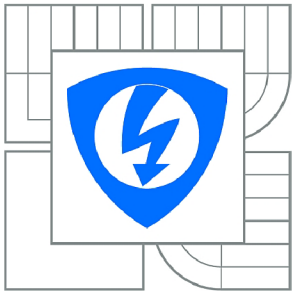
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF RADIO ELECTRONICS

BEZDRÁTOVÉ KOMUNIKAČNÍ MODULY PRO MIKROKONTROLÉRY

WIRELESS COMMUNICATION MODULES FOR MICROCONTROLLERS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

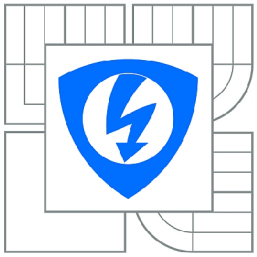
AUTOR PRÁCE — Bc. JAN KLÍMA

AUTHOR

VEDOUCÍ PRÁCE — Ing. ZBYNĚK FEDRA, Ph.D.

SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Diplomová práce

magisterský navazující studijní obor
Elektronika a sdělovací technika

Student:	Bc. Jan Klíma	***ID:***	83638
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Bezdrátové komunikační moduly pro mikrokontroléry

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte možnosti modulů pro bezdrátovou komunikaci dostupných pro mikrokontorléry. Zaměřte se na jednoduché moduly komunikující v ISM pásmu a na ZigBee moduly. Srovnejte možnosti jednotlivých řešení a vyberte dostupné moduly.

Navrhněte zapojení rozšiřujících desek s komunikačním modulem (přijímač i vysílač) pro využití se stávajícím vybavením v laboratoři mikroprocesorové techniky. Otestujte základní komunikace jednotlivých modulů.

Realizujte rozšiřující moduly pro bezdrátovou komunikaci a ověřte jejich charakteristiky. Vytvořte knihovnu funkcí pro dané komunikační moduly a navrhněte demonstrační úlohu. Charakterizujte možnosti použití modulů pro bateriově napájená zařízeni a chování při přechodu procesoru do sleep módu.

DOPORUČENÁ LITERATURA:

[1] MATOUŠEK, D. Práce s mikrokontroléry Atmel AVR. BEN - technická literatura, Praha, 2003
[2] MANN, B. C pro mikrokontroléry. BEN - technická literatura, Praha, 2003

Termín zadání:	7.2.2011	***Termín odevzdání:***	20.5.2011

Vedoucí práce: Ing. Zbyněk Fedra, Ph.D.

prof. Dr. Ing. Zbyněk Raida
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Cíl diplomové práce je rozebrání problematiky bezdrátové komunikace mezi mikrokontroléry.

V první části práce je rozebraná problematika bezdrátového ISM pásma. Hlavně moduly RFM12B a ZigBee, které v tomto pásmu pracují a ZigBee standart

V druhé části je popsaná realizace desky pro oba moduly. Dále jak naprogramovat ZigBee modul. A nakonec programy pro testování modulů.

ABSTRACT

The aim of this master's thesis is analysis of wireless communication between mikroprocesors.

The first part of the work are analysed the problems the wireless free ISM band. Mainly RFM12B and ZigBee module, which i these band work and ZigBee standart.

The second part are described relazation board for both module. Further how programming ZigBee module. And at last programs for tested module.

KLÍČOVÁ SLOVA

ISM pásmo, bezdrátové moduly pro ISM pásmo, ZigBee standard, topologie sítě, BitCloud, ZigBit, RFM12B

KEY WORDS

ISM band, wireless module for ISM band, ZigBee standard, network topology, BitCloud, ZigBit, RFM12B

KLÍMA, J. Bezdrátové komunikační moduly pro mikrokontroléry. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 56 s. Vedoucí diplomové práce Ing. Zbyněk Fedra, Ph.D.

.

Prohlášení

Jako autor uvedené diplomové práce na téma „Bezdrátové komunikační moduly pro mikrokontroléry“ dále prohlašuji, že v souvislosti s vytvořením této diplomové 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 části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb..

V Brně dne 19. května 2011 .....

podpis autora

Poděkování

Děkuji, vedoucímu diplomové práce Ing. Zbyňkovi Ferdovi Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne 19. května 2011 .....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1.Úvod

V dnešní době, kdy většina komunikace probíhá bezdrátově, je používání bezdrátových modulů pomalu nezbytná. Ať už se jedná o jednoduché rf moduly, které se starají pouze o modulaci, vysílání a příjem. Anebo o složitější, kdy se moduly spojují do větších sítí jako je např. Bluetooth nebo ZigBee.

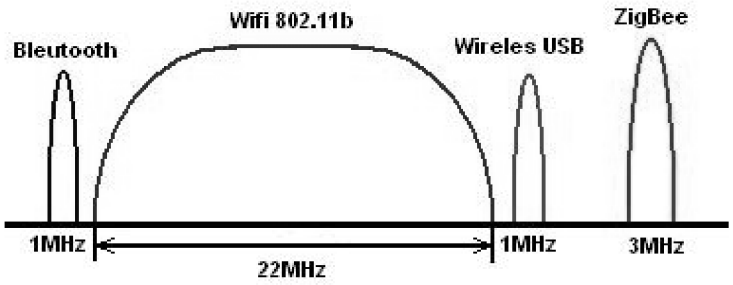
Cílem této diplomové práce je navrhnout desku pro moduly ZigBee a RFM12B, které pracují v ISM pásmu. Největší pozornost je věnována standartu ZigBee a jeho programování pomocí softwarového stacku BitCloud.

Diplomová práce je rozdělena do několika hlavních částí. Nejprve se seznámíme s oběma rf moduly. Další část je věnovaná standardu ZigBee. Poté je probraná realizace komunikační desky modulů. Ke konci jsou popsány programy pro komunikaci a přenosu dat mezi jednotlivými moduly.

2. ISM pásmo

Jedná se o volná radiová pásma - ISM (Industrial - průmyslový, Scientific - vědecký, Medical - medicínský), ve kterých je povolen provoz homologovaných bezdrátových zařízení. Avšak na rozdíl od licencovaných pásem zde není zaručen bezproblémový provoz a může se vyskytnout vzájemné rušení. Mezi nejvíce používaná ISM pásma patří tyto: 433MHz, 868MHz, 915MHz a 2,4GHz.

Z těchto pasem je dnes nejpoužívanější a nejznámější ta na frekvenci 2,4 GHz viz. obr.1. Vždyť na tomto pásmu pracuje všem známá bezdrátová spojení wifi a bluetooth. A v průmyslu je také znám ZigBee. Z důvodu minimalizace rušení se používá rozprostření spektra. Bluetooth používá FHSS (Frequency Hopping Spread Spectrum) zatímco Wifi (802.11 b/g/a) a ZigBee (802.15.4) používá DSSS (Direct Sequence Spread Spectrum).



Obr. 1 Srovnání signálů systémů pracujících v pásmu 2,4 GHz (převzato z [1])

Bluetooth 802.15.1
Rozdělí pásmo do 79. 1MHz kanálů. Mezi těmito kanály se pohybuje 1600 krát za sekundu. Jednotlivé zařízení jsou seskupeny do tzv. piconetů – každý piconet obsahuje jeden master a až 7 aktivních slave.

Wifi 802.11b
Vymezuje 13 kanálů, kde je každý široký 22MHz. Každý kanál používá Wifi přístupový bod, ke kterému se připojují jednotliví Wifi klienti. Max. rychlost až 11Mbit/s

Wireles USB
Šířka kanálu je 1MHz což umožňuje rozdělit pásmo na 79 kanálů. Rychlost až 62,5kbit/s

ZigBee 802.15.4
Radiový signál v pásmu 868MHz (Evropa), 915MHz (Severní Amerika) a 2,4GHz (celý svět). V pásmu 2,4 GHz je vymezeno 16 kanálů. Každý kanál zabírá 3MHz a kanály jsou centrovány na 5MHz od sebe. Mezi dvojicí kanálů je tedy mezera 2MHz. Používá se 11-čipový PN kód. Maximální rychlost přenosu je 128kbps.

[1]

3. RF modul - RFM 12B

Tento modul je díky svému lehkému připojení k mikrokontroléru, frekvenčním rozsahům a velikostí, výborným řešením.

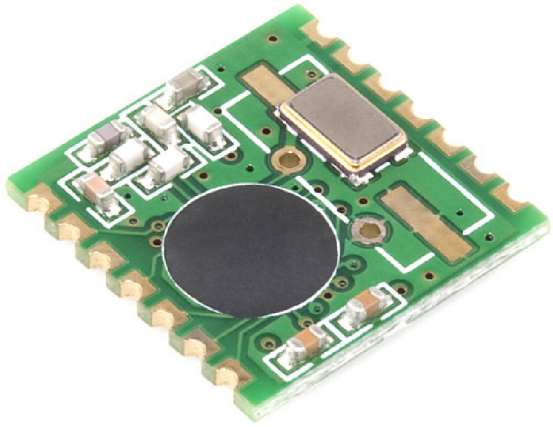
Při propojení s procesorem modul přijaté data namoduluje pomocí FSK modulace a vyšle k dalšímu moduly, který se chová jako přijímač. Ten data demoduluje a připojený procesor může data přijmout a vyhodnotit.

Frekvenční rozsah	433/868/915 MHz	Min. [MHz]	Max. [MHz]
Frekvenční kroky	433 MHz ,2.5KHz	430.24	439.75
	868 MHz ,5KHz	860.48	879.51
	915 MHz ,7.5KHz	900.72	929.27

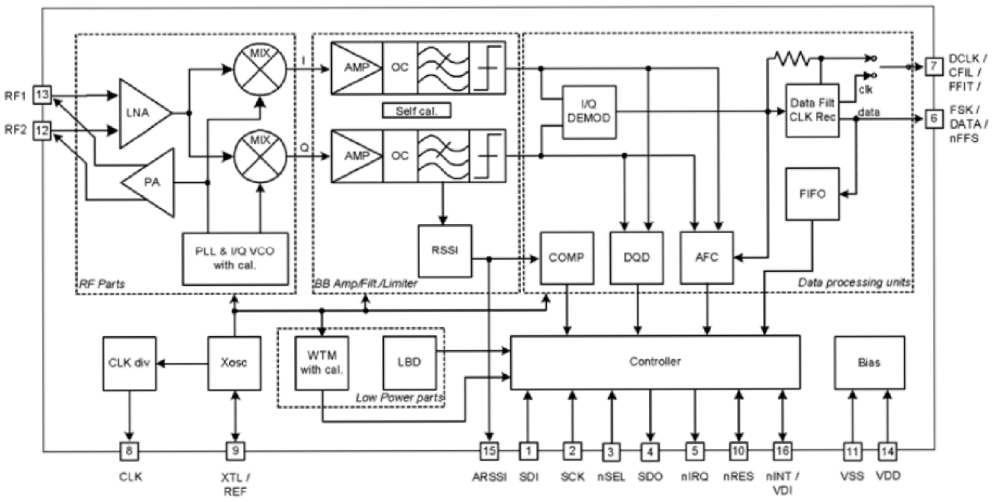
Tab. 1 Frekvenční rozsah modulu RFM12B (převzato z [9])

Datová rychlost int. demod.	115,2 kbps
Datová rychlost s ext. RC filtrem	256 kbps
Rozlišení PLL	2,5kHz
Automatická kontrola frekvence AFC	Ano
Detekce kvality dat (DQD)	Ano
Napájecí napětí	2.2 - 3.8V
Proudový odběr RX	záleží na použitém rozsahu
Proudový odběr TX	
Proudový odběr - spící režim	0.3 uA
RX data FIFO	16 bitů
TX data registry	2 x 8 bitů
Modulace	FSK
Rozměry	14 x 18 mm

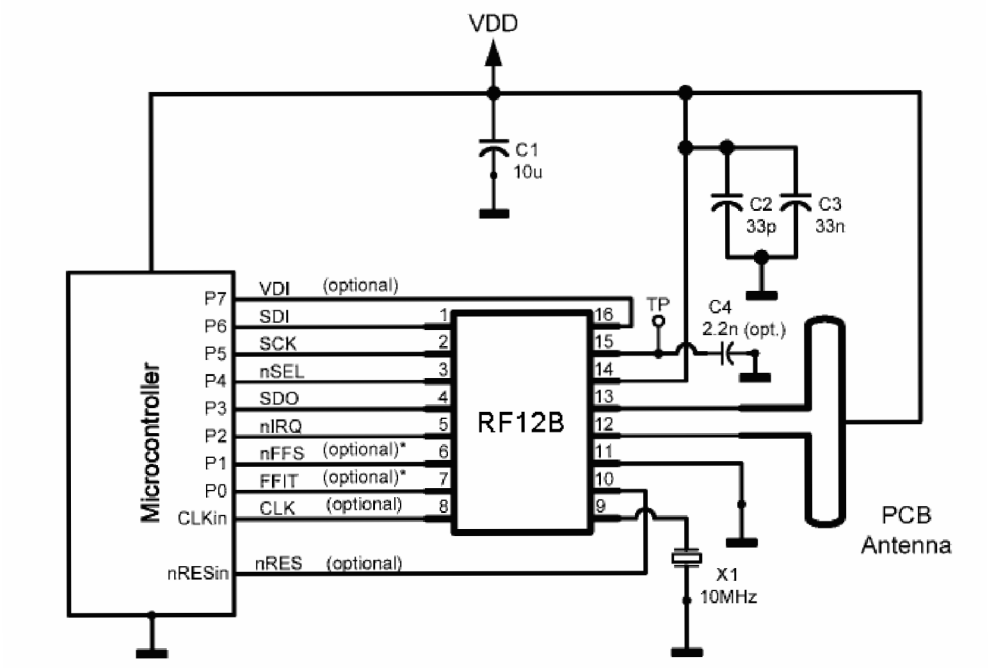
Tab. 2 Parametry modulu RFM 12B (převzato z [9])



Obr. 2 Modul RFM12B (převzato z [8])



Obr. 3 Funkční blokový diagram (převzato z [10])



Obr. 4 Zapojení rf modulu s mikrokontolérem ([10])

4. Programování modulu RFM12B

4.1 Způsob programování modulu

Aby modul správně fungoval, musí být připojen k mikroprocesoru, který obstarává obsluhu modulu. Proto způsob programování záleží na zvoleném mikroprocesoru, a jeho možnostech.

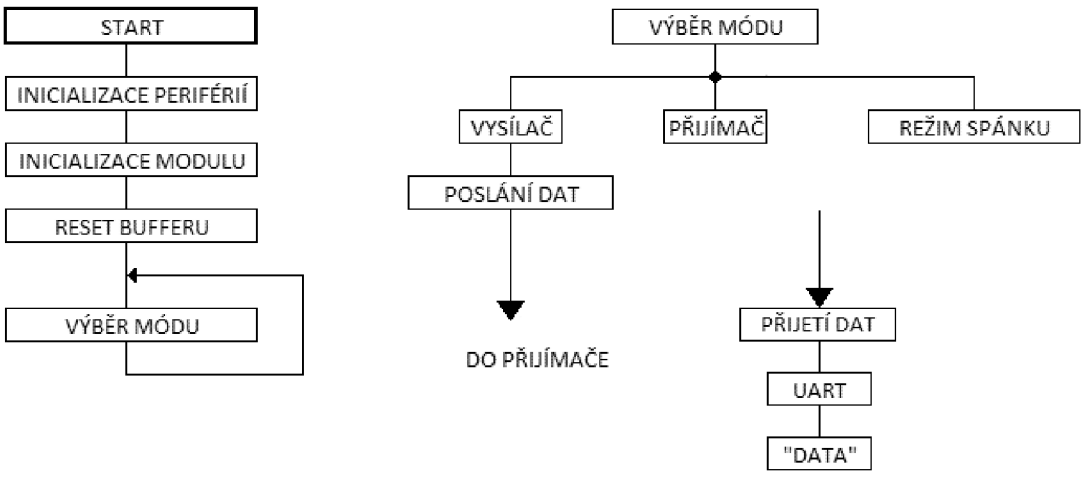
V tomto případě je modul připojen k modulu ZigBee, který využívá jako vnitřní procesor ATmega1281. Procesor podporuje způsob programování pomocí jazyka C, nebo Asembleru. Z těchto možností jsem vybral první z výše jmenovaných.

4.2 Popis programu pro modul RFM12B

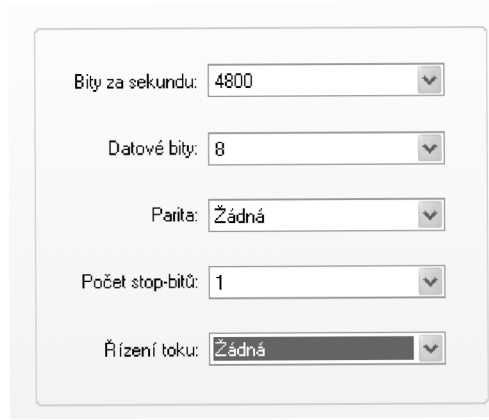
Po zapnutí desky a připojení komunikace je důležité zapnout napájení modulu, pomocí přepínače vedle modulu. Dále je potřeba nastavit jestli se bude modul chovat jako vysílač, přijímač, nebo bude v režimu spánku. Nastavení se provádí pomocí DIL přepínače. Pokud je nastaven na 1 jedná se o vysílač, 2 jde o přijímač, a pokud nebude nastaven žádný přepínač je modul v režimu spánku.

Po startu programu se provede inicializace periferií (porty, UART, vnější přerušení atd.) a inicializace rf modulu (nastavení modulu). Dále se resetuje buffer modulu a provede se výběr módu. Podle toho jaký mód je vybrán dokončí se nastavení modulu a modul začne plnit svou funkci. Vysílač vyšle data do přijímače, který je o příjmu dat informován vnějším přerušením. Potom se načte obsah bufferu z modulu a tyto data jsou odeslány přes UART kanál do počítače.

Podrobnější popsání nastavení modulu je uvedeno v příloze.



Obr. 5 Blokové schéma programu pro modul RFM12B



Obr. 6 Nastavení komunikace uzlu

5. ZigBee modul - ZigBit A2

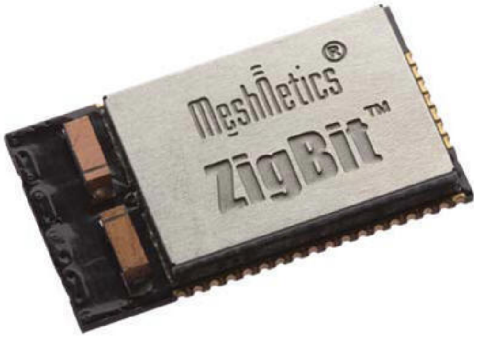
ZigBee modul transceiveru od firmy Meshnetics, později od firmy Atmel
ZDM-A1281-A2 - starší označení firmy Meshnetics
ATZB-24-A2 - novější označení firmy Atmel

Podporované externí rozhraní:

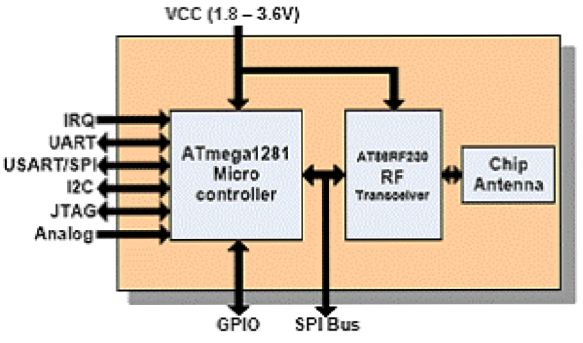
- USART/SPI, I2C, 1-wire
- UART s CTS/RTS kontrolou
- JTAG, ISP
- 9 GPIO (až 25 GPIO dohromady)
- 2 IRQ linie
- 4 ADC linie

Frekvence	868 MHz
Datová rychlost	20 kbps
Max. výstupní výkon	3 dBm
Citlivost	-101 dBm
Napájecí napětí	1.8 - 3,6V
Proudový odběr RX	19 mA
Proudový odběr TX	18 mA
Proudový odběr - spící režim	6 uA
Paměť Flash	128 kB
RAM	8 kB
EEPROM	4kB
Teplota	-40 + 85°C
Rozměry	13.5mm x 24mm

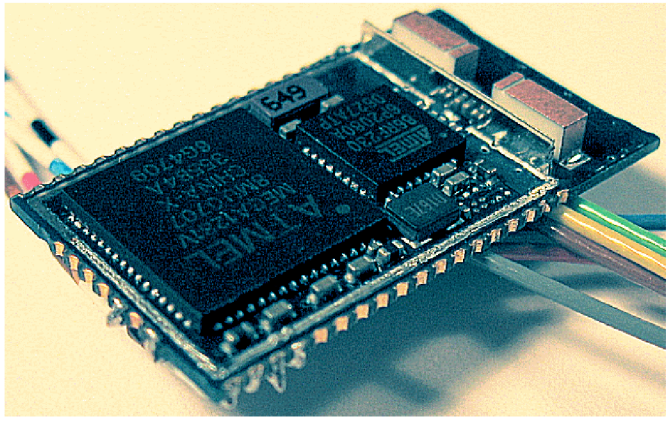
Tab. 3 Parametry modulu ZigBit-A2 (převzato z [7])



Obr. 7 Modul ZigBit-A2 (převzato z [4])



Obr. 8 Blokové schéma (převzato z [10])



Obr. 9 Pohled do útrob modulu (převzato z [14])

ZigBit piny		ATmega 1281 piny
1	SPI_CLK	PB1
2	SPI_MISO	PB3
3	SPI_MOSI	PB2
4	GPIO0	PB5
5	GPIO1	PB6
6	GPIO2	PB7
7	OSC32K_OUT	PG3
8	RESET	RESET
9	DGND	GND
10	CPU_CLK	XTAL_In
11	I2C_CLK	PD0
12	I2C_DATA	PD1
13	UART1_TXD	PD2
14	UART1_RXD	PD3
15	UART1_RTS	PD4
16	UART1_CTS	PD5
17	GPIO6	PD6
18	GPIO7	PD7
19	GPIO3	PG0
20	GPIO4	PG1
21	GPIO5	PG2

ZigBit piny		ATmega 1281 piny
22	DGND	GND
23	DGND	GND
24	D_VCC	VCC
25	D_VCC	VCC
26	JTAG_TMS	PF5
27	JTAG_TDI	PF7
28	JTAG_TDO	PF6
29	JTAG_TCK	PF4
30	ADC_INPUT_3	PF3
31	ADC_INPUT_2	PF2
32	ADC_INPUT_1	PF1
33	BAT	PF0
34	A_VREF	AREF
35	AGND	GND
36	GPIO_1WR	PG5
37	UART1_DTR	PE4
38	UART0_RXD	PE0
39	UART0_TXD	PE1
40	UART0_CLK	PE2
41	GPIO8	PE3
42	IRQ_7	PE7
43	IRQ_6	PE6

Tab. 4 Propojení ZigBit pinů a ATmega 1281 (převzato z [7])

Připojení pomocí spi

Piny spi komunikace (SPI_MISO a SPI_MOSI) Atmegy1281 jsou použity ke komunikaci s rf modulem AT86RF230 a proto není možné je využít jako vnější piny. Proto pro spi komunikaci musíme použít USART0, ale v tomto režimu lze použít spi pouze v režimu master.

Ze stejného důvodu nelze použít tyto piny k připojení isp programátoru a opět musíme využít piny USART0. Viz. schéma desky modulů v příloze.

6. Standard ZigBee

6.1 Základní vlastnosti

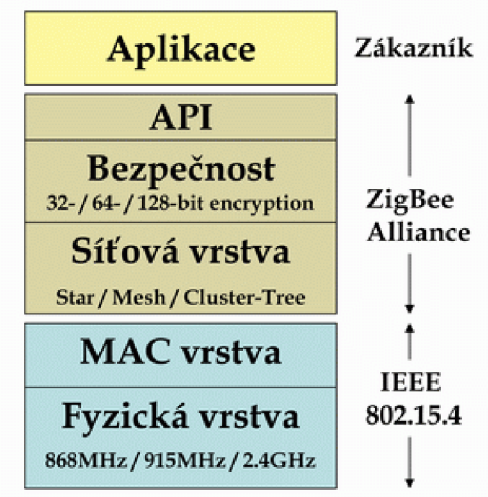
Jedná se o jednoduchý bezdrátový komunikační standard, který umožňuje komunikaci několika zařízení na vzdálenost 10 až 50 metrů. Tento bezdrátový komunikační standard spravuje organizace ZigBee Alliance a označuje se také jako IEEE 802.15.4.

6.2 Struktura

Fyzická struktura

Tři základní bloky OSI modelu (obr. 10):

- fyzická a linková vrstva je definovaná standardem IEEE 802.15.4
- síťovou a transportní vrstvu definuje ZigBee Alliance
- aplikační vrstvu definuje zákazník



Obr. 10 OSI model komunikačního protokolu ZigBee (převzato z [4])

Standard 802.15.4 definuje fyzickou a linkovou vrstvu (MAC vrstva). A právě fyzická vrstva určuje frekvenci, na které bude vysíláno.

Pracuje v tzv. ISM pásmech: 868MHz / 915MHz / 2,4GHz

- pásmo 868 MHz, 1 kanál, přenosová rychlost 20kb/s, (Evropa)
- pásmo 915 MHz, 10 kanálů, přenosová rychlost 40kb/s, (americký kontinent)
- pásmo 2.4 GHz, 16 kanálů, přenosová rychlost 250kb/s

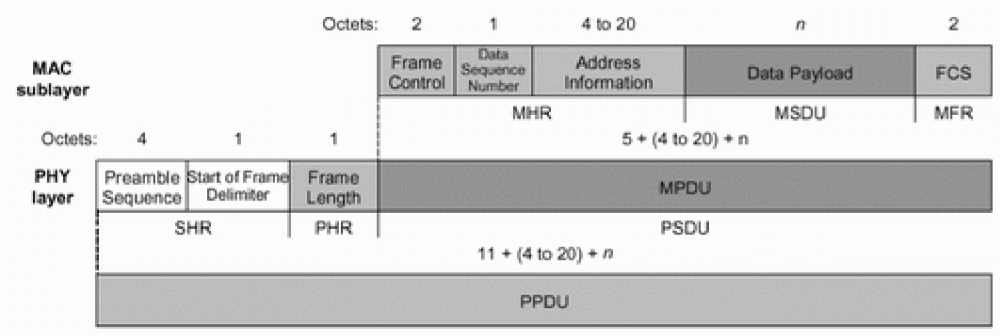
Vysílaný signál je modulován metodou BPSK nebo O-QPSK a je přenášen prostřednictvím DSSS (Direct Sequence Spread Spectrum). Pro přístup ke kanálu se využívá metoda CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance and optional time slotting)

Datová struktura

Linková vrstva přímo definuje komunikaci mezi jednotlivými zařízeními pomocí rámců. Tyto rámce jsou základní čtyři typy, které jsou používané buď pro přenos datových informací, nebo pro účely sestavování, správou a řízením sítě.

Čtyři základní rámce (obr. 11):

Data Frame – rámec pro přenos užitečné informace pro všechny datové přenosy
Acknowledgement Frame – rámec pro přenos potvrzovacích informací a je využitelný pouze na úrovni MAC pro potvrzovanou komunikaci
MAC Command Frame – rámec k centralizovanému konfigurování, nastavení a řízení klientských zařízení v síti
Beacon Frame – rámec k synchronizaci zařízení v síti. Je využíván hlavně při konfiguraci sítě a v módu v němž umožňuje uvádění klientských zařízení do spánkových režimů s extrémně sníženou spotřebou.



Obr. 11 Datový rámec standardu ZigBee (převzato z [4])

Na obr. 11 je zobrazen datový rámec paketu PPDU. Tento paket je definovaný standardem IEEE 802.11.15.4 a složený z MPDU (MAC vrstva) a SHR, PHR (fyzická vrstva).

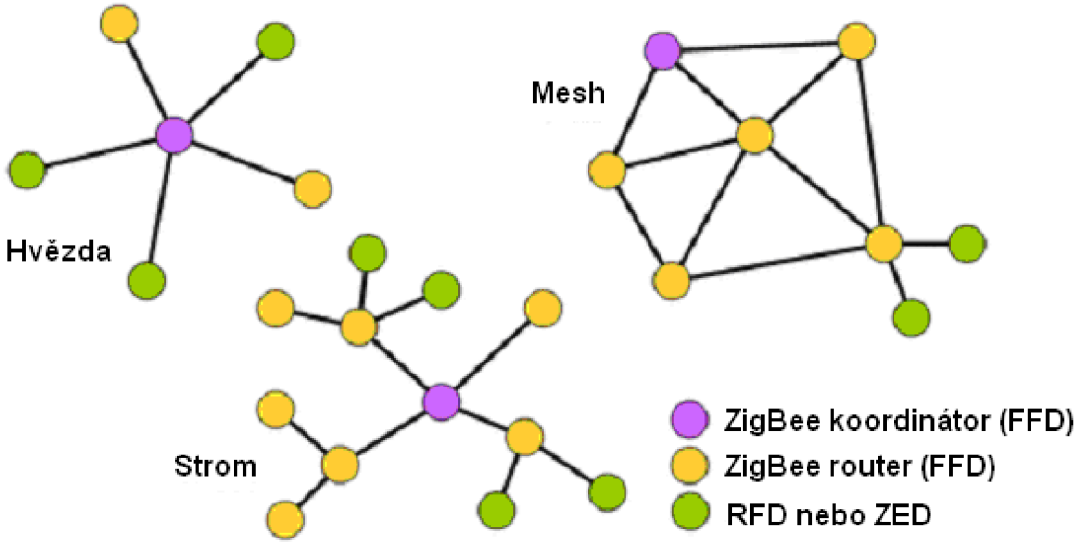
- MPDU obsahuje přenášená data, informace o adrese přijímající stanice, pořadové číslo datového paketu, řízení rámu a kontrolní mechanismus rámce (FCS – Frame Check Sequence).
- SHR, PHR obsahuje informace pro správný přenos paketu (např. délka rámce)

6.3 Topologie

Technologie ZigBee definuje tři síťové topologie (obr. 12). Základní topologií je hvězdicová s centrálním řídícím uzlem. Další typ je stromová struktura, která umožňuje zvětšit vzdálenost mezi řídícím uzlem a koncovým zařízením. Díky další topologii, kterou je Mesh síť, je možné prakticky libovolného uspořádání.

Standard ZigBee dělí na jednotlivá zařízení:

- FFD (Full Functional Device) – obsahují kompletní protokolový rámec a veškeré služby
- RFD (Reduced Functionality Device) – obsahují pouze nezbytné knihovny z důvodu omezené hardwarové obtížnosti.
- ZED (ZigBee End Device) – spící koncová zařízení



Obr. 12 Topologie realizovatelné standardem ZigBee (převzato z [4])

Zařízení jsou adresovaná pomocí binárního adresného kódu o délce 64 bitů, nebo ve zkrácené délce 16 bitů. Lokální zkrácená adresa umožňuje v jedné síti adresovat až 65535 zařízení. Každá takto vytvořená síť je ještě identifikovaná 16 bitovými PAN ID, které slouží pro rozlišení překrývajících se sítí.

Základem každé sítě je koordinátor, který přidělí PAN ID.

Koordinátor:
Hlavní odpovědnost koordinátora uzlu (Coordinator) je vytvořit síť s požadovanými vlastnostmi. Po vzniku sítě se mohou další uzly připojit pomocí koordinátoru nebo routeru, které, jsou již v síti. Koordinátor je schopen vykonávat funkci směrování dat. V síti je povolen pouze jeden.

Router:
Směrovač uzlu neboli router (Router) zajišťuje transparentní předávání údajů na ostatní uzly s cílovou adresou. Může také sloužit jako zdroj dat. Stejně jako koordinátor uzlu, jsou routery schopné fungovat jako vstupní bod sítě pro jiná zařízení a může sloužit jako přímý rodič pro koncové uzly zařízení.

Koncové zařízení:
Koncového zařízení (End Device) má nejméně organizačních schopností. Může pouze přijímat a odesílat data, které jsou vždy předány do / z destinace přes rodiče uzlu do koncového zařízení. Pouze koncová zařízení jsou schopné režimu spánku (viz dále).

6.4 Zabezpečení

V síťové vrstvě je implementováno základní zabezpečení AES (Advanced Encryption Standard) s klíčem o délce 64 nebo 128 bitů. [4]

6.5 Vývoj

ZigBee prošlo od svého "narození" v roce 2004 postupným vývojem.

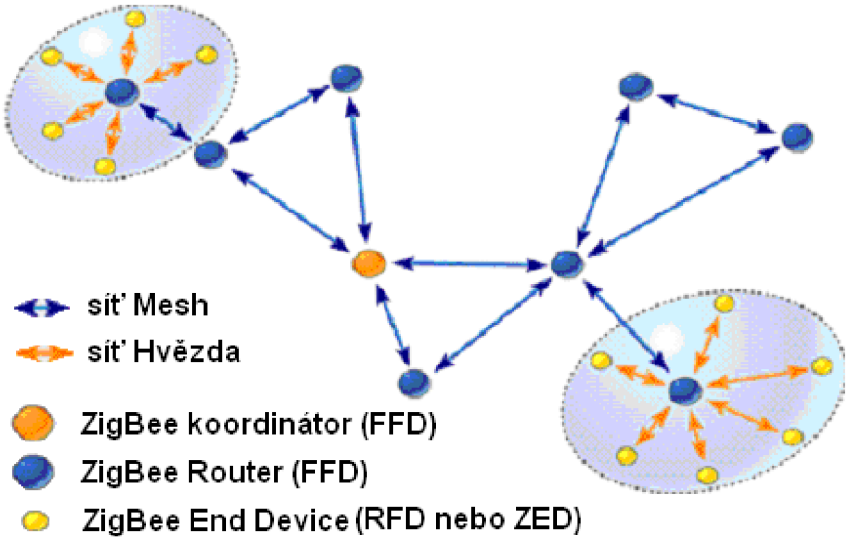
- ZigBee verze 1.0 (prosinec 2004)
- ZigBee verze 1.1
- ZigBee verze 2006
- ZigBee PRO (ZigBee 2007)

Pět komunikačních platforem (Golden Units):

- EmberZNet společnosti Ember
- BeeStack společnosti Freescale
- USB Dongle společnosti Integration Associates
- Z-Stack společnosti Texas Instruments
- AirBee firmy Airbee Wireless využívající hardwarovou platformu Texas Instruments
- BitCloud stack společnosti Atmel

6.6 ZigBee PRO

Postupným vývojem vznikly dvě verze ZigBee, které spolu neměly původně spolupracovat (klasické ZigBee x ZigBee PRO). Obě verze jsou však spolu za určitých podmínek použitelné a mohou spolu komunikovat. Základní rozdíl je v adresování.



Obr. 13 Topologie ZigBee PRO využívá jen MESH sítě (převzato z [4])

ZigBee PRO topologii strom zcela zavrhlo (obr. 13) a používá náhodné přidělování adres s mechanismem detekce kolizí adresování. Proto je potřeba neustále monitorovat sítě kvůli případným konfliktům. To má hlavní výhodu v tom, že pokud se dosáhne limitu adres lze snadno síť dále rozšířit. Nebo pokud by bylo použito několik mobilních koncových zařízení, které při pohybu přechází ze sítě do sítě. Náhodné přidělování adres umožňuje jednoduchou rozšiřitelnost, ale vyžaduje více času pro potřebu zjištěných možných konfliktů adres.

Dále podporuje tzv. limitování (omezení) adresování skupin. Jedná se o to, že chrání celou síť před zaplavením, když jsou všichni členové sítě umístěny v těsné blízkosti.

Proměnná vysílací frekvence:
Pokud je signál v ISM pásmu rušen (např. Wifi nebo Bleutooth) umožňuje ZigBee Pro zmírnit negativní dopady kolizí pomocí metody proměnné vysílací frekvence.

Správa - řízení napájení
ZigBee Pro neposkytuje síťový synchronizační mechanizmus pro koncová ZED zařízení. Místo toho se mohou koncová zařízení na určitý pevný časový usek (spící perioda) úplně vypnout.

Zpětná kompatibilita ZigBee a ZigBee PRO
Pokud už existuje síť složená ze ZigBee tak lze ZigBee PRO jednotky použít jako koncová zařízení. V sítích ZigBee, ZigBee PRO komunikuje jen přes směrovací zařízení, ale sami nemohou mít funkci směrování.
Nelze zaměnit ZigBee router se ZigBee PRO routerem. [5]

7. Programování modulu ZigBee

7.1 Způsob programování modulu pomocí BitCloud

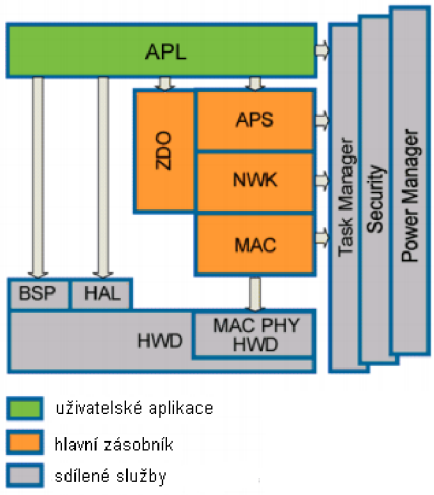
BitCloud stack je softwarový stack pro vývoj firmware pro bezdrátové aplikace od firmy Atmel. Tento software je určen jak pro domácí, tak i pro průmyslové aplikace. BitCloud je plně kompatibilní jak s ZigBee PRO tak i s ZigBee standard. Poskytuje také rozšířené rozhraní API, pomocí kterého lze použít rozšířené funkce.

Klíčové funkce:

- kompletní ZigBee PRO a ZigBee standardy
- jednoduché C používání API a sériových AT příkazů
- spolehlivé směrování v topologii Mesh
- velká podpora sítě – až 100 zařízení
- optimalizováno pro velmi nízkou spotřebu
- rozsáhlá bezpečnost API
- aktualizace software
- jednoduché použití vývojových nástrojů

Podporovaný hardware:

- SDK for ATAVRRZRAVEN: – kit
- SDK for ZigBit: – ATZB-DK-24
- SDK for ZigBit Amp – ATZB-DK-A24
- SDK for ZigBit 900 – ATZB-DK-900



Obr. 14 Architektura BitCloud (převzato z [6])

Architektura BitCloud

- APL (Application) - aplikace
- APS (Application support sub-layer) - podpora aplikací sub-vrstvy
- BPS (Board support package) - referenční ovladače pro podporu desky
- HAL (Hardware abstraction layer) - referenční ovladače pro podporované platformy
- ZDO (ZigBee Device Object) - objekt zařízení ZigBee
- MAC PHY (Media access control) - řízení přístupu a fyzická vrstva
- NWK - síťová vrstva
- SECURITY - bezpečnostní služba
- TASK MANAGER - správce úloh, zprostředkovává použití MCU mezi vnitřními komponenty a uživatelskými aplikacemi
- POWER MANAGEMENT ROUTINES - zodpovědné za vypínání všech komponent ukládání stavu systému při přípravě ke spánku a obnovení stavu systému při probuzení

[6]

Programovací styly:
1.Řízené události (event driven)
- pro systémy s omezenou pamětí
- vyvolání funkce pomocí API, asynchronním voláním
- dokončení, pomocí zpětného volání spojené s původní žádostí

2.Základní události (event based)

Žádost/potvrzení – indikační mechanizmus
Každá vrstva definuje počet volání na nižší vrstvy a ty naopak vyvolají funkci zpětného volání definované vyšší úrovní. Jednoduše řečeno, žádost je asynchronní volání do základního zásobníku, kde provede danou akci jménem uživatele aplikace a potvrdí je zpětným voláním, které se spustí, když je tato akce dokončena.
Uvažujme například: ZDO_StartNetworkReq(NetworkParams) je volání, které žádá ZDO vrstvu o spuštění sítě. NetworkParams je argument struktury definováné v zdo.h jako ZDO_StartNetworkReq_t. Poté funkce ZDO_StartNetworkConf() s argumentem typu ZDO_StartNetworkConf_t informuje zdali je zařízení k síti připojeno.
Kromě párů žádost - potvrzení párů, existují případy, kdy žádost musí být oznámena na vnější události, které nejsou odpovědí na konkrétní žádost. Pro toto, zde je řada uživatelsky definovaných volání s pevným jménem, která jsou uplatňována ve frontě asynchronně. Jedná se např. o připravenosti na spaní, nebo oznamující, že systém je nyní vzhůru.

Rozvrhové událostí
Hlavním aspektem vývoje aplikací je zajistit, aby různé volání nebyli v rozporu s provedením. Aby se neprovádělo další volání, pokud ještě nebude známa odpověď na první volání. Jedná se o to, že se nejprve vykonají krátké volání a ty delší se odloží.
Pro uživatelské aplikace je vyslané volání vždy označeno APL_TASK_ID. Vysílání úkolu vyústí v odložené volání handleru úkolu, APL_TaskHandler, který, na rozdíl od jiných volání, běží pod úrovní priority. V ostatních SYS_PostTask (APL_TASK_ID), je spuštěna pouze tehdy, když jsou dokončeny všechny s vyšší prioritou. To umožňuje delší dobu provedení úkolu v handleru.

Souběžnost a přerušení
Souběžnost odkazuje ve stejnou dobu na několik nezávislých vláken kontroly výkonu. V systému s timeslicing s kontrolou více vlákny, může být provedení jedné funkce, přerušena plánovacím systémem. Vzhledem k nepředvídatelnosti přerušení a skutečnosti, že obě funkce mohou sdílet data, musí se zajistit, aby aplikace měla okamžitý přístup ke všem sdíleným datům.
V BitCloud stack je jediné vlákno kontroly rozdělené mezi aplikaci a fronty. Spuštěním úlohy v dané vrstvě zásobníku, vlákno získává určitou prioritu.

Typická struktura BitCloud (významně se liší ve své organizaci od typického zakotveného C jazyku)
- Každá žádost definuje jeden úkol, který obsahuje část kódu (včetně kódu, přístupné prostřednictvím vnořených volání funkcí).
- Každá žádost definuje počet zpětných volání funkcí
- Každá žádost definuje počet volání se známými jmény
- Každá žádost globálně sdílí informace o stavu mezi voláním a obsluhou

Rozhraní konfigurace serveru:
Rozhraní BitCloud poskytuje rozsáhlý soubor konfiguračních parametrů, které určují chování sítě a uzlu. Tyto parametry jsou přístupné pro aplikace přes konfigurační-serverové rozhraní (ConfigServer, CS parametry).
Všechny CS parametry lze rozdělit do dvou kategorií: trvalá a netrvalá. Trvalé parametry jsou uloženy v paměti EEPROM a jejich hodnoty jsou dostupné pro použití i po HW resetu. Netrvalé parametry jsou uloženy v paměti RAM a po HW resetu se musí znovu inicializovat s jejich výchozími hodnotami.
ConfigServer.h soubor obsahuje poznámky parametru ID, zda jsou některé CS parametry trvalé, nebo ne.

Definice CS parametru v Makefile
Nejjednodušší metodou přiřazení hodnoty parametru CS je definovat ji v Makefile nebo Configuration soubvoru. V takovém případě je výchozí nastavení hodnot CS parametru v souboru configServer.h je přeskočen a hodnota je přidělena v Makefile.

Čtení a zápis CS parametrů
Funkce rozhraní API: CS_ReadParameter a CS_WriteParameter.
Obě funkce vyžadují parametr ID a ukazatel na hodnotu parametru jako argumenty. Parametr ID, který identifikuje CS parametr je podán, a je doplněn "_ID" na konci parametru CS name.adding "_ID". [6]

7.2 Start sítě

Typ zařízení je dán parametrem CS_DEVICE_TYPE typu DeviceType_t (ten lze nastavit na jednu z následujících hodnot):

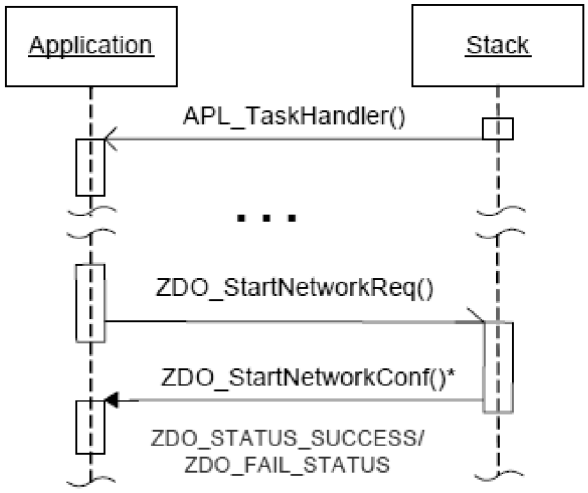
- Koondinátor: DEVICE_TYPE_COORDINATOR (nebo 0x00)
- Směrovač (router): DEVICE_TYPE_ROUTER (nebo 0x01)
- Koncové zařízení: DEVICE_TYPE_END_DEVICE (nebo 0x02)

Navíc logický parametr CS_RX_ON_WHEN_IDLE musí být nastaven na hodnotu true pro koordinátora a router sítě, zatímco u koncových zařízení, musí být nastavena na false.

Nastavení síťové adresy se provádí pomocí parametru CS_NWK_ADDR_ID. Zároveň musí být nastavena hodnota true v parametru CS_NWK_UNIQUE_ADDR_ID, aby adresa byla aktivovaná.

Žádost o start sítě inicializuje zahájení sítě provedením asynchronního volání ZDO_StartNetworkReq(). Po dokončení startu / připojení do sítě ZDO informuje o uplatnění výsledku zpětným volání s argumentem typu ZDO_StartNetworkConf_t() s parametrem ZDO_StartNetworkConf_t. Ten obsahuje stav provedených změn a informací o síti (např. síťovou adresu pro uzel). Stav ZDO_STATUS_SUCCESS se obdrží v případě, že postup je proveden úspěšně, zatímco status ZDO_FAIL_STATUS znamená, že start / připojení do sítě se nezdařil.

[6]



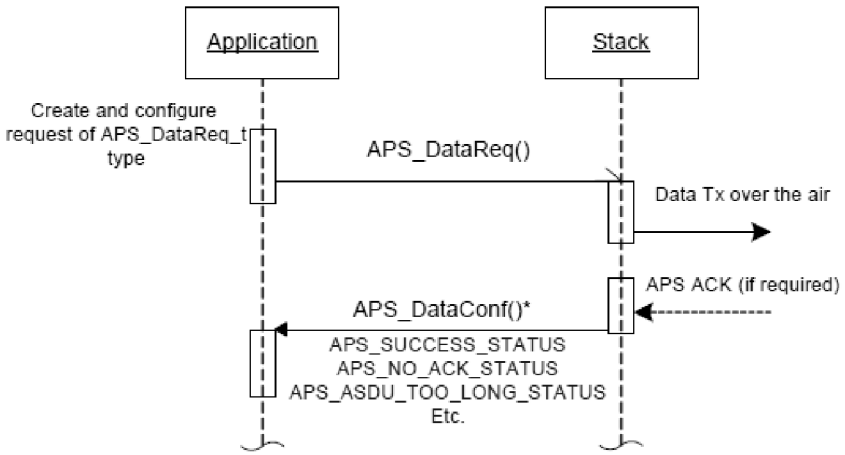
Obr. 15 Síťová startovní sekvence (převzato z [6])

Další funkce a datové typy jsou obsaženy v knihovně zdo.h.

7.3 Přenos dat v síti

Po připojení do sítě je další důležitou funkcí přenos dat mezi uzly. K uskutečnění přenosu je potřeba nastavit parametry uzlu, např. kam data přenášíme. To definuje parametr ve funkci žádosti o přenos dat APS_DataReq(&msgParams), kde je nastaveno: cílová adresa, koncový bod ID (1-240), aplikační profil ID, klastr ID, definice bufferu ASDU (obr. 18), požadavek o potvrzení příjmu atd. Funkce APS_DataInd určuje umístění zpětného volání funkce pro odeslání / přijetí dat.

Vysílání uzel - uzel
Po odeslání dat do místa určení, se automaticky najde nejspolehlivější cesta k požadovanému uzlu. Tyto cesty, tj. spojení mezi jednotlivými uzly jsou průběžně aktualizovány. Lze nastavit maximální počet skoků pro přenos dat. Nastavením čísla rádiusu v poli žádosti o přenos dat APS_DataReg_t. Jelikož protokol ZigBee je obousměrný, lze požadovat potvrzení o příjmu. To se docílí nastavením txOptions.acknowledgedTransmission na 1. Oznámení zda bylo doručení úspěšné je obsaženo v rámci APS_SUCCESS_STATUS, který nabývá:
APS_NO_ACK_STATUS nebo APS_SUCCESS_STATUS.



Obr. 16 Sekvence přenosu dat v síti (převzato z [6])

Všesměrové vysílání
Kromě přenosu uzel-uzel lze přenášet data tzv. všesměrovým vysíláním. Jedná se o to, že uzel vysílá data pro:

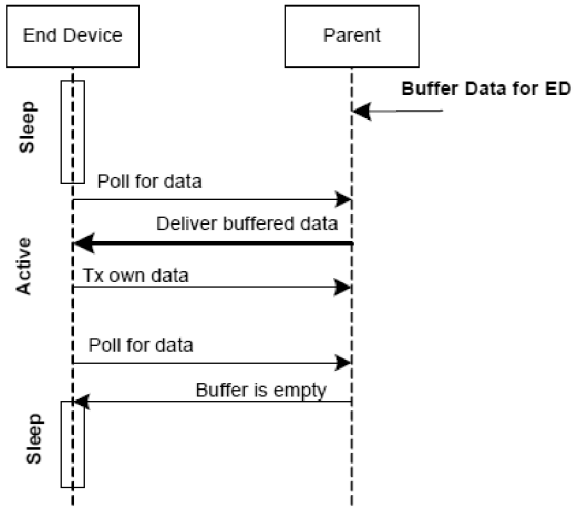
- všechny uzly v síti (BROADCAST_ADDR_ALL nebo 0xFFFF)
- všechny routery v síti (BROADCAST_ADDR_ROUTERS nebo 0xFFFC)

V případě všesměrového vysílání nelze použít potvrzení příjmu (txOptions.acknowledgedTransmission musí být 0).
Postup vysílání probíhá tak, že se datový rámec pošle 3 krát do "vzduchu". Každý uzel po příjmu jedné kopie datového rámce (ostatní jsou ignorovány) sníží přenosový rádius o jeden, a pokud je rádius stále větší, než jedna pokračuje ve vysílání "sousedům". Takto přenos pokračuje, dokud nebude přenosový rádius 0. Stejně jako při přenosu uzel-uzel, lze nastavit u rozhlasového vysílání počet skoků (nastavením velikosti rádiusu).

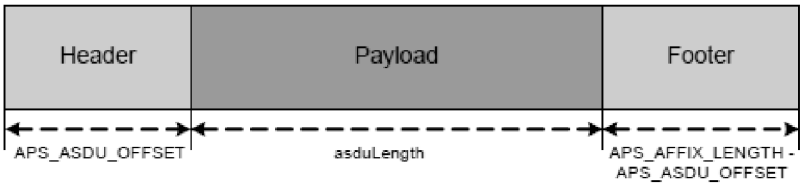
Vysílání dotazováním

Další způsob přenosu je tzv. dotazování. Tento typ přenosu může využívat pouze koncové zařízení, např. koncové zařízení a jeho nadřazený uzel (koordinátor nebo router). Jelikož koncové zařízení může být v době vysílání v režimu spánku a mohlo by dojít ke ztrátě dat, dotazuje se nadřazeného uzlu. Přenos probíhá takto: zařízení se po určitém čase probudí a dotazuje se nadřazeného uzlu, jestli nejsou data k doručení, pokud ano, provede se přenos. Poté se zařízení znovu dotazuje, a pokud již žádné data k příjmu nejsou, zařízení přejde opět do režimu spánku.

Vysílání dotazováním je definováno pomocí CS parametru CS_INDIRECT_POLL_RATE. Parametr je definován v ms a jeho standardní hodnota je 1000 ms. Hodnotu lze měnit v doporučeném rozmezí 200 ms až 10s. Toto vysílání je aktivováno zápisem tohoto parametru.



Obr. 17 Sekvence přenosu dat koncovému zařízení (převzato z [6])



Obr. 18 Formát ASDU pole (převzato z [6])

Maximální velikost přenášených dat je 84 bytů v případě nezabezpečeného přenosu a 53 bytů při zapnutí standardního zabezpečení.

[6]

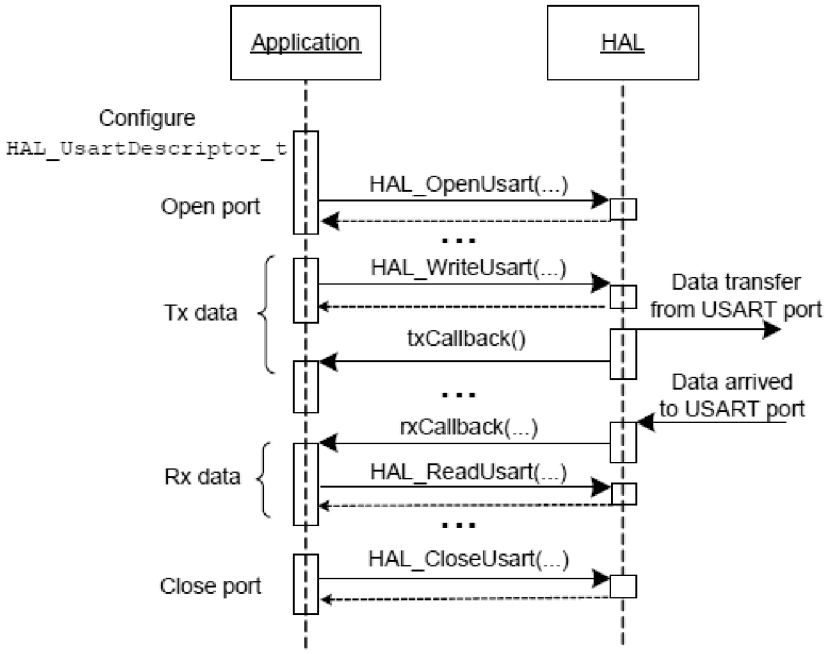
Další funkce a datové typy jsou obsaženy v knihovně aps.h.

7.4 Přenos dat pomocí USART

Pro přenos dat pomocí USART je potřeba nejdříve nastavit jeho parametry v globální proměnné typu HAL_UsartDescriptor_t. Zde je nastaveno: přenosová rychlost, synchronní nebo asynchronní mód, flow control, zpětné volání, módy parity, buffery, atd. Pro otevření USART kanálu je poté nutné použít funkci HAL_OpenUsart() s argumentem typu HAL_UsartDescriptor_t. Pro uzavření kanálu se použije funkce HAL_CloseUsart(). USART lze využívat ve dvou módech, mód callbacku a mód dotazování.

Mód callbacku

Funkce přenosu dat HAL_WriteUsart() je volaná s argumentem ukazatelem přenášených dat a délkou dat. Pokud je vrácena hodnota větší než nula, je provedena funkce definovaná jako txCallback v HAL_UsartDescriptor_t. USART je schopen přijímat data pokud není v polích rxBuffer a rxBufferLength NULL (0 je respektovaná). Volání režimu rxCalback je pokaždé když jsou v rxBufferu přijatá data. Pro přenos přijatých dat z rxBufferu do vyrovnávací paměti se použije funkce HAL_ReadUsart(), která vrací data a délku dat.

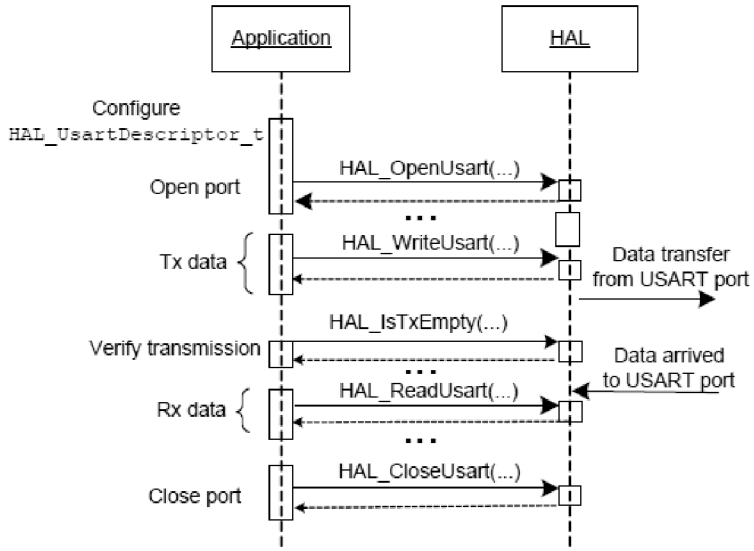


Obr. 19 Přenos dat pomocí USART - callback mode (převzato z [6])

Mód dotazování

V režimu dotazování se cyklicky využívají příslušné buffery, definované v HAL_UsartDescriptor_t, ty nesmí být nastaveny na NULL, délka bufferů musí být nenulová a callback musí být NULL. Hlavní rozdíl je v tom, že po volání funkce HAL_WriteUsart() se data cyklicky přesouvají to txBufferu. Potom může hned aplikace obsadit volné místo daty. Poté funkce HAL_IsTxEmpty() ověří, jestli je dostatek místa v bufferu nebo ověří, kolik bytů bylo skutečně přeneseno.
Na rozdíl od módu callbacku aplikace neupozorní na příjem dat, avšak data jsou cyklicky uložena do rxBufferu a aplikace k nim má přístup pomocí funkce HAL_ReadUsart().

[6]



Obr. 20 Přenos dat pomocí USART - mód dotazování (převzato z [6])

Další funkce a datové typy jsou obsaženy v knihovně usart.h.

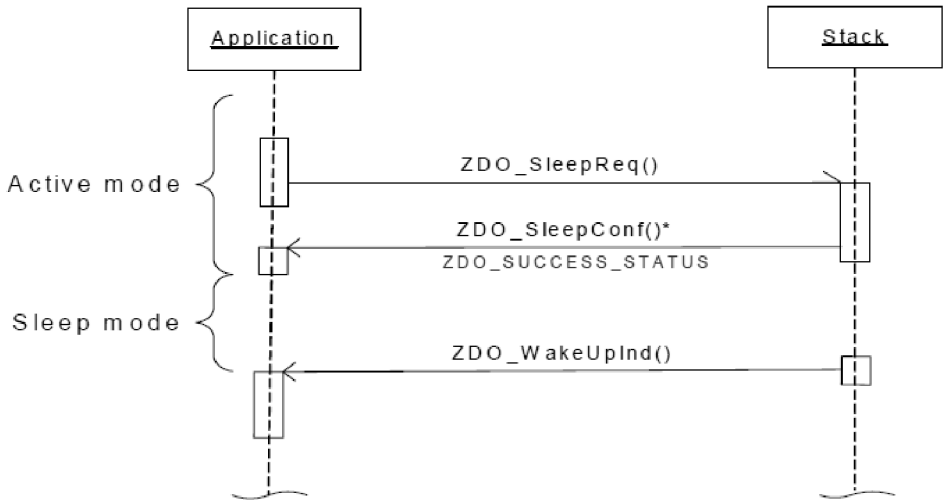
7.5 Správa napájení

V sítích ZigBee lze jednoduše přepínat mezi režimy spánky a aktivity. BitCloud v základní verzi podporuje mechanizmy správy napájení pouze u koncových zařízení.

Koncový uzel může být buď v aktivním režimu, nebo v režimu spánku, nezávisle na jeho stavu vůči síti (připojen k síti nebo ne). Po zapnutí je uzel vždy v aktivním režimu, MCU je plně zapnutý a RF čip je také zapnutý. V klidovém režimu je RF čip vypnutý a MCU pracuje ve speciálním stavu nízké spotřeby. Nelze provádět funkce přenosu a obsluhovat vnější obvody.
Funkce uspání uzlu je ZDO_SleepReq() s argumentem typu ZDO_SleepReq_t. Pokud je po potvrzujícím zpětným voláním vrácen status ZDO_SUCCESS_STATUS vstoupí uzel do režimu spánku.
Rozlišují se dva druhy probuzení ze spánku: časový rozvrh aktivity a zapnutí vstupním přerušením (např. pomocí tlačítka).

Časový rozvrh aktivity

Automaticky po uplynutí časového intervalu CS_END_DEVICE_SLEEP_PERIOD se uzel aktivuje. Interval je uveden v milisekundách a je specifikován v ConfigServer nebo v souboru Configuration. Aplikace je upozorněna na přepnutí na aktivní mód pomocí funkce ZDO_WakeUpInd. CS_END_DEVICE_SLEEP_PERIOD se nedá měnit v během programu.

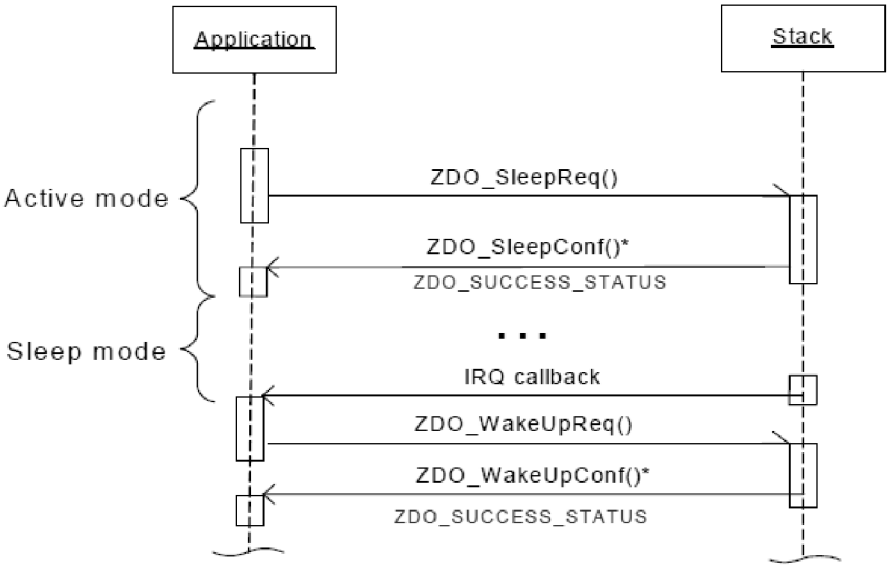


Obr. 21 Vzbuzení uzlu časovým rozvrhem aktivity (převzato z [6])

Zapnutí vstupním přerušením

Po sepnutí IRQ se MCU přepne do aktivního režimu a provede se definovaná IRQ událost. Protože se to, ale týká pouze HAL komponentů, je potřeba pro aktivaci RF čipu informovat síťovou vrstvu. To lze pomocí funkce ZDO_WakeUpReq. Když zpětné volání tohoto požadavku vrátí status ZDO_SUCCESS_STATUS je celý uzel v aktivním režimu. Pokud je uzel vzbuzen pomocí IRQ přerušení, je časovač CS_END_DEVICE_SLEEP_PERIOD zastaven a bude restartován po příštím volání funkce ZDO_SleepReq().

Obě dvě možnosti vzbuzení uzlu lze kombinovat.
Např. je-li CS_END_DEVICE_SLEEP_PERIOD nastavena na nulu, pak lze uzel vzbudit pouze pomocí IRQ přerušení.



Obr. 22 Vzbuzení uzlu zapnutím vstupním přerušením (převzato z [6])

Vypnutí pouze RF čipu

V některých případech je potřeba, aby MCU zůstal pracovat (např. při výpočtech) a je potřeba pouze vypnout RF čip. Funkce pro vypnutí je ZDO_StopSyncReq() a pro zapnutí je ZDO_StartSyncReq().

[6]

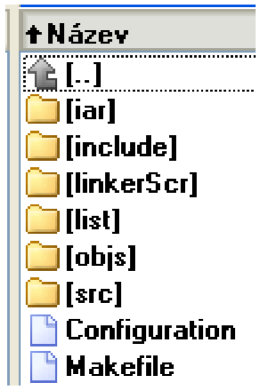
Další funkce a datové typy jsou obsaženy v knihovně zdo.h.

7.6 Vytvoření programu

Struktura složek:
Pro tvorbu nového programu využijeme složku „BlankProject“, v které jsou obsaženy soubory pro obsluhu všech jednotlivých periferií a programového vybavení.

V této složce je pro nás důležitá složka “include“, kde najdeme knihovnu “Periferie.h“ pro základní obsluhu ZigBee. Dále ve složce „src“ jsou soubory *.c, které obsahují obslužné programy pro ZigBee. Dále např. složka „linkerScr“ obsahuje definici atmelu 1281.

V Souboru „Configuration“ jsou obsaženy tyto informace:
- cesta ke knihovnám BitCloud
- jméno projektu (**je potřeba změnit!**)
- typ kompilátoru
- typ a nastavení desky (co obsahuje deska modulu)
- typy uzlů (C, R, ED)
- parametry nastavované do Config Serveru
- adresace uzlů
- nastavení zabezpečení



Obr. 23 Struktura složek

V Souboru „Makefile“ jsou obsaženy tyto informace:
- konfigurační parametry
- parametry kompatibility
- aplikační parametry
- parametry nastavované do Config Serveru
- specifikace kompilátoru
- cesta ke knihovnám BitCloud
- cíle kompilace projektu

Po kompilaci projektu se do kořenové složky vygenerují soubory *.hex, *.srec, *.elf a *.bin.
V programu Avrstudio se v nastavení konfigurace nastaví, aby se používal externí makefile. Složka s programem musí být v adresáři BitClout.

Alias
Pro funkce a datové typy uvedené v předchozích kapitolách jsou vytvořené alias programovacího kódu c., aby bylo programování pomocí BitCloud přehlednější a jednodušší. Jenomže z důvodu zjednodušení je používáno alias alias. Např. funkce BSP_OpenLeds() definovaná v knihovně leds.h má pro zjednodušení alias appOpenLeds(). Některé funkce mají až několik alias. Použití alias v programu není nijak limitováno, program zpracuje jakékoli definované alias. Jen z důvodu přehlednosti je nejlepší používat podobné alias. Změnu alias lze najít v knihovnách programu. Proto v ukázkovém programu nenajdete základní alias HAL_WriteUsart(), ale appWriteUsart(). To a ostatní použité alias jsou předefinovány v knihovně Periferie.h. V příloze popisuji vytvoření programu pomocí alias z knihovny periferi.h.

7.7 Popis programu pro modul ZigBee

Nastavení desky:

Nejdříve je potřeba připojit napájení a komunikaci, pokud ji budeme používat. Dále je nutné správně nastavit, jaký uzel bude modul realizovat. To se provede pomocí DIL přepínače. Koordinátor je 1, router je 2 a koncové zařízení je 3. Dále z důvodu snížení spotřeby vypnout modul RFM 12B.

Funkce jednotlivých uzlů
Koordinátor má za úkol vytvořit síť a nastavit bezpečnost, poté se stará o přenos dat v síti a obsluhu periférií. Jak router, tak i koncové zařízení se připojí do sítě vytvořené koordinátorem. A poté obsluhují své periférie a přenosy dat. Rozdíl mezi routerem a koncovým zařízením je to, že router nemůže přejít do režimu spánku.

Program se dělí na šest základních částí:
- inicializace uzlu
- start sítě
- uzel je v síti
- uzel mimo síť
- ovládání pomocí tlačítek
- příjem dat

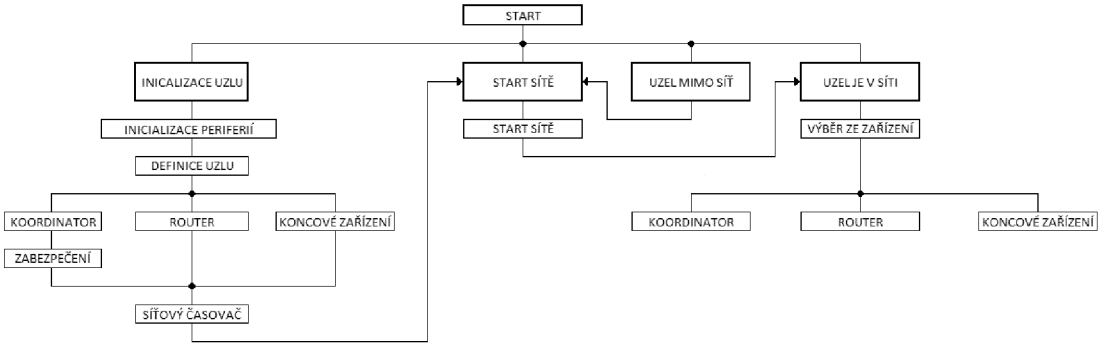
Průběh programu:

Nejprve proběhne inicializace uzlu. Jedná se o inicializaci periferií (přepínač, tlačítka, UART, led diody, atd.), poté je zjištěno nastavení DIL přepínače a podle toho nastavena úloha modulu (typ zařízení, síťová adresa a bezpečnost). Inicializace končí spuštěním síťového časovače. Další částí se spustí start sítě. Jedná se buď o vytvoření sítě, nebo o připojení do sítě. Pokud je uzel připojen do sítě, pokračuje se další částí - uzel je v síti. Zde se provede výběr funkce modulu podle jeho předešlého nastavení. Pokud se uzel z nějakého důvodu dostane mimo síť, pokouší se uzel o opětovné připojení do sítě.

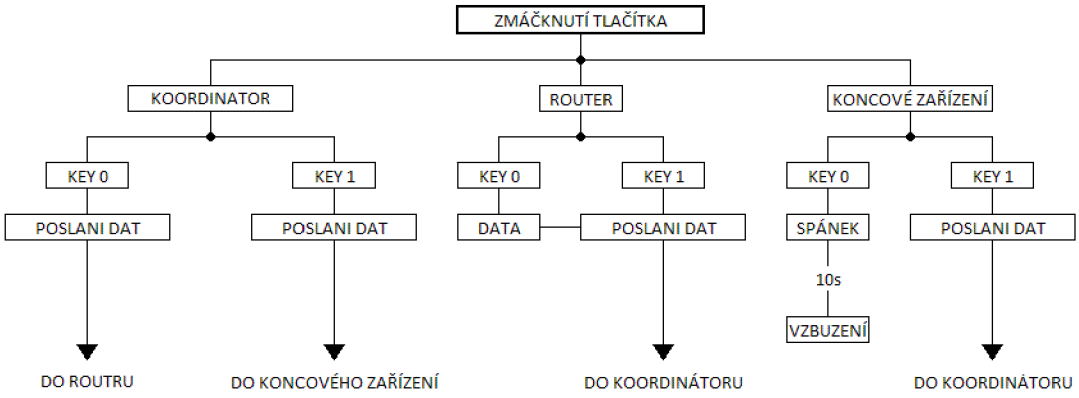
Po zmáčknutí jednoho, či druhého tlačítka se provede výběr, jaké funkce budou použity. Výběr opět záleží podle nastavení modulu v uzlu. V případě, že se jedná o koordinátor, jsou data poslána do routeru nebo koncového zařízení, podle stisknutého tlačítka. U routeru se nastaví počtem stisknutí tlačítka 0 číslo, které se druhým tlačítkem odešle do koordinátoru. A pokud je využit modul jako koncové zařízení je uzel možné pomocí tlačítka 0 na 10 sekund uspat. Poté se probudí a je možné pomocí druhého tlačítka posílat data do koordinátora.

Poslední část se věnuje přijetí dat jednotlivými uzly a další zpracování dat. Data jsou odeslaná pomocí UART kanálu do počítače, kde je lze zobrazit např. pomocí terminálu. Nastavení terminálu pro připojení modulů je vidět na obr. 27.

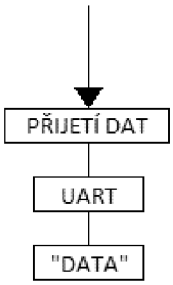
Blokové schémata programu:



Obr. 24 Základní části programu



Obr. 25 Ovládání pomocí tlačítek



Obr. 26 Přijetí a zobrazení dat

Obr. 27 Nastavení komunikace uzlu

8. Deska s osazenými rf moduly

Napájení desky
Lze zvolit mezi napájením z usb, z baterie anebo z adaptéru. Volba se provede pomocí jumperu JP5. Bateriové napájení lze řešit např. pomocí dvou článků AAA 1,5V baterií. Baterie je také připojena na modul ZigBee, pin BAT, pomocí kterého lze hlídat velikost napětí baterie. Pokud je deska propojena s počítačem pomocí usb lze zvolit toto napájení. Napájení z adaptéru je řešeno pomocí nízkopříkonového stabilizátoru LE33. Jeho maximální vstupní napětí je však max. 20 DCV, které je stabilizováno na 3,3V. Obvod je zapojen dle datasheetu [11].

Komunikace desky a počítače
Desku lze připojit k počítači pomocí konektoru USB nebo COM. Každý konektor má samostatnou desku, která se připojí na konektor JP4. Pro komunikaci přes usb je použit obvod FT232RL. Jedná se o obvod realizující rozhraní mezi USB a UART. Jelikož deska pracuje na napětí 3,3V je obvod využit i jako dc měnič. Deska je propojena pomocí signálů RS232 (RXD, TXD, RTS, CTS) a signálem PWREN#, který může sloužit k probouzení modulu ZigBee. Obvod je zapojen dle datasheetu [12]. Pro komunikaci přes sériovou linku, je použit obvod MAX3232CPE. Obvod transformuje napětí z 3,3V na 5V logiku linky RS232. Pro komunikaci slouží signály linky RS232(RXD, TXD, RTS, STC). Zapojení obvodu vychází z datasheetu [13].

Způsoby programování desky
Desku lze naprogramovat dvěma způsoby, buďto pomocí JTAG (Joint Test Action Group), nebo pomocí ISP (In-System Programming). JTAG konektor SV1 je 10-pinový a ISP konektor SV2 JE 6-pinový.

RF modul RFM12B
Přepínačem S5 se připojí napájení k modulu - zapne se. Modul RFM12B je připojen na porty PB a PD procesoru ATmega1281, který je obsažen v modulu ZigBee. Modul je připojen přes odpory, aby nedošlo k poškození jak modulu RFM12B nebo ZigBee. Komunikace mezi procesorem a modulem je řešena pomocí softwarové spi komunikace.

Obslužné a signalizační prvky desky
Deska se zapíná pomocí přepínače S6. Zapnutí desky signalizuje zelená dioda (LED4). Další tři diody, červená, žlutá a zelená slouží k signalizaci chodu programu, např. hledání sítě, připojení, odeslaní zprávy atd. K obsluze desky slouží DIL přepínač, dvě tlačítka. K resetu desky slouží tlačítko S3.

Ostatní možnosti desky
Aby byla deska co nejvíce univerzální, jsou piny, které nebyly využity pro předchozí funkce vyvedeny na konektory.

- GPIO piny JP1
- Tři ADC vstupy JP2
- USART0 piny JP3 (UART/SPI)

Pro použití ADC vstupů, je třeba zapojení referenčního napětí na pin A_VREF o velikosti 1,2V. Tato reference je získána pomocí nízkopříkonového stabilizátoru LE12, který je připojen k napětí V_CC. Obvod je zapojen dle doporučení výrobce [11]

Spotřeba desky

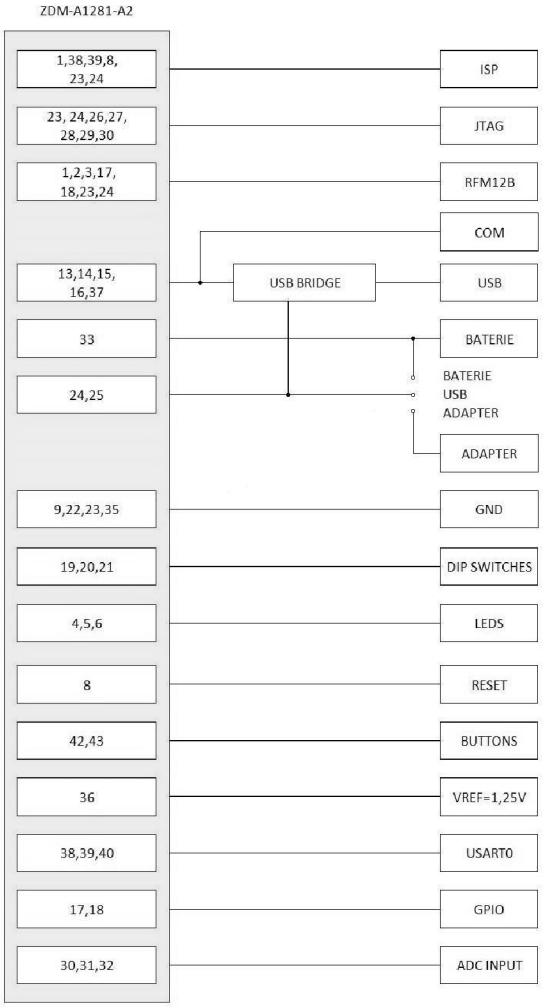
mód uzlu	Coodinator	Router	Koncové zařízení
po startu	41 mA	41 mA	29 mA
všechny tři diody	44 mA	44 mA	39 mA
režim spánku	-	-	16 mA

Tab. 5 Tabulka spotřeby desky při používání ZigBee

mód uzlu	Vysílač	Přijímač
min.	35 mA	38 mA
max.	42 mA	44 mA

Tab. 6 Tabulka spotřeby desky při používání RFM12B

Blokové schéma desky



Obr. 28 Blokové schéma desky modulu

9. Závěr

Nejdříve sem prostudoval volné tzv. ISM pásmo. Toto pásmo má několik frekvenčních rozsahů, které se liší podle toho, na jaké části planety Země se nacházíme. Pro Evropu platí frekvence: 433 MHz, 868 MHz a 2,4GHz. Pokud bychom se nacházeli v Severní Americe, můžeme použít frekvenční pásmo 915 MHz. V těchto pásmech se mohou používat pouze homologované bezdrátové zařízení. Hlavní nevýhodou těchto pásem je to že se může vyskytnout rušení, které zhorší přenos dat. V dnešní době je hodně používané pásmo 2,4 GHz. Zde se objevují zařízení typu Bluetooth, Wifi, ZigBee, atd.

Jako první modul jsem zvolil RFM12B. Jedná se o bezdrátový modul, který pro komunikaci potřebuje procesor. Jako procesor jsem zvolil Atmega1281, který řídí celý přenos a modul pouze zakóduje a vyšle data do dalšího modulu. Tento modul jsem vyzkoušel v praxi a použitý zdrojový kód je přiložen.

V další části jsem prozkoumal standard ZigBee, který je často používán v průmyslu. Jedná se už o složitější moduly, které již obsahují několik vrstev. Tyto moduly už mají svůj procesor a také svůj protokol. Oproti jednoduchým ISM modulům mohou zařízení v ZigBee standardu spolu navazovat spojení v několika topologiích. Základní topologií je hvězdicová s centrálním řídícím uzlem, další typy jsou stromová struktura a Mesh síť. Zařízení v tomto standardu mohou využívat tzv. *režim spánku*, kdy má zařízení velmi malý odběr a při potřebě komunikace se aktivuje, vykoná potřebné instrukce a pak opět přejde do *režimu spánku*. Díky tomuto režimu se může použít u těchto zařízení, bateriové napájení. Dalším velkým přínosem ZigBee je tzv. funkce *proměnné vysílací frekvence.* Ta umožňuje v případě rušení vysílací frekvence tuto frekvenci změnit.

Po seznámení se standardem ZigBee, jsem se vrhnul na možnosti programování a nastavení tohoto modulu. K jeho programování doporučuje firma ATMEL použít vývojové prostředí BitCloud stack. Pro komunikaci s modulem je možné použít jak JTAG rozhraní, tak i ISP. Způsob programování je odlišný od typického C jazyku. Programuje se speciálními příkazy, z nichž některé fungují jako žádost/potvrzení. Proto se nemalá část práce zaměřuje právě na způsob a možnosti programování.

Výsledkem práce je realizovaná univerzální deska s moduly. Desku lze možno napájet pomocí baterie, adaptéru nebo z usb. Dále ji lze připojit přes usb nebo com k počítači. Lze ji naprogramovat pomocí ISP nebo JTAG programátoru. Hlavní části je modul ZigBee a jeho periferie, ke kterým jsou připojeny obslužné prvky. Jedná se o signalizační diody, tlačítka, či přepínače atd. Jako mikroprocesor pro obsluhu modulu RFM12B je využit procesor modulu ZigBee, ATmega1281. Program pro modul ZigBee obsahuje základní funkce pro obsluhu modulu. Jedná se např. o definici uzlu, vytvoření a přijetí sítě, přenos dat mezi jednotlivými uzly, komunikace s počítačem přes USART kanál. Program pro RFM12B ukazuje jednoduchou výměnu dat mezi mikroprocesory. Pro oba moduly jsou vytvořeny knihovny funkcí Periferie.h.

Tato práce se dá použít jako návod na programování modulů ZigBee od firmy Atmel. Protože deska je kompatibilní i s ukázkovými programy firmy Atmel.

Literatura

[1] Wirelessnetdesignline.com server. Avoiding Interference in the 2.4-GHz ISM Band. [Online]. [cit. 19. května 2011].Dostupné na WWW: http://www.wirelessnet designline.com/howto/60401206

[2] Hope RF. RFM12b datasheet. [Online]. [cit. 19. května 2011]. Dostupné na WWW: http://www.hoperf.com/upfile/RF12B_code.pdf

[3] Atmel.com server - BitCloud - ZigBee PRO download. [Online]. [cit. 19. května 2011].Dostupné na WWW:http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4495

[4] Hw.cz server. ZigBee - novinka na poli bezdrátové komunikace. [Online]. [cit. 19. května 2011]. Dostupné na WWW: http://hw.cz/Rozhrani/ART1299-ZigBee---novinka-na-poli-bezdratove-komunikace.html

[5] Automatizace.hw.cz server. ZigBee PRO - nová vylepšená verze bezdrátové komunikace ZigBee. [Online]. [cit. 19. května 2011].Dostupné na WWW: http://automatizace.hw.cz/zigbee-pro-nova-vylepsena-verze-bezdratove-komunikace-zigbee

[6] AVR2050: BitCloud User Guide. [Online]. [cit. 19. května 2011].Dostupné na WWW: www.atmel.com/dyn/resources/prod_documents/doc8199.pdf

[7] MeshNetics ZigBit™ Modules datasheet. [Online]. [cit. 19. května 2011]. Dostupné na WWW: http://obchod.hw.cz/img.asp?attid=6060

[8] Obrázek modulu RFM12BS. [Online]. [cit. 19. května 2011]. Dostupné na WWW: http://pvelectronic.inshop.cz/inshop/catalogue/products/pictures/09582-01.jpg

[9] Hope RF. RFM12b datasheet. [Online]. [cit. 19. května 2011]. Dostupné na WWW: http://www.hoperf.com/upfile/RF12B.pdf

[10] MeshNetics ZigBit™ Modules datasheet. [Online]. [cit. 19. května 2011]. Dostupné na W:http://adaptivemodules.com/assets/File/zigbit_oem_module_zdm-a1281_datasheet.pdf

[11] Datasheet obvodů LE33 a LE12. [Online]. [cit. 19. května 2011]. Dostupné na WWW: http://www.datasheetarchive.com/pdf-datasheets/Datasheets-31/DSA-605053.html

[12] Datasheet obvodu FT232R od firmy FTDI chip. [Online]. [cit. 19. května 2011]. Dostupné na WWW: http://www.ftdichip.com/Support/Documents/DataSheets/ICs/ DS_FT232R.pdf

[13] Datasheet obvodu MAX3232 od firmy MAXIM. [Online]. [cit. 19. května 2011]. Dostupné na WWW: http://datasheets.maxim-ic.com/en/ds/MAX3222-MAX3241.pdf

[14] Obrázek modulu ZigBee. [Online]. [cit. 19. května 2011]. Dostupné na WWW: http://www.mikrocontroller.net/wikifiles/3/33/ZDM-A1281-A2_offen.gif

Seznam symbolů, veličin a zkratek

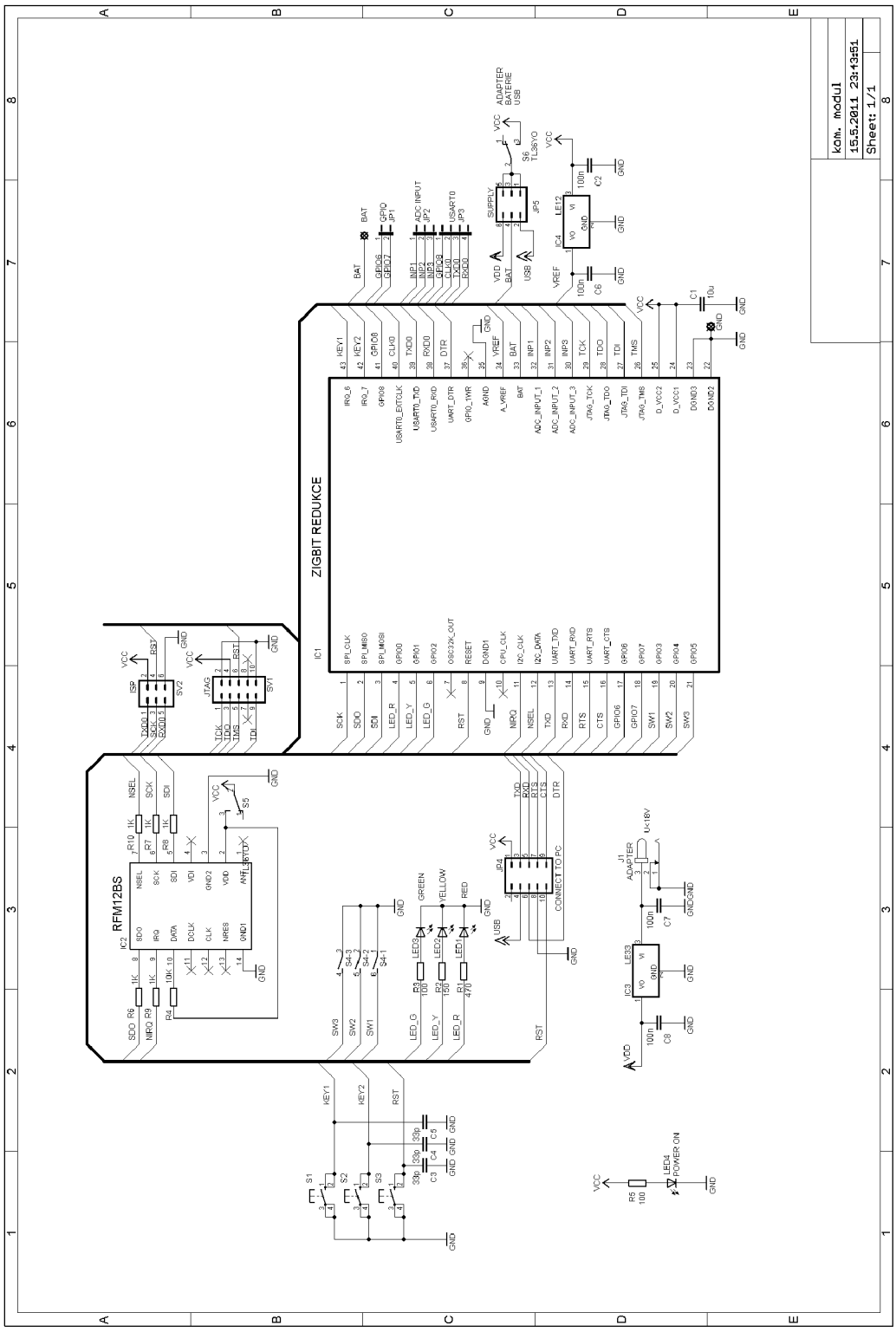
C	- koordinátor sítě (Coordinator)
°C	- jednotka teploty
1-wire	- připojeni pomocí jednoho vodiče
ADC	- analogově/digitální převodník
AES	- standard zabezpečení (Advanced Encryption Standard)
API	- rozhraní pro programování rozhraní (Application Programming Interface)
APL	- aplikace (Application)
APS	- podpora aplikací sub-vrstvy (Application support sub-layer)
AREF	- referenční napětí pro ADC převodník
ASDU	- buffer pro přenos dat v síti (Application-layer Service Data Unit)
AT	- příkaz pro naplánování příkazu (AT příkaz)
bit	- jednotka velikosti dat
BitCloud	- softwarový stack pro programování ZigBee
BPS	- referenční ovladače pro podporu desky (Board Support Package)
BPSK	- binární – fázové klíčování (Binary-Phase Shift Keying)
COM	- standard RS-232
CSMA/CA	- vícecestný přístup (Carrier Sense Multiple Access with Collision Avoidance)
CTS	- pin uart komunikace (Clear To Send)
dBm	- jednotka výkonu
DGND	- digitální zem (Digital Ground)
DIL	- přepínač na desce
DSSS	- přímé rozprostřené spektrum (Direct Sequence Spread Spectrum)
EEPROM	- paměť procesoru (Electrically Erasable Programmable Read-Only Memory)
ED	- koncové zařízení (End Device)
FCS	- kontrolní mechanismus rámce (Frame Check Sequence)
FFD	- zařízení se všemi funkcemi (Full Functional Device)
FHSS	- frekvenční rozprostřené spektrum (frequency hopping spread spectrum)
FSK	- frekvenční modulace (Frequency Shift Keying)
GHz	- jednotka frekvence
GND	- zem (Ground)
GPIO	- vstupně výstupní piny (General Pin Input Output)
HAL	- referenční ovladače pro podporované platformy (Hardware abstraction layer)
HW	- hardware
I2C	- připojení pomocí dvou vodičů (data + hodiny)
IEEE802.15.4	- standard ZigBee
ID	- identifikace
IRQ	- požadavek na přerušení (Interrupt ReQuest)
ISM	- celosvětová bezlicenční frekvenční pásmo (industrial, scientific and medical)
ISP	- způsob programování zařízení (In-system Serial Programming)
JTAG	- způsob programování zařízení (Joint Test Action Group)
kB	- jednotka velikosti dat

kb/s	- jednotka přenosové rychlosti
kbps	- jednotka přenosové rychlosti
kHz	- jednotka frekvence
m	- jednotka délky
mA	- jednotka proudu
MAC	- řízení přístupu (Media Access Control)
MAC PHY	- řízení přístupu fyzické vrstvy (Media Access Control PHYsical)
MCU	- mikroprocesor
mm	- milimetr - jednotka délky
MHz	- jednotka frekvence
MPDU	- část datového rámce ZigBee
NWK	- síťová vrstva (NetWorK)
O-QPSK	- Offset Quadrature Phase Shift Keying
OSI	- model systému (Open Systems Interconnection)
PA	- port mikroprocesoru ATmega1281
PB	- port mikroprocesoru ATmega1281
PC	- port mikroprocesoru ATmega1281
PD	- port mikroprocesoru ATmega1281
PE	- port mikroprocesoru ATmega1281
PF	- port mikroprocesoru ATmega1281
PG	- port mikroprocesoru ATmega1281
PAN ID	- 16-bit unikátní identifikátor
PHR	- část datového rámce ZigBee
PPDU	- datový rámec ZigBee
R	- router
RAM	- paměť procesoru (Random-Access Memory)
RFD	- zařízení s redukovanými funkcemi
Router	- směrovač sítě
RS232	- telekomunikační standard (Recommended Standard 232)
RTS	- pin uart komunikace (Request to Send)
RX	- příjem (Receive)
RXD	- pin uart komunikace (Receive Data)
SDK	- softwarový vývojový kit (Software Development Kit)
SHR	- část datového rámce ZigBee
SPI	- sériové periferní rozhraní (Serial Peripheral Interface)
TX	- vysílání (Transmit)
TXD	- pin uart komunikace (Transmit Data)
uA	- jednotka elektrického proudu
UART	- sériová komunikace (Universal Asynchronous Receiver / Transmitter)
USART	- sériová komunikace (Universal Synchronous / Asynchronous Receiver / Transmitter)
USB	- univerzální sériová sběrnice (Universal Serial Bus)
V	- jednotka elektrického napětí
VCC	- napájecí napětí (Voltage Common-collector)
ZED	- koncové zařízení (ZigBee End Device)
ZDO	- objekt zařízení ZigBee (ZigBee Device Object)

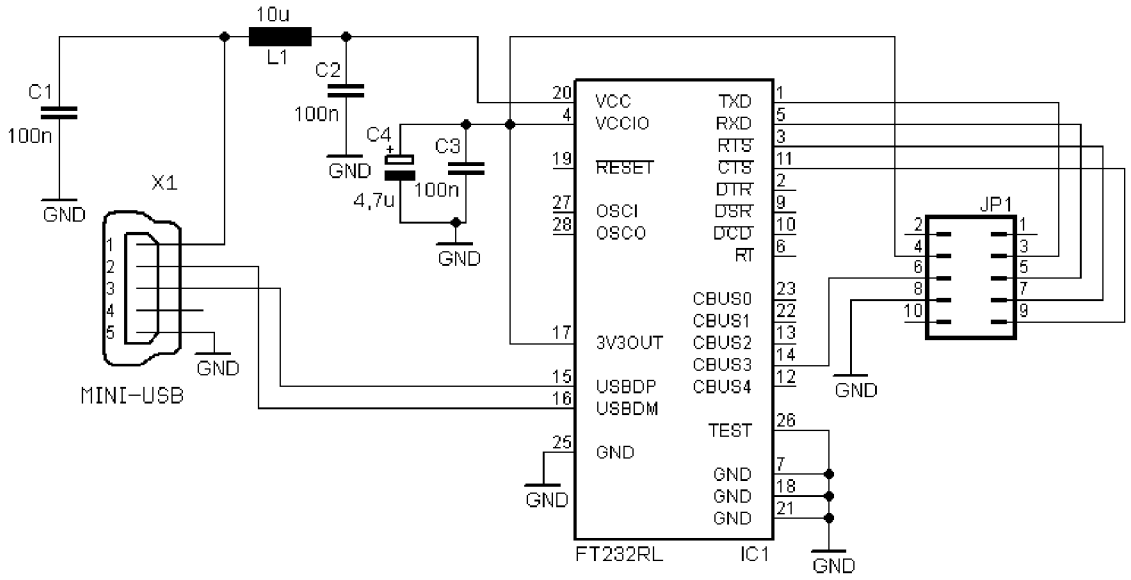
Přílohy

Příloha A - Schéma zapojení

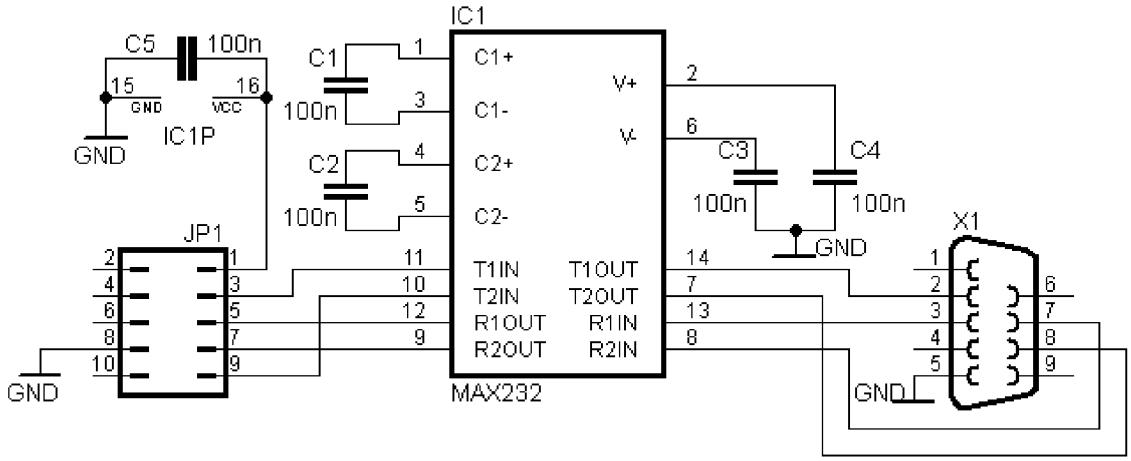
A.1 Deska modulů



A.2 USB komunikace



A.3 COM komunikace

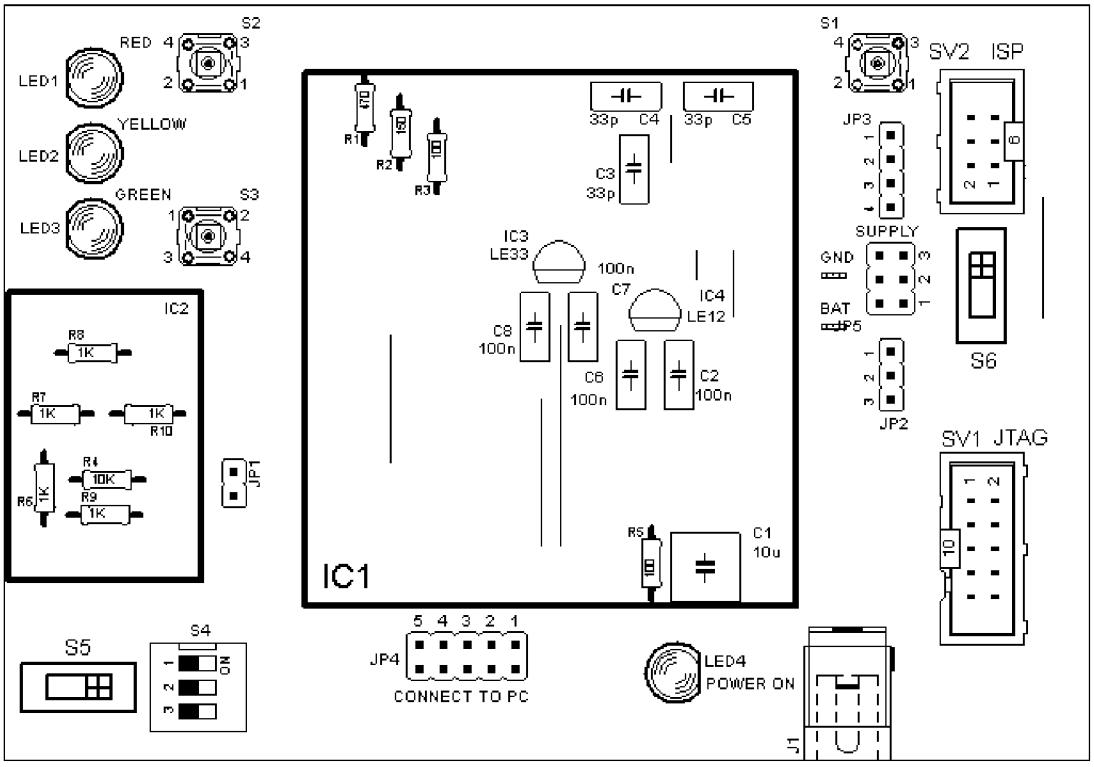


Příloha B - Desky plošných spojů

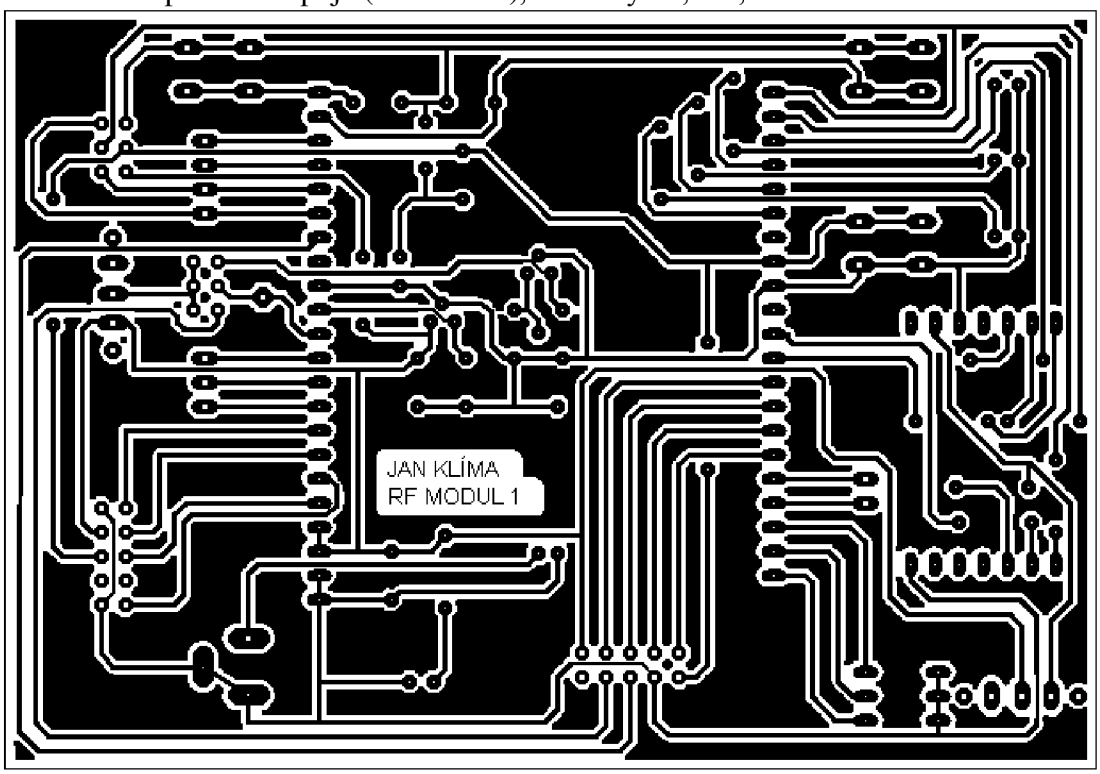
- nejsou v měřítku 1:1

B.1 Deska modulů

- osazovací plán

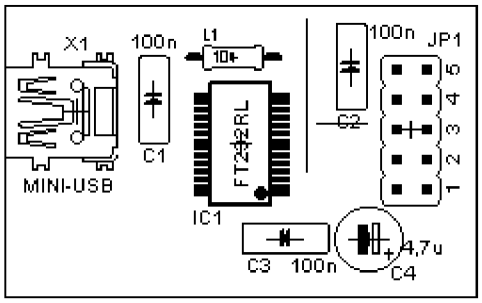


- motiv plošného spoje (BOTTOM), rozměry 11,4x8,0 cm

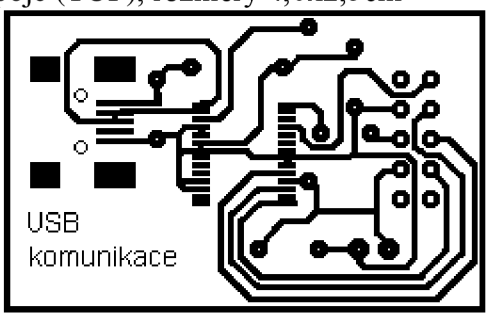


B.2 USB komunikace

- osazovací plán

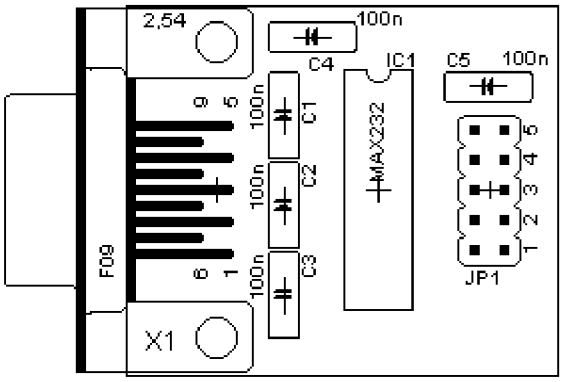


- motiv plošného spoje (TOP), rozměry 4,0x2,5cm

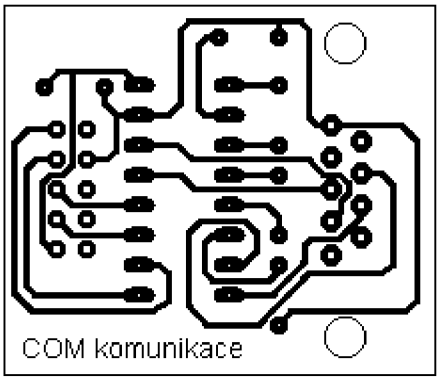


B.3 COM komunikace

- osazovací plán



- motiv plošného spoje (BOTTOM), rozměry 3,7x3,2 cm



Příloha C - Seznam součástek

C.1 Deska modulů

Označení	Hodnota	Pouzdro	Popis
R1	470R	0204/7	ODPOR
R2	150R	0204/7	ODPOR
R3, R5	100R	0204/7	ODPOR
R4	10K	0204/7	ODPOR
R6, R7, R8, R8, R9, R10	1K	0204/7	ODPOR
C1	10uF	C050-075X075	FOLIOVÝ KONDENZÁTOR
C2, C6, C7, C8	100nF	C050-030X075	KONDENZATOR
C3, C4, C5	33pF	C050-030X075	KONDENZATOR
JP1	1x2 piny	JP1	PATICE
JP2	1x3 piny	JP2	PATICE
JP3	1x4 piny	JP3	PATICE
JP4	2x5 pinů	JP5Q	PATICE
JP5	2x3 piny	JP3Q	KONEKTOR
IC1	ZIGBIT	REDUKCE	ZIGBEE MOODUL
IC2	RFM12B	REDUKCE	RFM12B MODUL
IC3	LE33	TO92	STABILIZATOR
IC4	LE12	TO92	STABILIZATOR
LED1	červená	LED5MM	DIODA
LED2	žlutá	LED5MM	DIODA
LED3, LED4	zelená	LED5MM	DIODA
J1	-	SPC4078	KONEKTOR ADAPTÉRU
S1, S2, S3	-	B3F-10XX	TLAČÍTKO
S4	DIL	DS-03	DIL PŘEPÍNAČ
S5, S6	-	TL36YO	PŘEPÍNAČ
SV1	2x5 pinů	ML10	KONEKTOR
SV2	2x3 piny	ML6	KONEKTOR

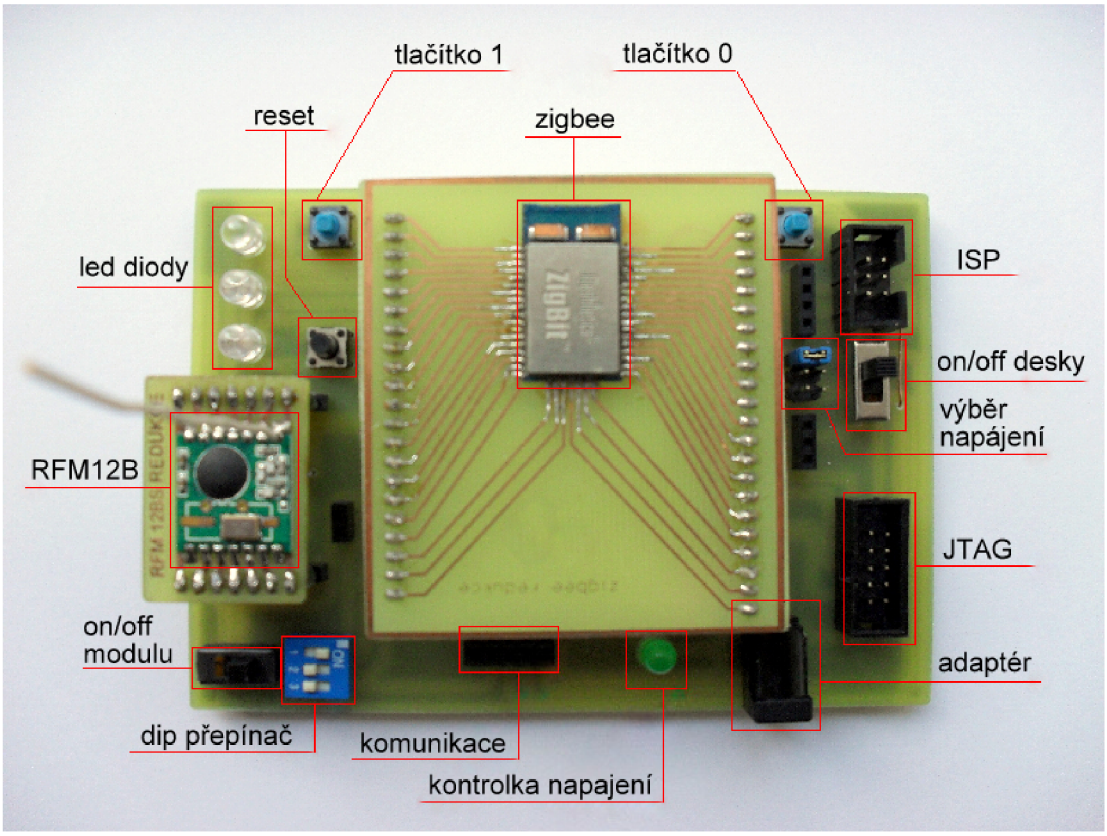
C.2 USB komunikace

Označení	Hodnota	Pouzdro	Popis
C1, C2, C3	100nF	C050-025X075	KONDENZÁTOR
C4	4,7 Uf	E2,5-6	ELEKTROLYT. KONDENZÁTOR
JP1	2x5 pinů	JP5Q	KONEKTOR
IC1	FT232RL	SSOP28	FT232RL
X1	MINI USB	32005-201	KONEKTOR
L1	10uH	0204/7	CÍVKA

C.3 COM komunikace

Označení	Hodnota	Pouzdro	Popis
C1, C2, C3,C4, C5	100nF	C050-025X075	KONDENZÁTOR
JP1	2x5 pinů	JP5Q	KONEKTOR
IC1	MAX3232	DIL16	MAX3232
X1	COM	F09H	COM KONEKTOR

Příloha D - Fotodokumentace



Deska modulů

Příloha E - Návod pro vytvoření programu pomocí BitCloud

Než začnete psát program

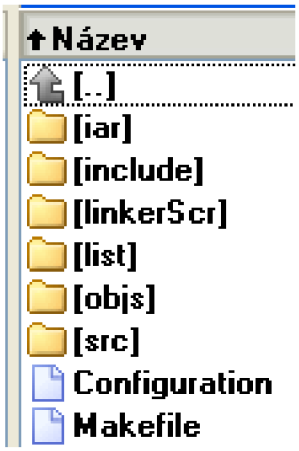
Pro tvorbu nového programu využijeme složku „BlankProject“, v které jsou obsaženy soubory pro obsluhu všech jednotlivých periferií a programového vybavení.

V této složce je pro nás důležitá složka „include“ kde najdeme knihovnu “Periferie.h“ pro základní obsluhu ZigBee. Dále ve složce „src“ jsou soubory *.c, které obsahují obslužné programy pro ZigBee. Zde je soubor periferie.c, který mimo jiné obsahuje funkce pro komunikaci, na které se můžeme odkazovat. Dále např. složka „linkerScr“ obsahuje definici atmelu 1281. Složka s programem musí být v adresáři BitClout

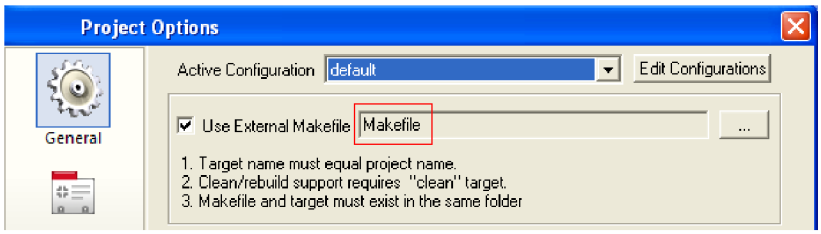
Před vytvořením nového projektu pomocí programu avrstudio je nutné **v souboru „Configuration“ pojmenovat projekt** stejně jak ho vytvoříme v avrstudiu.
V programu Avrstudio se v nastavení konfigurace nastaví, aby se používal externí makefile. A nastavíme Makefile, který je ve složce, kterou použijeme k tvorbě projektu. Po kompilaci projektu se do kořenové složky vygenerují soubory *.hex, *.srec, *.elf, *.bin.

```
#----------------------
# Application path
#----------------------
APP_PATH = .

#----------------------
# Project name
#----------------------
PROJNAME =

#----------------------
# Compiler type:
#----------------------
COMPILER_TYPE = GCC

#----------------------
```

Soubor Configuration - jméno projektu



Struktura složek projektu



Použití externího makefile

Nastavení CS parametrů v souboru Configuration

```
#----------------------------------------------------------------------------
# Stack parameters being set to Config Server
#----------------------------------------------------------------------------
CS_NWK_UNIQUE_ADDR = true
CS_RX_ON_WHEN_IDLE = true
CS_EXT_PANID = 0xAAAAAAAAAAAAAAAALL
CS_MAX_NETWORK_DEPTH = 3
CS_NEIB_TABLE_SIZE = 5
CS_MAX_CHILDREN_AMOUNT = 4
CS_MAX_CHILDREN_ROUTER_AMOUNT = 4
CS_ROUTE_TABLE_SIZE = 8
CS_CHANNEL_MASK = "(1l<<0x0f)"
CS_CHANNEL_PAGE = 0
CS_POWER_FAILURE = false
CS_APS_MAX_BLOCKS_AMOUNT = 4

#----------------------------------------------------------------------------
# Settings for security
#----------------------------------------------------------------------------
CS_NETWORK_KEY =
"{0xCC,0xCC,0xCC,0xCC,0xCC,0xCC,0xCC,0xCC,0xCC,0xCC,0xCC,0xCC,0
xCC,0xCC,0xCC}"
CS_ZDO_SECURITY_STATUS = 0
CS_APS_TRUST_CENTER_ADDRESS = 0xAAAAAAAAAAAAAAAALL
CS_APS_SECURITY_TIMEOUT_PERIOD = 10000
```

Struktura zdrojového kódu

Zde je ukázána struktura doporučená výrobcem:

Popis projektu:
Autor, popsání funkce

Definice knihoven:
#include <stdio.h>
#include <periferie.h>
...
…

Definice datových typů:
typedef enum //datový typ AppState_t
{
} AppState_t;
…
…

Funkce programu:
void Coordinator(void); //obsluha coordinatoru
…
…

Proměné programu:
static uint16_t nwkAddr; //proměnná adresy uzlu
…
…

Hlavní tělo programu:
void APL_TaskHandler(void)
{
…
}

Obslužné programy, callbacky:
void Coordinator(void)
{
}
…
…

Obsah knihovny Periferie.h

Rozdíly alias součástí mezi základními knihovnami a knihovnou Periferie.h. V původních knihovnách jsou názvy alias uvedeny zdlouhavými a ne zrovna jednotnými názvy. Pro zrychlení psaní kódu jsou aliasy předefinovány následující. S většinou aliasů se setkáte i v „Sample aplikacích“ od výrobce.

Definice názvu led diod (“id“)
- LED_G - zelená dioda
- LED_R - červená dioda
- LED_Y - žlutá dioda

Práce s diodami:
- appOpenLeds() - inicializace a otevření portu led diod
- appCloseLeds() - uzavření portu led diod
- appOnLed(“id“) - Zapnutí led diody “id“
- appOffLed(“id“) - Vypnutí led diody “id“
- appToggleLed(“id“) - Změna stavu led diody “id“ na opačný

(id je označení diody)

Práce s přepínačem (Sliders):
- appReadSliders() - načtení stavu přepínače - výstup: Sliders0, Sliders1, Sliders2
  vystup: 0 - 7

Práce s tlačítky:
- appOpenButtons(“obsluha stisknutí“, “obsluha uvolnění“) - otevření portu tlačítek, nastavení události přerušení a obsluhy přerušení - výstup BSP_KEY0 nebo BSP_KEY1, podle stisknutého tlačítka
- appCloseButtons() - uzavření potu tlačítek

Práce s kanálem USART
- OPEN_USART(„deskriptor“) - otevření kanálu USART
- CLOSE_USART() - uzavření kanálu USART,
- WRITE_USART(„deskriptor“) - zápis na kanál USART
- READ_USART(„deskriptor“) - čtení z kanálu USART
  (deskriptor je typu HAL_UsartDescriptor_t)

Knihovna dále obsahuje několik globálních funkcí. Jsou to funkce používány v hlavním programu, které jsou definovány v souboru periferie.c, ve kterém jsou obsaženy funkce pro komunikaci a další obslužné programy.

Praktické ukázky použití příkazů:

Nastavení uzlu

Nastavení uzlu na určitý typ zařízení se provádí pomocí zápisu cs parametru *CS_DEVICE_TYPE_ID*. Tento parametr je typu *DeviceType_t* a nabývá následujících hodnot. Dále je potřeba ještě definovat parametr *CS_RX_ON_WHEN_IDLE_ID*, ten je *false* v pokud se jedná o koncové zařízení anebo *true* pokud je zařízení koordinátor nebo router. Nastavení síťové adresy se provede pomocí parametru *CS_NWK_ADDR_ID.* Zde se zapíše číslo adresy, *0* je rezervovaná pro koordinátor. Aby tato adresa byla aktivovaná, musí být nastaven parametr *CS_NWK_UNIQUE_ADDR_ID* nastaven na *true.*

DeviceType_t deviceType
- *datový typ typu zařízení*

DEVICE_TYPE_COORDINATOR; bool rxOnWhenIlde = true;
DEVICE_TYPE_ROUTER; bool rxOnWhenIlde = true;
DEVICE_TYPE_END_DEVICE; bool rxOnWhenIlde = false;

```
CS_WriteParameter(CS_DEVICE_TYPE_ID,&deviceType);
CS_WriteParameter(CS_RX_ON_WHEN_IDLE_ID, &rxOnWhenIdle);
CS_WriteParameter(CS_NWK_ADDR_ID, &nwkAddr);
CS_WriteParameter(CS_NWK_UNIQUE_ADDR_ID, &(bool){true});
```
- *zápis CS parametrů pro definici uzlů*

Start sítě

static ZDO_StartNetworkReq_t networkParams;
- *datový typ startovních síťových parametrů*

networkParams.ZDO_StartNetworkConf = ZDO_StartNetworkConf;
- *parametry konfigurace startu site*

ZDO_StartNetworkReq(&networkParams);
- *start sítě*

Nastavení časovače

HAL_AppTimer_t Timer
- *datový typ časovače*

Timer.interval = *doba v ms*
Timer.mode = *počet opakování* - TIMER_REPEAT_MODE - opakovaně
- TIMER_ONE_SHOT_MODE - pouze jednou

Timer.callback = *zpětné volání* - nastavení obsluhy

HAL_StartAppTimer(&Timer)
- *spuštění časovače*

HAL_StopAppTimer(&Timer)
- *ukončení časovače*

Nastavení USART kanálu

Před požitím USART kanálu je potřeba nejdříve nastavit jednotlivé parametry:

static HAL_UsartDescriptor_t appUartDescriptor;
- *datový typ nastavení USART kanálu*

---

appUartDescriptor.tty = číslo kanálu - USART_CHANNEL_0
- USART_CHANNEL_1

appUartDescriptor.mode = *mód synchronizace kanálu* - USART_MODE_ASYNC
- USART_MODE_SYNC

appUartDescriptor.baudrate = *baudrate rychlost* - USART_BAUDRATE_1200
- USART_BAUDRATE_2400
- USART_BAUDRATE_4800
- USART_BAUDRATE_9600
- USART_BAUDRATE_19200
- USART_BAUDRATE_38400

appUartDescriptor.dataLength = *délka dat* - USART_DATA5
- USART_DATA6
- USART_DATA7
- USART_DATA8

appUartDescriptor.parity = *parita dat* - USART_PARITY_NONE
- USART_PARITY_EVEN
- USART_PARITY_ODD

appUartDescriptor.stopbits = *počet stopbitů* - USART_STOPBIT_1
- USART_STOPBIT_2

appUartDescriptor.flowControl = *flow kontrol* - USART_FLOW_CONTROL_NONE
-
USART_FLOW_CONTROL_HARDWARE

appUartDescriptor.rxBuffer = *ukazatel na rxBuffer* - NULL (callback mód)

appUartDescriptor.rxBufferLength = *velikost rxBufferu*

appUartDescriptor.rxCallback = *zpětné volání* - nastavení obsluhy
- NULL (dotazovací mód)

appUartDescriptor.txBuffer = *ukazatel na txBuffer* - NULL (callback mód)

appUartDescriptor.txBufferLength = *velikost txBufferu*

appUartDescriptor.txCallback = *zpětné volání* - nastavení obsluhy
- NULL (dotazovací mód)

OPEN_USART(&appUartDescriptor)
- *otevření kanálu USART danými parametry*

WRITE_USART(&appUartDescriptor, (void*) aData, aLength)
- *přenos dat (aData) o délce (aLenght) do UART*

READ_USART(&appUartDescriptor, (void*) aData, aLength)
- *příjem dat (aData) o délce (aLenght) z UART*

CLOSE_USART(&appUartDescriptor)
- *uzavření kanálu USART danými parametry*

Přenos dat v síti mezi uzly

Pro přenos dat v síti je nutné nastavit parametry v datovém typu APS_DataReq_t.
static APS_DataReq_t msgParams;
- *datový typ pro přenos dat*

static SimpleDescriptor_t simpleDescriptor = {1, 1, 1, 1, 0, 0 , NULL, 0, NULL};
- *nastavení deskriptoru*

APS_RegisterEndpointReq_t endpointParams;
- *datový typ koncového parametru*

BEGIN_PACK
typedef struct
{
uint8_t header[APS_ASDU_OFFSET]; - *Hlavička*
uint8_t data[APP_ASDU_SIZE]; - *Aplikační data*
uint8_t footer[APS_AFFIX_LENGTH - APS_ASDU_OFFSET]; - *Zápatí zprávy*
} PACK AppMessageBuffer_t;
END_PACK

static AppMessageBuffer_t appMessageBuffer;
- *datový typ bafru zprávy*

---

msgParams.dstAddrMode = *mód adresování* - APS_NO_ADDRESS = 0x00
- APS_SHORT_ADDRESS = 0x02
- APS_EXT_ADDRESS = 0x03

msgParams.dstAddress.shortAddress = *adresa cílu* - koordinator je vždy 0
- router je 1
- koncové zařízení je 2

msgParams.dstEndpoint = *cílový koncový bod* - počet jednotlivých koncových bodů
- koncový bod vysílání

msgParams.profileId = *identifikátor profilu*-endpointParams.simpleDescriptor
->AppProfileId;

msgParams.clusterId = *cílové číslo klastru*
msgParams.srcEndpoint = *zdrojový koncový bod*
msgParams.asduLength = *velikost ASDU bafru* - sizeof(appMessageBuffer.data)
msgParams.asdu = *ASDU bafr* - appMessageBuffer.data
msgParams.txOptions.acknowledgedTransmission = *potvrzení příjmu* - 1
= *nepotvrzovat příjem* - 0
msgParams.radius = *poloměr vysílání*;
msgParams.APS_DataConf = *obsluha potvrzení odeslání*

APS_DataReq(&msgParams);
- *poslání dat*

endpointParams.APS_DataInd = APS_DataIndCoord;
- *callback pro přijatá data v koordinatoru*

endpointParams.APS_DataInd = APS_DataIndRout;
- *callback pro přijatá data v routru*

endpointParams.APS_DataInd = APS_DataIndEnd;
- *callback pro přijatá data v end device*

Správa napájení - uspání a probuzení uzlu

Koncové zařízení má možnost snížit svou spotřebu, uvedením uzlu do režimu spánku.

ZDO_WakeUpInd()
- *funkce indikující probuzení uzlu*

ZDO_WakeUpConf(ZDO_WakeUpConf_t *conf)
- *funkce potvrzující probuzení uzlu*

static ZDO_SleepReq_t zdoSleepReq;
- *datový typ pro uspání uzlu*

static void ZDO_SleepConf(ZDO_SleepConf_t *conf)
- *funkce potvrzující uspání uzlu*

---

zdoSleepReq.ZDO_SleepConf = ZDO_SleepConf;
- *nastavení parametru pro uspání uzlu*

ZDO_SleepReq(&zdoSleepReq);
- *uspání uzlu*

Příloha F - Návod pro vytvoření programu pro RFM12B

Pro jednodušší práci s tímto modulem jsem vytvořil knihovnu Periferie.h a její zdrojový soubor Periferie.c. Tento soubor obsahuje definici periférií, softwarovou spi, obsluhu vnějšího přerušení, inicializace a komunikace modulu, vyhodnocení dat atd. V další části jsou popsány základní funkce pro tento modul.

Nastavení modulu

Inicializace modulu

	Příkaz	Popis
1	Konfigurační nastavení	frekvenční pásmo, kapacita, filtr, atd.
2	Správa výkonu	přijímač/vysílač, syntetizátor, xtal osc, časovač probuzení, atd.
3	Nastavení frekvence	nastavení frekvenčního kroku, rozsah
4	Datová rychlost	bitová rychlost
5	Kontrola příjmu	indikace použitelných dat (vdi), frekvenční rozsah
6	Filter dat	typ filtru, zotavovací parametr
7	Mód FIFO a Reset	Úroveň dat FIFO, start FIFO, povolení FIFO
8	Čtení FIFO přijímače	RX FIFO bude načten pomocí tohoto příkazu
9	AFC	AFC parametry
10	TX nastavení	parametry modulace, výstupní výkon
11	Vysílací registr	TX data budou vyslány pomocí tohoto příkazu
12	Probouzecí příkaz	časová perioda probuzení
13	Nižší povinnosti cyklu	zapnutí nižší povinnosti cyklu
14	Detekce nízkého napětí baterie a hodiny mikrokontroléru	zapnutí detekce a rozsah hodin
15	Status načtení příkazu	načtení statusových bitů

Nastavení jednotlivých příkazů:

1	(0x80E7)	EL,EF,868pásmo,12.0pF
2	(0x8299)	er,!ebb,ET,ES,EX,!eb,!ew,DC
3	(0xA640)	výběr frekvence
4	(0xC647)	4.8kbps
5	(0x94A0)	VDI,FAST,134kHz,0dBm,-103dBm
6	(0xC2AC)	AL,!ml,DIG,DQD4
7	(0xCA81)	FIFO8,SYNC,!ff,DR (FIFO level = 8)
8	(0xCED4)	SYNC=2DD4;
9	(0xC483)	@PWR,NO RSTRIC,!st,!fi,OE,EN
10	(0x9850)	!mp,90kHz,MAX OUT
11	(0xCC17)	!OB1,!OB0, LPX,!ddy,DDIT,BW0
12	(0xE000)	nevyužito
13	(0xC800)	nevyužito
14	(0xC040)	1.66MHz,2.2V

Nastavení modulu jako vysílač:
(0x8239)
(0x0000)

Nastavení modulu jako přijímač:
(0x8299)
(0x0000)

Přechod modulu do režimu spánku:
(0x8201)
(0x0000)

Přesné vysvětlení jednotlivých pojmů a možnosti nastavení jednotlivých příkazů je v[9].

Délka antény:

Doporučené délky antén pro modul pracující na frekvenci 868MHz:
¼ vlnové délky: 82,2mm
½ vlnové délky: 164,3 mm
Plná vlnová délka: 345,5 mm