak, že klikneme na záložku `Build` a následně opět na stejnojmenné tlačítko `Build`. Tím se provede zkompilování projektu. V dialogu `Build log` je naznačen průběh kompilování. Po jeho skončení musí být na konci tohoto dialogu napsáno `0 errors` a nejlépe i `0 warnings`. Pokud se tak stalo, bude vytvořen soubor s koncovkou `.bin`, ke kterému vede obvykle následující cesta:

```
\nazev_projektu\JN5139_Build\Release\nazev_projektu.bin
```

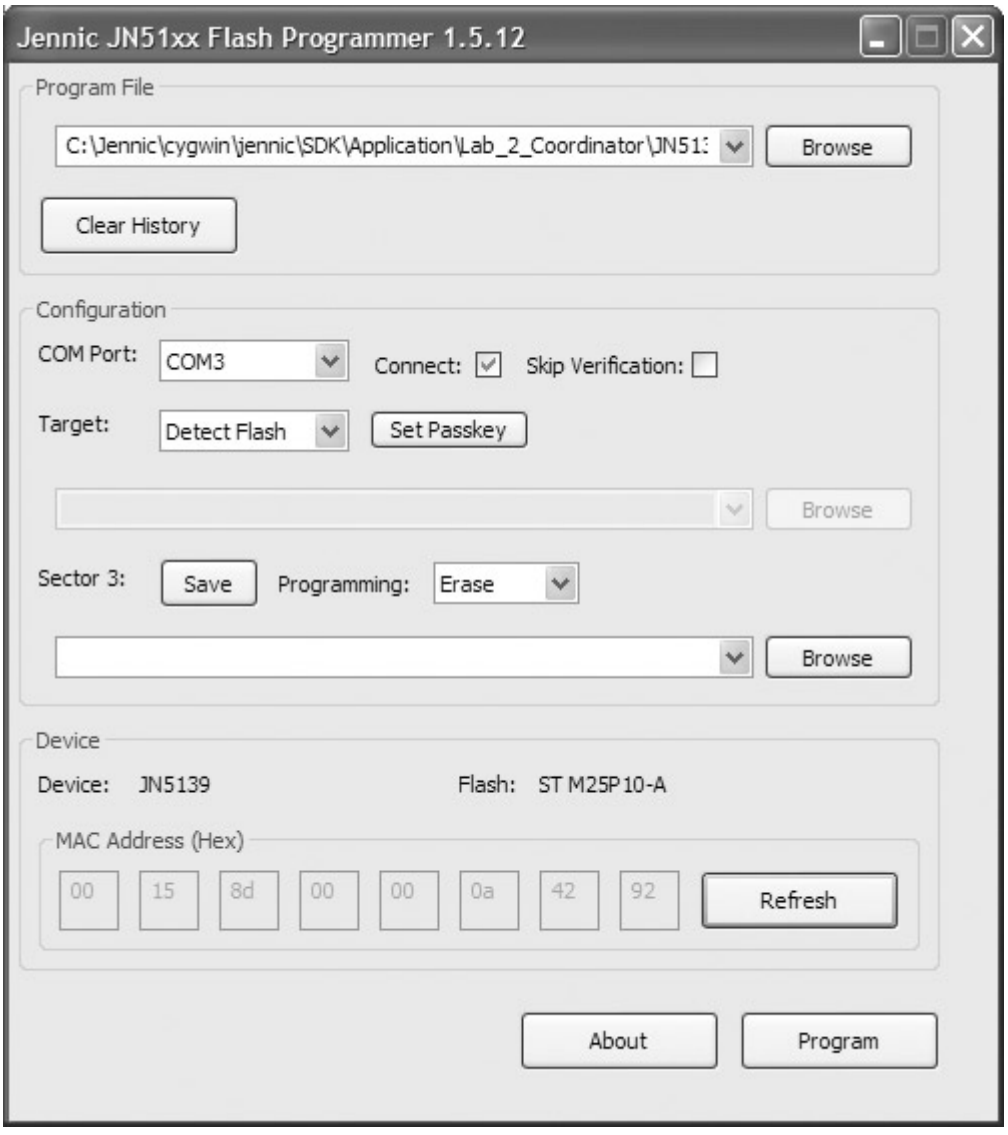
Následně je potřeba spustit program Jennic Flash Programmer a připojit zařízení na které chceme nahrávat pomocí USB kabelu k počítači. Kabel má z jedné strany konektor pro připojení k UART portu. Při připojování k UART portu musíme dbát na správnou orientaci kabelu. Na pinu číslo 1 na desce bude připojen černý drát z USB kabelu (Obr. 5.3). Následně zapneme zařízení. Poté je potřeba zmáčknout a držet tlačítko `PROGRAM`, mezi tím zmáčknout a pustit tlačítko `RESET` a následně pustit tlačítko `PROGRAM`. Tím je zařízení připraveno k nahrávání do paměti.



Obr. 5.3: Připojení USB kabelu k portu UART

V programu Jennic Flash Programmer (Obr. 5.4) vybereme v části `Program File` cestu k souboru s koncovkou `.bin`. V další části `Configuration` vybereme

port, ke kterému je USB kabel připojen a zbytek necháme v původním stavu. V části `Device` je možné kliknutím na tlačítko `Refresh` zkontrolovat připravenost zařízení k nahrávání do paměti. Pokud je zařízení připraveno, zobrazí se název zařízení, typ paměti a MAC adresa. Pro provedení nahrání programu do paměti klikneme na tlačítko `Program`. Po úspěšném nahrání programu do paměti, je ještě potřeba celé zařízení restartovat tlačítkem `RESET`. Nyní již běží na zařízení nový program.



Obr. 5.4: Program Jennic Flash Programmer

6 POPIS PRVNÍ LABORATORNÍ ÚLOHY

V první laboratorní úloze se studenti seznámí s parametry daného standardu, jako jsou frekvenční pásma, přenosové rychlosti, dále pak s typy sítí a zařízení. V praktickém měření pak budou mít za úkol ověřit dosah v uzavřených prostorách školy, kvalitu spojení v závislosti na okolních podmínkách, jako je zejména rušení jinými technologiemi, pracujícími ve stejném frekvenčním pásmu a změřit spotřebu různých typů zařízení a následně ji pak porovnat se známými hodnotami spotřeby jiných typů technologií.

6.1 Návrh řešení

Při řešení této úlohy bude třeba vytvořit programy pro koordinátora sítě a koncové zařízení. Vytvoření programu pro směrovač nebude třeba, protože by byl v této laboratorní úloze stejný, jako program pro koncové zařízení. Dále je z důvodu zjištění dosahu a kvality spojení v závislosti na okolních podmínkách nutné, aby bylo možné ručně nastavit frekvenci vysílání, a ještě aby koncové zařízení a směrovač posílaly informace o kvalitě spojení, které se budou zobrazovat na LCD displeji koordinátora sítě.

Z časových a prostorových důvodů bude dostačující použití topologie sítě typu hvězda. Výhodněji se jeví topologie typu síť či strom, avšak z důvodu časově omezené výuky a velikosti prostor není nutné, abychom zvětšovali dosah sítě za pomoci směrovačů. Pro zjištění dosahu budou použita dvě zařízení, a to koordinátor sítě a koncové zařízení. Koncové zařízení bude ponecháno v laboratorní místnosti a s koordinátorem sítě se budeme pomalu vzdalovat, dokud se připojení úplně neztratí. Podle plánů budovy se pak odhadne vzdálenost mezi koordinátorem sítě a koncovým zařízením.

Pro rušení na stejném frekvenčním pásmu bude použita, z důvodu dostupnosti ve škole, technologie Wifi. Při tomto testu se nastaví stejné frekvenční pásmo, jak na ZigBee, tak na Wifi zařízení. Aby byl zajištěn reálný provoz, musíme ještě zatížit Wifi síť. To bude zajištěno přenosem dat z jednoho počítače do druhého. Na LCD displeji budeme sledovat kvalitu spojení ZigBee sítě. Na počítači budeme sledovat rychlost přenosu dat, popřípadě počet ztracených paketů.

V závěrečném testu studenti zjistí spotřebu koncového zařízení. Digitálním osciloskopem nebo z teoretických hodnot se zjísti spotřeba a porovná s dalšími typy standardů (Wifi, Bluetooth). Zařízení umožňuje také režim snížené spotřeby, ve kterém může zůstat v nečinnosti až 15 minut a tím ještě snížit spotřebu. S tímto režimem budou studenti seznámeni jen teoreticky.

6.2 Programy pro moduly Jennic

Programy budou psány v již popsaném vývojovém prostředí Code::Blocks IDE (kapitola 5). Po vytvoření nového projektu se vygeneruje základní šablona. Ta je rozdílná pro různé typy zařízení, proto je důležité zvolit při vytváření správný typ. Do této šablony budeme dopisovat veškeré vlastní funkce.

Šablonu musíme rozšířit tak, aby se na koordinátorovi sítě dal nastavit kanál. Dále aby bylo možné ke koordinátorovi připojit všechny zařízení z vývojového kitu a mohly spolu komunikovat zároveň.

6.2.1 Koordinátor sítě

Před vlastním vytvářením funkcí je dobré vložit do programu odkazy na soubory, ve kterých jsou vloženy předpřipravené aplikace. V našem případě je potřeba vložit navíc tyto soubory:

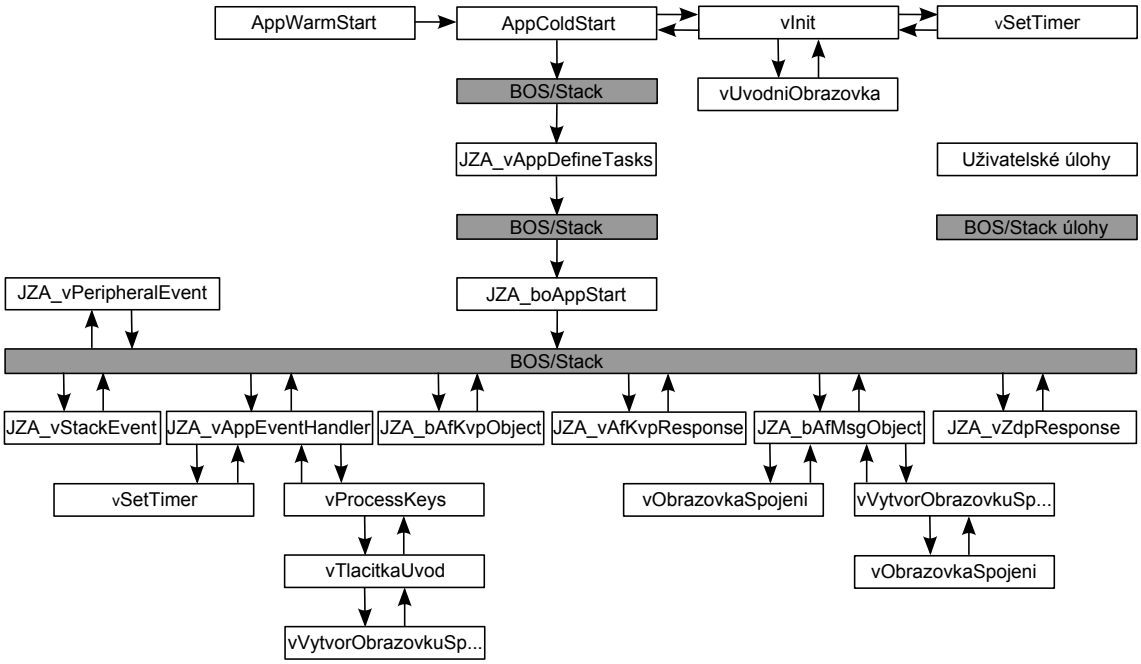
- `gdb.h` – knihovna pro debuger;
- `LcdDriver.h` – knihovna, ve které jsou umístěny aplikace pro ovládání LCD displeje;
- `LedControl.h` – knihovna pro ovládání LED diod;
- `Button.h` – aplikace pro zachytávání zmáčknutí tlačítek;
- `header.h` – vlastní knihovna s předdefinovanými proměnnými.

Následně si vytvořit datové struktury, do kterých bude možné uložit všechny data a také stavy zařízení:

- `teTlacitka` – datová struktura, která přiřadí názvy jednotlivým tlačítkům;
- `teObrazovka` – zde budou uloženy informace o tom, v jakém okně se právě nacházíme;
- `tsDataKZ` – sem se budou ukládat veškeré hodnoty z koncových zařízení;
- `tsSystem` – v této struktuře budou uloženy systémová data;
- `tsFronta` – struktura s frontou pro hardwarové přerušení.

Nyní již můžeme začít psát vlastní funkce. Na obrázku (Obr. 6.1) vidíme vývojový diagram hotového programu, který si teď popíšeme. Po zapnutí zařízení postupně probíhají tyto funkce:

- `AppWarmStart` – tato funkce je volána při zmáčknutí tlačítka reset. Můžou se v ní objevit speciální nastavení pro restart. Zde to však není potřeba, proto zavolá `AppColdStart`;



Obr. 6.1: Vývojový diagram koordinátora sítě v úloze č. 1

- `AppColdStart` – je volána po zapnutí zařízení nebo popřípadě z aplikace `AppWarmStart`. Probíhají zde inicializace funkcí pro debuger a jsou nastavovány informace o síti. Kanál je nastaven na hodnotu proměnné `KANAL`, nastavené v knihovně `header.h`. Stejně tak je nastaveno PAN_ID podle proměnné `PAN_ID`. Následně ještě volá funkci `vInit` a poté předá řízení do `BOS/Stack`;
- `vInit` – provádí inicializaci Zigbee stacku, LED diod, tlačítek, nastavení výchozích hodnot všech proměnných a povoluje přerušení po vypršení časovače. Dále volá `vSetTimer`. Poté vytvoří úvodní obrazovku a spustí `vUvodniObrazovka` a následně BOS;
- `vSetTimer` – spouští časovač, který po vypršení vyvolá přerušení a následně se opět spustí;
- `vUvodniObrazovka` – je volána při inicializaci, kdy vypíše na displej právě nastavený kanál. Může být také zavolána z funkce `vTlacitkaUvod` při změně nastaveného kanálu;
- `JZA_vAppDefineTasks` – je volána z `BOS/Stack`. Zde můžou být definovány další uživatelské úlohy, což však není v naší úloze potřeba;
- `JZA_boAppStart` – také je zavolána z `BOS/Stack`. Zde je vytvořen jednoduchý testovací časovač, který problikne LED diodou. Touto funkcí končí proces inicializace zařízení. Kontrola je předána do `BOS/Stack`, který v pravidelných

intervalech volá další funkce;

- `JZA_vPeripheralEvent` – je volána pouze když dojde k hardwarovému přerušení z periferie, jako je například časovač. Do proměnné `sFronta` zapíše typ přerušení, podle kterého je ve funkci `JZA_vAppEventHandler` provedena určitá operace;
- `JZA_vAppEventHandler` – v pravidelných intervalech ji volá BOS, můžou v ní být umístěny jakékoliv uživatelské aplikace, které je nutné pravidelně spouštět. Obsluhuje také frontu událostí z proměnné `sFronta`. V našem případě se v ní opět spouští časovač a kontroluje se zde, jestli nebylo zmáčknuto některé z tlačítek. Pokud ano, tak spustí funkci `vProcessKeys` a předá jí identifikátor zmáčknutého tlačítka;
- `vProcessKeys` – v této funkci je přeložen identifikátor zmáčknutého tlačítka na jméno definované v proměnné `eTlacitka`. Následně je vyhodnoceno, v jakém stavu se nachází program (podle proměnné `eObrazovka`) a spuštěna odpovídající funkce (v našem případě `vTlacitkaUvod`);
- `vTlacitkaUvod` – zde je již podle názvu tlačítka provedena další operace. První tlačítko vyobrazí na displeji doplňující informace. Další dvě tlačítka slouží ke změně hodnoty kanálu. Poslední z tlačítek spustí síť a zavolá funkci `vVytvorObrazovkuSpojeni`;
- `vVytvorObrazovkuSpojeni` – tato funkce zkontroluje počet připojených koncových zařízení a vytvoří pole pro doplnění hodnot (teplota, LQI, osvětlení a napětí na bateriích). Následně zavolá funkci `vObrazovkaSpojeni`;
- `vObrazovkaSpojeni` – doplní do pole vyobrazeném na LCD displeji hodnoty, které byly přijaty z koncových zařízení;
- `JZA_vStackEvent` – je volána po vytvoření sítě a vytváří pomocí funkce `vAddDesc` jednoduchý deskriptor, pro kontrolu a nastavení hodnot při přijímání a odesílání dat;
- `JZA_bAfMsgObject` – je spuštěna v případě přijetí MSG rámce (MSG rámec obsahuje jen data a informace o délce datového pole) od jiného uzlu v síti. V našem případě jsou přes tyto MSG rámce posílána všechna data. Nejprve je zkontrolováno, jestli číslo cílového uzlu sedí s tím, který je nastaven v rámci. Pokud ano, uloží se všechny hodnoty do proměnných. Poté je zkontrolováno podle MAC adresy, jestli již bylo od stejného uzlu něco přijato. Pokud ano, přepíší se v proměnné `sDataKZ` nově přijata data za starší, které pomocí funkce `vObrazovkaSpojeni` zobrazí na displeji. Jestliže koordinátor nezná MAC adresu, uloží si ji do proměnné `sDataKZ` i spolu s dalšími hodnotami a poté po-

mocí funkcí `vVytvorObrazovkuSpojeni` a `vObrazovkaSpojeni` vytvoří nové podle na displeji a zapíše do něj přijaté hodnoty;

- `vStringCopy`, `vValToDec`, `vValu16ToDec`, `vVali16ToDec` – před vyobrazením hodnot na LCD displeji musejí být nejprve převedeny do správného tvaru. K tomuto účelu právě slouží tyto 4 funkce;
- `JZA_bAfKvpObject`, `JZA_vAfKvpResponse` a `JZA_vZdpResponse` – nejsou využity.

6.2.2 Koncové zařízení

Do programu, stejně jako v případě koordinátora sítě, vložíme odkazy na soubory, ve kterých jsou vloženy předpřipravené aplikace. V tomto případě vložíme navíc tyto soubory:

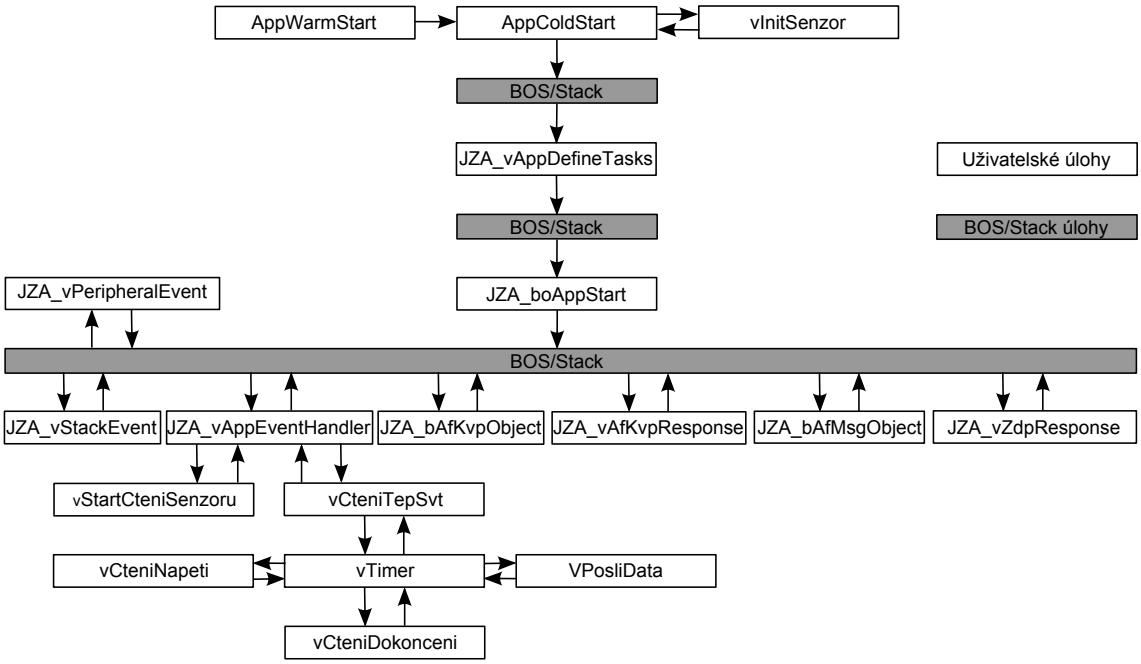
- `gdb.h` – knihovna pro debuger;
- `LedControl.h` – knihovna pro ovládání LED diod;
- `Button.h` – aplikace pro zachytávání zmáčknutí tlačítek;
- `HTSdriver.h` – knihovna pro senzor teploty a vlhkosti;
- `ALSdriver.h` – knihovna s funkcemi pro světelný senzor;
- `header.h` – vlastní knihovna s předdefinovanými proměnnými.

Datové struktury pro uložení veškerých proměnných:

- `teTlacitka` – datová struktura, která přiřadí názvy jednotlivým tlačítkům;
- `teStav` – zde budou uloženy informace o stavu měření na senzorech;
- `tsHodnoty` – sem se budou ukládat veškeré naměřené hodnoty;
- `tsSystem` – v této struktuře budou uloženy systémová data;
- `tsFronta` – struktura s frontou pro hardwarové přerušení.

Nyní již opět můžeme začít psát vlastní funkce. Na obrázku (Obr. 6.2) vidíme vývojový diagram programu koncového zařízení. Po zapnutí postupně probíhají tyto funkce:

- `AppWarmStart` – stejná funkce, jako v případě koordinátora sítě. Zde také není potřeba, proto zavolá `AppColdStart`;
- `AppColdStart` – je volána po zapnutí zařízení nebo popřípadě z aplikace `AppWarmStart`. Probíhají zde inicializace funkcí pro debuger a jsou nastavovány informace o síti. Kanál je nastaven na hodnotu 0, což znamená, že bude



Obr. 6.2: Vývojový diagram koncového zařízení v úloze č. 1

vybrán ten, který má nejsilnější signál. PAN_ID je nastaveno podle proměnné `PAN_ID`. Tato hodnota musí být stejná, jako v případě koordinátora sítě. Následně ještě volá funkci `vInit` a poté předá řízení do `BOS/Stack`;

- `vInit` – provádí inicializaci Zigbee stacku, LED diod, tlačítek, časovače, nastavení výchozích hodnot všech proměnných a povoluje přerušení po vypršení časovače a po stisknutí tlačítka. Dále volá `vInitSensor` a poté spustí BOS;
- `vInitSensor` – zde jsou inicializovány a povoleny veškeré funkce a přerušení, které jsou potřeba pro měření na senzorech;
- `JZA_vAppDefineTasks` – stejně jako v případě koncového zařízení, zde není potřebná;
- `JZA_boAppStart` – je zavolána z `BOS/Stack`. Zde je vytvořen jednoduchý testovací časovač, který problikne LED diodou. Touto funkcí končí proces inicializace zařízení. Kontrola je předána do `BOS/Stack`, který v pravidelných intervalech volá další funkce;
- `JZA_vStackEvent` – je volána po připojení do sítě. Kontroluje, zda se zařízení připojilo jako koncové a vytváří pomocí funkce `vAddDesc` jednoduchý deskriptor, pro kontrolu a nastavení hodnot při přijímání a odesílání dat;
- `JZA_vPeripheralEvent` – funguje stejně, jako v případě koordinátora sítě;

- `JZA_vAppEventHandler` – v pravidelných intervalech ji volá BOS, můžou v ní být umístěny jakékoliv uživatelské aplikace, které je nutné pravidelně spouštět. Obsluhuje také frontu událostí z proměnné `sFronta`. V našem případě může být spuštěna při vypršení časovače nebo při přerušení z některého senzoru při měření. V prvním případě spustí funkci `vStartCteniSenzoru`. V druhém případě po přerušení ze senzoru spustí funkci `vCteniTepSvt`;
- `vStartCteniSenzoru` – první funkce z pěti, kterou je třeba ke čtení hodnot ze senzorů. Spouští čtení teploty a startuje přerušení v případě, kdy je čtení teploty dokončeno;
- `vCteniTepSvt` – tato funkce je zavolána po skončení čtení teploty. Uloží si její hodnotu do paměti, vymaže předchozí přerušení a uloží si do paměti hodnotu ze senzoru osvětlení. Dále spouští čtení napětí a zároveň s tím spouští časovač `vTimer`;
- `vTimer` – tato funkce je volána po vypršení časovače `vTimer`. Podle hodnoty proměnné `eStav` spouští jednu ze tří funkcí (`vCteniNapeti`, `vCteniDokonceni`, `vPosliData`;
- `vCteniNapeti` – tato funkce je volána tak dlouho, dokud není ukončena konverze v ADC;
- `vCteniDokonceni` – po ukončení konverze je volána tato funkce, ve které je hodnota s ADC konvertoru upravena do výsledného tvaru a uložena;
- `vPosliData` – je poslední v řetězci funkcí z `vTimer`. Nejprve zjistí vlastní MAC adresu a tím zkompletuje všechna data, která je třeba odeslat do koordinátora sítě. Data jsou odesílána ve tvaru `Napeti//Teplota//Osvetleni//MAC adresa`. Spolu s těmito hodnotami je také odesílána délka MSG rámce. Po odeslání dat, zařízení čeká na přerušení z časovače a ten opět spustí celý řetězec čtení ze senzorů znovu;
- `JZA_bAfMsgObject`, `JZA_bAfKvpObject`, `JZA_vAfKvpResponse` a `JZA_vZdpResponse` – nejsou využity.

6.3 Postup a výsledky

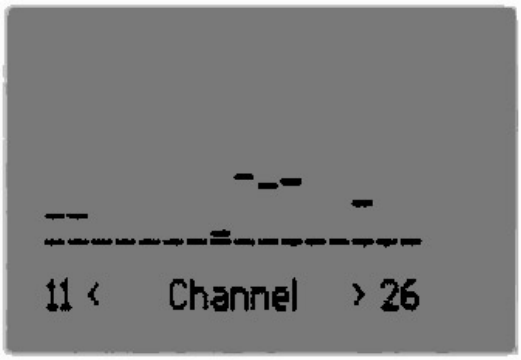
Do všech ZigBee zařízení, které budeme používat, musíme nejdříve nahrát již předpřipravené programy. Ty budeme do zařízení nahrávat pomocí aplikace Jennic Flash Programmer, popsané v kapitole (5.3).

6.3.1 Kvalita spojení

Při měření kvality spojení musíme nejdříve prozkoumat obsazení kmitočtového pásma. K tomuto úkolu využijeme aplikaci LCD Site Survey Tool, která nám na LCD displeji zobrazuje aktivitu v celém používaném frekvenčním pásmu. V aplikaci Jennic Flash Programmer zvolíme cestu k souboru `LCDSiteSurveyTool_JN5139_DEVKIT2.bin`, která je:

```
\Jennic\cygwin\jennic\SDK\Application\JN-AP-1014-Site-Survey-Tool\Build\
```

K počítači připojíme zařízení s LCD displejem a nahrajeme do něj tento program. Po spuštění zařízení se na LCD displeji zobrazí aktivita ve frekvenčním pásmu od 2 380 MHz do 2 505 MHz. Pomocí tlačítka č. 1 můžeme toto zobrazení přepnout zobrazení aktivity na jednotlivých kanálech 11–26. Pokud zmáčkneme tlačítko č. 1 znovu, zobrazí se nám aktivita pouze na jednom kanále. Ten vybíráme pomocí tlačítka č. 2. K prozkoumání aktivity využijeme obrazovku se zobrazením všech kanálů (Obr. 6.3) a poznamenáme si 3 kanály. Jeden s největší aktivitou, jeden s nejmenší aktivitou a třetí se střední aktivitou. Kanály musejí být z rozsahu 12–24, protože rozsah pásma ZigBee sítě je jiný, než rozsah pásma WiFi sítě.



Obr. 6.3: LCD Site Survey Tool

Nyní do koordinátora sítě a do koncového zařízení již nahrajeme programy popsané výše. Ty najdeme v následujících třech cestách:

```
\Jennic\cygwin\jennic\SDK\Application\Lab_1_Coordinator\JN5139_Build\Release\
\Jennic\cygwin\jennic\SDK\Application\Lab_1_EndDevice\JN5139_Build\Release\
\Jennic\cygwin\jennic\SDK\Application\Lab_1_EndDeviceHighPower\JN5139_Build\...
```

Programy do zařízení nahrajeme opět pomocí aplikace Jennic Flash Programmer. Do zařízení s High Power modulem (M02R1) musíme nahrát upravený program

Obr. 6.4: LCD Obrazovka - stav spojeni

přímo pro něj. Poté zapneme koordinátora sítě a zvolíme pomocí tlačítek 2 a 3 jeden ze zvolených kanálů. Volbu potvrdíme pomocí tlačítka č. 4 a tím zapneme síť.

Nyní koordinátor čeká na připojení koncových zařízení. Zapneme postupně 3 koncové zařízení s různými moduly (jiný typ antény – M00R1, M01R1 a M02R1) a zapíšeme si hodnoty LQI (Obr. 6.4). Poté vypneme koncová zařízení, restartujeme koordinátora sítě a zvolíme další ze zvolených kanálů, potvrdíme a opět zapneme koncová zařízení. Opět si opíšeme hodnotu LQI a to samé opakujeme i pro poslední kanál. Poté všechny zařízení vypneme.

Tab. 6.1: Frekvenční pásma používané v ZigBee a WiFi

ZigBee		**WiFi**	
Kanál	Frekvence [MHz]	Kanál	Frekvence [MHz]
11	2 405	1	2 412
12	2 410	2	2 417
13	2 415	3	2 422
14	2 420	4	2 427
15	2 425	5	2 432
16	2 430	6	2 437
17	2 435	7	2 442
18	2 440	8	2 447
19	2 445	9	2 452
20	2 450	10	2 457
21	2 455	11	2 462
22	2 460	12	2 467
23	2 465	13	2 472
24	2 470	14	2 484
25	2 475	-	-
26	2 480	-	-

V tabulce (6.1) pak najdeme ke zvoleným ZigBee kanálům adekvátní, dle frekvence co nejbližší, WiFi kanály. U WiFi nelze vybrat 14. kanál, protože jej není možno používat v Evropě. Nyní na WiFi zařízení nastavíme jeden z kanálů a spustíme přenos souboru z jednoho počítače do druhého. Zapíšeme si rychlost přenosu bez spuštěných ZigBee zařízení. Následně spustíme ZigBee síť s adekvátním kanálem a zapíšeme si hodnoty LQI z koncových zařízení a rychlost přenos WiFi sítě. Stejný postup zopakujeme opět pro další zvolené kanály. Vzorové naměřené hodnoty pro místnost s jednou WiFi sítí v dosahu jsou v tabulce (6.2). K přepočítání hodnoty LQI na výkon v dBm pak slouží vzorec

$$P = (LQI - 305)/3.$$

Přepočítaná hodnota výkonu je však spíše orientační, jelikož hodnota LQI zahrnuje i jiné proměnné.

Tab. 6.2: Kvalita spojení v ZigBee síti

ZigBee						**WiFi**	
Kanál	**Zařízení**	**LQI**	**P [dBm]**	**LQI**	**P [dBm]**	**Kanál**	**Rychlost**
		WiFi vypnuté		WiFi zapnuté			
14	M00R1	132	-57,66	126	-59,66	3	2 650 kB/s
	M01R1	180	-41,66	180	-41,66		
	M02R1	214	-30,33	208	-32,33		
18	M00R1	126	-59,66	120	-61,66	7	2 640 kB/s
	M01R1	180	-41,66	180	-41,66		
	M02R1	210	-31,66	204	-33,66		
22	M00R1	120	-61,66	120	-61,66	14	2 590 kB/s
	M01R1	180	-41,66	180	-41,66		
	M02R1	204	-33,66	204	-33,66		

Z tabulky (6.2) můžeme vypozorovat, že jedna WiFi síť na stejné frekvenci nemá žádný vliv na ZigBee síť, stejně tak je tomu i naopak.

6.3.2 Dosah

Dosah ZigBee sítě budeme měřit pomocí stejných programů, jako kvalitu spojení. Proto není nutné do zařízení nahrávat jiný program. Opět využijeme 3 různá koncová zařízení a jednoho koordinátora sítě. Vybereme si pásmo, na kterém není příliš velká aktivita a nastavíme ho na koordinátorovi sítě. Nyní zapneme i koncová zařízení a budeme si zapisovat hodnoty LQI pro jednotlivé vzdálenosti. Vzdalovat se budeme s koordinátorem sítě a koncová zařízení zůstanou na místě. Do tabulky pak zapíšeme

jako první hodnotu pro nulovou vzdálenost a jako poslední hodnotu, která byla naměřena jako poslední před ztracením signálu. Mezi tyto hodnoty zapíšeme další 3 vzdálenosti s hodnotou LQI. V tabulce (6.3) pak jsou vzorové naměřené hodnoty pro otevřené prostranství. Z praktického měření vyplývá, že na dosah má velký vliv přímá viditelnost, pokud se něco dostane do cesty signálu (třeba jen strom), dosah se velmi sníží.

Tab. 6.3: Dosah ZigBee sítě

Zařízení	LQI	dBm	LQI	dBm	LQI	dBm	LQI	dBm	LQI	dBm
M00R1	150	-51,7	90	-71,7	54	-83,7	36	-89,7	6	-99,7
Vzdálenost	1 m		10 m		30 m		50 m		80 m	
M01R1	114	-63,7	90	-71,7	66	-79,7	48	-85,7	6	-99,7
Vzdálenost	1 m		30 m		70 m		100 m		140 m	
M02R1	240	-21,7	84	-73,7	60	-81,7	42	-87,7	12	-97,7
Vzdálenost	1 m		150 m		230 m		350 m		500 m	

6.3.3 Spotřeba

Pro koordinátora sítě a směrovač je doporučeno používat napájení ze sítě. Hlavním důvodem je hlavně nutnost permanentního běhu zařízení a tím stálé spotřeby. Tyto zařízení musejí být stále spuštěna, protože s nimi můžou komunikovat i jiná zařízení. V případě koncového zařízení je situace jiná. Tyto zařízení mohou být na delší časové úseky uspávána a tím snižována spotřeba na minimum.

Máme dvě základní možnosti jak získat výslednou spotřebu. První z nich je změření pomocí digitálního osciloskopu, kde budeme měřit úbytek napětí na odporu zařazeném do napájení. Pokud zvolíme odpor $1\,\Omega$, bude hodnota napětí odpovídat hodnotě proudového odběru (např. $10\,\mathrm{mV} = 10\,\mathrm{mA}$). Výslednou spotřebu pak vypočítáme jako integrál proudu v čase. Druhá z možností je využít hodnot proudových odběrů a časů uváděných výrobcem. Výrobce udávané proudové odběry, včetně předpokládaného časového intervalu, jsou v první části tabulky (6.4).

Pokud budeme předpokládat, že zařízení se bude každých 10 minut probouzet a vysílat naměřené data, dojdeme k tomu, že průměrný odběr je $2{,}031\,\mu\mathrm{A} \,/\, 1\,\mathrm{ms}$. Výpočty pro 10minutový cyklus jsou v druhé části tabulky (6.4).

Hodnotu jednoho 10minutového cyklu $1\,218,463\,\mu\mathrm{C}$ pak můžeme vyjádřit v mAh. Jednotka μC je vlastně složená jednotka mA×ms. Proto když hodnotu náboje převedeme do hodin ($1\,218{,}463\,\mu\mathrm{C} \,/\, 1\,000 \,/\, 3\,600$), získáme hodnotu $0,33846 \times 10^{-3}\,\mathrm{mAh}$, což je $1{,}4621\,\mathrm{mAh}$ za měsíc. Z té pak již snad můžeme vypočítat výdrž zařízení na baterie.

Tab. 6.4: Spotřeba koncového zařízení za 10minutový cyklus

Typ odběru	**I [mA] × t [ms]**	**Náboj [μC]**
vysílání	$37 \times 2,5$	$92,5$
přijímání	37×1	37
probouzení	9×16	144
senzory	9×5	45
sleep mód	$0,0015 \times 599\,975,5$	$899,9633$
	Celkově:	$1\,218,463$

Zvolíme si kapacitu baterií 1 500 mAh. Také musíme počítat se samovolným vybíjením baterií. Budeme počítat s 0,2 % z celkové kapacity, což je 3 mAh. Vývoj kapacity baterií si můžete prohlédnout v tabulce (6.5).

Tab. 6.5: Vývoj kapacity baterií

Počet roků	1	3	5	10	15	20	25
Kapacita [mAh]	1 446,5	1 339,4	1 232,3	964,5	696,8	429,1	161,4

Tyto hodnoty jsou však pouze teoretické a v praxi by se výdrž pohybovala spíše kolem 10 roků. Je to způsobeno zejména nedokonalostí dnešních baterií a také jiným proudovým odběrem zařízení, například z důvodu jiné provozní teploty, než je ideální.

Pokud by jsme uvažovali, že zařízení nebude ve sleep módu 10 minut, ale 10 vteřin, zvýšila by se spotřeba za měsíc na 24 mAh a tím se snížila výdrž na 4 roky.

Pro maximální čas ve sleep módu (15 minut) je spotřeba za měsíc 1,3349 mAh a tím pádem výdrž na baterie teoreticky až 28 roků.

7 POPIS DRUHÉ LABORATORNÍ ÚLOHY

Druhá laboratorní úloha bude seznamovat zejména s programováním. Studenti se seznámí s vývojovým prostředím Code::Blocks IDE, se základními programovými funkcemi (kapitola 5.2) a za pomocí návodu se pokusí rozšířit aplikace pro koordinátora sítě a koncové zařízení tak, aby spolu dokázali komunikovat a vytvořili síť pro automatizované ovládání domácnosti.

7.1 Návrh řešení

Jelikož senzorové jednotky můžou měřit teplotu a osvětlení, můžeme v této úloze ovládat klimatizaci, topení a osvětlení. Koncové zařízení bude v tomto případě plnit úlohu senzoru a automatizovaného ovládání a pomocí koordinátora sítě bude možné ovládat zařízení na dálku. Bude také poskytovat informace o hodnotách naměřených senzory.

Pro tuto síť, ve které pro jednoduchost použijeme opět topologii typu hvězda, bude potřeba napsat aplikace pro koordinátora sítě a koncové zařízení. Koordinátor sítě bude sdružovat veškeré informace do jednoho místa a podle nich se uživatel bude moci rozhodnout, jakým způsobem bude zařízení (např. klimatizace) ovládána. Budeme počítat se 3 základními stavy, a to zapnuto / vypnuto / automaticky. V režimu automaticky přebere kontrolu nad ovládáním koncové zařízení. V případě klimatizace ji bude podle teploty zapínat či vypínat. To samé se bude dít s osvětlením.

Není reálné, abychom v této laboratorní úloze ovládali skutečné osvětlení či klimatizaci, proto budou funkci zapnuto / vypnuto signalizovat diody na koncovém zařízení. Pro jednoduchost také vypustím ovládání topení, tak nám zůstane jen ovládání osvětlení a klimatizace, na což nám budou stačit dvě volitelné diody na koncovém zařízení.

Jelikož napsání takovéhoto programu je časově náročné, můžeme využít aplikaci z předchozí laboratorní úlohy a postupně ji rozšířit tak, aby vše fungovalo podle zadání. Studenti budou mít k dispozici náhled do již hotového programu a budou vedeni podle podrobného postupu. Po dokončení programování oba programy zkompilují a nahrají do zařízení. Následně zařízení spustí a ověří si funkčnost jejich aplikací spuštěním sítě a pokusným zapnutím a vypnutím klimatizace a osvětlení.

7.2 Programy pro moduly Jennic

Rozšíření aplikací z první laboratorní úlohy bude v případě koordinátora sítě složitější, než v případě koncového zařízení. Program opět budeme psát ve vývojovém

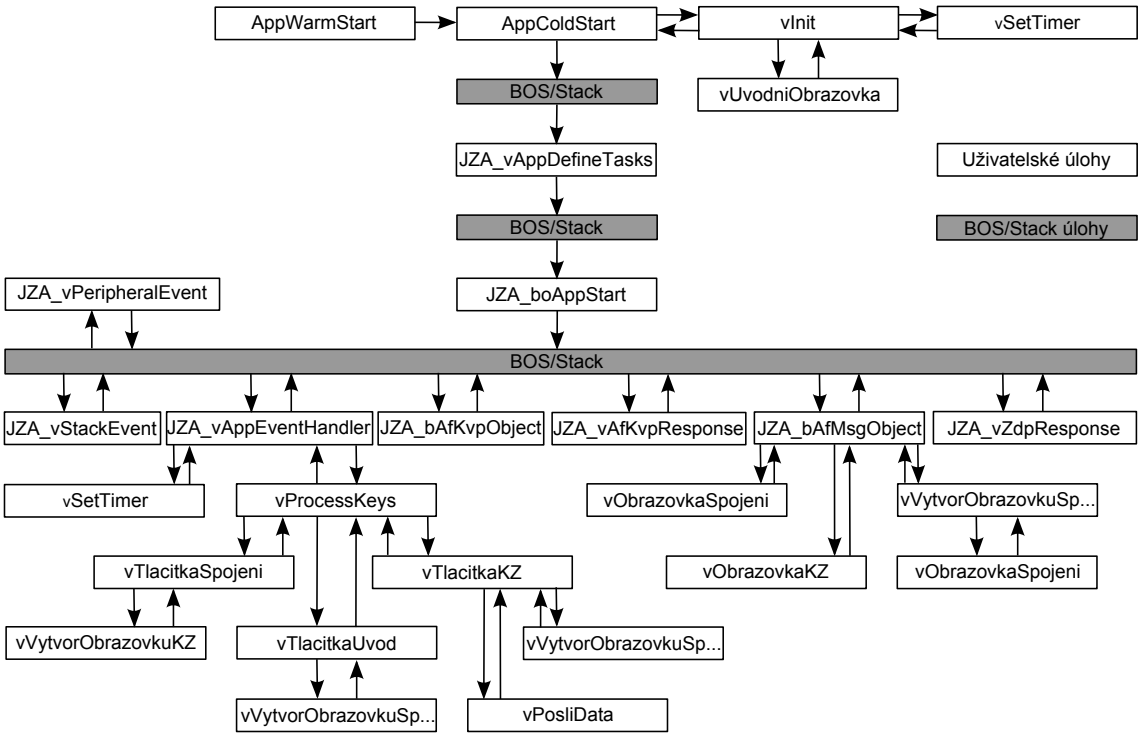
prostředí Code::Blocks IDE. Není třeba popisovat celý program opět znovu, proto budou popsány jen části a funkce, které byly změněny nebo nově vytvořeny.

7.2.1 Koordinátor sítě

Opět je třeba nejprve si zkontrolovat vložené knihovny s předpřipravenými funkcemi. Oproti úloze č. 1 však není potřeba žádné další přidávat, proto ponecháme původní.

Datové struktury je ale třeba doplnit. Musíme vytvořit proměnné, do kterých bude možné uložit stav klimatizace, osvětlení, automatického ovládání a dalších hodnot. Zde jsou uvedeny všechny změny:

- `teObrazovka` – jelikož bude potřeba vytvořit ještě jednu ovládací obrazovku, musíme přidat další stav do této proměnné – `E_STATE_KZ`;
- `tsDataKZ` – zde budou uloženy informace o klimatizaci, osvětlení a automatickém ovládání. Je potřeba doplnit tyto proměnné – `bool_t boKlima`, `bool_t boOsvetleni` a `bool_t boAuto`;
- `tsSystem` – do této struktury musíme přidat proměnnou `uint8 u8ZvoleneKZ`, ve které bude uloženo číslo právě ovládaného koncového zařízení.



Obr. 7.1: Vývojový diagram koordinátora sítě v úloze č. 2

Na obrázku (7.1) je vývojový diagram rozšířeného programu. Můžeme zde vidět veškeré funkce, které budeme muset dále vytvořit. Nyní budou popsány veškeré změny:

- `vProcessKeys` – do této funkce musíme přidat nové reakce na zmáčknutí tlačítka ve stavu `E_STATE_SPOJENI` a nový stav `E_STATE_KZ` včetně reakce. Výsledný kód cyklu `switch` vypadá následovně:

```
switch (sSystem.eObrazovka)
  {
  case E_STATE_UVOD:
    vTlacitkaUvod(u8KeysDown);
    break;
  case E_STATE_SPOJENI:
    vTlacitkaSpojeni(u8KeysDown);
    break;
  case E_STATE_KZ:
    vTlacitkaKZ(u8KeysDown);
    break;
  default:
  break;
  }
```

- `vTlacitkaSpojeni` – při zmáčknutí tlačítka ve stavu `E_STATE_SPOJENI` zvolíme zařízení, které chceme ovládat a vytvoříme novou ovládací obrazovku. Při zmáčknutí prvního tlačítka (`E_KEY_0`) vybereme první zařízení. Stejně tak je tomu v případě dalších tlačítek (`E_KEY_1`, `E_KEY_2` a `E_KEY_3`). Funkci vytvoříme následovně:

```
PRIVATE void vTlacitkaSpojeni(uint8 u8KeyMap)
{
  switch (u8KeyMap)
    {
    case E_KEY_0:
      if(sSystem.u8PripojeneKZ > 0)
        {
        sSystem.u8ZvoleneKZ = 1;
        sSystem.eObrazovka = E_STATE_KZ;
        vVytvorObrazovkuKZ();
        }
      break;
    case E_KEY_1:
      if(sSystem.u8PripojeneKZ > 1)
        {
        sSystem.u8ZvoleneKZ = 2;
        sSystem.eObrazovka = E_STATE_KZ;
        vVytvorObrazovkuKZ();
        }
      break;
    default:
      break;}}
```

Do cyklu `switch` je ještě potřeba doplnit reakce na zmáčknutí tlačítka č. 3 a č. 4, které vypadají stejně jako v případě předchozích tlačítek. Mění se pouze název zmáčknutého tlačítka, číslo zvoleného zařízení a počet připojených koncových zařízení;

- `vTlacitkaKZ` – podobný cyklus jako u předchozí funkce bude i zde. Po zmáčknutí tlačítek 1–3 se bude zapínat či vypínat klimatizace, osvětlení a automatické ovládání. Po zmáčknutí posledního tlačítka se vrátíme zpět na obrazovku spojení. Kód vypadá takto:

```
PRIVATE void vTlacitkaKZ(uint8 u8KeyMap)
{
  switch (u8KeyMap)
    {
    case E_KEY_0:
      sDataKZ[sSystem.u8ZvoleneKZ-1].boKlima =
                      !sDataKZ[sSystem.u8ZvoleneKZ-1].boKlima;
      vObrazovkaKZ(sSystem.u8ZvoleneKZ-1);
      vPosliData();
      break;
    case E_KEY_3:
      sSystem.eObrazovka = E_STATE_SPOJENI;
      vVytvorObrazovkuSpojeni();
      break;
    default:
        break;
    }
}
```

  Nyní je ještě třeba doplnit reakce na tlačítka `E_KEY_1` a `E_KEY_2`. Budou vypadat naprosto stejně jako v případě `E_KEY_0`, pouze místo proměnné `boKlima` bude proměnná `boOsvetleni`, respektive `boAuto`;

- `vVytvorObrazovkuKZ` – tato funkce vytvoří ovládací obrazovku zvoleného koncového zařízení. Zdrojový kód je zde:

```
PRIVATE void vVytvorObrazovkuKZ(void)
{
  char acString[10];
  vLcdClear();
  vLcdWriteText("Koncove zarizeni: ", 0, 0);
  vValToDec(acString, sSystem.u8ZvoleneKZ, " ");
  vLcdWriteText(acString, 0, 95);
  vLcdWriteText("Temp:", 1, 15);
  vLcdWriteText("LQI:", 1, 75);
  vLcdWriteText("Light:", 2, 15);
  vLcdWriteText("U:", 2, 75);
  vLcdWriteText("Klimatizace:", 3, 15);
  vLcdWriteText("Osvetleni:", 4, 15);
  vLcdWriteText("Automaticky:", 5, 15);
```

```
  vObrazovkaKZ(sSystem.u8ZvoleneKZ - 1);
  vLcdWriteText("Klima", 7, 0);
  vLcdWriteText("Svetlo", 7, 30);
  vLcdWriteText("Auto", 7, 72);
  vLcdWriteText("Zpet", 7, 103);
  vLcdRefreshAll();
}
```

V prvním případě si vytvoříme pomocnou proměnnou `char acString[10]`, následně vymažeme obsah celého LCD displeje a vypíšeme na obrazovku pole pro zobrazení hodnot. Funkce `vLcdWriteText(acString, 0, 95)` má tři parametry. První z nich je to, co se má na displeji zobrazit. Další dva značí řádek a sloupec displeje. Na konec je pro správné zobrazení potřeba LCD displej obnovit;

- `vObrazovkaKZ` – pomocí této funkce vypíše do předpřipravených polí hodnoty a informace o zvoleném koncovém zařízení. Zdrojový kód vypadá následovně:

```
PRIVATE void vObrazovkaKZ(uint8 u8AktualniKZ)
{
char acString[10];
vValu16ToDec(acString, sDataKZ[u8AktualniKZ].u16Teplota, "[C ");
vLcdWriteText(acString, 1, 45);
vValToDec(acString, sDataKZ[u8AktualniKZ].u8LQI, " ");
vLcdWriteText(acString, 1, 95);
vValu16ToDec(acString, sDataKZ[u8AktualniKZ].u16Svetlo, " ");
vLcdWriteText(acString, 2, 45);
vValu16ToDec(acString, sDataKZ[u8AktualniKZ].u16Napeti, "mV ");
vLcdWriteText(acString, 2, 85);
vValToDec(acString, sDataKZ[u8AktualniKZ].boKlima, " ");
vLcdWriteText(acString, 3, 85);
vValToDec(acString, sDataKZ[u8AktualniKZ].boOsvetleni, " ");
vLcdWriteText(acString, 4, 85);
vValToDec(acString, sDataKZ[u8AktualniKZ].boAuto, " ");
vLcdWriteText(acString, 5, 85);
vLcdRefreshAll();
}
```

Funkce `vValToDec` a `vValu16ToDec` převádí hodnoty koncového zařízení do textové podoby pro zobrazení na displeji;

- `vPosliData` – zdrojový kód této funkce je na následujících řádcích:

```
PRIVATE void vPosliData(void)
{
AF_Transaction_s asPRENASENA_DATA[1];
asPRENASENA_DATA[0].u8SequenceNum = u8AfGetTransactionSequence(TRUE);
asPRENASENA_DATA[0].uFrame.sMsg.u8TransactionDataLen = 3;
asPRENASENA_DATA[0].uFrame.sMsg.au8TransactionData[0] =
                sDataKZ[sSystem.u8ZvoleneKZ-1].boKlima;
asPRENASENA_DATA[0].uFrame.sMsg.au8TransactionData[1] =
```

```
                sDataKZ[sSystem.u8ZvoleneKZ-1].boOsvetleni;
asPRENASENA_DATA[0].uFrame.sMsg.au8TransactionData[2] =
                sDataKZ[sSystem.u8ZvoleneKZ-1].boAuto;
(void)afdeDataRequest(APS_ADDRMODE_SHORT,
                sDataKZ[sSystem.u8ZvoleneKZ-1].u16ShortAdr,
                CISLO_KONC_BODU, CISLO_ZDROJ_BODU, PROFILE_ID,
                INPUT_CLUSTER2, AF_MSG, 1, asPRENASENA_DATA,
                APS_TXOPTION_NONE, ENABLE_ROUTE_DISCOVERY, 4);
}
```

Pomocí `AF_Transaction_s asPRENASENA_DATA[1]` si vytvoříme jednoduché pole pro odesílání dat. Poté do této proměnné uložíme všechny potřebná data a pomocí příkazu `(void)afdeDataRequest()` data odešleme. V parametrech tohoto příkazu nastavujeme mimo jiné adresu a číslo cílového bodu, typ zprávy a samozřejmě určujeme, která data se mají přenést;

- `JZA_bAfMsgObject` – zdrojový kód této funkce je příliš dlouhý, proto je zde uvedena jen nejdůležitější část:

```
if(u8ClusterID == INPUT_CLUSTER)
  {
  u16BatNapeti = puTransactionInd->uFrame.sMsg.au8TransactionData[1];
  u16BatNapeti  = u16BatNapeti << 8;
  u16BatNapeti |= puTransactionInd->uFrame.sMsg.au8TransactionData[0];
  u16Teplota = puTransactionInd->uFrame.sMsg.au8TransactionData[3];
  u16Teplota  = u16Teplota << 8;
  u16Teplota |= puTransactionInd->uFrame.sMsg.au8TransactionData[2];
  u16Svetlo = puTransactionInd->uFrame.sMsg.au8TransactionData[5];
  u16Svetlo = u16Svetlo << 8;
  u16Svetlo |= puTransactionInd->uFrame.sMsg.au8TransactionData[4];
  boKlima = puTransactionInd->uFrame.sMsg.au8TransactionData[6];
  boOsvetleni = puTransactionInd->uFrame.sMsg.au8TransactionData[7];
  boAuto = puTransactionInd->uFrame.sMsg.au8TransactionData[8];
  for (i = 9; i < 17; i++)
    {
    MAC = MAC << 8;
    MAC |= puTransactionInd->uFrame.sMsg.au8TransactionData[i];
    }
```

Tato část zdrojového kódu ukládá data z přijatého rámce do jednotlivých proměnných. V dalších částech zdrojového kódu je ještě potřeba vytvořit pomocné proměnné pro nové hodnoty (klimatizace, osvětlení, automatické ovládání) a dále rozlišit, v jakém stavu se zařízení nachází. Podle toho stavu pak data zobrazit na displeji nebo jen uložit do paměti. Výsledný zdrojový kód je možné si prohlédnout v příloze.

Pro správnou funkci je ještě potřeba nově přidané funkce zapsat do seznamu funkcí v části `Local Function Prototypes`. Na toto místo doplníme, nejlépe před všechny ostatní funkce, tyto řádky:

```
PRIVATE void vTlacitkaSpojeni(uint8 u8KeyMap);
PRIVATE void vTlacitkaKZ(uint8 u8KeyMap);
PRIVATE void vVytvorObrazovkuKZ(void);
PRIVATE void vObrazovkaKZ(uint8 u8AktualniKZ);
PRIVATE void vPosliData(void);
```

Tím máme doplněny všechny funkce potřebné na straně koordinátora sítě, nyní ještě musíme doplnit další funkce do programu koncového zařízení.

7.2.2 Koncové zařízení

Na straně koncového zařízení nebude již doplnění funkcí tak těžké. Není potřeba vkládat žádné nové knihovny. Všechny potřebné jsou již v programu uvedeny.

Je však třeba vytvořit novou datovou strukturu, ve které budou uchovány data o stavech klimatizace, osvětlení a automatického ovládání. Nová struktura vypadá následovně:

```
typedef struct
{
  bool_t boKlima;
  bool_t boOsvetleni;
  bool_t boAuto;
} tsOvladani;
```

Dále je třeba vytvořit proměnnou založenou na nově vytvořené datové struktuře. Do seznamu proměnných (`Local Variables`) doplníme novou s názvem `sOvladani`:

```
PRIVATE tsOvladani sOvladani;
```

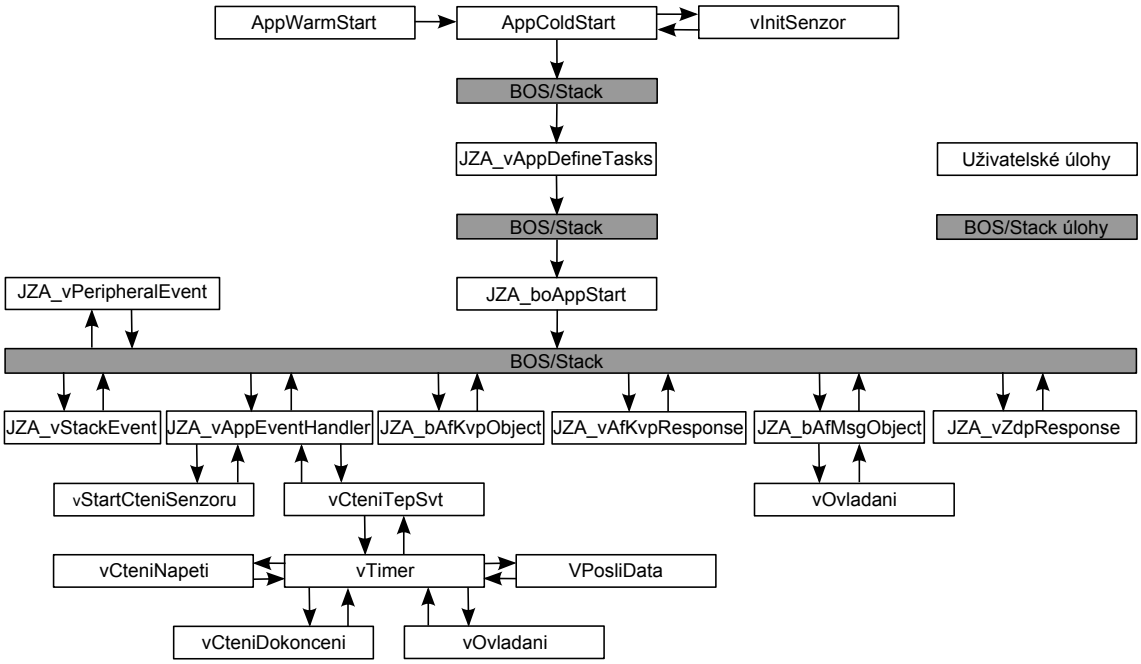
Nyní již můžeme doplnit do programu novou funkci pro ovládání klimatizace a osvětlení a rozšířit stávající funkce podle vývojového diagramu na obrázku (7.2):

- `vInit` – při inicializaci je potřeba vhodně nastavit proměnné v nové datové struktuře:

  ```
  sOvladani.boKlima = FALSE;
  sOvladani.boOsvetleni = FALSE;
  sOvladani.boAuto = FALSE;
  ```

  Tyto 3 řádky je třeba doplnit do sekce nastavení výchozích parametrů. Hodnoty `FALSE` nám říkají, že jak klimatizace, tak osvětlení, bude vypnuto. Automatické ovládání bude také vypnuto;

- `vOvladani` – tato funkce je třeba vytvořit pro ovládání klimatizace a osvětlení:



Obr. 7.2: Vývojový diagram koncového zařízení v úloze č. 2

```
PRIVATE void vOvladani(void)
{
if(sOvladani.boAuto == TRUE)
  {
  if(sHodnoty.u16Teplota > 23)
    {
    sOvladani.boKlima = TRUE;
    }
  else {sOvladani.boKlima = FALSE;}
  if(sHodnoty.u16Svetlo < 50)
    {
    sOvladani.boOsvetleni = TRUE;
    }
  else {sOvladani.boOsvetleni = FALSE;}
  }
vLedControl(0, sOvladani.boKlima);
vLedControl(1, sOvladani.boOsvetleni);
}
```

Na začátku je nejprve zkontrolováno, jestli má program ovládat zařízení sám nebo ne. Pokud je automatické ovládání povoleno, tak je dále zkontrolována hodnota teploty. Pokud je větší než 23, je spuštěna klimatizace. Obdobně je tomu u hodnoty osvětlení. Pokud je menší než 50, spustí se osvětlení. Na konci funkce se zapnou nebo vypnou LED diody, které odpovídají klimatizaci (LED č. 1) a osvětlení (LED č. 2);

- `vTimer` – v časovači musíme před odesláním dat spustit novou funkci `vOvladani`,

které po každém naměření hodnot zkontroluje nastavení klimatizace a osvětlení. Před rádkem `vPosliData();` vytvoříme nový řádek `vOvladani();` a až poté odešleme data;

- `vPosliData` – v datovém rámci musíme poslat i informace o stavu klimatizace, osvětlení a pro kontrolu i o stavu automatické ovládání. Část zdrojového kódu s načítáním dat do datové struktury pro odeslání vypadá následovně:

```
AF_Transaction_s asPRENASENA_DATA[1];
asPRENASENA_DATA[0].u8SequenceNum = u8AfGetTransactionSequence(TRUE);
asPRENASENA_DATA[0].uFrame.sMsg.u8TransactionDataLen = 17;
asPRENASENA_DATA[0].uFrame.sMsg.au8TransactionData[0] = sHodnoty.u16Napeti;
asPRENASENA_DATA[0].uFrame.sMsg.au8TransactionData[1] =
                sHodnoty.u16Napeti >> 8;
asPRENASENA_DATA[0].uFrame.sMsg.au8TransactionData[2] =
                sHodnoty.u16Teplota;
asPRENASENA_DATA[0].uFrame.sMsg.au8TransactionData[3] =
                sHodnoty.u16Teplota >> 8;
asPRENASENA_DATA[0].uFrame.sMsg.au8TransactionData[4] = sHodnoty.u16Svetlo;
asPRENASENA_DATA[0].uFrame.sMsg.au8TransactionData[5] =
                sHodnoty.u16Svetlo >> 8;
asPRENASENA_DATA[0].uFrame.sMsg.au8TransactionData[6] = sOvladani.boKlima;
asPRENASENA_DATA[0].uFrame.sMsg.au8TransactionData[7] =
                sOvladani.boOsvetleni;
asPRENASENA_DATA[0].uFrame.sMsg.au8TransactionData[8] = sOvladani.boAuto;
  for (i = 0; i < 8; i++)
    {
    asPRENASENA_DATA[0].uFrame.sMsg.au8TransactionData[i + 9] =
                    *(pu8ExtAdr + i);
    }
```

- `JZA_bAfMsgObject` – jelikož budeme od koordinátora sítě přijímat data, musíme tuto funkci upravit tak, aby je dokázala zpracovat:

```
if ((eAddrMode == APS_ADDRMODE_SHORT)
                && (u8DstEP = CISLO_KONC_BODU))
  {
  if(u8ClusterID == INPUT_CLUSTER2)
    {
    sOvladani.boKlima =
      puTransactionInd->uFrame.sMsg.au8TransactionData[0];
    sOvladani.boOsvetleni =
      puTransactionInd->uFrame.sMsg.au8TransactionData[1];
    sOvladani.boAuto =
      puTransactionInd->uFrame.sMsg.au8TransactionData[2];
    }
  vOvladani();
  }
return 0;
```

Tento zdrojový kód musí být ve funkci obsažen. Nejdříve je zkontrolováno, jestli je správa určená pro toto zařízení a následně jsou data s MSG rámce

uložena do proměnných. Poté je volána funkce `vOvladani`, která provede nastavení podle přijatých dat.

Nově přidanou funkci je ještě třeba zapsat do seznamu funkcí v části `Local Function Prototypes`. Na toto místo doplníme, nejlépe opět před všechny ostatní funkce, tento řádek:

```
PRIVATE void vOvladani(void);
```

Nyní máme programy, jak pro koordinátora sítě, tak pro koncové zařízení hotové a připravené ke kompilování a následně k nahrání do zařízení.

7.3 Postup a výsledky

Ke kompilování využijeme program Jennic Flash Programmer a postupovat budeme dle kapitoly (5.3). Nejprve program pro koordinátora sítě zkompilujeme a zkontrolujeme jestli nehlásí žádné chyby, následně připojíme zařízení pomocí USB kabelu k počítači a nahrajeme do něj výsledný zkompilovaný program s koncovkou `.bin`. To samé provedeme i s programem pro koncové zařízení.

7.3.1 Spuštění zařízení

Nyní již máme oba typy zařízení připravené ke spuštění. Nejprve spustíme koordinátora sítě a nastavíme pomocí tlačítek 2 a 3 frekvenční kanál. Poté potvrdíme volbu pomocí tlačítka 4. Tím se dostaneme na obrazovku se seznamem koncových zařízení. Pokud žádné takové zařízení není spuštěno, na obrazovce je napsáno `Nenalezeny KZ`. Teď můžeme spustit i koncová zařízení. Postupně se začnou objevovat na LCD displeji s pořadovými čísly jako v případě laboratorní úlohy č. 1. Až do tohoto okamžiku jsme nevyužili žádnou námi nově naprogramovanou funkci.

Pomocí tlačítek 1 až 4 vybereme jedno z koncových zařízení, čímž již využijeme naše nové funkce. Tímto se dostaneme na novou obrazovku (Obr. 7.3), kde vidíme i stav ovládání zařízení. Všechny proměnné (klimatizace, osvětlení a automatické ovládání) by měly mít hodnotu 0, což znamená vypnuto. Pomocí tlačítek 1 a 2 můžeme zkusit pokusně zapnout osvětlení a klimatizaci. Zmáčknutí těchto tlačítek by se mělo projevit změnou hodnoty na 1 (zapnuto) a také na koncovém zařízení rozsvícením LED diod.

Pokud vše funguje, můžeme ještě vyzkoušet automatické ovládání. To zapneme tlačítkem 3. Tím přebere kontrolu nad ovládáním koncové zařízení a dle přednastavených prahových hodnot zapne či vypne klimatizaci a osvětlení.

Obr. 7.3: LCD Obrazovka - stav ovládání

Pokud vše funguje jak má, můžeme se pomocí tlačítka 4 vrátit do seznamu koncových zařízení a stejným postupem vyzkoušet ovládání i na jiném zařízení.

7.3.2 Analýza síťové komunikace

Pro analýzu komunikace využijeme ZENA Wireless Network Analyzer (kapitola 3). Zařízení připojíme k počítači pomocí USB portu a spustíme program pro analýzu. V tomto programu klikneme na `ZigBee(TM) 2006 Tools` a následně na `Network Traffic Monitor`. Tím se otevře aplikace pro monitorování ZigBee sítě. V této aplikaci zvolíme stejný kanál, jaký je zvolený na koordinátorovi sítě a všechny položky ve `Verboseness Level` nastavíme na `Verbose`. Tím je konfigurace hotova. Nyní ještě pomocí tlačítka `Network Configuration Display` otevřeme okno, ve kterém se nám graficky zobrazí topologie sítě. Zachytávání spustíme tlačítkem `Start Sniffing`.

Nyní spustíme i Zigbee zařízení, vyzkoušíme veškeré funkce a sledujeme například, jakým způsobem je odesílán příkaz k vypnutí osvětlení. Ze zachycených rámců můžeme vyčíst, kromě jiného, zdrojovou a cílovou adresu, PAN ID sítě a v políčku `AF Data` také vlastní data. V okně s topologií sítě můžeme graficky sledovat směr dat v síti.

8 ZÁVĚR

Cílem této bakalářské práce bylo důkladně se seznámit s komunikačním standardem IEEE 802.15.4 ZigBee, zejména s jeho praktickou částí a vytvořit dvě laboratorní úlohy, které budou seznamovat s možnostmi tohoto standardu a použitého vývojového kitu od firmy Jennic.

Z dostupné literatury jsem nastudoval vlastnosti protokolu ZigBee. V první kapitole jsem pak čtenáře seznámil se standardem IEEE 802.15.4, na kterém je protokol ZigBee postaven. Uvedl jsem frekvenční rozdělení pásem, ve kterých pracuje. Dále typy modulací a možnosti uspořádání sítě.

V dalších kapitolách jsem popsal vlastní protokol ZigBee, zejména rozdělení na jednotlivé vrstvy referenčního modelu, dále pak s možnostmi adresování a zabezpečení sítě. Uvedl jsem informace o použitém analyzátoru ZENA a vývojovém kitu JN5139 od firmy Jennic. Popsal jsem také způsoby vytváření aplikací pro moduly tohoto kitu ve vývojovém prostředí Code::Blocks IDE.

Na základě zadání jsem první laboratorní úlohu sestavil tak, aby se v ní studenti seznámili celkově s daným standardem a pochopili jeho výhody a nevýhody. Při praktickém měření této úlohy mají studenti za úkol ověřit kvalitu spojení v závislosti na rušení jinými technologiemi pracujícími na stejném frekvenčním pásmu, dále pak dosah a spotřebu zařízení z vývojového kitu Jennic.

Druhá úloha je pak koncipována tak, aby studenty seznámila s vytvářením aplikací pro dané moduly. Mají zde za úkol rozšířit aplikaci z předchozí úlohy tak, aby se dalo za pomoci koncových zařízení ovládat osvětlení a klimatizace. Funkčnost aplikací pak ověří spuštěním sítě.

Při praktickém ověřování těchto úloh jsem nenarazil na žádné výrazné problémy, pouze některé údaje výrobce, jako například dosah až 4 kilometry neodpovídají a dle měření vychází maximální dosah do 500 metrů. Jinak zařízení fungují spolehlivě a dají se bez problémů využít v praxi, s přihlédnutím na reálný dosah a podmínky provozu. Návody k laboratorním úlohám, včetně hotových programů pro ZigBee moduly, jsou v elektronické podobě na přiloženém DVD.

LITERATURA

[1] BRADÁČ, Z. Bezdrátový komunikační standard ZigBee. *Automatizace* [online]. 2005, ročník 48, číslo 4, strana 261, [cit. 5. 12. 2009]. Dostupné z URL: <http://www.automatizace.cz/article.php?a=638>.

[2] IEEE COMPUTER SOCIETY. *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)* [online]. 2006 [cit. 5. 12. 2009]. Dostupné z URL: <http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf>.

[3] JENNIC. *Data Sheet: JN5139-001 and JN5139-Z01* [online]. 2009 [cit. 5. 12. 2009]. Dostupné z URL: <http://www.jennic.com/support/...>.

[4] JENNIC. *DR1047 Controller Board Reference Manual* [online]. 2009 [cit. 5. 12. 2009]. Dostupné z URL: <http://www.jennic.com/support/...>.

[5] JENNIC. *DR1048 Sensor Board Reference Manual* [online]. 2009 [cit. 5. 12. 2009]. Dostupné z URL: <http://www.jennic.com/support/...>.

[6] JENNIC. *Internetové stránky firmy* [online]. [cit. 5. 12. 2009]. Dostupné z URL: <http://www.jennic.com>.

[7] JENNIC. *ZigBee Stack User Guide* [online]. 2008 [cit. 5. 12. 2009]. Dostupné z URL: <http://www.jennic.com/support/...>.

[8] LABIOD, H. *H. et al.: Wi-Fi, Bluetooth, ZigBee and WiMax* 2007. 316 s. ISBN 978-1-4020-5396-2.

[9] MICROCHIP. *Internetové stránky firmy* [online]. [cit. 13. 4. 2010]. Dostupné z URL: <http://www.microchip.com>.

[10] MICROCHIP. *ZENA 3.0 Network Analyzer* [online]. 2008 [cit. 13. 4. 2010]. Dostupné z URL: <http://www.microchip.com/stellent/...>.

[11] ZIGBEE ALLIANCE. *Internetové stránky sdružení* [online]. [cit. 5. 12. 2009]. Dostupné z URL: <http://www.zigbee.org/>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ACK Acknowledgement

AES Advanced Encryption Standard

APL Application Layer

APS Application Support Sublayer

ASK Amplitude Shift Keying

BOS Basic Operating System

BPSK Binary Phase Shift Keying

CBC Cipher Block Chaining

CLI Command Line Interface

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance and Optional Time Slotting

CTR Counter

EEPROM Electronically Erasable Programmable Read-Only Memory

FFD Full Function Device

IDE Integrated Development Environment

IEEE Institute of Electrical and Electronics Engineers

MAC Media Access Control

MIC Message Integrity Code

NWK Network

O-QPSK Offset - Quadrature Phase Shift Keying

OSI Open Systems Interconnection

PAN ID Personal Area Network Identification

PHY Physical Layer

QPSK Quadrature Phase Shift Keying

RFD Reduced Function Device

SSP Security Services Provider

UART Universal Asynchronous Receiver / Transmitter

WPAN Wireless Personal Area Network

ZDO ZigBee Device Object

SEZNAM PŘÍLOH

Adresář na DVD	**Obsah**
\Bakalářská práce	Elektronická podoba bakalářské práce
\Laboratorní úloha č. 1	Návod k úloze a aplikace k ZigBee modulům
\Laboratorní úloha č. 2	Návod k úloze a aplikace k ZigBee modulům
\Literatura	Elektronická část použité literatury
\Programy	Veškeré potřebné programy pro PC
\Ostatní	Další programy pro moduly ZigBee