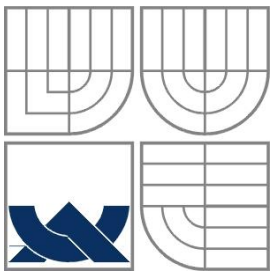
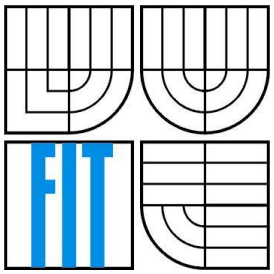


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

BEZDRÁTOVÁ SENZOROVÁ SÍŤ Z KOMPONENT ARDUINO

WIRELESS SENSOR NETWORK WITH ARDUINO COMPONENTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

FILIP VÝBORNÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAN SAMEK, Ph.D.

BRNO 2012

Abstrakt

Cílem této bakalářské práce je představení možnosti využití hardwarové platformy a vývojového prostředí Arduino při vytváření bezdrátových senzorových sítí WSN (Wireless Sensor Networks), se zaměřením zejména na problematiku směrování naměřených dat ze senzorových uzlů do základního uzlu. V práci byla navržena a realizována bezdrátová senzorová síť z vhodných komponent platformy Arduino, založená na bezdrátových komunikačních modulech XBee, které budou představeny ve druhé části. Třetí část pojednává o výběru vhodného komunikačního protokolu a topologie, kde budete seznámeni s dosud poměrně neznámým komunikačním protokolem DigiMesh. Ve čtvrté části se zabývám samotnou realizací bezdrátové sítě, která je v závěru práce otestována.

Klíčová slova

Bezdrátové senzorové sítě, Arduino, XBee moduly, protokol DigiMesh, senzory, synchronní periodické uspávání sítě.

Abstract

The goal of this bachelor's thesis is to introduce the possibility of using Arduino hardware platform as an aid to the creation of wireless sensor networks, focusing especially on the routing of measured data issues from sensor nodes to the base station. My thesis deals with the design and implementation of wireless sensor networks from suitable Arduino components, based on the XBee wireless communication modules which are introduced in the second chapter. The third chapter discusses the selection of a suitable communication protocol, the relatively unknown protocol DigiMesh. The final chapter covers the design of the actual wireless network which has been tested.

Keywords

Wireless sensor networks, Arduino, XBee modules, protocol DigiMesh, sensors, synchronous cyclic network sleep.

Citace

Filip Výborný: Bezdrátová senzorová síť z komponent Arduino. Brno 2012, bakalářská práce, FIT VUT v Brně.

Bezdrátová senzorová síť z komponent Arduino

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jana Samka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Filip Výborný
16. 5. 2012

Poděkování

Děkuji vedoucímu práce Ing. Janu Samkovi, Ph.D. za jeho výjimečnou ochotu a odbornou pomoc.

Obsah

1 Úvod

V této bakalářské práci budete seznámeni s možnostmi využití hardwarové platformy Arduino v oblasti bezdrátových senzorových sítí WSN (Wireless Sensor Networks). V rámci práce byla navržena a realizována bezdrátová senzorová síť z vhodných modelů platformy Arduino, založená na RF (radiofrekvenčních) komunikačních modulech XBee od firmy Digi International. Při návrhu sítě jsem se zabýval výběrem vhodné topologie a vhodného komunikačního protokolu, se zaměřením zejména na problematiku směrování dat ze senzorových uzlů do základního uzlu tzv. base station. Na základě získaných poznatků tato síť byla sestavena a následně došlo k otestování její funkčnosti a schopnosti směrování naměřených dat do základního uzlu. Bakalářská práce může sloužit čtenáři jako praktický návod pro vytvoření uspávané bezdrátové senzorové sítě a poskytne mu přehled všeho potřebného tak, aby mohl pochopit problematiku vytváření senzorových sítí z komponent Arduino a RF modulů XBee.

Druhá kapitola představuje teoretický úvod do tvorby bezdrátových senzorových sítí WSN. Je zde představena samotná hardwarová platforma Arduino i její vývojové prostředí Arduino IDE. Dále budou podrobně popsány dva konkrétní typy použitých komponent (Arduino Uno a Arduino Fio) a radiofrekvenční moduly XBee, které budou zajišťovat jejich vzájemnou bezdrátovou komunikaci.

Třetí kapitola se věnuje samotnému návrhu senzorové sítě. Zabývá se výběrem vhodné topologie a vhodného komunikačního protokolu pro senzorové uzly. Zde budete seznámeni s dosud poměrně neznámým a málo rozšířeným protokolem DigiMesh od firmy Digi International, alternativou široce používaného protokolu ZigBee.

Ve čtvrté kapitole je popsána výsledná realizace navrhnuté sítě a závěrečné testování její funkčnosti. Jsou zde popsány konkrétní implementace jednotlivých typů senzorových uzlů, jejich senzorů, dílčích programů a softwaru, který bude sloužit pro výslednou vizualizaci naměřených dat.

V poslední kapitole dojde ke shrnutí dosažených výsledků a zhodnocení možností dalšího vývoje tohoto projektu.

2 Senzorová síť z komponent Arduino

2.1 Úvod do bezdrátových senzorových sítí

Progresivní vývoj v oblasti elektroniky a bezdrátových komunikačních technologií umožnil prudký rozvoj a vývoj v oblasti multifunkčních bezdrátových senzorových sítí WSN (Wireless Sensor Networks). Tento typ sítí je obecně navrhován pro monitorování různých fyzikálních veličin (např. sledování teploty, tlaku, osvětlení nebo vlhkosti) a řízení procesů s využitím senzorů umístěných na jednotlivých uzlech. Spojením těchto uzlů do jediné sítě můžeme vytvořit autonomní distribuovaný a decentralizovaný systém typu ad-hoc [9], který má obrovský potenciál nasazení v mnoha oblastech, jako je např. řízení průmyslu a výrobních procesů, monitorování domácností nebo sledování životního prostředí. Tento druh sítí se může uplatnit zejména ve špatně přístupných nebo pro člověka nebezpečných oblastech.

Inteligentní senzorové uzly jsou obecně tvořeny mikrokontrolerem, základní deskou, sadou měřících senzorů a RF (radiofrekvenčním) modulem, nebo jinou bezdrátovou technologií, která umožní komunikaci uzlu se svým vzdáleným okolím. Uzly sítě jsou většinou napájeny z externího energetického zdroje, např. z baterie, proto jsou jejich výpočetní a vysílací možnosti značně omezeny. Při návrhu senzorové sítě tedy musíme klást důraz na její energetickou efektivitu a tak je běžné, že se jednotlivé uzly na dobu, kdy aktivně nepracují, úplně vypínají, nebo přepínají do režimu spánku, kdy nespotřebovávají téměř žádnou energii. Nižší výpočetní výkon uzlů je logickým důsledkem požadavku na jejich malou velikost, hmotnost a nízkou cenu.

Vzhledem k tomu, že se jedná o rozsáhlé distribuované sítě (řádově desítky až tisíce uzlů), jednotlivé senzorové uzly mezi sebou musí spolupracovat a předávat si vzájemně zprávy. Komunikace mezi uzly probíhá podle předem stanovených pravidel, které označujeme jako komunikační protokol. Výběr vhodného protokolu pro použití v bezdrátových senzorových sítích ovlivňuje zejména použitý hardware, cílová topologie, celková spolehlivost, energetická náročnost, datová propustnost vzhledem k počtu měřených hodnot a počtu předpokládaných uzlů v síti a v neposlední řadě jeho další možnosti, např. šifrování komunikace, uzamykání sítě vůči vstupu nových uzlů, její celková diagnostika a uspávání. [9]

2.2 Platforma Arduino

Arduino je otevřená hardwarová platforma disponující vlastním programovacím jazykem a vlastním grafickým vývojovým prostředím tzv. IDE (Integrated Development Environment). Za jejím vznikem stojí italská firma Smart Projects, která v roce 2005 představila svou koncepci pro novou, jednodeskovou platformu osazenou výkonným mikrokontrolerem ATMega AVR firmy Atmel. Cílem

tohoto projektu bylo vytvořit jednoduchou prototypovací platformu, která by studentům umožnila rychlý a snadný návrh hardwaru pro nová zařízení. Arduino zaznamenalo velký úspěch nejen mezi studenty, méně zkušenými programátory a kutily, ale díky svému univerzálnímu uplatnění v mnoha oblastech oslovilo i spoustu profesionálních vývojářů. Vzhledem k tomu, že se jedná o otevřenou platformu, čili autoři Arduina poskytli podrobnou dokumentaci ke všem modelům hardwaru a dali výslovný souhlas k jejímu volnému šíření a použití, vznikl nespočet různých klonů a dalších doplňků (např. FreeDuino, Seeduino, Boarduino a Bare Bones Boards). Tyto klony jsou většinou označeny jako „arduino-kompatibilní“, jsou tedy nějakým způsobem upraveny, ale lze je programovat stejným způsobem jako klasické Arduino a lze k nim zpravidla připojovat stejné periferie, pokud dodržují vzorové rozmístění pinů a signálů na nich. Po uvedení prvního modelu Arduina na trh v roce 2005 došlo k jeho rychlému rozšíření a popularizaci. Následně vzniklo ještě mnoho jeho dalších různě vylepšených nástupců a k začátku roku 2010 se jich prodalo více jak 120 tisíc kusů.

Při počátečním návrhu Arduina autoři vycházeli ze dvou podobně zaměřených projektů Wiring a Processing. Projekt Wiring [19] inspiroval návrh hardware a vlastního programovacího jazyka Arduina, zatímco grafické vývojové prostředí bylo značně inspirováno projektem Processing [20].

Jednotlivé modely hardwaru Arduina jsou diferencované pro odlišné oblasti nasazení. Primárně se liší svými rozměry, hmotností, výpočetním výkonem, mikrokontrolery, počtem digitálních vstupně-výstupních pinů, stylem napájení, komunikačními rozhraními a dalšími přídavnými periferiemi. Tím se logicky liší i jejich cena. Z poměrně širokého spektra nabízených komponentů si tedy můžeme vybrat modely vhodné pro nasazení v oblasti bezdrátových senzorových sítí, které splňují nejvíce našich požadavků.

K základním deskám Arduina, tzv. základnám, je možné připojit řadu různých periferních zařízení a rozšiřujících desek, což značně zvyšuje jejich flexibilitu a možnosti použití celé platformy. Právě to, že tyto periferie nejsou přímo součástí základen Arduina výrazně snižuje náklady na pořizovaný hardware. Všechny komponenty a periferie je možné zakoupit jako hotový produkt, nebo si je každý může sestavit z jednotlivých dílů sám. Platforma Arduino může být použita k vytváření samostatných interaktivních zařízení, nebo může přímo komunikovat se softwarem na počítači. [1, 11]

2.2.1 Hardware

Srdcem každého Arduina jsou mikrokontrolery ATmega z rodiny AVR. Výrobcem těchto čipů je norská firma Atmel, zabývající se jejich výrobou od roku 1996. Mikrokontrolery ATmega AVR jsou osmibitové procesory typu RISC (Reduced Instruction Set Computer) s harvardskou architekturou, čili mají oddělený paměťový prostor pro data a vlastní program. Mikrokontrolery ATmega právě jako

jedny z prvních použily pro uložení svého programu paměť typu FLASH, na rozdíl od konkurenčních mikrokontrolerů, které v té době používaly paměti typu EPROM, EEPROM nebo ROM.

Oficiální vydání Arduina používají čipy z řady megaAVR, konkrétně tedy ATmega8, ATmega168, ATmega328, ATmega 1280 a ATmega 2560, které se liší především velikostí vnitřní paměti RAM pro data a vnitřní paměti FLASH pro program. Na většině desek nalezneme kromě mikrokontroleru několik LED diod, resetovací tlačítko, konektory pro ICSP programování, analogové a digitální vstupně-výstupní piny, napájecí konektor, krystalový oscilátor (některé varianty disponují keramickým rezonátorem), stabilizátor vstupního napětí a u novějších verzí USB konektor.

Hlavní mikrokontroler má již předprogramovaný zavaděč systému, tzv. boot loader, což značně zjednodušuje nahrávání programů přímo do FLASH paměti čipu, na rozdíl od ostatních zařízení, která typicky vyžadují externí programátor. Zavaděč se postará o základní nastavení mikrokontroleru, jako jsou např. interní časovače, nastavení rozhraní UART a nastavení vnitřních registrů, které upravují vlastnosti čipu. Mikrokontroler můžeme k počítači připojit pomocí USB portu, na kterém je virtuální (softwarově simulovaná) sériová komunikace přes linku RS-232 (sériový port). U starších typů desek se pro tyto účely používají FTDI čipy.

Každá deska má většinu vstupně-výstupních pinů přístupných přes precizní patice, do kterých lze připojit řadu periferií, zvaných „shieldy" (kryty). Tyto rozšiřující desky se dají upevnit na vršek základny Arduina do příslušných pinů a umožnit tak rozšíření funkčnosti celého zařízení. To nám umožní např. manipulaci s krokovými motory (Arduino Motor Shield), připojení desky k ethernetové síti pomocí RJ-45 konektoru (Ethernet Shield), výstup na LCD display (Color LCD Shield), přehrávání zvuku (Wave Shield), bezdrátovou komunikaci (XBee Shield), vstup od uživatele (Touch Shield) a mnoho dalších funkcí. [1, 11]

2.2.2 Software

Vývoj programů a vlastní programování mikrokontrolerů Arduina probíhá stejně jako s jakoukoli jinou deskou s MCU (Microprocessor Control Unit) z řady AVR. Programy pro Arduino lze vyvíjet v jazyce „Arduino Programmable Language". Tento jazyk používá standardní knihovny jazyka Wiring [19], jenž vznikl zjednodušením a modifikací jazyka C++, odrážející specifické požadavky na vývoj software pro jednočipová zařízení. Tento jazyk je používaný pro vizualizaci, animaci a řízení hardware z prostředí osobního počítače.

Pro psaní programů, tzv. „skečů" a programování mikrokontrolerů autoři Arduina vyvinuli přenositelné grafické vývojové prostředí „Arduino IDE", napsané v javě, čímž zajistili možnost vývoje programů pro Arduino napříč různými operačními systémy. Arduino IDE (viz obr. 2.1) je od prvního pohledu velice přehledné a intuitivní. Jeho součástí je editor kódu, zvýrazňovač syntaxe a automatické odsazování kódu. Vývojové prostředí umožňuje automatické sestavení programu a jeho nahrání do mikrokontroleru pomocí jediného kliknutí. Stačí v horním menu vybrat správný typ

základní desky a port, ke kterému je zařízení připojeno. Před zahájením programování je nutné z desky odstranit všechny periferie a rozšiřující desky (včetně XBee modulů). Při vývoji Arduino IDE se autoři značně inspirovali z otevřeného projektu Processing [20], který byl původně navržen pro výuku programování a tak nováčkům v programování poskytl velice přehledné a jednoduché vývojové prostředí, pro vytváření jejich nových aplikací. Vývojové prostředí, včetně všech potřebných ovladačů, je volně ke stažení z oficiálních stránek Arduina [15], kde zároveň naleznete podrobné instrukce k jeho instalaci na různé platformy (Windows, Linux a Macintosh).

Obrázek 2.1: Arduino IDE.

Pro kompilování programů se využívá kolekce nástrojů vytvořených v rámci projektu GNU a knihovna AVR Libc, což je knihovna jazyka C, již je možno použít s překladačem GCC. K vlastnímu programování mikrokontrolerů se využívá volně dostupného softwaru Avrdude od firmy Atmel, který je schopen manipulovat s jejich ROM nebo EEPROM pamětí.

Nově vytvořené programy se ukládají s příponou „ino“, odvozenou z posledních tří písmen slova Arduino. Součástí IDE je několik základních knihoven, které poskytují rozšiřující funkcionalitu nejen našim programů, ale zjednodušují práci s různým hardware a manipulaci s daty. Příkladem mohou být knihovny EEPROM (manipulace s vestavěnou EEPROM pamětí), ETHERNET (komunikace desky s internetem), WIRE (pro komunikaci se sítí senzorů nebo zařízení). Mimo tyto

běžné knihovny existuje mnoho neoficiálních knihoven, nabízejících daleko více nadstandardních funkcí a možností. [1, 4]

2.3 Hardware pro uzly WSN

Platforma Arduino nabízí široké spektrum různorodých komponent, od miniaturních Arduino Nano a Pro Mini, až po velké plnohodnotné modely Arduino Mega s větším výpočetním výkonem a s více možnostmi připojení. Některé modely jsou vyráběny zcela účelně pro aplikačně specifické nasazení, např. senzorové uzly (Arduino Fio), vestavěné aplikace (Arduino Pro), nebo mohou být všity do oblečení (LilyPad). Osobně jsem měl k dispozici modely Arduino Fio a Arduino Uno a proto se budu podrobněji věnovat pouze jim.

2.3.1 Arduino Uno

Základní deska Arduina Una s rozměry 7,1 x 8,5 cm (viz obr. 2.2) je osazena MCU (Micro Controller Unit) řady ATmega328 z rodiny AVR. Mikrokontroler pracuje s taktovací frekvenci 16 MHz, jenž poskytuje integrovaný krystalový oscilátor. Deska disponuje USB konektorem, který ji umožňuje jednoduchým způsobem připojit k počítači pomocí USB kabelu typu A/B. Arduino Uno poskytuje 14 digitálních vstupně-výstupních pinů, operujících s napětím 5 V. Proud protékající každým pinem může být maximálně 40 mA. Součástí těchto pinů je vnitřní pull-up rezistor s odporem 20-50 kOhm, který je standardně odpojen. Některé piny mají speciální funkci. Šest z nich (pin 3, 5, 6, 9, 10 a 11) může být použito jako 8bitový PWM (Puls Width Modulation) výstup. Piny 2 a 3 mohou být nastaveny tak, aby vyvolaly externí přerušení při nízké hodnotě na nástupnou hranu, sestupnou hranu nebo při změně úrovně hodnoty na vstupu pinu. K třináctému pinu je připojena vestavěná LED, která svítí, pokud je hodnota úrovně na tomto pinu vysoká. Arduino Uno disponuje šesti analogovými piny označenými A0 až A5, které jsou vybaveny analogově-digitálním převodníkem s 10bitovým rozlišením a tak jsou schopny převodem analogové hodnoty napětí, standardně rozsah od 0 do 5 V, poskytnout 1024 různých digitálních hodnot. Horní hranice rozsahu převodu se může snížit použitím referenčního napětí na pinu AREF. Na desce dále nalezneme napájecí konektor a resetovací tlačítko.

Tento model se od všech předchozích verzí stejné třídy desek liší především tím, že jeho mikrokontroler, pro sériovou komunikaci, nepoužívá převodník z USB na RS232 od firmy FTDI. Namísto tohoto převodníku se používá samostatný mikrokontroler ATmega16U2, naprogramovaný tak, aby zastoupil jeho funkci. V jeho firmware jsou uloženy standardní USB COM ovladače a žádné další ovladače nejsou třeba. Ke komunikaci s počítačem proto postačí pouze USB kabel. ATmega328 poskytuje UART rozhraní, pro klasickou sériovou komunikaci, které je dostupné na pinech 0 (RX) a 1 (TX). ATmega16U2 tuto sériovou linku tuneluje přes USB port a softwaru počítače se tak jeví jako virtuální COM port. LED diody RX a TX svým rozsvěcením značí probíhající komunikaci na

tomto rozhraní. Pro komunikaci s dalším zařízením, nebo s jiným mikrokontrolerem se nabízí další rozhraní podporující SPI (Serial Peripheral Device) a I2C sběrnici.

Mikrokontroler Arduina Una disponuje pamětí typu FLASH o velikosti 32 KB pro uložení našeho programu (z nichž je 0,5 KB vyhrazeno pro zavaděč programu), 2 KB pamětí typu SRAM a 1 KB pamětí EEPROM (k níž lze přistupovat s použitím knihovny EEPROM od Arduina). Před jeho samotným programováním dochází k automatickému resetu desky prostřednictvím MCU ATmega16U2 a tak ji není nutné resetovat manuálně.

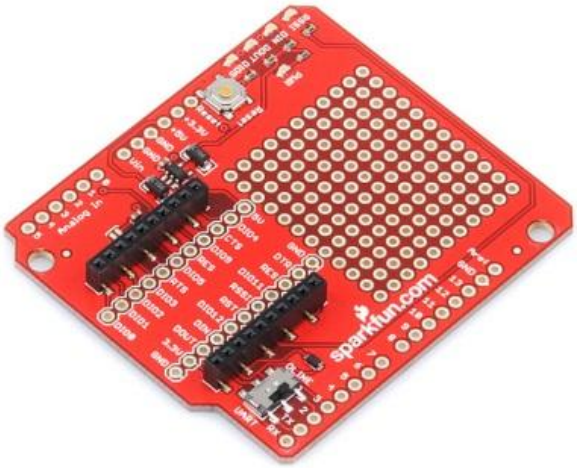
Deska může být napájena přímo USB kabelem, adaptérem z rozvodné sítě nebo baterií. Zdroj napájení se vybere automaticky a není ho nutné manuálně nastavovat. Pokud je deska napájena z externího zdroje (adaptérem nebo baterií), může pracovat při napětí 6 až 20 V. Přesto je doporučena bezpečná hodnota napájení v rozmezí 7 až 12 v z důvodu přehřívání integrovaného regulátoru napětí. Alternativou Arduina Una může být model Arduino Mega s větším výpočetním výkonem a větším počtem vstupně-výstupních pinů. [12]

2.3.1.1 XBee Shield

K některým základním deskám Arduina je možné připojit řadu periferií prostřednictvím rozšiřujících krytů. Jedním z těchto nepostradatelných rozšíření je XBee Shield (kryt), jenž nám umožní rozšířit desku Arduina Una o patici (viz obr. 2.3), do které můžeme připojit RF modul XBee. XBee je rádiový modul, vyráběný firmou Digi International, který zajišťuje bezdrátovou komunikaci zařízení s jeho okolím a bude mu věnována samostatná kapitola.

Kryt se nasadí seshora na základní desku Arduina a piny se vzájemně propojí např. pomocí dutinkových lišt. Na rozšiřující desku jsou tedy vyvedeny všechny signály z pinů základny Arduina. Xbee shield je napájen z 5 v pinu Arduina. XBee modul vyžaduje 3,3 v a proto je na desce integrován regulátor napětí, který vstupních 5 v snižuje na požadovaných 3,3 V. Piny DIN a DOUT XBee modulu jsou připojeny prostřednictvím sériové linky na piny hostitelského mikrokontroleru RX a TX. Na krytu je vyvedeno resetovací tlačítko ze základní desky a sada diod indikujících aktivitu na pinech DIN, DOUT, RSSI a DIO5 XBee modulu. [14]

Obrázek 2.2: Arduino Uno [12].



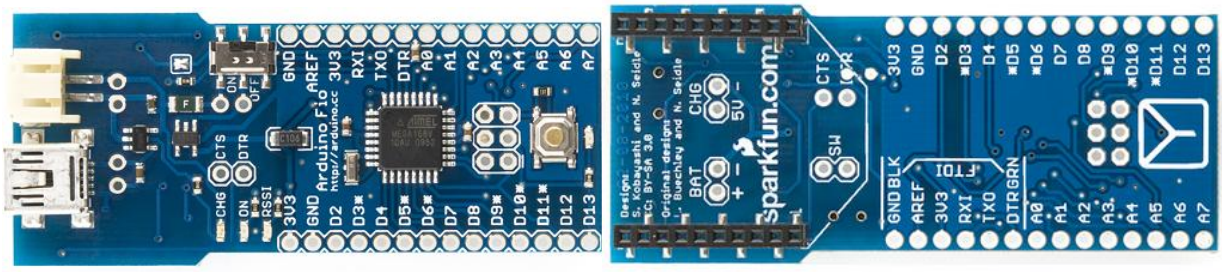
Obrázek 2.3: XBee Shield [14].

2.3.2 Arduino Fio

Arduino Fio bylo navrženo tak, aby sloužilo jako samostatné bezdrátové zařízení. Proto jeho základna, o poměrně malých rozměrech 2,8 x 6,6 cm, je zespodu (viz obr. 2.4) osazena paticí pro uchycení XBee modulu a konektorem pro připojení externí baterie.

Základní deska Fia je založena na mikrokontroleru ATmega328P, je napájena 3,3 V a pracuje s taktovací frekvencí 8 MHz. Nabízí nám 14 digitálních vstupně-výstupních pinů (z nichž může být 6 použito jako 8bitový PWM výstup), 8 analogových vstupů, rezonátor a resetovací tlačítko. Piny 2 a 3 mohou být nastaveny tak, aby vyvolaly externí přerušení při nízké hodnotě, na nástupnou popř. sestupnou hranu nebo při změně hodnoty na vstupu pinu. Každý ze 14 vstupně-výstupních pinů pracuje s napětím 3,3 V. Maximální proud protékající každým pinem je 40 mA. Součástí těchto pinů je pull-up rezistor 20-50 kOhm, který je standardně odpojen.

Na desce nalezneme USB rozhraní, které ale neslouží k programování mikrokontroleru, nýbrž pouze k nabíjení lithiovo polymerové baterie, kterou lze k desce připojit. Deska Arduina Fia totiž integruje obvod, který nám právě umožňuje nabíjet baterii přímo přes USB konektor. Baterie nemusí být jediným zdrojem napájení. Desku můžeme napájet FTDI kabelem nebo přivedením regulovaného napětí 3,3 V na 3V3 pin.

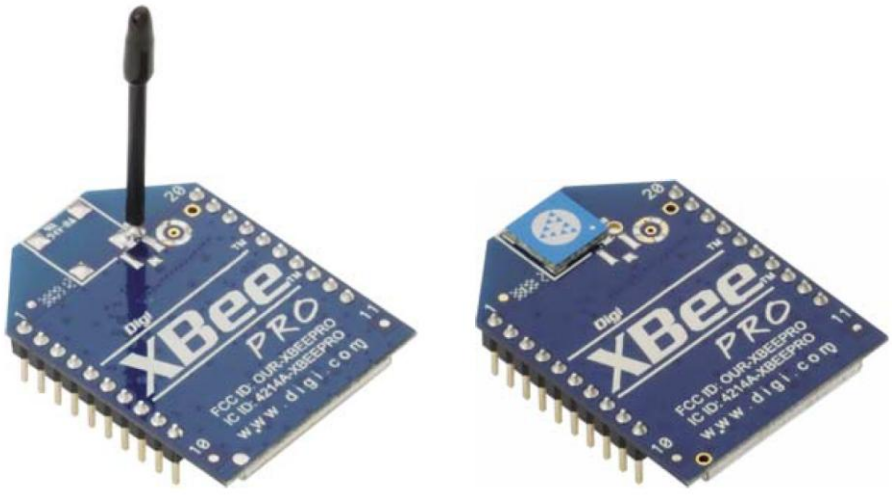


Obrázek 2.4: Arduino Fio [13].

Fio disponuje pamětí typu FLASH o velikosti 32 KB pro uložení vlastního kódu (z nichž jsou 2 KB vyhrazeny pro zavaděč programu), 2 KB pamětí typu SRAM a 1 KB pamětí typu EEPROM. Jsou dva způsoby, jakými můžeme do paměti nahrávat naše programy. Mikrokontroler je možné programovat FTDI kabelem (popř. FTDI programátorem), nebo ho můžeme programovat rovnou bezdátově, pomocí dvou XBee modulů a USB-XBee adaptéru. Bezdrátové programování je výhodné zejména ve chvíli, kdy je cílové zařízení umístěno mimo náš dosah a nemůžeme ho lehce fyzicky připojit k našemu počítači a naprogramovat. Arduino Fio bylo navrženo a je vyráběno firmou SparkFun Electronics. Alternativou k Arduinu Fiu může být např. model Arduino Nano s rozšiřující základnou pro patici XBee. [13]

2.4 XBee moduly

Aby mezi sebou mohly jednotlivé modely Arduina vzájemně bezdrátově komunikovat, budeme muset jejich základny obohatit o RF komunikační moduly. Firma Digi International vyrábí celou řadu těchto různorodých bezdrátových modulů, primárně určených pro komunikaci v přenosovém pásmu 2,4GHz, známých jako XBee. Moduly XBee (viz obr. 2.5) byly navrženy tak, aby splnily mezinárodní normu IEEE 802.15.4 a podpořily unikátní potřeby levných, nízko-výkonových senzorových sítí. Vyžadují minimální výkon pro vzájemnou komunikaci a spolehlivé doručení dat k jejich cíli. Na trhu momentálně existuje přes několik desítek různých kombinací hardwaru a firmwaru XBee, které můžeme momentálně rozdělit v rámci výrobních sérií na dvě základní skupiny **XBee série 1** a **XBee série 2**. Obě dvě série jsou si fyzicky velmi podobné. Přestože mají téměř stejné rozložení pinů a signálů, nelze je vzájemně kombinovat v rámci jedné senzorové sítě. Všechny uzly sítě tedy musí být vybaveny buď moduly první nebo druhé série. Jednotlivé moduly se mohou dále lišit typem antény nebo typem anténního konektoru a tím i svým vysílacím výkonem.



Obrázek 2.5: XBee PRO modul s prutovou anténou (vlevo) a čipovou anténou (vpravo) [3].

Firmware XBee modulů si uchovává jejich lokální nastavení, které se může dodatečně upravovat specifickými příkazy. Každá z obou sérií podporuje různé varianty firmwarů a tím i různé komunikační protokoly. Firmware v modulech můžeme libovolně zaměňovat jeho přeprogramováním jiným kompatibilním firmwarem. Důrazně se ovšem doporučuje, aby všechny moduly operující ve stejné síti používaly totožnou verzi firmware. [3, 10]

2.4.1 XBee 1. série

Moduly první série používají čip vyrobený firmou Freescale. Nativní firmware těchto rádií podporuje pouze sadu standardů síťového protokolu IEEE 802.15.4 a tak poskytuje jednoduchou a standardní topologii typu hvězda a point-to-point, která nahrazuje přímé propojení dvou zařízení kabelem. Přesto pokud potřebujeme s moduly první série vytvořit složitější topologie typu mesh, která je pro senzorové sítě mnohem vhodnější, nabízí se nám možnost použít jiný firmware. Přímo výrobce těchto modulů vyvinul alternativní firmware a komunikační protokol podporující topologie typu mesh zvaný DigiMesh, kterému bude věnována zvláštní kapitola. [1]

2.4.2 XBee 2. série

Čipy pro druhou sérii XBee modulů vyrábí firma Ember Networks. Jsou navrženy pro rozsáhlejší senzorové sítě a ve svém nativním firmware plně podporují topologie typu mesh a obecně velmi známý protokol ZigBee. Moduly druhé série mají delší rádiový dosah a menší celkovou spotřebu energie, než moduly první série. [1]

2.4.3 XBee PRO

Obě dvě série XBee modulů se vyrábějí ve dvou odlišných formách, jako klasická nebo PRO (professional). Klasická verze se běžně označuje jen XBee, resp. XBee 2, zatímco se verze PRO označuje jako XBee PRO, resp. XBee PRO 2. Klasická i PRO verze jsou pinově kompatibilní, s mírnými rozdíly v rozměrech, kde PRO verze je mírně delší. Rozšířená verze PRO má podstatně větší vysílací výkon až 60 mW a tím i celkovou spotřebu energie oproti klasické verzi s vysílacím výkonem 1 mW. Její přijímač je také podstatně citlivější a je schopný zpracovat mnohem slabší signály než běžná verze XBee. To znamená, že jeho rádiový dosah je mnohonásobně delší než u klasické verze. Proto je vhodné rozšířenou verzi modulů použít v situacích, kde vzdálenosti mezi jednotlivými senzorovými uzly jsou řádově delší než několik desítek metrů. PRO verze je pochopitelně cenově dražší. Srovnání všech verzí XBee modulů naleznete v tabulce 2.1. [1, 3]

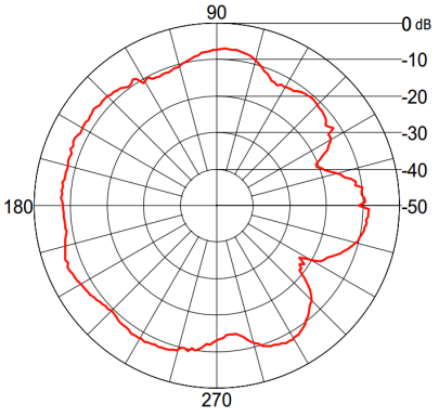
	XBee 1. série	XBee 2. série	XBee PRO
Dosah uvnitř budov	až 30 m	až 40 m	až 90 m
Dosah na volném prostranství	až 100 m	až 120 m	až 1600 m
Vysílací výkon	1 mW (0 dbm)	2mW (+3 dbm)	63mW (+18 dBm)
Hrubá přenosová rychlost	250 kbps	250 kbps	250 kbps
Citlivost přijímače	-92 dBm	-98 dBm	-100 dBm
Napájecí napětí	2,8 – 3,4 V	2,8 – 3,6 V	2,8 – 3,4 V
Spotřeba při vysílání	45 mA / 3,3 V	40 mA / 3,3 V	250 mA / 3,3 V
Spotřeba při přijímání	50 mA / 3,3 V	40 mA / 3,3 V	55 mA / 3.3 V
Spotřeba při vypnutí	< 10 µA	< 1 µA	< 10 µA
Operační frekvence	2,4 GHz	2,4 GHz	2.4 GHz
Operační teplota	-40 až 85 °C	-40 až 85 °C	-40 až 85 °C
Počet kanálů	16	16	12
Standardně podporované topologie	point-to-point, star	point-to-point, star, mesh	point-to-point, star, mesh
Chipset od firmy	Freescale	Ember networks	Ember networks
Typy antén	Integrovaná, prutová, U.FL	Integrovaná, prutová, U.FL, RPSMA	Integrovaná, prutová, U.FL, RPSMA

Tabulka 2.1: Srovnání XBee 1., 2. a PRO série [1, 3, 5].

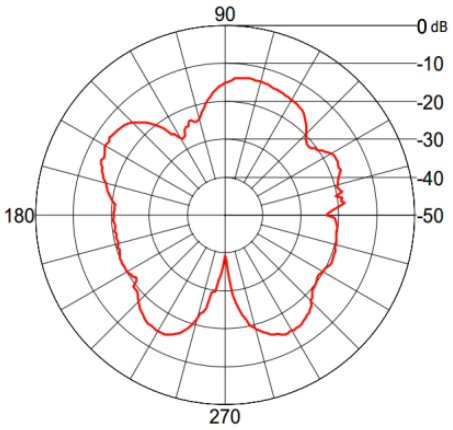
2.4.4 Typy antén XBee modulů

Typ antény, kterou je XBee modul vybaven, ovlivňuje nejen jeho celkový dosah, ale i oblast vyzařování a pokrytí signálem. Všechny XBee moduly jsou dostupné ve variantách s prutovou anténou, nízko-profilovou čipovou anténou nebo s U.FL konektorem, ke kterému lze připojit vlastní externí anténu. XBee PRO moduly se navíc vyrábějí ve variantě s RPSMA konektorem pro všesměrovou anténu. Výkon modulu s prutovou anténou je podle vyzařovacího diagramu (viz obr. 2.6) poměrně necitlivý na natočení k rovině kolmé na anténu. Naopak čipová anténa je (viz obr. 2.7) poměrně citlivá na úhel natočení a při některých polohách tak můžeme docílit většího výkonu vysílače. Podle dostupných měření [6] můžeme vyvodit, že prutová anténa má sice oproti čipové anténě delší dosah, ale pouze ve volném prostranství. Prutová anténa tedy zvyšuje celkový dosah modulu mimo budovy, ale zároveň také zabírá více místa. Pokud tedy požadujeme delší vysílací dosah uzlů a zároveň jejich minimální velikost, je vhodnější vybrat rozšířený modul XBee PRO s čipovou anténou. [6, 1]

,



Obrázek 2.6: Vyzařovací diagram - prutová anténa [6].



Obrázek 2.7: Vyzařovací diagram - čipová anténa [6].

2.4.5 Programování XBee

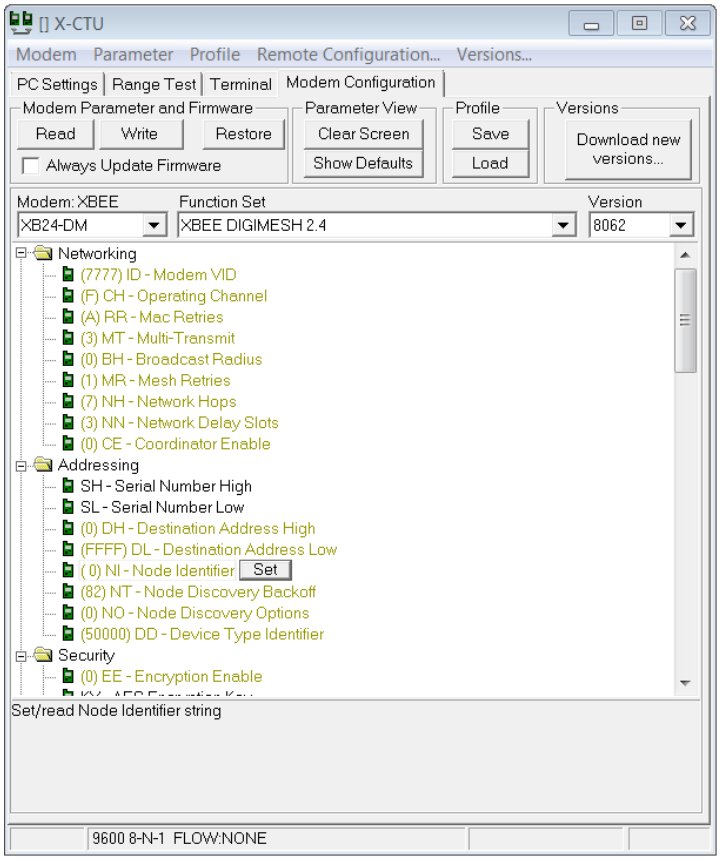
Samotné XBee moduly můžeme programovat několika způsoby. Pro úpravu jejich základního nastavení bude plně dostačovat jakýkoli sériový terminál, např. HyperTerminal, TeraTerm nebo CoolTerm. Pokud budeme chtít změnit celý firmware modulu, budeme potřebovat program X-CTU od výrobce těchto modulů firmy Digi International.

XBee se programuje přes jeho sériové rozhraní. Abychom ho mohli připojit k počítači, budeme k tomu potřebovat patřičný adaptér. K dispozici je hned několik vhodných adaptérů, z nichž mezi nejpoužívanější patří XBee Explorer (obr. 2.8) s USB konektorem a XBee Adapter Kit (obr. 2.9) s konektorem pro FTDI kabel. Před samotným zahájením programování modulů je někdy nutné XBee resetovat, např. ve chvíli kdy se daný modul sám uspává a v režimu spánku by ho nebylo možno úspěšně naprogramovat. Proto jsou některé adaptéry opatřeny resetovacím tlačítkem. Pokud jím adaptér není vybaven, jako v případě XBee Exploreru a XBee Adapter Kitu, můžeme XBee modul resetovat manuálně propojením resetovacího (RST) a zemnícího (GND) pinu nějakým vodičem.

Obrázek 2.8: XBee explorer [4].

Obrázek 2.9: Xbee adapter [4].

Software X-CTU je oficiální program pro konfigurování XBee modulů a je volně ke stažení na webových stránkách výrobce těchto modulů firmy Digi International [16]. Na první pohled (viz obr. 2.10) nabízí velice přehledné a na použití jednoduché rozhraní, jehož součástí je jednoduchý sériový terminál a editor pro konfiguraci nebo změnu firmware. Po připojení adaptéru s XBee modulem k počítači stačí v programu vybrat COM port, ke kterému se adaptér připojil a ověřit funkčnost spojení tlačítkem „Test/Query“. V této chvíli už můžeme s modulem komunikovat prostřednictvím terminálu (záložka Terminal) nebo nástrojem pro úpravu nastavení firmwaru (záložka Modem Configuration). Při editaci firmwaru musíme nejdříve tlačítkem „Read“ načíst současnou konfiguraci modulu. Jednotlivé parametry a rozsahy možných vstupních hodnot jsou dobře popsány vysvětlivkou v dolní liště programu. Úprava jednotlivých parametrů probíhá jednoduchým kliknutím na hodnotu, kterou máme v úmyslu změnit. Následně se objeví editační okno, ve kterém můžeme hodnotu změnit. Pokud chceme změnit kompletně celý firmware modulu, stačí vybrat z roletkového menu nový firmware a dále postupovat jako při klasické změně parametrů. Upravené parametry nahrajeme zpět do modulu stiskem tlačítka „Write“. [1, 4, 7]



.

Obrázek 2.10: Program X-CTU.

2.4.6 XBee modul jako senzorový uzel

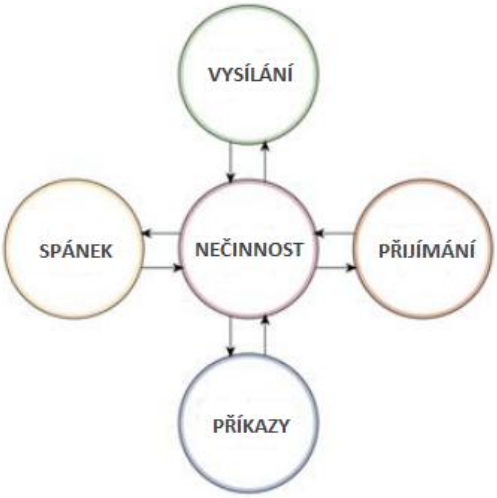
Každý XBee modul má schopnost pracovat přímo jako senzorový uzel a shromážděná data ze senzorů předávat dál bez externího mikrokontroleru. Kromě toho implementuje několik základních ovládacích funkcí, např. je možné XBee modulu zaslat příkaz pro rozsvícení LED diody nebo zapnutí motoru. Bez použití přídavného mikrokontroleru takto můžeme získat velmi malé senzorové uzly, schopné mezi sebou vytvořit bezdrátovou senzorovou síť a posílat naměřená data do základního uzlu. Použitím samostatného XBee také rapidně snížíme hmotnost, energetickou spotřebu a celkovou cenu senzorového uzlu.

XBee moduly disponují 10 digitálními vstupně-výstupními piny. Některé piny zároveň zastupují funkce sériové komunikace, signalizace přenosu dat, spánku a aktivity na ostatních pinech, které ale z většiny nejsou u samostatných modulů třeba. Většina pinů může být nastavena v jednom z pěti módů: vypnutí, vestavěná funkce, analogový vstup, digitální vstup, digitální výstup 0 a digitální výstup 1. Nastavení módu digitálních pinů probíhá v příkazovém módu XBee pomocí vybraných konfiguračních AT příkazů. Naměřená data jsou cílovým uzlům předávána v přenosovém API módu pomocí speciálních paketů. Příkazový mód a API mód budou probrány v další části textu.

Na druhou stranu připojením externího mikrokontroleru k XBee modulu získáme hned několik výhod. Zatímco samotné XBee může být zdrojem senzorových dat, použití externího mikrokontroleru umožní senzorovému uzlu přidat lokální rozhodovací logiku. Mohou se tak použít i digitální senzory, které vyžadují složitější konfiguraci a manipulaci při měření dat. Zároveň s daty můžeme pracovat ještě před jejich samotným přenosem (např. odesílání pouze průměru z několika naměřených hodnot nebo jen hodnot, které překročily stanovenou mez). Desky Arduina navíc poskytují další přídavné vstupně-výstupní piny, ke kterým lze připojit mnoho dalších senzorů. Kromě toho můžeme k Arduinu připojit několik dalších periferních zařízení a tak naše senzorové uzly můžeme obohatit o funkce ovládání krokových motorů, lokalizace s GPS nebo výstup na LCD display. Nedávno bylo oznámeno zahájení vývoje speciálních XBee modulů, vybavených dalším mikrokontrolerem, který by měl umožnit modulu pracovat s určitou základní lokální logikou. Každopádně programování těchto modulů bude zřejmě vyžadovat pokročilé znalosti programovacích jazyků C a Assembler. [1, 5]

2.5 Režimy provozu XBee

Moduly XBee se při provozu vždy nacházejí v jednom z pěti funkčních režimů: vysílání dat, přijímání dat, příkazový režim, režim spánku a režim nečinnosti. Modul se po zapnutí automaticky přepne do režimu nečinnosti. Všechny možné přechody mezi jednotlivými režimy jsou znázorněny na obrázku 2.11.



Obrázek 2.11: Režimy provozu XBee. Převzato a upraveno z [5].

2.5.1 Režim nečinnosti

Standardně se modul XBee, pokud nevysílá ani nepřijímá žádná data, nachází v režimu nečinnosti, kdy nevykonává žádnou užitečnou činnost. Z režimu nečinnosti (viz obr. 2.11) se modul může přepnout do vysílacího, resp. přijímacího módu za předpokladu, že jsou obdržena nová sériová data na pinu DI, resp. výstupním pinu DO. Do spánkového módu se modul může přepnout ve chvíli, kdy jsou splněny všechny podmínky pro přejití do spánkového režimu, dané lokálním nastavením firmwaru XBee. Příkazový mód je vyvolán po obdržení speciální sekvence znaků v transparentním módu nebo po obdržení speciálního paketu v API módu. [5]

2.5.2 Režim vysílání a přijímání dat

Ve chvíli kdy jsou sériová data připravena k odeslání, modul se přepne do vysílacího módu a pokusí se data odeslat. Podle cílové adresy se určí, kterému uzlu budou data předána, aby byla úspěšně doručena k cíli. Pro sítě typu mesh dojde, ve chvíli kdy není známa cesta k cílovému uzlu, k iniciaci vyhledání vhodné cesty k cílovému uzlu podle typu používaného protokolu. Když uzel s odpovídající adresou v síti není nalezen, dojde k zahození paketu. Pokud se nejedná o přenos typu broadcast, modul po odeslání dat očekává potvrzovací paket ACK (acknowledgment), který je odeslán cílovým uzlem, aby potvrdil přijetí dat. V případě, že modul potvrzovací paket neobdrží, dojde k jeho

opětovnému zaslání. Parametr RR (Mac Retries) XBee modulu udává počet neúspěšných pokusů, po kterých se už modul nebude snažit paket opětovně zaslat. Nastavením parametru RR na nulu, funkcionalitu potvrzování paketů vypneme úplně. [5]

2.5.3 Spánkový režim XBee (sleep mode)

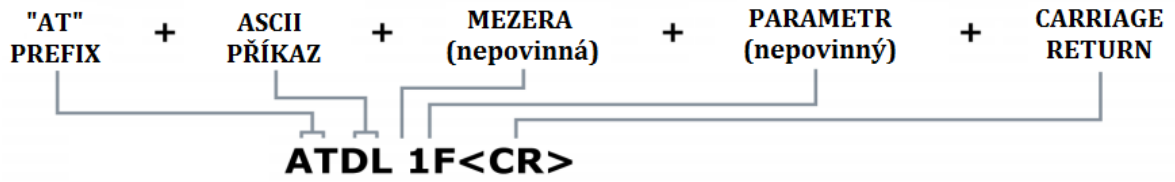
Při práci s bezdrátovými zařízeními je obvyklé, že celé zařízení je mobilní a je tak napájeno z omezeného externího zdroje. XBee moduly se mohou, stejně tak jako mnoho dalších komunikačních zařízení a mikrokontrolerů, dočasně přepnout do tzv. „spánkového režimu“, kdy je XBee modul odpojen a nespotřebovává téměř žádnou energii. Cenou za uspávání XBee modulů je fakt, že v tomto režimu je zařízení téměř kompletně vypnuté a není tak schopné přijímat, přeposílat ani odesílat žádné zprávy až do doby, kdy se opět zase plně aktivuje. Z tohoto důvodu není vhodné uspávat XBee moduly tam, kde je toto chování nežádoucí, např. XBee sloužící jako tzv. směrovače, které musí být schopné kdykoli zprávy přeposílat dalším uzlům, aby dosáhly svého cíle. Většina moderních komunikačních protokolů pro bezdrátové sítě jako je ZigBee a DigiMesh jsou speciálně navrženy tak, aby hladce zvládly komunikaci v síti s takto uspávanými RF moduly. Pravidelným uspáváním XBee modulů můžeme při stejném odběru dat ze senzorů prodloužit provozuschopnost baterií až o několik měsíců.

XBee moduly nabízejí 3 asynchronní a 2 synchronní spánkové módy. Spánkové režimy se nastavují parametrem SM (Sleep Mode). Asynchronní spánkové režimy dovolují ovládat spánkový režim na základě externího signálu (např. řízení mikrokontrolerem). Naopak synchronní spánkové režimy dovolují uzlům uspávání na základě předem stanovených pravidel, kdy je uspávání celého uzlu plně v moci XBee modulu. [5]

2.5.4 Příkazový mód

V některých případech nebudeme chtít data zasílat vůbec, ale budeme potřebovat komunikovat přímo se samotným XBee modulem, např. pokud chceme změnit jeho nastavení. K tomuto účelu slouží příkazový AT mód. Do příkazového módu se XBee může přepnout pouze z transparentního módu po obdržení sekvence GT CC GT. Kde parametr GT (Guard Time) značí tzv. ochranný čas, po který XBee modul nesmí obdržet ani odeslat žádné další zprávy vnějším rozhraním, ten je standardně nastaven na jednu vteřinu. Parametr CC (Command Sequence Character) označuje unikátní sekvenci znaků, standardně tři znaky plus „+++“, za kterými následuje opět ochranný čas, po který modul nesmí přijmout ani odeslat žádné další zprávy. Před samotným přepnutím do příkazového módu, modul nejdříve odešle všechny čekající zprávy ze vstupní fronty. Ihned po přejití do příkazového módu, modul odešle zprávu „OK\r“ výstupním DOUT rozhraním. Parametry GT a CC jsou uloženy v nastavení firmwaru každého XBee a lze je dodatečně měnit.

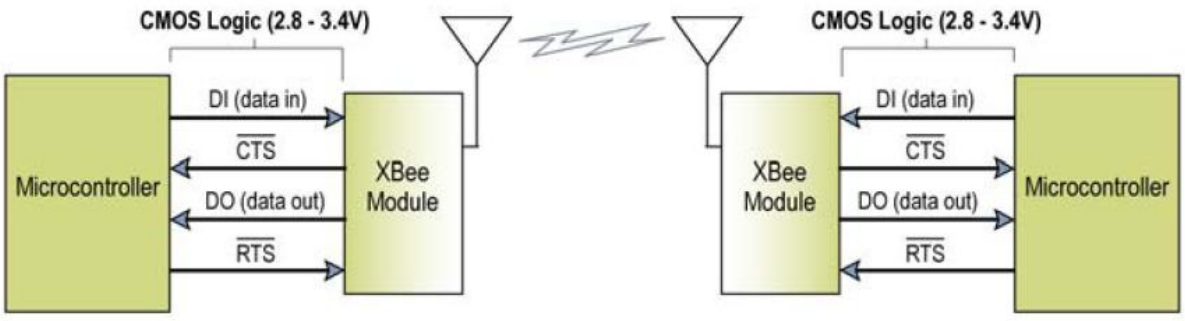
V příkazovém módu už XBee modul nepřeposílá žádná data vnějším rozhraním, ale reprezentuje všechny příchozí zprávy jako příkazy, které se vzápětí snaží vykonat. Tyto příkazy označujeme jako AT ze slova „Attention“ a slouží ke konfiguraci firmware lokálního XBee. Struktura příkazů odpovídá obr. 2.12. Každý příkaz začíná řetězcem „AT“ bezprostředně následovaným sekvencí dvou znaků označujících konkrétní příkaz. Za tímto řetězcem může následovat nepovinná mezera a nepovinná hodnota v závislosti na typu příkazu. AT příkaz musí končit řídícím znakem carriage return. Z příkazového módu se modul opět přepne zpět do transparentního módu po uplynutí 10 vteřin od posledního zpracovaného příkazu nebo po obdržení speciálního AT příkazu CN (Exit Command Mode). [5]



Obrázek 2.12: Struktura AT příkazů. Převzato a upraveno z [5] .

2.6 Komunikace XBee modulů

XBee komunikuje se svým hostitelským zařízením (např. mikrokontrolerem desky Arduina) přes jeho asynchronní sériový port UART. Tímto portem modul může komunikovat s jakýmkoli UART kompatibilním zařízením nebo s jakýmkoli jiným sériovým zařízením využitím úrovňového převodníku. Zařízení, která disponují UART rozhraním, se mohou připojit přímo k pinům XBee modulu (viz obr. 2.13). Podle obrázku data vstupují do modulu přes pin DI (Data In) jako asynchronní sériový signál a vystupují přes pin DO (Data Out). Sériová komunikace tedy probíhá mezi UART rozhraním hostitelského zařízení (např. mikrokontrolerem) a UART rozhraním XBee modulu. Obě rozhraní musí pracovat se stejným nastavením přenosové rychlosti, parity, startovního bitu, koncového bitu a ostatních datových bitů. Nastavení UART rozhraní XBee modulu je uloženo v jeho firmware (parametry BR, NB).



Obrázek 2.13: Propojení XBee a zařízení s UART rozhraním [5].

Příchozí sériová data, od hostitelského zařízení z pinu DI, jsou nejdříve uložena ve vstupní frontě DI, dokud nejsou XBee modulem úplně zpracována a odeslána. Aby nedošlo k zaplnění fronty a ke ztrátě doručených dat, byl do tohoto procesu zaveden signál $\overline{CTS}$ (Clear To Send Flow Control). Ve chvíli kdy ve vstupní frontě DI zbývá už pouze 17 bytů do jejího úplného zaplnění, signál $\overline{CTS}$ se změní do logické jedničky, která hostitelskému zařízení signalizuje k zastavení zasílání dalších dat. Signál $\overline{CTS}$ se opět změní do logické nuly ve chvíli, kdy se ve vstupní frontě DI uvolní alespoň 34 bytů místa. Celková velikost vstupní fronty je 202 bytů.

Stejná logika je uplatněna i pro přijatá data zaslaná hostitelskému zařízení z pinu DO. Příchozí data jsou XBee modulem uložena do výstupní fronty DO, dokud nejsou úspěšně odeslána hostitelskému zařízení. Hostitelské zařízení může obdobně využít signál $\overline{RTS}$ (Request To Send Flow Control), k signalizování XBee modulu pro zastavení zasílání dat pinem DO.

Čip v modulech XBee nám nabízí dvě varianty datových přenosů. Podle zvoleného způsobu přenosu pak modul dokáže v různých situacích pracovat efektivněji. K dispozici máme na výběr transparentní a API (Application Programing Interface) mód. Oba tyto módy jsou lokální k danému modulu a pro každý z nich je vydán odlišný firmware. Aby spolu mohly moduly úspěšně komunikovat, je nutné, aby všechny pracovaly se stejnou variantou datového přenosu, tzn. buď s transparentním nebo API módem. [5]

2.6.1 Transparentní mód

Standardní mód, ve kterém se XBee rádio nativně nachází, označujeme jako transparentní. Nazýváme ho transparentní právě proto, že modul data jednoduše přenáší přesně ve tvaru a formě, v jakém je přijme. Z hlediska složitosti řídícího programu, přenosu dat a výpočetních nároků na mikrokontroler je transparentní mód ze všech ostatních módů tím nejjednodušším. v transparentním módu XBee nahrazuje jednoduchou sériovou linku a všechna přijatá data z DI pinu jsou odeslána sériovým rozhraním přes pin DO. Příchozí data jsou řazena ve vstupní frontě, dokud nedojde k jejich úplnému zpracování a odeslání. To může být iniciováno např. dosažením časového limitu, během kterého do vstupní fronty už nepřibyla žádná další data, naplněním celkové kapacity paketu (100 bytů), nebo při přijetí sekvence pro přejití do příkazového módu. Časový limit lze nastavit parametrem RO (packetization timeout). [5]

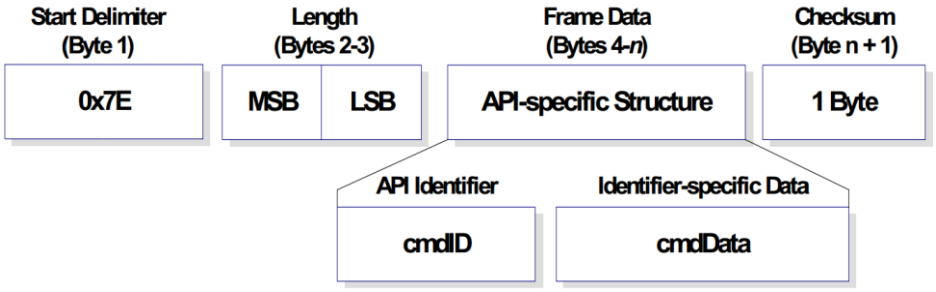
2.6.2 API mód

Hlavní podstatou API módu je co nejrychlejší a nejspolehlivější přenos vysoce strukturovaných dat. Využívá se speciálních rámců (viz obr. 2.14), které zvyšují úroveň interakce hostitelské aplikace se síťovými možnostmi modulu. Protože XBee rádia nemají dostatečnou paměť pro všechny instrukce transparentního a API módu, je nutné pro používání API módu změnit firmware modulu na verzi podporující API mód. v tomto módu jsou všechna příchozí a odchozí data zabalena ve

strukturovaných paketech obsahujících mj. řídící informace (AT příkazy). Díky tomu se zvyšuje celková komplexnost a náročnost řídících programů, které pracují s pakety na místo se surovými daty.

Užitečnost API módu se projeví zejména ve chvílích, kdy adresujeme více cílových uzlů, nebo když v aplikaci potřebujeme zjistit adresu zdrojového uzlu paketu. Součástí paketů je totiž zdrojová i cílová adresa a tak ji můžeme jednoduše zjišťovat, nebo měnit naší aplikací bez nutnosti měnit nastavení XBee modulu. API mód využijeme i v případě, že potřebujeme vzdáleně měnit konfiguraci cílových uzlů. Dovoluje nám totiž vzdáleně posílat AT příkazy a číst, nebo měnit jejich nastavení. Pro zjednodušení práce s API módem bylo vydáno několik oficiálních i neoficiálních knihoven, které můžeme v našich aplikacích použít.

Z důvodu komplexnosti API módu a faktu, že pro jednoduchou senzorovou síť s modely Arduino nepřináší nic užitečného, budu pro přenos dat používat transparentní mód. [5]



Obrázek 2.14: Obecná struktura API paketu [5] .

3 Popis technologií pro WSN

3.1 Výběr topologie pro WSN

Topologie popisuje reálné i logické umístění všech uzlů v síti a styl jejich vzájemného propojení. V bezdrátových senzorových sítích se zpravidla vyskytují maximálně tři druhy uzlů a to koordinátor, směrovače a koncové uzly. Jednotlivé typy uzlů spadají do dvou kategorií, plně funkční zařízení FFD (Full Function Device) nebo zařízení se sníženou funkcí RFD (Reduced Function Device). RFD uzly mohou přecházet do režimu spánku, kdy jsou téměř vypnuté a šetřit tak celkovou spotřebovanou energii. Ukládání zpráv a další potřebný servis jim zajišťuje jejich nadřazený spárovaný uzel, jenž je typu FFD. Koordinátor a směrovače musí být nutně typu FFD, naopak koncové uzely mohou být typu RFD. Všechny běžně používané topologie budou podrobněji popsány dále a jejich vizualizaci naleznete na obr. 3.1. Převážná část této podkapitoly byla převzata z textu [1]. [2, 9]

3.1.1 Prvky topologie

Koordinátor (Coordinator)

V bezdrátových senzorových sítích se většinou vyskytuje alespoň jedno zařízení fungující jako tzv. koordinátor. Koordinátor neboli základní uzel zajišťuje formování sítě, přidělování adres koncovým uzlům, zabezpečení sítě a udržení její stability. Je to nejdůležitější uzel, který shromažďuje všechna měřená data z koncových uzlů a dál je zpracovává.

Směrovač (Router)

Směrovač je plně funkční uzel, který zajišťuje bezpečné doručení dat z koncových uzlů sítě do základního uzlu. Může se připojit do již existující sítě, posílat zprávy, přijímat zprávy a směrovat je. Směrování je určení nejvhodnější cesty mezi dvěma uzly, které nejsou vzájemně v dosahu, s pomocí ostatních uzlů v síti. Většinou jsou umístěny stacionárně a jsou trvale napájeny. Směrovače se v síti nemusejí vůbec vyskytovat, pokud jsou všechna koncová zařízení v dosahu koordinátoru.

Koncové zařízení (End device)

Koncová zařízení jsou uzly vybavené sadou senzorů, kterými měří různé fyzikální veličiny. Mohou se připojit do již existujících sítí, přijímat a posílat zprávy, ale nepřeposílají zprávy od jiných uzlů. Jsou to zařízení se sníženou funkcí RFD. To znamená, že se mohou dočasně přepnout do tzv. spánkového módu, kdy jsou téměř vypnuty a nespotřebovávají téměř žádnou energii. Koncové zařízení musí být vždy spárováno se směrovačem popř. koordinátorem, který jim pomáhá se do sítě připojit a uchovává pro ně zprávy, ve chvílích kdy jsou ve spánkovém režimu a nemůžou je přijmout. Koncové uzly jsou

většinou mobilní a jsou napájeny z externího omezeného zdroje např. baterií. V některých případech také rozhoduje jejich velikost a odolnost, kdy je potřeba, aby byly co nejmenší a bylo je možno použít i na nezvyklých nebo špatně dostupných místech.

3.1.2 Možné síťové topologie

Pár (Pair)

Nejjednodušším seskupením jsou jen dvě zařízení, koordinátor a koncové zařízení. Koordinátor zformuje síť a koncové zařízení se do ní připojí.

Hvězda (Star)

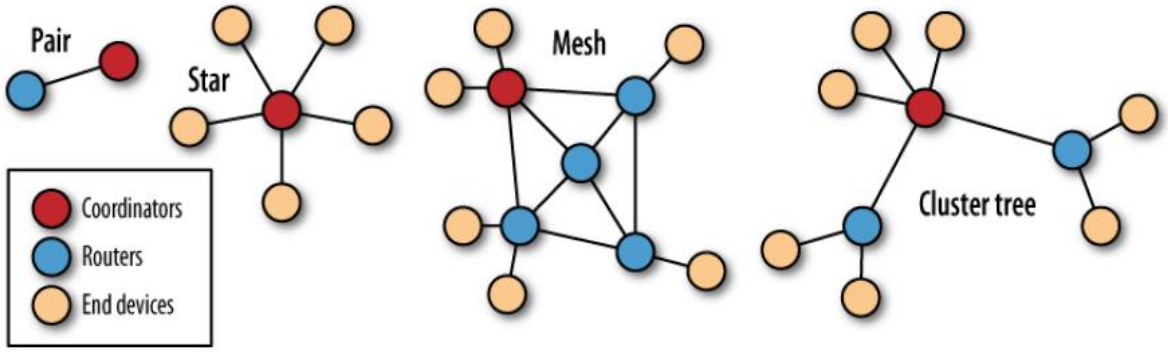
Propojení všech uzlů vzdáleně připomíná hvězdu. V jejím centru je koordinátor, který zformuje síť. Ostatní uzly jsou koncová zařízení, umístěná kolem centrálního uzlu, která se do sítě postupně připojují. Jedná se o běžně používanou topologii pro sítě s malým počtem uzlů.

Strom (Cluster Tree)

Topologie stromu se zvláště neliší od topologie mesh. Směrovače a koordinátor tvoří tzv. páteřní síť, ke které se připojují koncová zařízení.

Pletivo (Mesh)

Mesh topologie vzájemně libovolně propojuje koordinátor s okolními směrovači a vytváří tzv. „pletivo“, ke kterému se připojují ostatní koncová zařízení. Bezdrátové senzorové sítě založené na klasických topologiích mají komunikační páteř tvořenou propojením několika směrovacích zařízení, ke kterým se připojují koncové uzly sítě. Naopak sítě s mesh topologií žádnou komunikační páteř nemají. Představují distribuovaný systém, kde všechny uzly navzájem spolupracují a předávají si zprávy. V síti je mnoho redundantních spojení, které při výpadku uzlu nahradí jeho funkci. Zpráva se tak k cílovému uzlu může dostat různými cestami. Komunikace v sítích mesh se ustanovuje jen v případě potřeby a jen na nezbytně nutnou dobu, což podstatně méně zatěžuje přenosové komunikační pásmo. Jsou jednoduché na výstavbu a levné na náklady, protože není nutné složitě budovat páteřní síť. Takto se pro senzorové sítě jeví topologie mesh jako ideální volba a proto ji budu chtít použít. [2]



Obrázek 3.1: Síťové topologie [1] .

3.2 Výběr protokolu pro WSN

S bezdrátovými senzorovými sítěmi obvykle spojujeme jen nejrozšířenější síťový protokol ZigBee vybudovaný na standardu IEEE 802.15.4. Přesto ZigBee není jediným protokolem schopným pracovat v topologii typu mesh. Protokol DigiMesh, navržený firmou Digi International, je schopen tvořit mesh sítě stejně dobře jako protokol ZigBee a přitom je svým návrhem podstatně jednodušší. Všechny tři zmíněné protokoly budou popsány dále. Vzhledem k tomu, že jsem měl k dispozici pouze XBee moduly první série, jenž nepodporují protokol ZigBee, jedinou dostupnou alternativou schopnou operovat s topologií sítě typu mesh, je pouze protokol DigiMesh a proto byl vybrán a použit při implementaci sítě.

3.2.1 Protokol IEEE 802.15.4

Tento protokol zahrnuje sadu standardů, které definují řízení spotřeby, adresování, opravu chyb, formáty zpráv a další specifikace potřebné k řádné komunikaci mezi dvěma rádiovými zařízeními. Podporuje pouze topologie typu hvězda a pár**.** Poskytl základ, na kterém jsou vystavěny protokoly ZigBee a DigiMesh. [2, 5]

3.2.2 ZigBee protokol

Bezdrátový komunikační standard ZigBee je určen zejména pro nízko-výkonová zařízení v sítích PAN (Personal Area Network), pracujících na malé vzdálenosti, řádově desítky metrů. ZigBee je navržen jako jednoduchá a flexibilní technologie pro sestavení i rozsáhlejších bezdrátových sítí, u nichž není požadován přenos velkého objemu dat. K jeho přednostem patří především spolehlivost, jednoduchá a nenáročná implementace, velmi nízká spotřeba energie a příznivá cena. Díky nízkým nárokům na hardware a minimální spotřebě elektrické energie, tento protokol najde uplatnění v oblasti řízení budov, spotřební elektroniky, průmyslu a senzorových sítích např. právě v podobě

bateriově napájených senzorů. ZigBee je otevřený standard, vystavěný na fyzické a linkové vrstvě standardu IEEE 802.15.4. Na jeho vývoji se momentálně podílí více než 60 firem z celého světa. Protokol ZigBee definuje tři typy uzlů a to koncové uzly, koordinátor a směrovací uzly. [2]

Struktura komunikačního protokolu

Z důvodů nutnosti podpory ZigBee i v málo výkonných 8bitových mikrokontrolerech, byl kladen důraz na maximální jednoduchost implementace jeho protokolu, díky čemuž nezabere více než 30 kB programové paměti. Celý protokol se skládá ze tří základních vrstev, nejspodnější vrstvy standardu IEEE 802.15.4 (fyzická vrstva a kontrola přístupu k médiu MAC), nad kterými je definovaná síťová vrstva (NWK) a aplikační vrstva (APL). Tyto přídavné vrstvy přidávají standardu IEEE 802.15.4 dvě zásadní rozšíření funkčnosti a to směrování mezi více uzly a vytváření samo-uzdravujících se mesh sítí.

Podle modelu OSI fyzická vrstva specifikuje přístup k přenosovému médiu. MAC vrstva je podvrstvou linkové vrstvy a poskytuje mechanismus adresování zařízení v rámci sítě. Síťová vrstva realizuje připojení k síti, zabezpečení a směrování paketů mezi jednotlivými uzly. Aplikační vrstva zajišťuje ostatní potřebné služby. Skládá se z aplikační podvrstvy, ZigBee objektů a uživatelských aplikačních objektů [2].

3.2.3 DigiMesh

Protokol DigiMesh je novou alternativou protokolu ZigBee v oblasti bezdrátových senzorových sítích s topologií mesh. Standard DigiMesh je vyvinutý a patentovaný firmou Digi International. Přestože se na první pohled může jevit podobně jako protokol ZigBee, při jeho bližším prozkoumání zjistíme, že se jedná o zcela odlišný protokol. DigiMesh, na rozdíl od ZigBee, definuje pouze jeden typ uzlů. Jakožto silně homogenní síť, všechny uzly mohou vzájemně směrovat data a jsou mezi sebou volně zaměnitelné. Neexistují mezi nimi žádné rodičovské vztahy a všechny se mohou nakonfigurovat jako nízko-výkonová zařízení. Celou síť (resp. všechny uzly v ní) je možno vzájemně synchronizovat a synchronně cyklicky uspávat. V tu chvíli se všechna zařízení téměř vypnou a rapidně se sníží jejích příkon, což značně snižuje nároky na spotřebovanou energii jednotlivých uzlů.

Oproti ZigBee nám tento protokol nabízí se svými většími datovými rámci větší propustnost sítě a zároveň poskytuje delší dosah. DigiMesh umožňuje vytvářet daleko jednodušší strukturu mesh sítí, což přináší své výhody i nevýhody. Každý uzel se standardně chová jako směrovač a zároveň jako koncové zařízení. Změnou parametru CE (Coordinator Enable) v XBee modulu, můžeme uzel přepnout do módu koncového zařízení a vypnout tak jeho funkci jako směrovače. Tento protokol vytváří samo-uzdravující se síť, čili jakýkoli uzel se může kdykoli odpojit, nebo připojit k síti aniž by způsobil její nestabilitu nebo pád. Spolehlivost doručení paketů k cíli je zaručena zpětným zasíláním potvrzení o doručení zprávy od cílového zařízení, tzv. potvrzovacích ACK (acknowledgment) paketů. Uzly si neudržují žádnou trvalou mapu sítě, namísto toho k vyhledání optimální cesty dochází pouze

v případě potřeby. Při hledání optimální cesty se využívá algoritmu podobnému AODV (Ad hoc On Demand Distance Vector) [1]. DigiMesh není otevřeným standardem jako je ZigBee, ale je patentován firmou Digi International a proto pravděpodobně nebude nikdy tak rozšířen. V dalších podkapitolách se budeme protokolem DigiMesh zabývat podrobněji, jelikož bude implementován ve všech senzorových uzlech. [5, 10]

	ZigBee	DigiMesh
Typy uzlů v síti	koordinátor, směrovač, koncové uzly	jeden univerzální typ
Adresování v síti	MAC adresa (64 bitů) a síťová adresa (16 bitů)	MAC adresa (64 bitů)
Režim spánku	pouze koncové uzly	všechny uzly v síti
Zabezpečení	128bit AES, síť lze uzavřít proti přidání dalších uzlů	128bit AES
Velikost datové části rámce	80 B	256 B
Přenosová pásma a tok dat	2.4 GHz (250 kb/s), 900 MHz (40 kb/s) 868 MHz (20 kb/s)	2.4 GHz (250 kb/s), 900 MHz (150 kb/s)
Odolnost proti interferencím	DSSS - Direct sequence spread spectrum,.	DSSS - Direct sequence spread spectrum (2,4 Ghz), FHSS - Frequency hopping spread spectrum (900 Mhz)

Tabulka 3.1: Rozdíly mezi protokoly ZigBee a DigiMesh [10].

3.3 Adresování XBee v DigiMesh

Stejně tak jako má každý z nás svojí poštovní adresu i XBee modulům je přiděleno několik unikátních adres. Všechny rádiové moduly mají pevně přiřazené svoje unikátní 64bitové sériové číslo, jenž žádný jiný XBee modul na světě nemá. Sériové číslo je modulu přiřazeno při výrobě v továrně a je uloženo v parametrech SL (Serial Number Low) a SH (Serial Number High) modulu, které se zároveň automaticky používá jako zdrojová adresa**.**

Obecně nám nestačí znát pouze adresy jednotlivých zařízení, ale potřebujeme nějakým způsobem označit samotnou síť, ve které operují. K tomuto účelu slouží adresa sítě „Network ID“. Je to další 16bitová adresa, se kterou můžeme adresovat až 32768 různých sítí, kde každá z těchto sítí může obsahovat až 65536 dalších různorodých zařízení. Síťová adresa je uložena ve firmware XBee modulů jako parametr ID a může nabývat hodnot od 0x0 (0) do 0x7FFF (32767). Standardně je

adresa sítě nastavena na hodnotu 0x7FFF. Obecně se doporučuje tuto hodnotu změnit, aby nedocházelo k nechtěným kolizím s dalšími sítěmi v okolí.

Znát adresu sítě a všech zařízení nebude stačit, pokud nebudou všechny XBee moduly uzlů v síti pracovat na stejné frekvenci. Při zakládání sítě je obvyklé, že koordinátor automaticky vyhledá volnou frekvenci, kterou pak používá i zbytek uzlů v síti, např. u ZigBee. V případě DigiMesh výběr vhodného kanálu musíme zajistit sami nastavením parametru CH (Channel) ve firmwaru XBee modulu, který daný kanál bude nadále pokaždé používat. Pokud chceme se sítí přejít na jiný kanál, musíme ho manuálně změnit v nastavení každého XBee modulu. Klasické XBee moduly podporují všech 16 kanálů, zatímco XBee PRO podporují pouze 12 kanálů. Aby spolu mohly uzly vzájemně komunikovat, adresa sítě a zvolený kanál musejí být ve všech XBee modulech nastaveny stejně. [5]

3.3.1 Adresování unicastem

Při přenosu dat za použití unicast adresování, čili když data posíláme pouze jednomu koncovému zařízení, je spolehlivé doručení dat zajištěno pomocí opakovaného zasílání a potvrzování přijatých dat tzv. ACK paket. Počet pokusů pro opětovné zaslání paketu je dán hodnotou parametru NR (Network Retries). Pakety jsou zasílány maximálně NR + 1 krát. Potvrzení o přijetí paketu zasílá pouze cílový uzel. Pokud potvrzení o přijetí nebylo přijato v čase, ve kterém by trvala dojnásobná cesta paketu z jednoho konce sítě na druhý, paket je znovu odeslán. Adresa cílového zařízení (parametry SH a SL cílového zařízení) musí být uložena v parametrech DH a DL vysílajícího zařízení. Všechny koncové uzly sítě budou zasílat unicastové zprávy s naměřenými hodnotami ze senzorů do základního uzlu a proto jako cílovou adresu budou mít napevno nastavenou adresu základního uzlu. [5]

3.3.2 Adresování broadcastem

Broadcastové přenosy jsou přijímány a přeposílány všemi směrovači v síti. Vzhledem k tomu, že se při broadcastu nepoužívá potvrzování o přijetí paketu, zdrojový uzel vysílá paket vícekrát. Parametr MT (Broadcast Multi-Transmit) určuje počet opakování vysílání broadcastového paketu, který se vysílá MT + 1 krát (standardně čtyřikrát). Podobně tak všechny ostatní směrovače v síti tento paket přepošlou všem sousedním uzlům MT + 1 krát. S cílem vyhnout se nechtěným kolizím je v každém uzlu generováno náhodné zpoždění, se kterým každý směrovač daný paket přepošle. Proto časté zasílání broadcastových zpráv může celkem rapidně snížit propustnost sítě a tak by se mělo využívat jen v opravdu nezbytných případech.

Broadcastová adresa je 64bitová adresa s 16ti nejnižšími bity nastavenými na 1 a horními bity na 0. Pro vyslání takovéto zprávy se musí nastavit parametr DH na 0x0 a parametr DL na 0xFFFF. V API módu by cílová adresa musela být nastavena na hodnotu 0x000000000000FFFF. Broadcastové

vysílání bude převážně využíváno při zasílání synchronizačních zpráv pro pravidelné uspávání všech uzlů. [5]

3.4 Směrování v DigiMesh

Každý uzel v síti je schopen objevit spolehlivou cestu k cílovému uzlu pomocí směrovacího algoritmu a lokální asociativní směrovací tabulky. Směrovací algoritmus využívá reaktivní metodu odvozenou od algoritmu AODV (Ad hoc On Demand Distance Vector). Pokud cílové zařízení není v dosahu zdrojového zařízení, je paket potřeba poslat přes jiné dostupné směrovače v síti. K mapování cílové adresy zařízení na tzv. next-hop adresu, čili adresu sousedního směrovače nebo přímo adresu koncového zařízení pro zaslání paketu, slouží směrovací tabulka. Posláním paketu na next-hop adresu zpráva buď rovnou dosáhne svého cíle, nebo je přeposlána na další směrovač, který se obdobným způsobem postará o její doručení do cílového zařízení.

Zpráva s adresou typu broadcast je vyslána všem sousedním uzlům. Každý směrovač příchozí broadcastovou zprávu opět přepošle MT + 1 krát (standardně čtyřikrát) všem sousedním uzlům až nakonec obsáhne celou síť. Sledováním paketů se zabrání přeposílání broadcastové zprávy více než MT + 1 krát. Parametr MT (Broadcast Multi-Transmit) je možné změnit v nastavení XBee modulu. [5]

3.4.1 Získání optimální cesty k cíli

Proces zjišťování optimální cesty (route discovery) ze zdrojového do cílového uzlu je zahájen v případě, že zdrojový uzel nezná cestu k cílovému zařízení právě zpracovávaného paketu. Potom je takový paket přesunut do fronty, kde čeká, dokud se nedokončí proces vyhledání cesty k cílovému zařízení. Tento proces je zahájen i v případě, kdy se vyčerpá počet opětovných vyslání paketu bez úspěšného potvrzení o jeho přijetí cílovým uzlem. Vyhledání cesty začíná vysláním broadcastové zprávy ze zdrojového uzlu tzv. požadavek na cestu RREQ (route request). Jakýkoliv směrovač, který tento požadavek obdrží, ho buď zahodí, nebo přepošle na základě toho, zda má výhodnější cestu zpět ke zdrojovému uzlu. Pokud ano, uzel si uloží informaci z RREQ, požadavek upraví a vyšle ho jako broadcast zpět do sítě. Ve chvíli kdy cílový uzel obdrží RREQ, zašle odpověď unicastem zpět zdrojovému uzlu spolu s celou cestou z RREQ, nehledě na kvalitu cesty a počet předtím obdržených RREQ. Zdrojový uzel tak obdrží více odpovědí s různými cestami k cíli, z nichž vybere tu nejlepší s ohledem na kvalitu cesty, kterou použije pro odeslání čekajícího paketu. Část této adresy si zdrojové zařízení zároveň uloží do směrovací tabulky. [5]

3.5 Uspávání senzorové sítě s DigiMesh

Žádná senzorová síť se nemůže obejít bez řádného uspávání svých uzlů, neboť tak můžeme podstatně prodloužit jejich provozuschopnost. Uspávání koncových uzlů bych rozdělil na dvě části a to uspávání mikrokontroleru na desce Arduina a uspávání samotného XBee modulu.

Pro uspávání mikrokontroleru na desce Arduina můžeme použít knihovnu „sleep“ od firmy Atmel, jenž nám nabízí těchto pět spánkových režimů [18], seřazených od nejmenší energetické úspory po tu největší.

- režim nečinnosti (SLEEP_MODE_IDLE)
- režim redukce šumu (SLEEP_MODE_ADC)
- režim úspory energie (SLEEP_MODE_PWR_SAVE)
- pohotovostní režim (SLEEP_MODE_STANDBY)
- režim úplného vypnutí (SLEEP_MODE_PWR_DOWN)

Jednotlivé režimy jsou vhodné pro odlišné situace, zejména proto, že se liší dobou probuzení z tohoto režimu, kde samozřejmě režimy s větší úsporou energie mají delší dobu zotavení.

XBee moduly nabízejí řadu spánkových módů, čili režimů, kdy zařízení dočasně přejde do stavu, kdy téměř nespotřebovává žádnou energii. Tyto módy můžeme rozdělit do dvou základních skupin na asynchronní a synchronní. Asynchronní spánkový mód je používán pro ovládání režimu spánku na základě externího signálu (např. od mikrokontroleru). XBee moduly takto uspávané by tudíž neměly být používány pro směrování dat, protože dochází k jejich nepravidelnému uspávání, kdy nepřijímají ani neodesílají žádná data a nejsou tak schopny úspěšně propagovat zprávy po síti. Periodický synchronní spánkový mód je rysem protokolu DigiMesh, který umožnuje všechny uzly sítě vzájemně synchronizovat a uspávat tak celou síť současně v jeden moment. Uzly se synchronizují obdržením tzv. synchronizační zprávy, kterou pravidelně zasílá uzel pracující jako tzv. koordinátor spánkového režimu. Spánkový koordinátor zasílá tuto synchronizační zprávu ihned po probuzení na začátku každého spánkového cyklu. Zpráva je zasílána jako typ broadcast a je tudíž opakovaně přeposílána všemi uzly v síti. Pro změnu doby spánku a doby běhu celé sítě tedy stačí pouze změnit nastavení jednoho konkrétního uzlu, který pracuje jako koordinátor spánku [5, 1].

3.5.1 Spánkové režimy

Každý modul XBee může pracovat v jednom ze šesti spánkových režimů, které budou popsány dále. Pokud chceme využít synchronní spánkový mód, který nám nabízí protokol DigiMesh, budeme muset uzly sítě nakonfigurovat pouze s periodickým spánkovým módem a spánek podporujícím módem. Spánkové režimy se nastavují příslušnou hodnotou parametru SM (Sleep Mode) v XBee modulu. Všechny uzly sítě by měly být nastaveny se stejným režimem usínání, neboť uzly s asynchronními

spánkovými režimy nejsou kompatibilní s těmi se synchronními. Převážná část této podkapitoly byla převzata z textu [5].

Normální spánkový mód (SM = 0)

Ihned po zapnutí uzlu dojde k jeho automatickému přepnutí do normálního spánkového módu. V tuto chvíli se daný uzel nemůže uspat, dokud se nesynchronizuje se sítí, aby se mohl uspávat a probouzet právě ve stejnou chvíli jako zbytek ostatních uzlů v síti. Ve chvíli kdy dojde k synchronizaci uzlu se zbytkem sítě, bude pouze předávat synchronizační zprávy dalším uzlům sítě, ale nebude je sám generovat. Jakmile je uzel správně synchronizován, může kdykoli přejít do spánkového módu.

Asynchronní spánkový mód s pinem (SM = 1)

Tento režim umožní XBee modulu přejít ze stavu běhu do stavu spánku na základě stavu jeho pinu Sleep_RQ (pin 9). Ve chvíli kdy se na pinu Sleep_RQ objeví stav logické jedničky, modul dokončí přenos všech zpráv a přepne se do spánkového módu. Opět se probudí ve chvíli, kdy se na pinu Sleep_RQ změní hodnota do logické nuly.

Asynchronní cyklický spánkový mód (SM = 4)

Cyklický spánkový mód umožní modulu periodicky spát po předem stanovenou dobu a opět se na krátkou dobu probudit. Pokud modul po probuzení obdrží jakákoliv data, prodlouží se jeho doba běhu o hodnotu parametru ST (Wake Time), jinak je opět okamžitě uspán. Ve chvíli kdy je modul v režimu spánku, se na pinu On_SLEEP objeví logická nula. Naopak po probuzení se tato hodnota změní do logické jedničky.

Asynchronní cyklický spánkový mód s pinem (SM = 5)

Jak název napovídá, jedná se o variaci asynchronního spánkového módu s pinem a asynchronního cyklického spánkového módu. To umožňuje, aby mohl být v asynchronním cyklickém spánkovém módu probuzen na základě hodnoty signálu na pinu Sleep_RQ (pin 9).

Spánek podporující mód (SM = 7)

V tomto módu dochází k synchronizaci uzlu se zbytkem sítě, ale nedochází k vlastnímu uspávání uzlu. Vhodné zejména pro uzly, které jsou trvale napájeny, nebo slouží jako tzv. spánkový koordinátor. Uzel je kdykoli schopen zaslat synchronizační zprávu nově připojeným uzlům do sítě. K přenosu ostatních zpráv dochází pouze ve chvíli kdy je zbytek sítě probuzen.

Periodický spánkový mód (SM = 8)

Uzly v periodickém režimu spánku jsou pravidelně uspávány na koordinátorem definovanou dobu a probouzí se naráz se zbytkem sítě. Okamžitě po probuzení uzlu dojde k výměně dat a synchronizačních zpráv s ostatními uzly sítě a také se odešlou všechny zprávy uložené ve

vyrovnávací paměti XBee modulu. Potom uzel opět přejde do režimu spánku, kdy nemůže vysílat, ani přijímat žádné zprávy z UART portu od mikrokontroleru, ani od ostatních uzlů v síti. V době, kdy je modul v režimu spánku, je na pinu CTS nastavena logická jednička a naopak logická nula, když se opět probudí. Doba spánku (SP) a doba probuzení (ST) jsou definovány spánkovým koordinátorem v synchronizačních zprávách. Nesynchronizovaný uzel se bude pravidelně probouzet a pokoušet se získat synchronizační zprávu. Potom opět usne na dobu specifikovanou jako SP v lokálním XBee modulu, tak dlouho, dokud neobdrží synchronizační zprávu od spánkového koordinátora a nedojde k synchronizování se zbytkem sítě.

3.5.2 Výběr spánkového koordinátora

Spánkový koordinátor je standardně volen manuálně nebo procesem tzv. nominace. Manuální výběr koordinátoru probíhá nastavením bitu „preferovaný koordinátor spánku“ (bit 0) v parametru XBee modulu SO (sleep operations) na 1. Takto nastavený uzel bude vždy zasílat synchronizační zprávy při probuzení po každém spánkovém cyklu. Z tohoto důvodu je důležité, aby byl v síti vždy pouze jen jeden takto zvolený uzel. Jako spánkový koordinátor je vždy výhodné volit uzel, který je lokalizován v centru sítě, aby se minimalizovala cesta, kterou musí synchronizační zpráva urazit napříč sítí i k nejvzdálenějším uzlům. V naší síti bude jako spánkový koordinátor zvolen základní uzel, který pro to splňuje všechny základní požadavky.

Proces nominace nového spánkového koordinátora je zahájen po ztrátě kontaktu s původním spánkovým koordinátorem. Standardně je tento proces vypnut, nastavení tohoto chování je v parametru SO. Pokud uzel po sobě neobdrží tři a více synchronizačních zpráv může se stát, v závislosti na lokální konfiguraci, že se uzel začne po několika cyklech bez synchronizace chovat jako nový spánkový koordinátor. Obdobně je možné, např. při odpojení původního spánkového koordinátoru, že se více uzlů v síti začne chovat jako nový koordinátor. Pokud tato situace nastane, zahájí se proces zvolení hlavního koordinátora podle definovaných priorit. Nejvyšší prioritu mají uzly, které se nikdy neuspávají. Dále uzly, jenž mají nastaven bit preferovaný spánkový koordinátor v parametru SO (Sleep Options) na 1 a na konec, pokud se žádné takové zařízení v síti nevyskytuje, je zvolen uzel s nejvyšším sériovým číslem.

4 Realizace WSN

V této kapitole se budu věnovat sestavení bezdrátové senzorové sítě na základě dosud získaných poznatků. Cílová senzorová síť bude tvořena modely Arduino Fio pro koncové uzly a jedním modelem Arduino Uno, jenž bude sloužit jako hlavní uzel. Všechny uzly budou vybaveny XBee moduly první série, se kterými vytvoří bezdrátovou síť s topologií typu mesh. XBee moduly první série sice standardně topologie typu mesh, resp. protokol ZigBee nepodporují, ale můžeme ho nahradit alternativou firmy Digi International, změnou firmwaru všech zúčastněných modulů na firmware podporující protokol DigiMesh.

Abychom maximalizovali efektivitu a energetickou nenáročnost celé sítě, všechny XBee moduly, včetně mikrokontrolerů na hostujících deskách Arduina, se budou cyklicky synchronně uspávat. Synchronní cyklický spánek je specialitou protokolu DigiMesh, který je schopen s pomocí tzv. koordinátoru spánku vzájemně synchronizovat uzly sítě a všechny je v jeden moment uspat. Jako koordinátor spánku bude sloužit hlavní uzel, který je pro to ideální vzhledem k faktu, že je trvale napájen a většinou bude lokalizován v centru sítě.

Koncové uzly budou napájeny z lithiovo polymerových baterií a budou opatřeny sadou senzorů (světelných a teplotních) připojených na analogové a digitální vstupní piny, ze kterých se budou snímat měřená data a odesílat do základního uzlu. Základní uzel je bude přes sériovou linku počítačem, kde se budou dál zpracovávat a vizualizovat.

4.1 Realizace uzlů sítě

Všechny senzorové uzly pro vzájemnou bezdrátovou komunikaci používají XBee moduly první série s firmwarem XB24-DM (XBee DigiMesh 2.4), který se musel do modulů nahrát s použitím programu X-CTU od firmy Digi International (viz kap. 2.4.5). S tímto firmwarem XBee moduly operují s komunikačním protokolem DigiMesh v transparentním datovém módu a proto jsou řídící programy pro mikrokontrolery Arduina velmi snadné na pochopení. Aby se mohly všechny uzly úspěšně spojit, každý z nich musí mít nutně nastavené stejné ID sítě a operující kanál.

4.1.1 Koordinátor

Základní uzel neboli koordinátor shromažďuje naměřená data ze senzorů koncových uzlů a dále je zpracovává. Po dobu provozu sítě je připojen k počítačové stanici, kde je spuštěn klientský program, kterému předává příchozí naměřená data. Jako koordinátor je použit model Arduino Uno, jenž je připojen prostým USB kabelem přímo k počítači. XBee modul základního uzlu je třeba naprogramovat dle parametrů uvedených v tabulce 4, s pomocí obyčejného sériového terminálu nebo programu X-CTU (viz kap. 2.4.5). Pro ostatní parametry se předpokládá, že jsou v továrním nastavení

a proto je vhodné před samotným programováním XBee modulu uvést jeho firmware, příkazem RE (Restore Defaults), do továrního nastavení a následně tovární hodnoty uložit příkazem WR (Write). Podrobnější popis jednotlivých parametrů naleznete v tabulkách přílohy 3.

Základní uzel zároveň slouží jako tzv. spánkový koordinátor, který podle svých parametrů SP (doba spánku) a ST (doba provozu) synchronně uspává a probouzí ostatní uzly sítě. Jako spánkový koordinátor se základní uzel jako jediný neuspává a pracuje pouze jako zařízení podporující spánkový mód, jenž rozesílá synchronizační zprávy všem ostatním uzlům. K nastavení doby spánku a provozu všech uzlů v síti tedy stačí pouze změnit parametry spánkového koordinátora.

Mikrokontroler základního uzlu je třeba naprogramovat (viz kap. 2.2.2), pomocí vývojového prostředí Arduino IDE, přiloženým programem „coordinator.ino“, který naleznete ve složce „/Arduino/Uno/“ na přiloženém DVD.

Tento program, ihned po inicializaci sériového spojení s klientským programem běžícím na počítači, přepne XBee modul koordinátoru do příkazového módu. V příkazovém módu se z nastavení modulu získá aktuální ID sítě a používaný kanál. Následně se odešle zpráva o inicializaci koordinátoru klientskému programu, ve kterém se předá konkrétní ID sítě a kanál. Klientský program vyhledá, popř. vytvoří záznam této bezdrátové sítě v databázi a následně čeká na příchozí data ze senzorů. V této chvíli je veškerá příchozí komunikace od koncových uzlů sítě předávána klientskému programu, kde se dále zpracovává a ukládá do databáze pro pozdější vizualizaci.

Parametr	Název a popis	Hodnota
ID	Adresa sítě.	7777
CH	Kanál.	F
SO	Možnosti spánku.	5
SM	Spánkový režim.	7
SP	Doba spánku.	BB8 (3 sek.)
ST	Doba provozu.	9C4 (2,5 sek.)

4.1: Nastavení XBee modulu koordinátoru.

4.1.2 Koncové uzly

Koncové uzly jsou vybaveny senzory, kterými měří různé fyzikální veličiny ze svého okolí. Naměřená data jsou odesílána do základního uzlu, kde jsou dále zpracovávána. Pro koncové uzly je použit model Arduino Fio. Přesto, že tyto senzorové uzly zároveň slouží jako směrovače, jsou díky schopnostem protokolu DigiMesh synchronně cyklicky uspávány. Ihned po zapnutí začne XBee modul očekávat synchronizační paket od spánkového koordinátora, podle kterého se nastaví, na jak dlouho se bude uspávat a opět probouzet. Přestože XBee modul tvoří největší část spotřebované

energie senzorového uzlu, uspává se nejenom samotné XBee, ale na základě jeho stavu dochází i k uspávání mikrokontroleru Arduina. Abychom mohli propojit spánkové cykly XBee a mikrokontroleru, musíme nejdříve vodičem propojit pin CTS (Clear To Send) se třetím digitálním pinem, který umožňuje vyvolat externí přerušení. Ve chvíli, kdy je XBee modul v režimu spánku, je na pinu CTS nastavena logická jednička, v opačném případě logická nula. Toho využijeme a funkcí „`attachInterrupt(1, wakeUp, FALLING)`“ připojíme ke třetímu digitálnímu pinu funkci „`wakeUp`“, která se zavolá po vyvolání externího přerušení na tomto pinu (na padající hranu signálu CTS). V této funkci dojde ke změření hodnot ze senzorů, k jejich odeslání a následně k okamžitému uspání mikrokontroleru bez ohledu na stav XBee modulu. Čili mikrokontroler Arduina je vypnut do chvíle, dokud není XBee modul připraven k vysílání. Mikrokontroler se uspává do režimu s minimální spotřebou a je téměř vypnut (režim SLEEP_MODE_PWR_DOWN).

Senzorové uzly sítě byly vybaveny digitálními teplotními čidly (DS18B20) a analogovými světelnými čidly (LDR 05-75), které budou popsány v následující podkapitole.

XBee moduly koncových uzlů je třeba naprogramovat, obdobně jako u základního uzlu, dle parametrů uvedených v tabulce 4. Taktéž mikrokontrolery koncových uzlů je třeba naprogramovat (viz kap. 2.2.2) přiloženým programem „`end_node.ino`“, který naleznete ve složce „`/Arduino/Fio/`“ na přiloženém DVD. Před samotným programováním je třeba v programu nastavit proměnnou DEV_ID, která charakterizuje identifikaci uzlu. Musí se jednat o unikátní hodnotu (v rámci sítě) z písmen a číslic o velikosti 3 znaků. Dále je nutné povolit typ připojených senzorů (proměnné ENABLE_I pro světelný senzor a ENABLE_T pro teplotní senzor).

Program používá standardní volně dostupné knihovny pro práci s 1-Wire (OneWire) sběrnicí (ta bude popsána v další části), digitálními teplotními čidly (DallasTemperature) a uspávání mikrokontroleru (Avr/Sleep). Po jeho inicializaci dojde k nastavení rozlišení všech připojených digitálních teploměrů na jejich maximum. Poté se mikrokontroler uspí a čeká až se XBee modul synchronizuje se zbytkem sítě. Následně dochází k pravidelnému probouzení mikrokontroleru, změření hodnot ze senzorů, jejich odeslání do základního uzlu a jeho opětovnému uspání.

Aby se mohl spánkově nesynchronizovaný uzel připojit k již existující síti, musí obdržet synchronizační zprávu od uzlu již synchronizovaného se zbytkem sítě. Po zapnutí uzel pravidelně zasílá broadcastové zprávy s požadavkem o zaslání synchronizační zprávy, následně se uzel uspí na dobu definovanou parametrem SP. Jakýkoli uzel v síti po obdržení tohoto požadavku ihned odešle odpovídající synchronizační zprávu. Vzhledem k tomu, že síť může být uspávána po delší dobu, ve které uzly nejsou schopny odpovědět na požadavek o synchronizační zprávu, jsou zde metody pro synchronizaci uzlu ve chvíli kdy jsou ostatní uzly uspány. Jednou z těchto metod je zapnutí uzlu v dosahu koordinátoru. Koordinátor je vždy probuzen a schopen ihned odpovědět na požadavek o synchronizační zprávu.

Parametr	Název a popis	Hodnota
ID	Adresa sítě.	7777
CH	Kanál.	F
SO	Možnosti spánku.	2
SM	Spánkový režim.	8

4.2: Nastavení XBee modulu koncového uzlu.

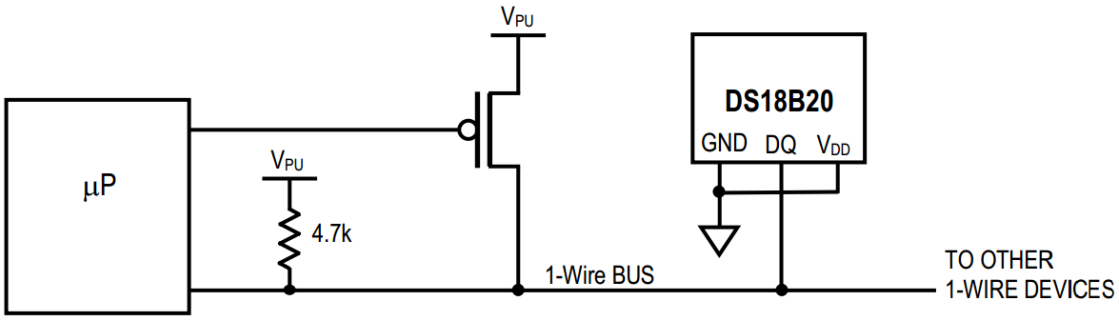
4.2 Senzory

Senzory jsou nezbytnou součástí každé senzorové sítě. Jsou zdrojem analogových nebo digitálních hodnot různých měřených fyzikálních veličin. Výstupní signál analogových senzorů je udán např. hodnotou napětí, proudu nebo frekvencí. Naopak digitální senzory naměřené hodnoty poskytují rovnou v digitální číslicové formě např. jako binární hodnotu.

4.2.1 Teplotní senzor DS18B20

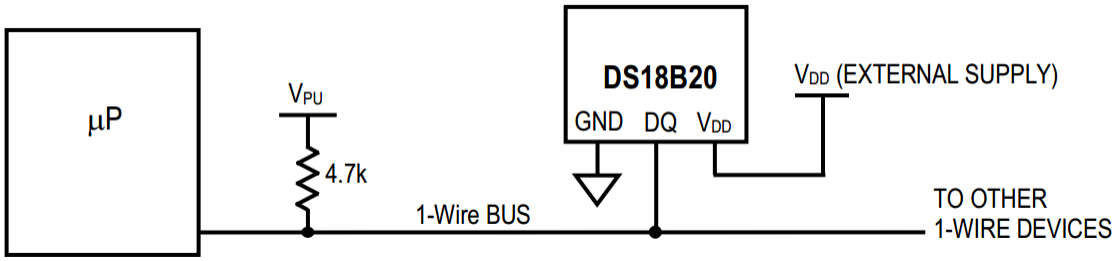
Digitální teplotní senzor DS18B20 od společnosti Dallas Semiconductor, patří mezi velmi přesné, strukturálně jednoduché a přitom výkonné teploměry. Tento senzor spadá do rodiny programovatelných digitálních teploměrů s možností funkce jako termostatu. Poskytuje přímo číslicový ekvivalent naměřené teploty s nastavitelnou přesností 9 až 12 bitů, čímž odpadají problémy s jeho kalibrací, linearizací a A/D převodem do digitální podoby. Rozsah měřitelných teplot se pohybuje od -55 °C do 125 °C s maximální chybou $\pm 0{,}5$ °C. Výstupní hodnoty jsou kalibrovány rovnou na stupně Celsia. DS18B20 může být napájen přímo z datového vodiče v tzv. parasitním módu, což eliminuje nutnost dalšího napájecího vodiče (viz obr. 4.1).

Se senzorem můžeme komunikovat přes 1-Wire sběrnici, která k tomu byla účelně navržena firmou Dallas Semiconductor. Ve své podstatě je tato sběrnice podobná konceptu sběrnice I2C, ale je navržena pro delší dosah a menší datové přenosy. Obvykle se používá ke komunikaci se sítí drobných senzorů např. sada teploměrů. Jednotlivé senzory mají unikátní 64bitové sériové číslo, což umožňuje pracovat více senzorům zároveň na jedné sběrnici. Takto vytvořená síť senzorů, ovládaná mikrokontrolerem, se označuje jako MicroLan. Ke komunikaci přes sběrnici postačují pouze dva vodiče, jeden datový, který zároveň slouží k napájení senzorů a druhý zemnící. [9]



Obrázek 4.1: Parasitní zapojení DS18B20 [9].

Teploměr může být k desce senzorového uzlu zapojen v parazitním módu nebo s externím zdrojem napájení. Při parasitním módu je datový vodič připojen k digitálnímu pinu 2, zemnící vodič je připojen na zemnící pin GND a mezi datovým a napájecím vodičem desky Arduina je 4,7 kOhm odpor (viz obr. 4.1). Napájení teploměru z externího zdroje je dle zapojení na obr. 4.2.



Obrázek 4.2: Zapojení DS18B20 s externím zdrojem energie [9].

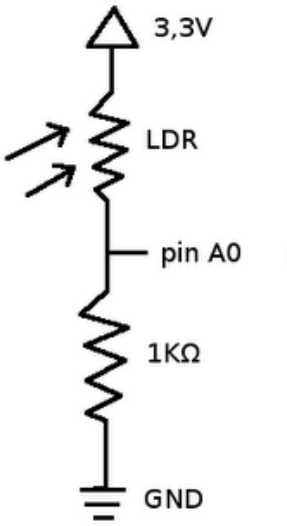
Pro komunikaci s teploměrem přes 1-Wire sběrnici byla použita volně dostupná knihovna OneWire od Arduina. Protože se jedná o digitální teploměr, změření a převod teploty do číslicové podoby vyžaduje určitý čas. Čím vyšší přesnost měření vyžadujeme, tím bude doba převodu delší viz tab. 3. Důležitou součástí teploměru je jeho konfigurační registr, ve kterém můžeme snížit přesnost měření. Ke konfiguraci a manipulaci s jednotlivými teplotními senzory firma Dallas Semiconductor vydala knihovnu DallasTemperature, kterou jsem použil pro nastavení maximálně přesného 12bitového měření a získání teploty z každého teplotního senzoru.

Přesnost (bit)	Doba převodu (max)	Jednotka
9	93,75 ms	0,5 °C
10	187,5 ms	0,25 °C
11	375 ms	0,125 °C
12	750 ms	0,0625 °C

Tabulka 4.3: Závislost přesnosti teploty na době převodu [9].

4.2.2 Fotorezistor LDR 05-75

Fotorezistor je polovodičová elektronická součástka, jejíž elektrický odpor se snižuje se zvyšující se intenzitou dopadajícího světla, čili jeho elektrická vodivost se ve tmě snižuje. LDR (Light Dependent Resistor) 05-75 zvyšuje svůj odpor s klesajícím osvětlením od 75 k0hm do 5 MOhm. K základně je zapojen podle schéma na obr. 17 jako tzv. odporový dělič a tak se s měnícím se odporem na fotorezistoru mění napětí na pinu A0. Toto napětí, v rozmezí 0 až 3,3 V, je 10bitovým analogově-digitálním převodníkem převedeno na ekvivalentní digitální hodnotu v rozpětí 0 až 1024, kde 1024 značí maximální hodnotu napětí 3,3 V. Čtení a převod hodnoty je iniciován poskytnutou funkcí analogRead(ID), kde ID značí číslo pinu, ze kterého chceme danou hodnotu získat. [17]



Obrázek 4.3: Zapojení LDR k Arduinu.

4.3 Vizualizace naměřených dat

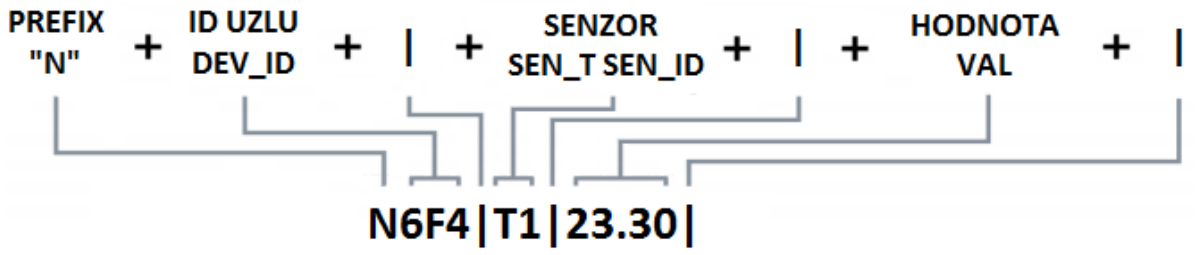
Všechna naměřená data ze senzorových uzlů jsou směrována do základního uzlu tzv. koordinátoru. Ten je připojen USB kabelem k počítači, na němž je spuštěn klientský program, který všechna příchozí data zpracovává a ukládá do databáze. Pro samotnou vizualizaci naměřených dat bylo vytvořeno webové rozhraní, ve kterém je z naměřených hodnot pro každý měřící senzor generován přehledný graf. Databáze a webové rozhraní mohou být umístěny lokálně nebo na vzdáleném serveru, čímž celou aplikaci můžeme zpřístupnit z internetu. Zdrojové kódy klientského programu i webového rozhraní jsou k dispozici na přiloženém DVD, kde rovněž naleznete postup pro zprovoznění celé aplikace.

4.3.1 Klient

Klientský program musí být po celou dobu provozu senzorové sítě trvale spuštěn. Zpracovává příchozí data ze základního uzlu a ukládá je do databáze pro jejich pozdější zpracování a vizualizaci. Je napsán v objektově orientovaném jazyce Ruby [21] verze 1.9.3p125, ve kterém je napsané i samotné webové rozhraní. Jsou v něm použity standardní knihovny (gemy) pro práci se sériovým portem, databází a parametry příkazové řádky. Klientský program „`klient.rb`“ ve složce „`/Klient/`“ na přiloženém DVD očekává před jeho spuštěním nastavení adresy portu, ke kterému je základní uzel připojen pomocí USB kabelu. Instrukce pro jeho nastavení a spuštění jsou přiloženy ve jeho složce.

Po spuštění programu se otevře sériové spojení se základním uzlem. Po inicializaci základního uzlu dojde k vytvoření, popř. vyhledání záznamu měřené sítě v databázi. Následně se začnou příchozí

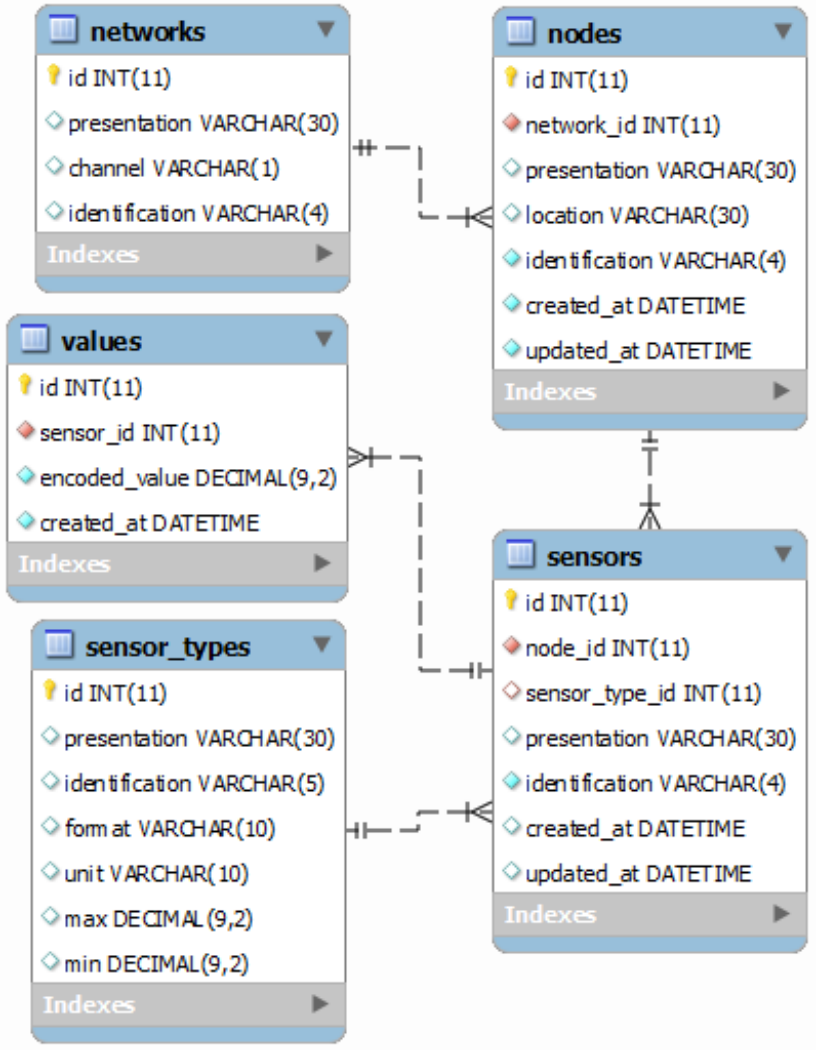
zprávy zpracovávat v nekonečné smyčce až do ukončení programu. Očekávaný formát zprávy odpovídá obr. 4.4 a je kontrolovaný sadou regulárních výrazů. DEV_ID je identifikátor označující každý konkrétní uzel sítě. Musí být explicitně uveden v kódu každého senzoru. SEN_T označuje typ senzoru, jenž může nabývat hodnoty T (pro teplotu) nebo I (pro osvětlení). SEN_ID je identifikátor konkrétního senzoru na senzorovém uzlu. VAL značí vlastní naměřenou hodnotu ze senzoru.



Obrázek 4.4: Tvar příchozí zprávy s naměřenými daty.

Pro ukládání naměřených dat se využívá databáze MySQL verze 5.5 s 5 tabulkami, které jsou ve vzájemných vztazích dle ER diagramu na obr. 4.5. Výběr některých datových typů sloupců a primárních klíčů tabulek je specifický pro framework použitý k vytvoření webového rozhraní, kde každý záznam v databázi má svoje unikátní ID (sloupec id). Struktura tabulek je univerzálním způsobem uložena v souborech tzv. „migracích“, což zajišťuje jejich přenositelnost v rámci různých databází. Následně bude popsána jen ta část tabulek sloužící k uložení naměřených dat.

V tabulce sítě „networks“ se ukládají záznamy o konkrétní měřené síti, čili její ID (identification), kanál (channel) a její název (presentation). Ke každému záznamu v tabulce sítě lze přiřadit uzly z tabulky „nodes“. V záznamu uzlu se uchovává jeho název (presentation), umístění (location), DEV_ID (identification), doba kdy byl vytvořen (created_at) a doba jeho poslední změny (updated_at). K jednotlivým uzlům lze přiřadit senzory z tabulky senzorů „sensors“, které si uchovávají název senzoru (presentation), SEN_ID (identification), dobu kdy byly přidány (created_at) a dobu kdy byly naposledy upraveny (updated_at). Každému senzoru lze přiřadit naměřené hodnoty v tabulce hodnot „values“, kde se ukládá naměřená hodnota (encoded_value) a doba kdy byla přijata (created_at). Senzorům jsou dále přiřazeny jejich typy z tabulky typů senzorů „sensor_types“, kde se uchovává rozsah měřených hodnot senzoru (min, max), jednotka (unit), typ ukládaných dat (format) a SEN_T (identification).



Obrázek 4.5: ER diagram tabulek pro uložení měřených dat v databázi.

4.3.2 Webové rozhraní

Webové rozhraní je napsané také v jazyku Ruby verze 1.9.3p125, konkrétně byl použit framework RoR (Ruby on Rails) [22] verze 3.2.2 a jeho MVC (Model View Controller) architektura. Rozhraní je možné umístit na externí server spolu s databází a zpřístupnit tak výsledky měření sítě na internetu.

Základním principem Rails aplikací je „Konvence má přednost před konfigurací“, tedy že programátor konfiguruje pouze ty části aplikace, které se liší od jejího běžného nastavení. Dalším principem je znovupoužitelnost napsaného kódu DRY „Don’t repeat yourself“. Architektura MVC odděluje části webové aplikace odpovědné za manipulaci dat (models), části řídící, kde se zpracovávají vstupy od uživatele a řídí jejich zobrazení (controllers) a části starající se o zobrazení dat (views). Tomu odpovídá i adresářová struktura výsledné webové aplikace. [22]

Webové rozhraní tvoří dvě hlavní sekce „Sítě“ a „Senzory“ viz náhledy v příloze 1. Sekce senzory umožňuje přidávání a úpravu typů fyzických senzorů umístěných na senzorových uzlech. Dle

tohoto nastavení se pak generují výsledné grafy. V sekci sítě nalezneme seznam všech dosud měřených sítí a detailní grafy naměřených dat z jednotlivých senzorů, které jsou roztříděny podle příslušnosti k jednotlivým senzorovým uzlům. Grafy se vykreslují a automaticky aktualizují s pomocí javascriptových knihoven jQuery verze 1.7.2 a jQuery Flot verze 0.7. Je u nich možné nastavit periodu jejich aktualizace a rozsah zobrazovaných hodnot formou výběru z rolovacího menu.

Provoz webového rozhraní na lokální počítačové stanici zajišťuje webový server Webrick, ten je jako knihovna součástí prostředí jazyka Ruby a tak není nutné instalovat žádný další server. Zdrojové soubory rozhraní jsou spolu s postupem jeho instalace a spuštění umístěny ve složce „`/Server/`“ na přiloženém DVD.

4.4 Testování sítě

Po navržení bezdrátové senzorové sítě je nedílnou součástí jejího vývoje otestování její funkčnosti. Navržená senzorová síť byla sestavena a čtyři dny testována v prostorách školy VUT FIT, v areálu bývalého kláštera Kartuziánského řádu, který poskytl ideální zátěžové podmínky pro vyzkoušení sítě.

Testovací síť byla složena z pěti nezávislých měřících uzlů a jedné základny, z nichž 4 měřící uzly byly vybaveny XBee modulem s prutovou anténou a zbylý uzel se základnou byly vybaveny XBee modulem s čipovou anténou. Senzorové uzly byly rozmístěny v dostupných kancelářích podle topologie na obr. 4.6, vzdálených od sebe řádově v jednotkách až desítkách metrů. Ke všem pěti senzorovým uzlům byl připojen alespoň jeden teplotní senzor (DS18B20). Další dva fotorezistory (LDR 05-75) byly připojeny k uzlům č. 2 a 4, jenž kvůli tomu byly umístěny na okenních parapetech. Základní uzel byl po celou dobu měření připojen k počítačové stanici s nainstalovaným prostředím, ve kterém byl spuštěn klientský program a webový server s databází. Senzorové uzly se probouzely a měřily data ze svých senzorů v pravidelných půlminutových intervalech. Po zbytek doby byly synchronně uspány a téměř vypnuty.

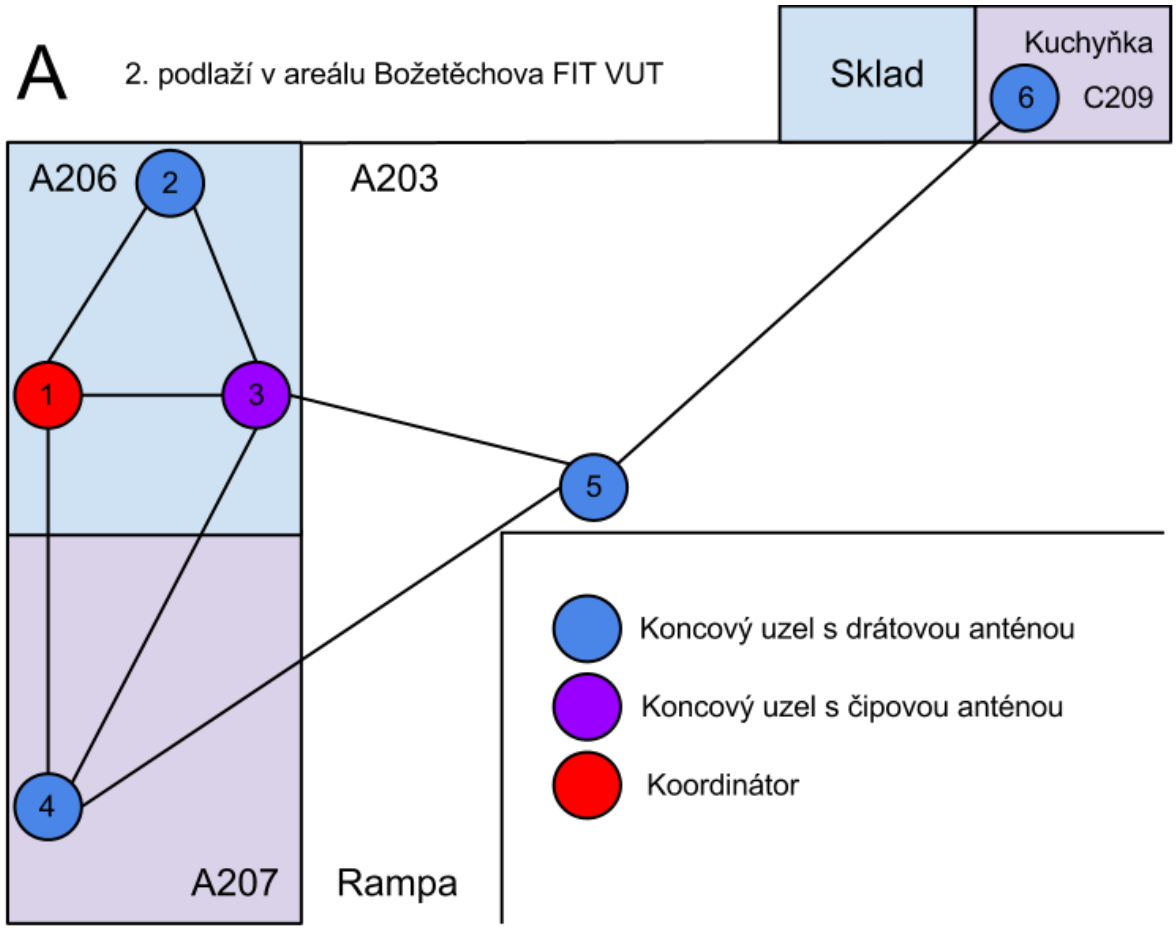
Testování sítě bylo rozděleno na dvě fáze. V první fázi se ověřovaly schopnosti směrování naměřených dat z koncových uzlů do základního uzlu, při změně topologie uzlů, popř. při výpadcích některých klíčových směrovacích uzlů. Ve druhé fázi se ověřovala dlouhodobá schopnost provozu sítě.

Z pohledu směrovacích schopností bezdrátové sítě se ukázalo, že protokol DigiMesh je plně schopen zvládnout problémy spojené s výpadky směrovacích uzlů a případným nalezením alternativní cesty k cíli (pokud nějaká existuje). Při testování docházelo k aktivním přesunům jednotlivých uzlů a změně topologie tzv. „on the fly“ za chodu sítě. Při simulovaných výpadcích uzlů (vypnutím konkrétního uzlu) byl protokol DigiMesh schopen vykompenzovat ztrátu tohoto uzlu a použít alternativní cestu.

To bylo odzkoušeno např. při testování schopnosti směrování naměřených dat z pátého a šestého uzlu, kdy se střídavě vypínaly uzly č. 3 a 4. Musela se tedy pokaždé najít náhradní cesta přes opačný uzel, který zůstal zapnutý.

Testování z pohledu dlouhodobějšího pozorování a měření ukázalo nepravidelné výpadky některých uzlů ze své spánkové synchronizace, čímždošlo ke ztrátě části naměřených dat v době výpadku. Přesto, že XBee moduly uzlů mají implementovanou vyrovnávací paměť pro neodeslaná data, při delších výpadcích ze spánkové synchronizace došlo k přetečení této paměti a tím i ke ztrátě většiny neodeslaných naměřených dat. Uzly se po určité době (někdy až po několika hodinách) znovu spárovaly a začaly opět zasílat naměřená data. Problémem může nastat ve chvíli, kdy ze synchonizace vypadne některý z klíčových směrovacích uzlů a zabrání se tak doručení dat od uzlů z jinak nepřístupné části sítě. Proto bych doporučil synchronní cyklické uspávání používat pouze tam, kde je výsledná síť robustní se spoustou redundantních spojení, jenž poskytnou množství alternativních cest.

Ukázku naměřených dat z uzlu č. 4 a jejich reprezentaci formou grafů teploty a intenzity osvětlení naleznete v příloze 2.



Obrázek 4.6: Testovací topologie.

5 Závěr

Cílem bakalářské práce bylo seznámit se s hardwarovou platformou Arduino a navrhnout z jejích komponent bezdrátovou senzorovou síť, založenou na radiofrekvenčních modulech XBee, se zaměřením zejména na schopnost sítě směrovat data ze senzorových uzlů do základního uzlu. Následně měla být navržená síť sestavena a otestována její směrovací schopnost v budově školy.

V první části práce jsem představil samotnou platformu Arduino, její vývojové prostředí a vhodné komponenty pro nasazení v oblasti WSN. Dále se zde čtenář seznámil s radiofrekvenčními komunikačními moduly XBee, kterými byly vybaveny všechny senzorové uzly sítě.

Výběru vhodné topologie a komunikačního protokolu se věnovala druhá část práce. Byl zde představen protokol DigiMesh od firmy Digi International, alternativa protokolu ZigBee, jenž byl implementován v jednotlivých uzlech sítě. Tento protokol definuje pouze jeden typ senzorových uzlů, jenž slouží jako koncové uzly a zároveň jako směrovače. DigiMesh umožňuje všechny tyto uzly vzájemně synchonizovat a cyklicky je naráz uspávat, což značně snižuje jejich celkovou energetickou náročnost.

V praktické části práce jsem z komponent platformy Arduino vytvořil bezdrátovou senzorovou síť, jejíž všechny uzly byly synchronně uspávány. Z pohledu požadavku zadání na ověření schopnosti směrování dat z koncových uzlů do základního uzlu, se při testování směrovacích schopností protokolu DigiMesh ověřilo, že je plně schopen vypořádat se s náhlými výpadky směrovacích uzlů a vyhledáním alternativní cesty k cíli. Testování z pohledu dlouhodobějšího provozu sítě ukázalo, že uzly nepravidelně ztrácely svoji spánkovou synchronizaci se zbytkem sítě a docházelo tak k částečným ztrátám měřených dat po dobu, než se opět synchronizovali se zbytkem sítě. Problém může nastat ve chvíli výpadku některého z klíčových směrovacích uzlů, který by zabránil doručení dat od uzlů z jinak nepřístupné části sítě. Proto bych doporučil používat uspávání směrovacích uzlů pouze v robustních sítích s dostatečným počtem redundantních spojení.

Oblast synchronního cyklického uspávání WSN si i nadále zasluhuje svoji pozornost a může být předmětem dalšího zkoumání. Jiným námětem pro následující vývoj může být rozšíření řídících a komunikačních schopností jednotlivých uzlů, které jsem v práci nastínil, využitím API módu a jejich případné vzdálené ovládání.

Mezi hlavní přínosy práce patří bezesporu to, že na reálné implementaci měla možnost ukázat použití platformy Arduino, komunikačních modulů XBee a protokol DigiMesh s jeho schopností synchronního uspávání uzlů. Díky této práci jsem získal mnoho nových znalostí a zkušeností zejména v oblasti tvorby WSN. Považuji ji za úvodní seznámení s problematikou WSN a chtěl bych se této oblasti i nadále věnovat v nastávajícím studiu na VUT FIT.

Literatura

[1] FALUDI, Robert. Building Wireless Sensor Networks. 1. vydání, O'Reilly Media, 2010, 293 s. ISBN 978-0-596-80773-3.

[2] EADY, F. Hands-On ZigBee: Implementin 802.15.4 with Microcontrollers. 1. vydání, Newnes., 2007, 352 s. ISBN 978-0-12-370887-8.

[3] HEBEL, M. - BRICKER, G. - Harrys D. Getting Started with XBee RF Modules. 1. vydání, Parallax Inc., 2010, 161 s. ISBN 978-1-928-98256-2.

[4] MARGOLIS, M. Arduino Cookbook. 1. vydání, O'Reilly Media, 2011, 662 s. ISBN 978-0-596-80247-9.

[5] XBee / XBee-PRO DigiMesh 2.4 RF Modules [pdf]. 6. 1. 2012, [cit. 8. 5. 2012]. Dostupné na URL: <http://ftp1.digi.com/support/documentation/90000991_E.pdf>

[6] XBee & Xbee-PRO OEM RF Module Antenna Considerations [pdf]. [cit. 9. 1. 2012]. Dostupné na URL: <http://ftp1.digi.com/support/images/XST-AN019a_XBeeAntennas.pdf>

[7] X-CTU Configuration & Test Utility Software [pdf]. [cit. 9. 1. 2012]. Dostupné na URL: <http://ftp1.digi.com/support/documentation/90001003_A.pdf>

[8] DS18B20 Programmable Resolution 1-Wire Digital Thermometer [pdf]. [cit. 9. 1. 2012]. Dostupné na URL: <http://datasheets.maxim-ic.com/en/ds/DS18B20.pdf>

[9] TARANNUM, S. Wireless Sensor Networks. 1. vydání, InTech., 2011, 342 s. ISBN 9789533073255.

[10] VOJÁČEK, A. DigiMesh – jednodušší bezdrátová síť podobná ZigBee [online]. 16. 11. 2008, [cit. 12. 4. 2012]. Dostupné na URL: <http://automatizace.hw.cz/digimesh-jednodussi-bezdratova-sit-podobna-zigbee>

[11] MALÝ, M. Arduino: Vývojový kit pro hrátky s hardware [online]. 30. 9. 2009 ,[cit. 9. 1. 2012]. Dostupné na URL: <http://www.root.cz/clanky/arduino-vyvojovy-kit-pro-hratky-s-hardware/>

[12] Arduino Uno [online]. [cit. 5. 5. 2012]. Dostupné na URL: <http://arduino.cc/en/Main/ArduinoBoardUno>

[13] Arduino Fio [online]. [cit. 5. 5. 2012]. Dostupné na URL: <http://arduino.cc/en/Main/ArduinoBoardFio >

[14] XBee Shield [online]. [cit. 2. 5. 2012]. Dostupné na URL: <http://www.sparkfun.com/products/9588>

[15] Arduino [online]. [cit. 2. 5. 2011].
Dostupné na URL: <http://www.arduino.cc>

[16] Digi International [online]. [cit. 2. 5. 2011].
Dostupné na URL: <http://www.digi.com>

[17] Fotorezistory [pdf]. [cit. 5. 5. 2011].
Dostupné na URL: http://www.ges.cz/sheets/k/k0760.pdf

[18] Arduino sleep code [online]. [cit. 5. 5. 2011].
Dostupné na URL: <http://arduino.cc/playground/Learning/ArduinoSleepCode>

[19] Wiring [online]. [cit. 7. 5. 2011].
Dostupné na URL: <http://wiring.org.co/>

[20] Processing [online]. [cit. 7. 5. 2011].
Dostupné na URL: <http://processing.org/>

[21] Ruby programming language [online]. [cit. 8. 5. 2011]
Dosupné na URL: <http://www.ruby-lang.org/en/>

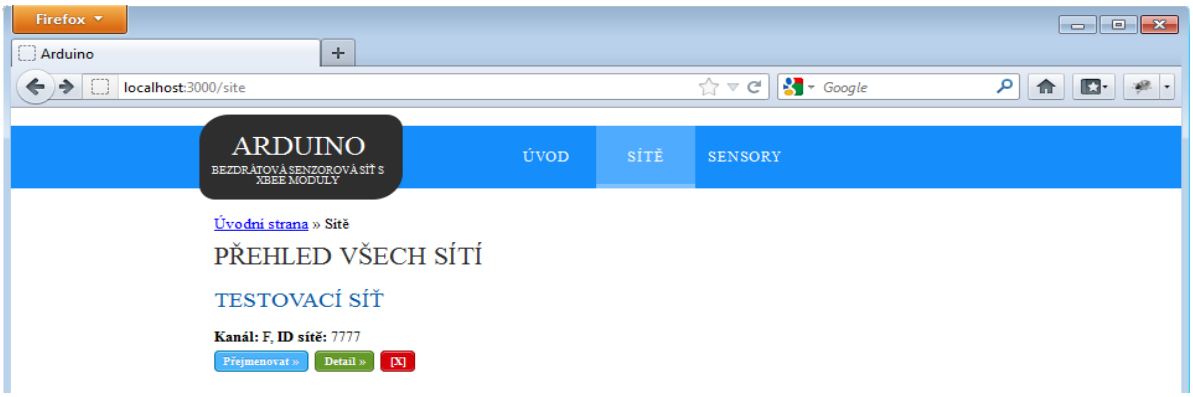
[22] Getting Started with Rails [online]. [cit. 8. 5. 2011]
Dosupné na URL: <http://guides.rubyonrails.org/getting_started.html>

Seznam použitých zkratek

ADC	Analog to Digital Converter
AES	Advanced Encryption Standard
AODV	Ad hoc On Demand Distance Vector
API	Application Programming Interface
APL	Application Layer
CTS	Clear To Send
COM	Component Object Model
EEPROM	Electrically Erasable Programmable Read Only Memory
EPROM	Erasable Programmable Read Only Memory
FFD	Full Function Device
FTDI	Future Technology Devices International
GCC	GNU Compiler Collection
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
LDR	Light Dependent Resistor
MAC	Media Access Control
MCU	Microprocessor Control Unit
MVC	Model View Controller
NWK	Network Layer
OSI	Open System Interconnection
PAN	Personal Area Network
RF	Radio-Frequency
RFD	Reduced Function Device
RISC	Reduced Instruction Set Computer
ROM	Read Only Memory
ROR	Ruby On Rails
RREQ	Route Request
RS 232	Recommended Standard 232
RTS	Request To Send
UART	Universal Asynchronous Reciever/Transmitter
USB	Universal Serial Bus
WSN	Wireless Sensor Network

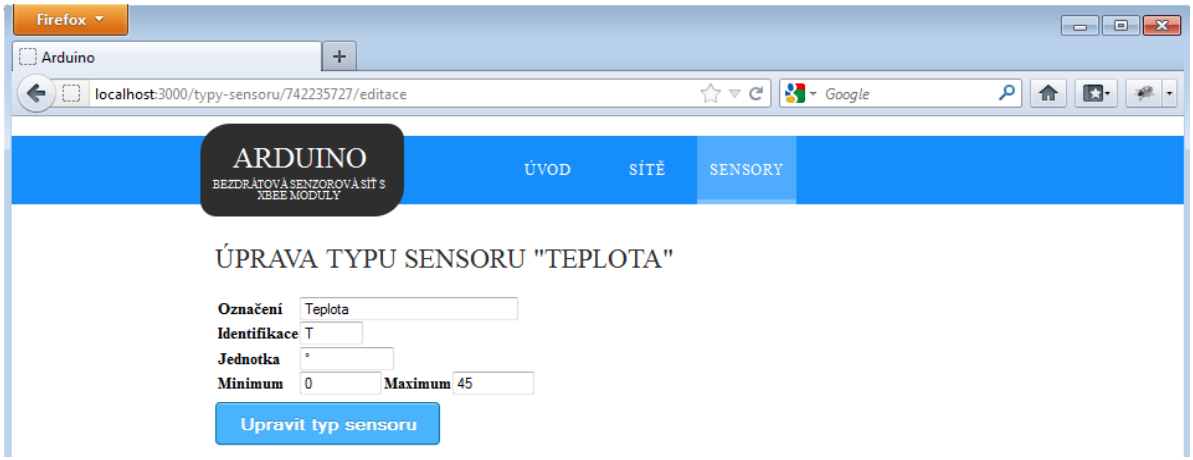
Seznam příloh

Příloha 1. Ukázka webového rozhraní



Náhled 1: Výběr sítě.

Náhled 2: Přehled měřených uzlů vybrané sítě.

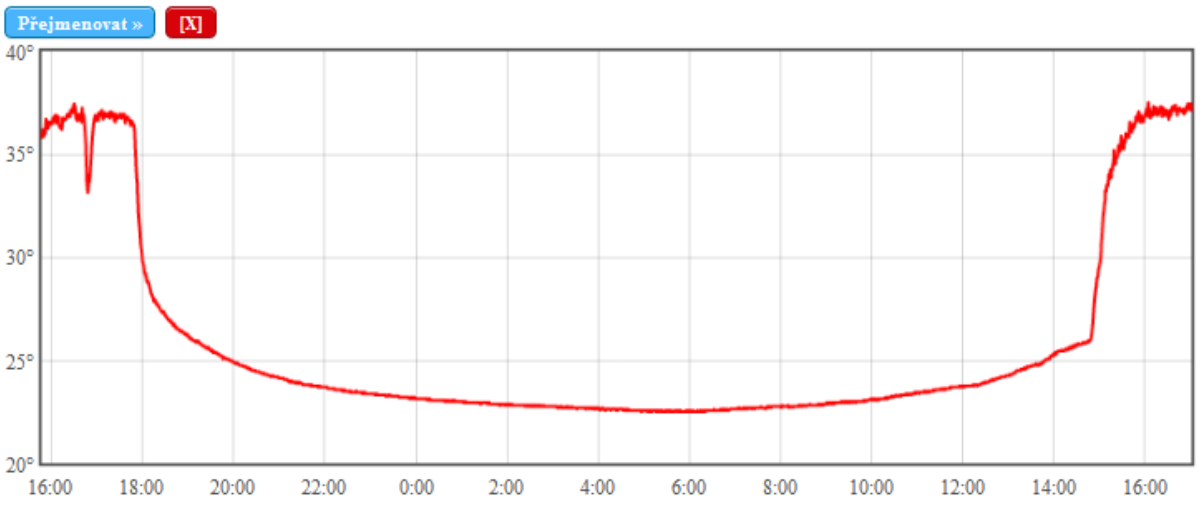


Náhled 3: Úprava nastevení teplotního senzoru.

Příloha 2. Ukázka grafů měření

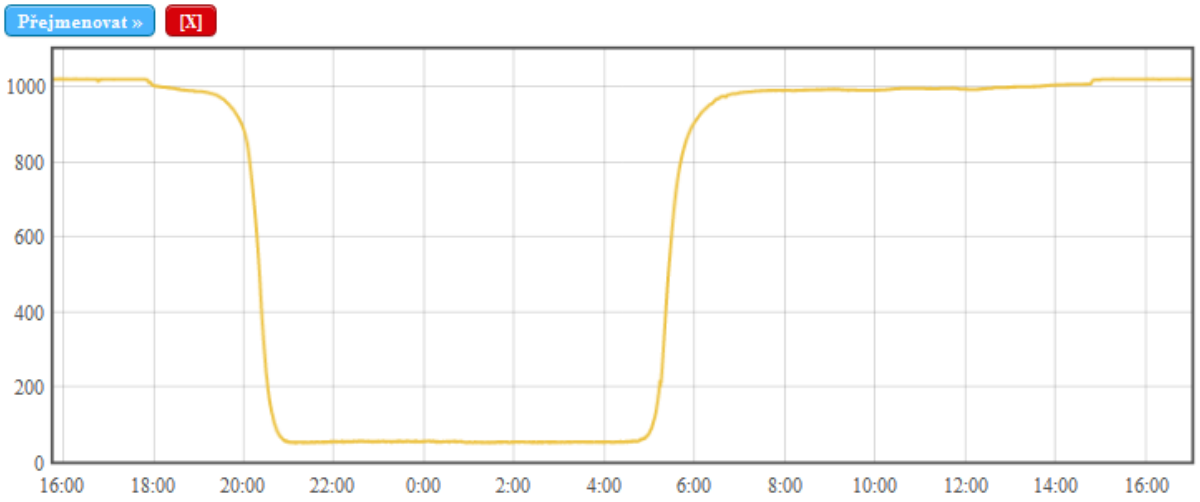
Denní vývoj teploty a osvětlení uzlu umístěného na okenním parapetu v kanceláři VUT FIT A207.

1) (T1) (TEPLOTA)



Aktualizace grafu: 4 sekundy, **Hodnoty:** Poslední den

2) (I1) (OSVĚTLENÍ)



Aktualizace grafu: 4 sekundy, **Hodnoty:** Poslední den

Příloha 3. Přehled důležitých AT příkazů

Příkazy byly převzaty a přeloženy z dokumentu [5], kde naleznete i všechny zbylé příkazy.

Speciální příkazy

AT příkaz	Název a popis	Rozsah	Standardně
WR	Zapsat (**Write**). Zapíše hodnoty všech parametrů do nevolatilní paměti tak, aby změny parametrů přetrvaly následné resetování zařízení.	-	-
RE	Obnovit výchozí nastavení **(Restore Defaults)**. Obnoví továrního nastavení parametrů modulu.	-	-
AC	Použít změny (**Apply Changes**). Ihned se použije aktuální nastavení bez ukončení příkazového módu.	-	-

Nastavení adresování

AT příkaz	Název a popis	Rozsah	Standardně
DH	Vrchní cílová adresa (**Destination Address High**). Vrchních 32 bitů z 64bitové cílové adresy.	0 – 0xFFFFFFFF	0
DL	Spodní cílová adresa (**Destination Address Low**). Spodních 32 bitů z 64 bitové cílové adresy.	0 – 0xFFFFFFFF	0x0000FFFF
SH	Vrchní sériové číslo (**Serial Number High**). Přečte vrchních 32 bitů sériového čísla.	0 – 0xFFFFFFFF	Z výroby
SL	Spodní sériové číslo (**Serial Number Low**). Přečte spodních 32 bitů sériového čísla.	0 – 0xFFFFFFFF	Z výroby
ID	ID sítě (**Network ID**). Nastaví/přečte identifikátor sítě.	0x0000 – 0x7FFF	0x7FFF
CH	Kanál (**Channel**). Nastaví/přečte aktuálně používaný kanál pro přenos dat.	0x0B – 0x1A (0x0C – 0x17 XBee-PRO)	0x0C
NH	ID sítě (**Network Hops**). Nastaví/přečte identifikátor sítě.	0x0000 až 0x7FFF	0x7FFF
CE	Koordinátor/Koncové zařízení (**Coordinator/End Device**). Nastaví/přečte směrovací nastavení uzlu. 0 – směrovač 2 – koncové zařízení	0, 2	0

Příkazy pro nastavení MAC vrstvy

AT příkaz	Název a popis	Rozsah	Standardně
RR	Počet pokusů doručení unicastu (**Unicast Mac Retries**). Nastaví/přečte počet pokusů o doručení unicast paketu. Pokud je RR nenulový, modul očekává potvrzovací paket o přijetí, jinak zasílá paket znovu až RR krát.	0 – 0xF	10
MT	Vysílání broadcastu (**Broadcast Multi-Transmit**). Nastaví/přečte aktuální počet přeposílání broadcastových zpráv.Všechny broadcastové zprávy jsou přeposlány MT + 1 krát.	0 – 0xF	3

Nastavení sériového rozhraní

AT příkaz	Název a popis	Rozsah	Standardně
AP	Mód API (**API Mode**). Nastaví/přečte aktuální API mód modulu. 0 – API mód vypnut, 1 – API mód zapnut, 2 – API mód v escapovací sekvenci	0, 1, 2	0
BD	Přenosová rychlost (**Baud Rate**). Nastaví/přečte přenosovou rychlost sériového rozhraní modulu.	0x39 – 0x1C9C38	0x03 (9600)
RO	Limit odeslání paketu **(Packetization Timeout**). Nastaví/přečte dobu od posledního přijatého znaku, po které se odešle paket.	0 až 0xFF	3

Nastavení příkazového módu

AT příkaz	Název a popis	Rozsah	Standardně
CT	Časový limit příkazového režimu (**Command Mode Timeout**). Nastaví/přečte dobu nečinnosti po které se modul přepne zpět z příkazového módu do základního.	2 – 0x1770	0x64
CN	Ukončit příkazový mód (**Exit Command Mode**). Modul okamžitě opustí příkazový mód.	-	-
CC	Příkazový znak (**Command Character**). Nastaví/přečte znak, který je nutný pro přechod modulu do příkazového módu.	0 – 0xFF	0x2B (+)
GT	Ochranná doba (**Guard Times**). Nastaví potřebnou dobu před a po obdržení příkazového znaku, pro přejití modulu do příkazového módu.	0 až 0xFFFF	0xE8

Nastavení spánkového režimu.

AT příkaz	Název a popis	Rozsah	Standardně
SM	Spánkový mód (**Sleep Mode**). Nastaví/přečte aktuální spánkový režim. 0 – Bez spánku, 1 – Pinem ovládaný spánek (pin SLEEP_RQ), 4 – Asynchronní cyklický spánek, 5 – Asynchronní cyklický spánek s probouzejícím pinem, 7 – Podpora spánku, 8 – Synchronní cyklický spánek.	0, 1, 4, 5, 7, 8	0
SO	Možnosti spánku (**Sleep Options**). Nastaví/přečte aktuální nastavení spánku. Pro synchronní: 0 – Preferovaný spánkový koordinátor 1 – Spánkový koordinátor bez uspávání 2 – Povolí API zprávy o stavu spánku 3 – Zakáže předčasné probuzení 4 – Povolí rovnost uzlů 5 – Zakáže opakování synchr. zpráv Pro asynchronní: 8 – Pokaždé probudit na dobu ST	Bity 0 až 8	0x02
ST	Doba provozu (**Wake Time**). Nastaví/přečte dobu provozu modulu po probuzení ze spánkového režimu.	0x45 – 0x36EE80	0x7D0 (2 sek.)
SP	Doba spánku (**Sleep Time**). Nastaví/přečte dobu periody spánku, po kterou bude modul spát.	1 – 1440000 (x 10 ms)	2 sek.
MS	Počet zmeškaných synchronizačních zpráv (**Number of Missed Syncs**).	-	-

Příloha 4. DVD se zdrojovými soubory.

Tato příloha obsahuje bakalářskou práci v elektronické podobě včetně všech použitých obrázků, soubory obsahující zdrojové kódy klientského programu, webové části, firmware koncových uzlů a základního uzlu.

Adresářová struktura DVD

- `/BP/bakalarska_prace.pdf`
  Text bakalářské práce ve formátu PDF.
- `/BP/Obrazky/`
  Všechny zdrojové obrázky uvedené v bakalářské práci.
- `/Klient/`
  Adresář obsahující zdrojové kódy implementace klientského programu.
- `/Server/`
  Adresář obsahující zdrojové kódy implementace serverové části.
- `/Arduino/Uno/`
  Adresář obsahující zdrojové kódy pro naprogramování základního uzlu (Arduino Uno).
- `/Arduino/Fio/`
  Adresář obsahující zdrojové kódy pro naprogramování koncových uzlů (Arduino Fio).