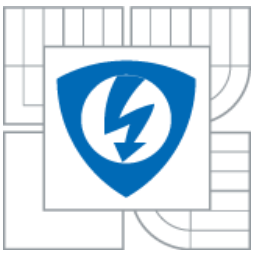
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

NÁVRH LABORATORNÍCH ÚLOH DO PŘEDMĚTU SENZOROVÉ SYSTÉMY

LABORATORY EXERCISES CREATION FOR COURSE SENSOR SYSTEMS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — VÍTĚZSLAV ŠTĚTINA
AUTHOR

VEDOUCÍ PRÁCE — ING. MILAN ŠIMEK
SUPERVISOR

BRNO 2010

Anotace

Bakalářská práce na téma Návrh laboratorních úloh do předmětu Senzorové systémy popisuje teoretickou část a praktické řešení zaměřené na vytvoření 5 laboratorních úloh. K laboratorním měření je použit programovací jazyk nesC, operační systémem TinyOS a další programy. V 5 kapitolách jsou popsány samotné návody laboratorních úloh. Ty jsou koncipovány od základní úlohy až po složitější zadání. V pracovních postupech je vždy zadání úlohy, seznam potřebného vybavení, postup pří vytváření hlavní části programu a následná instalace do uzlu nebo do základnové stanice. Součástí úloh je rozbor zdrojového kódu a popsání nových příkazů oproti předešlým úlohám. A part of the task is to analyze the source code and desription of new orders compared to previous exercises.

Abstract

Topic of bachelor thesis is Laboratory exercises creation for course Sensor systems and describes the theoretical and practical solutions focused on the creation of 5 laboratory experimets. To the laboratory exercises is used a programming language nesC, operating system TinyOS and other programs. In 5 chapters describe the actual instructions for laboratory exercises. They are composed of the basic tasks to complex assignment. The workflow is a task assignment, a list of necessary equipment, the procedure for creating the main part of program and subsequent installation to the node or base station.

Klíčová slova

jazyk nesC, operační systém TinyOS, senzor, síť, uzel, základnová stanice

Keywords

programming language nesC, operating system TinyOS, sensor, network, node, base station

Bibliografická citace práce

ŠTĚTINA, V. *Návrh laboratorních úloh do předmětu Senzorové systémy*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 40 s. Vedoucí bakalářské práce Ing. Milan Šimek.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma Návrh laboratorních úloh do předmětu Senzorové systémy jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

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

V Brně dne ....

(podpis autora)

OBSAH

Seznam obrázků

Seznam tabulek

Úvod

Cílem bakalářské práce, jak už z jejího názvu vyplývá, je návrh laboratorních úloh do předmětu Senzorové systémy. K dispozici v laboratoři je veškeré technické vybavení pro návrh a realizaci cvičení od firmy Crossbow. Hardwarová část obsahuje uzly IRIS XM2110, brány MIB520, senzorové desky MDA100 a MTS420 a kabely pro propojení základnové stanice s počítačem. Využívaný operační systém TinyOS, popsaný v kapitole 1 běží na bázi Unixu, proto je nezbytné v operačním systému Windows použít emulaci programem Cygwin. Pro tvorbu aplikací je použit jazyk nesC a Programmer's Notepad pro jejich vytváření, které se přes něj i do uzlů nahrávají. Komunikace s počítačem probíhá přes USB rozhraní emulované na sériové UART. Jelikož se porty přidělují podle obsazenosti, nejsou vždy stejné a student si je musí zjistit v nastavení, to je ovšem detailně popsáno v návodech. Pro rozbor paketových zpráv slouží program XServe, běžící v emulaci Cygwin a programy MoteView a XSniffer.

Práce je rozdělena do devíti kapitol. Úvodní 4 kapitoly popisují operační systém TinyOS, programovací jazyk nesC, technologii XMesh a potřebný hardware k práci v laboratořích. Následujících 5 kapitol obsahuje jednotlivé laboratorní návody včetně zadání, rozboru kódu a samostatné práce studentů. Úlohy jsou koncipovány od první základní, pro seznámení se s problematikou, až po přídavné funkce systému XMesh a práce s GPS modulem.

1 Operační systém TinyOS

1.1 *Využití v praxi*

Aplikace jsou psány v nesC, což je optimalizovaný programovací jazyk C určený pro paměti v senzorových sítích. Jako doplňky používá Java a „shell script“. Přidružené knihovny a nástroje, NesC kompilátor a Atmel AVR jsou většinou napsané v jazyce C.(1)

TinyOS pracuje asynchronně. Proto všechny I/O operace trvající déle než několik mikrosekund jsou neblokující a mají zpětné volání. Pokud je vyžadován nativní kompilátor, TinyOS využívá přídavné funkce nesC k propojení těchto zpětných volání nazývajících se události. I když neblokování umožňuje TinyOS udržet vysokou souběžnost se samotným zásobníkem, nutí programátory psát složité kódy spojené z mnoha malých událostí. Pro podporu lepších výpočtů stanoví operační systém úkoly, které jsou podobné odloženým voláním procedur a přeruší obsluhu spodní poloviny. Komponenta může poslat žádost o pozdějším vykonání operace. Žádosti jsou nepreventivní a spouští se v pořadí FIFO. Tento souběžný model obvykle dostačuje pro I/O orientované aplikace, ale jeho potíže s vytížením CPU, měly za důsledek začlenění linie výpočtu do OS. (2) (3)

Operační systém podporuje mikroprocesory od 8bitové architektury s pamětí 2 KB až po 32bitové procesory s operační pamětí 32 MB a více. Poskytuje také předdefinovanou sbírku funkcí a procedur (API), které slouží k výpočtu možností senzorů v rámci sítě. Díky těmto vlastnostem se data částečně zpracují dříve, než se odesílají do uzlu. Tím se optimalizuje propustnost sítě a zbytečné odesílání a přijímání zpráv, což šetří výdrž akumulátorů.(1)

2 Jazyk nesC

(Vyslovováno „NES-see”) je rozšíření programovacího jazyka C tak, aby začlenil koncepci strukturování a realizaci modelu operačního systému TinyOS.

2.1 *Základní pojmy*

2.1.1 **Oddělení konstrukce a kompozice.**

Programy jsou postaveny z komponent, které jsou sestaveny („drátově“) k tvorbě celých programů. Programy mají vnitřní souběžnost ve formě úkolů. Linii výpočtů kontroly můžeme dostat pomocí komponent přes jeho rozhraní. Tato osnova se řídí buď úkolem, nebo hardwarovým přerušením. (3)

2.1.2 **Specifikace chování komponent v sadě rozhraní.**

Rozhraní mohou být poskytnuta nebo použita komponentou. Využívání rozhraní je určeno k popsání funkčnosti, která z komponent bude sloužit uživateli. Použité rozhraní určuje komponenty potřebné k výkonu dané operace. (4)

2.1.3 **Rozhraní jsou obousměrná.**

Představují sadu funkcí, které mají být implementovány rozhraním poskytovatele (příkazy) a nastavení, která mají být provedená uživatelem (úkoly). To umožňuje jednotné rozhraní představující složitou interakci mezi komponentami (např. registrace zájmu o některou z událostí, následné zpětné volání, kdy se má daná událost vykonat). To je důležité proto, že všechny dlouhé příkazy v *TinyOS* (např. poslání paketu) jsou neblokující, jejich dokončení je oznámeno skrz událost (odesláno). Určením rozhraní nemůže komponenta poslat příkaz, jestliže nejde implementovat událost *„sendDone“*. Typické příkazy pro sestupné volání, tedy od aplikační komponenty blíže k hardwaru, zatímco události využívají volání směrem nahoru. Některé základní události jsou vázány k přerušení hardwaru. (3)(4)

Díky statickému propojení komponent přes jejich rozhraní se zvyšuje provozní efektivita a zlepšuje statická analýza programů.(3)

3 XMesh

3.1 Úvod

Bezdrátové sítě ve standardu IEEE 802.15 s menším rozsahem byly vyvinuty pro efektivní použití zařízení bez počítačů. Samostatně pracující síťová architektura umožňuje využít bezdrátové aplikace mezi dvěma zařízeními, včetně detekce pohybu senzorů v měřeném rozsahu. Sítě mohou být navrženy různě, v závislosti na konkrétních prioritách. Všechny bezdrátové systémy mají společné požadavky, mezi které patří: (4)(5)

- **Nízká spotřeba energie** – podpora dlouhodobého provozu je zajištěna tak, že se spotřeba energie rádiového spojení musí minimalizovat kompaktním lehkým akumulátorem s dlouhou výdrží.
- **Snadné použití** – síťový protokol dovoluje senzorové síti vlastní inicializaci v rámci způsobu vlastní samosprávy.
- **Rozšiřitelnost** – síť musí být schopna podporovat určitý počet uzlů požadovaných okamžitě a také budoucí rozšíření bez exponenciálního nárůstu přetížení.
- **Reakční doba** – topologie musí být v rozpoznávání nově přihlášených zařízení efektivní, zejména u aplikací, kde se uzly pohybují.
- **Rozsah** – více energeticky náročné je vysílat rádiový signál s větší silou na větší vzdálenost, než vysílat na kratší vzdálenosti v kratším čase. Síť je tvořena opakovači používající protokol pro podporu multi-skokového směrování, Jestliže je uzel daleko od základnové stanice, pošle pakety přes jiný.
- **Obousměrná komunikace mezi základnovou stanicí a senzorem** – aby mohla základnová stanice upravovat některé provozní parametry během provozu, musí být komunikace obousměrná.
- **Spolehlivost –** při monitorování například v lékařství je požadována kritická spolehlivost údajů.

Pro splnění výše uvedených požadavků se využívá pevný síťový protokol. Ten poskytuje podporu topologie a řídí směrování dat skrz síť. (5)

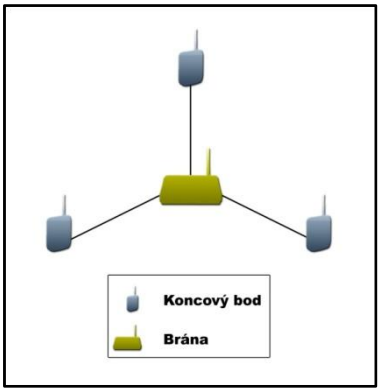
3.2 *Topologie*

K provedení aplikace bezdrátové sítě včetně hvězdy, smyčky a smíšené dohromady existuje několik řešení. Každá topologie představuje vlastní problémy, výhody a nevýhody. K lepšímu porozumění patří následující komponenty:

- **Koncové body** – sjednocení senzorů a akčních členů k zachycení dat ze snímačů, které jsou poskytovány uživateli. Při směrování mají specifickou adresu.
- **Směrovače** - jsou aktivní síťová zařízení pracující na síťové vrstvě, sloužící k přeposílání paketů směrem k určenému cíli. Umožňují také rozšíření oblasti sítě a zajištění náhradní trasy v případě přetížení nebo selhání zařízení. V některých případech můžou směrovače pracovat rovněž jako koncové body.
- **Brány** – jsou uzly, spojující dvě sítě s různými protokoly. Seskupují data, rozhraní hostitele, LAN nebo Internetu a pracují jako portál pro sledování výkonu a konfiguraci síťových parametrů.
- **Systémový software** – poskytuje síťový protokol umožňující samostatné řízení a sestavování ad hoc sítí. (4) (5)

3.2.1 Topologie hvězdy

Je jedno-skokový systém, kde jsou všechny uzly v přímém dosahu k bráně. Obvykle ve vzdálenosti 30 – 100 metrů. Všechny uzly jsou identické a zároveň jsou i koncovými body. Brány slouží pro datovou komunikaci a pro příkazy do koncových bodů (uzlů). Používá se také pro přenos dat na vyšší úrovni řízení nebo pro monitorovací systém. Koncové body využívající stejnou bránu spolu nekomunikují, jak lze vidět na Obrázek 1.(5)

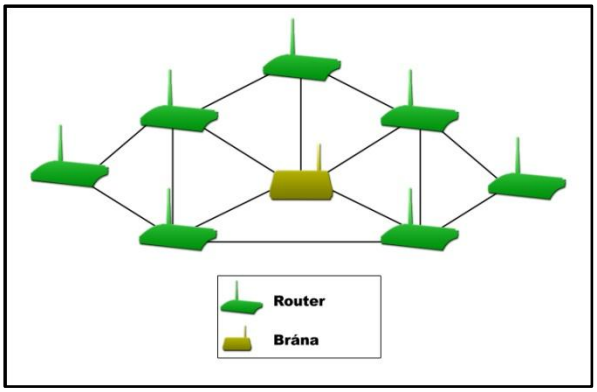


Obrázek 1 - Topologie hvězdy

Topologie hvězdy přináší nejnižší spotřebu energie, je ale omezena přenosovou vzdáleností rádiového signálu mezi koncovým bodem a bránou. Neexistují také žádné náhradní komunikační cesty do koncových bodů v případě poškození.(5)

3.2.2 Topologie smyčky

Multi-skokový systém obsahuje shodné zařízení plnící funkci směrovačů. Uzly komunikují se sousedními směrovači a posílají přes ně data do společné brány. Jedná se o standardní nastavení *XMesh*. Oproti topologii hvězdy se může zpráva poslat přes jiný uzel plnící funkci směrovače.

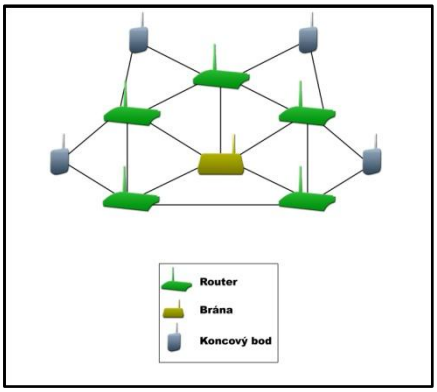


Obrázek 2 - Topologie smyčky

Díky tomu se může síť rozšiřovat, teoreticky až do nekonečna. Smyčková topologie je také velice odolná proti chybám, protože každý koncový bod, má několik možných datových cest k bráně. Pokud senzor selže, síť se sama rekonfiguruje. V závislosti na počtu uzlů a jejich vzdálenostem se může zvýšit latence posílání dat z uzlu do uzlu po cestě do brány. (5)

3.2.3 Topologie hvězda-smyčka

Kombinací dvou předchozích topologií lze dosáhnout nízkou energetickou náročnost s hvězdicovou jednoduchostí spolu s rozšiřitelným rozsahem hvězdicové topologie. Senzory se rozmístí kolem směrovačů, které se podle pořadí sdružují do sítě. Bezdrátové uzly mohou komunikovat s více směrovači a poskytují tak možnost růstu sítě a ochranu proti částečnému selhání nebo přetížení sítě. (5)



Obrázek 3 - Topologie hvězda-smyčka

3.3 *Technologie XMesh*

Je plně vybavený multi-skokový, ad-hoc a smyčkový protokol vyvinutý firmou *Crossbow* pro bezdrátové sítě. Síť *XMesh* obsahuje vzájemně bezdrátově komunikující uzly, které posílají zprávu do základnové stanice, a ta komunikuje s počítačem. Tato komunikace efektivně rozšiřuje rozsah sítě a snižuje energii potřebnou k přenosu zprávy. Další dvě výhody této komunikace jsou spolehlivost a lepší pokrytí. Dva uzly totiž nemusí být v přímém dosahu, aby mohly komunikovat, zpráva může být poslána přes směrovač. V případě špatné

komunikace mezi uzly se může zpráva poslat po jiné trase do stejného cíle přesměrováním. Obvykle se tak děje při dlouhodobějším zapnutí uzlu, který běží v úsporném režimu. (4) (5)

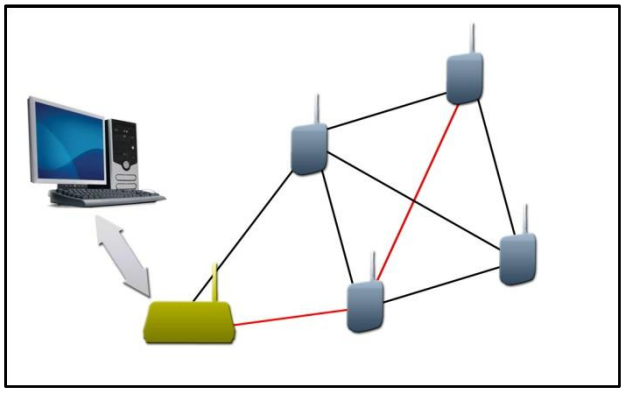
XMesh je softwarová knihovna využívající operační systém *TinyOS* běžící v zařízeních. Tyto zařízení se nazývají uzly a obsahují:

- Mikroprocesor – pro uzel *IRIS Atmel ATmega* 1281 s pamětí 128KB
- Rádiová komunikace – v pásmu 2,4 GHz
- Sériová paměť – externí paměť pro ukládání OTAP (ove-the-air-programming) a pro datový zápis

3.3.1 Síť *XMesh*

Kompletní síť *XMesh* obsahuje tyto části:

- Jeden nebo více uzlů pracující v síti – *IRIS XM2110*
- Základnová stanice - MIB520 je další uzel připojený k počítači a naprogramován programem *XMeshBase*. Slouží pro správu sítě a s její komunikaci
- Počítač s potřebnými programy



Obrázek 4 - Schéma systému XMesh

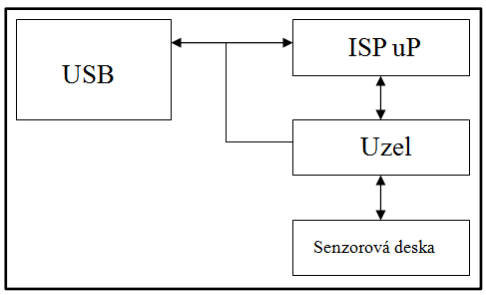
4 Potřebný hardware

Kapitola popisuje potřebný hardware potřebný pro práci v laboratořích. Nezabývá se však samotným vybavením učebny, tj. počítačem a kabely.

4.1 *MIB520*

Obrázek 5 - Programovací jednotka MIB520

MIB520 je propojovací brána pro bezdrátové senzorové sítě a slouží pro programování uzlů *IRIS*, *MICAz* a *MICA2* pomocí USB portu, pomocí kterého se i napájí. Kromě přenosu dat MIB520CB poskytuje také programové rozhraní USB rychlostí 57,6 baudů (57,6 Kb/s) a nabízí dva samostatné porty. Jeden je vyhrazený pro programování uzlů a druhý pro datovou komunikaci přes USB. MIB520 má na desce umístěný procesor, který je možno programovat pomocí programu *Mote Processor Radio Boards.* Připojení přes USB umožňuje i napájení, proto není potřebný externí zdroj. Dále obsahuje 51pinový konektor a 3 LED diody pro indikaci stavů.(6)



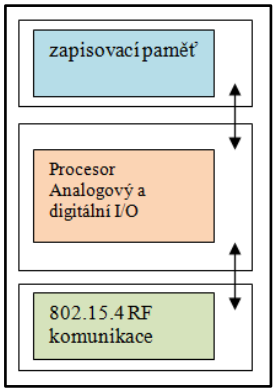
Obrázek 6 - Blokové schéma MIB520

4.2 *Uzel IRIS*

Obrázek 7 - Uzel IRIS XM2110

Nízko-výkonový uzel pracující v pásmu 2,4 GHz standardu IEEE 802.15 s přenosovou rychlostí až 250 Kb/s. Lze rozšířit na měření světelnosti, teploty, barometrického tlaku, polohy a jiných veličin, podle přídavné senzorové desky. Použití v praxi může být v budovách k monitorování a bezpečnosti. Paměť RAM je 8 Kb, kapacita 512 Kb a napájení 2xAA akumulátorem. (7)

Uzel *XM2110* je osazen procesorem *Atmel ATmega1281*, který spouští aplikace ze své vnitřní paměti. Jednoprocesorová deska umožňuje zpracování aplikace a zároveň bezdrátovou komunikaci. 51pinový konektor podporuje analogový vstup, digitální I/O, I2C, SPI a UART rozhraní. (7)

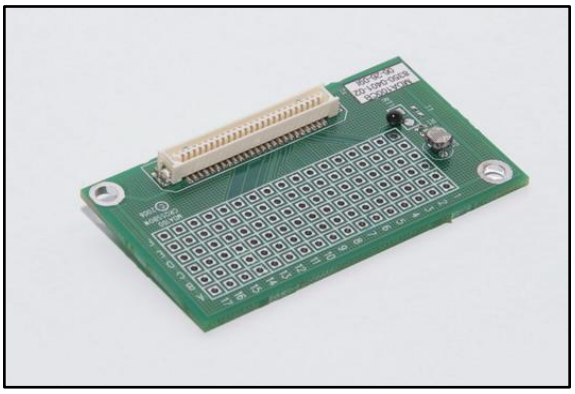


Obrázek 8 - Blokové schéma XM2110CA

4.3 *Senzorové desky*

4.3.1 MDA100CB

Snímač obsahuje světelné čidlo, termistor, fotobuňky a oblast určenou pro připojení dalších zařízení pomocí 42 pájecích bodů. Připojení k uzlu *IRIS* je zrealizováno 51pinovým konektorem. (8)



Obrázek 9 - Senzorová deska MDA100CB

4.3.2 MTS420

Tato senzorová deska umožňuje měření 5 základních parametrů z environmentálního prostředí a pomocí přídavného modulu také GPS souřadnice. Využívají nejnovější generaci, povrchově montovaných senzorů. Tyto energeticky efektivní digitální jednotky poskytují menší spotřebu a nízkou náročnost na údržbu. Pro uživatele poskytuje 64K paměti EEPROM.(8)

Jednotlivé veličiny a jejich specifikace:

- Teplota v rozsahu -10°C až 60°C
- Vlhkost od 0% až do 90% relativní vlhkosti
- Akcelerometr – analogová jednotka ADXL202JE poskytující měření dvou os
- Snímač barometrického tlaku v rozsahu od 300 do 1100 mbar s přesností ±1,5% na 25°C
- Senzor okolního světla s citlivostí 400-1000 nm
- GPS modul – 16 kanálů, pozice s přesností 10 metrů

5 Laboratorní úloha č. 1 - Rozblikání LED diody komunikačního uzlu

5.1 *Cíl úlohy*

Cílem této úlohy je pochopení základních principů při práci s *časovači ATmega 128L*, komunikačních jednotek *IRIS*, programovací brány *MIB520* a programu *Programmers Notepad 2 (PN 2)*. Výstup této úlohy je rozblikání LED diody v intervalu 1 vteřiny.

5.2 *Potřebný hardware*

- 1x programovací jednotka (brána) *MIB520*
- 1x uzel *IRIS XM2110*

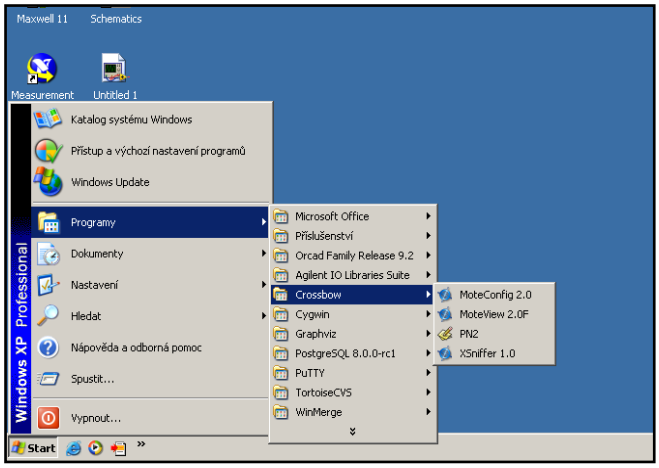
5.3 *Zadání*

- Vytvořit všechny potřebné soubory ke kompilaci
- Připojit potřebný HW
- Zkompilovat a nainstalovat program do uzlu a tím rozblikat LED diodu
- Na konci cvičení splnit samostatný úkol

5.4 *Pracovní postup*

a) V programu Total Commander vytvořte adresář s vaším jménem **C:\Crossbow\cygwin\opt\MoteWorks\apps\tutorials\MSSY\lab1\jmeno.**

b) Ve složce start – programy – Crossbow naleznete potřebné programy pro práci. Spusťte program *PN2.*



Obrázek 10 – Printscreen umístění programů

c) V pravém okně *Projects* otevřete složku s vaším jménem. Tato složka bude na začátku obsahovat 4 soubory (***Makefile, Makefile Components, MyApp.nc a MyAppM.nc***). Při každé změně musíte soubory obnovit stisknutím pravého tlačítka na danou složku a příkazem refresh.

d) Vytvořte nový soubor, vepište do něj následující příkazy a uložte pomocí CTRL+S

```
include Makefile.component
include $ (TOSROOT)/apps/MakeXbowlocal
include $ (MAKERULES)
```

e) Podobně vytvořte další soubor a vepište následující text a opět uložte.

```
COMPONENT=MyApp
SENSORBOARD=mts310
```

Detaily kódu: Tento soubor popisuje nejvyšší úroveň komponent aplikace, v našem případě komponenta *MyApp* a typ senzorové desky - *mts310*.

Uložte soubor CTRL+S.

f) V novém souboru, který bude uložen jako ***MyApp.nc*** je zapotřebí definovat následující strukturu komponent a závislostí.

Aplikace *MyApp* implementuje celkem 4 komponenty: *Main, MyAppM, TimerC, LedsC.* Komponenta *Main* se do aplikací vkládá pokaždé. Funguje na principu časovače a spouští úkoly definované ostatními komponenty. Spouští se

ve všech TinyOS aplikacích vždy jako první. V první aplikaci komponenta *Main* využívá rozhraní *StdControl* komponent *TimerC* a *MyAppM* (značí se pomocí symbolu ->):

```
MyApp_TimerM.StdControl -> MyAppM.StdControl
```

Rozhraní *StdControl* je běžně používané rozhraní ke spuštění a zastavení *TinyOS* komponent. Pokud se spustí funkce *Main.StdControl.start (),* díky této vazbě budou spuštěny i funkce *MyAppM.StdControl.start() a Timer.StdControl.start().*

Aplikační modul (což je i komponenta) *MyAppM* definuje chování navrhované aplikace a využívá rozhraní *Timer* komponenty TimerC a *Leds* komponenty LedsC. Modul *MyAppM* řídí časovač a LED diody voláním funkcí komponent *TimerC* a *LedsC.* Klíčové slovo *unique* definuje jedinečnost použití komponenty. Každé rozhraní může být implementováno několikrát, ale s různým identifikátorem. Příklad vytvoření dvou časovačů s různými identifikátory:

```
MyApp_TimerM.cervLed -> TimerC.Timer[unique("Timer")]
MyApp_TimerM.zelLed -> TimerC.Timer[unique("Timer")]
```

V případě zápisu posledního řádku kódu, modul *MyAppM* využívá rozhraní *Leds* komponenty LedsC. Jméno využívaného rozhraní se nemusí definovat, lze ho pouze zdědit, je-li stejné na levé i pravé straně. Zápis:

```
MyAppM.Leds->LedsC.Leds
```

Může být zkrácen:

```
MyAppM.Leds->LedsC
```

V níže uvedeném kódu je vidět použití časovače a komponty LED diod. Po ukončení uložte CTRL+S.

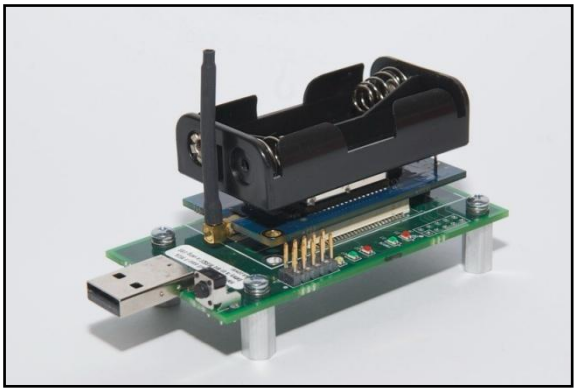
```
configuration MyApp {
}
implementation {
     components Main, MyAppM, TimerC, LedsC;

     Main.StdControl -> TimerC.StdControl;
     Main.StdControl -> MyAppM.StdControl;
     MyAppM.Timer                              ->
TimerC.Timer[unique("Timer")];
     MyAppM.Leds -> LedsC.Leds;
}
```

g) Vytvoření aplikačního modulu **MyAppM.nc**, který definuje vlastní chování aplikace.

Nejprve je třeba definovat jméno modulu a jaká rozhraní používá a která sdílí (klíčová slova *provides* a *uses*). V tomto případě sdílí rozhraní *StdControl* a používá *Timer* a *LedsC* rozhraní. Rohranní *StdControl* je potřebné pro inicializaci (*init()*) a spuštění (*start ()*) všech komponent modulu MyAppM. Příkaz *StdControl.init()* inicializuje LED diody spuštěním příkazu *Leds.init().* Rozhraní *TimerC* definuje příkazy *start()* a *stop()* a jednu událost *fired(),* která je volána při vypršení časovače.

5.4.1 Instalace aplikace *MyApp* do komunikační jednotky *IRIS*

a) Připojte programovací jednotku *MIB520* k počítači pomocí USB kabelu a poté ve správci hardwaru zjistěte, na kterých USB portech pracuje. (Ovládací panely – Systém – Záložka Hardware – Správce zařízení – Porty COM a LPT). Porty budou dva a to port COM X (slouží pro programování) a port COM X +1 (slouží pro komunikaci).
b) Nyní připojte uzel *IRIS* (pozor bez akumulátorů) k programovací jednotce, viz Obrázek 11. Uzel může být vypnutý, napájí se z brány.

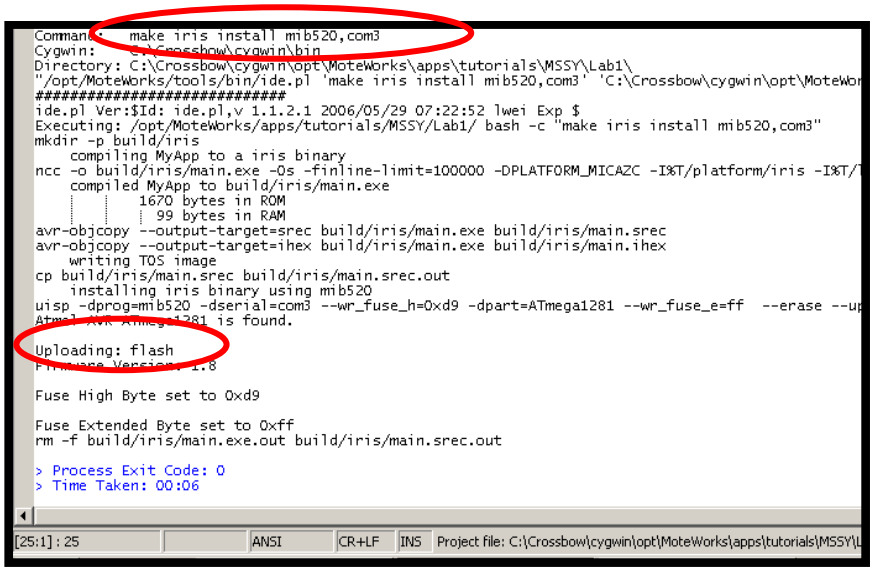


Obrázek 11 - Připojení komunikační jednotky k programovací bráně

c) Zkompilujte program a nahrajte do mikrokontroleru pomocí následujících kroků. Přepněte se do souboru *MyApp.nc*. Stiskněte F6 (nebo *Tools – Shell*) a pomocí následujícího příkazu zkompilujte a nainstalujte aplikaci do komunikační jednotky:

```
make iris install mib520,comX
```

d) Po úspěšném dokončení se na posledním řádku výstupní části objeví text: ***Uploading: flash,*** viz Obrázek 12. Jestliže ne, pravděpodobně jste udělali v jednom z vašich souborů chybu. Před nahráním aplikace se ujistěte, že kompilace proběhla v pořádku.



Obrázek 12 – Printscreen po instalaci

e) Nyní můžete odpojit uzel *IRIS* z programovací brány, vložit 2 AA akumulátor a zapnout postranní vypínač. Červená dioda by měla blikat v intervalu 1 vteřina.

5.4.2 Samostatný úkol

a) Rozblikejte červenou diodu v intervalu 2 vteřiny a zelenou diodu v intervalu 4 vteřiny.

b) Vytvořte grafickou reprezentaci vztahů jednotlivých komponent. Na souboru *MyApp.nc* spustě *shell* a zadejte:

```
make iris docs
```

V adresáři */MoteWorks/doc/nesdoc/iris.docs/nesdoc/iris/* se vygeneruje soubor *index.html*

c) Prostudujte grafickou reprezentaci aplikace a její jednotlivé vazby.

6 Laboratorní úloha č. 2 - Bezdrátový přenos dat do základnové stanice

6.1 *Cíl úlohy*

Cílem této úlohy je vytvoření aplikace, která navzorkuje data získaná ze světelného senzoru *MDA100CB* připojeného na uzlu *IRIS XM2110* v intervalu jedna vteřina, uloží je do zprávy a pošle tuto zprávu bezdrátovým prostředím do základnové stanice, která tato data přijme a zobrazí. Snímaná data budou vizualizovaná pomocí aplikace *XSniffer*.

6.2 *Potřebný hardware*

- 1x programovací jednotka (brána) *MIB520*
- 2x uzel *IRIS XM2110*
- 1x senzorová deska MDA100CB

6.3 *Zadání*

- Pomocí sezorové desky *MDA100CB* snímejte světlo.
- Pošlete data do základnové stanice pomocí sériového a bezdrátového rozhraní.
- Rozblikejte žlutou LED diodu jako indikaci vzorkování senzoru.
- Rozblikejte zelenou LED diodu jako indikaci odeslání dat do základnové stanice.

6.4 *Pracovní postup*

a) V programu Total Commander vytvořte adresář s vaším jménem **C:\Crossbow\cygwin\opt\MoteWorks\apps\tutorials\MSSY\lab2\jmeno.**

b) Stáhněte všechny patřičné soubory a uložte do vašeho adresáře. Všimněte si, že chybí hlavní soubor MyApp.nc, který je třeba vytvořit.

6.4.1 Vytvoření aplikace *MyApp.nc*

Otevřete soubor *MyApp_Sensing_nc.html* a důkladně prostudujte vazby mezi jednotlivými komponentami. Pomocí tohoto grafického vyjádření vytvořte soubor *MyApp.nc*. Detailně prostudujte seznam rozhraní, které modul *MyAppM.nc* používá. Důležitá jsou vyjádření, jak jsou jednotlivá rozhraní v modulu definována, abyste mohli tato rozhraní definovat v aplikaci *MyApp.nc*.

V soubor *MyApp.nc* přibyly oproti předešlé úloze dvě nové komponenty a to *Photo* a *GenericComm*. Komponenta *Photo* se používá ke komunikaci se světelným senzorem a komponenta *GenericComm* slouží bezdrátového posílání zpráv mezi základnovou stanicí a uzlem.

Všechny komponenty v souboru tedy jsou: *Main, MyAppM, TimerC, LedC, Photo, GenericComm* (jako *Comm*). Po dokončení uložte.

6.4.2 Instalace aplikace *MyApp* do komunikační jednotky *IRIS*

a) Připojte programovací jednotku MIB520 k počítači pomocí USB kabelu a zjistěte, na kterých USB portech pracuje. (Ovládací panely – Systém – Záložka Hardware – Správce zařízení – Porty COM a LPT).
b) Nyní připojte uzel *IRIS* bez akumulátorů k programovací jednotce, viz Obrázek 13. Uzel může být vypnutý, napájí se z brány.
c) Zkompilujte program a nahrajte jej do mikrokontroleru pomocí následujících kroků. Přepněte se do nově vytvořeného souboru ***MyApp.nc***. Stiskněte F6 (nebo *Tools – Shell*) a pomocí následujícího příkazu zkompilujte a nainstalujte aplikaci do komunikační jednotky:

```
make iris install, mib520,comX
```

d) Po úspěšném dokončení se na posledním řádku výstupní části objeví text: ***Uploading: flash.***
e) Nyní odpojte uzel *IRIS* z programovací brány, připojte senzorovou desku *MDA100CB*, vložte 2ks AA akumulátoru a zapněte postranní vypínač (viz

Obrázek 13). Po úspěšném nahrání by se měly diody rozblikat v intervalu 1 vteřina. Jednotlivé barvy udávají různé indikace, viz Tabulka 1 - Indikace LED diod níže.

Obrázek 13 - Uzel IRIS XM2110 s připojenou senzorovou deskou MDA100CB

Tabulka 1 - Indikace LED diod

LED dioda	Indikace
Červená	Interval 1 vteřina pro vzorkování
Žlutá	Vzorkování světelného senzoru
Zelená	Odeslání dat do základnové stanice

f) Připojte jiný uzel, opět bez akumulátoru, k programovací jednotce a v *PN 2* načtěte soubor ***TOSBase.nc*** umístěný v **/apps/general/XSniffer.** Stiskněte F6 (nebo *Tools – Shell*) a napište následující příkaz:

```
make iris install,2 mib520,comX
```

g) Po úspěšném dokončení nechte uzel připojený a postupujte pomocí následujících kroků.

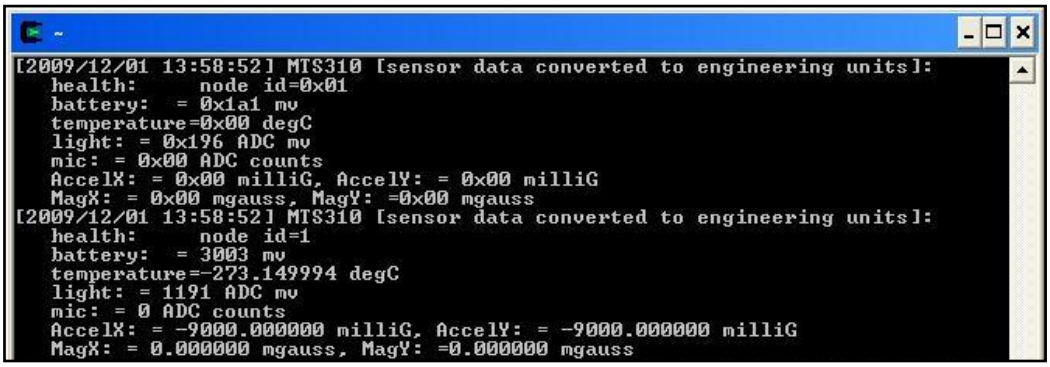
6.4.3 Rozbor senzorové zprávy pomocí XServe

K zobrazení senzorové zprávy, která přichází do PC přes sériový port použijte nástroj *XServe*, který je součástí *MoteWorks* a běží společně v programu *CYGWIN*.

a) Na pracovní ploše spusťte program *CYGWIN* a napište příkaz:

```
xserve -device=COMX
```

Jestliže je brána správně připojena a vybrán správný komunikační port, zobrazí se vám podobná zpráva jako na Obrázek 14.



Obrázek 14 - Cygwin a běžící XSereve

Všimněte si, že zpráva obsahuje datové pole pro ostatní senzory, které nejsou používány a jsou nastaveny na hodnotu 0. Níže jsou vidět hodnoty u světelného senzoru z vašeho uzlu. Z dalších hodnot jsou to například ID uzlu, stav akumulátoru a teplota.

6.4.4 Broadcastové odeslání senzorové zprávy

K bezdrátovému odesílání zpráv z jednoho uzlu do druhého, namísto odesílání zpráv přes sériový port (*UART*) se docílí změnou aplikace *MyAppM.nc*.

Původní:

```
if (call SendMsg.send(TOS_UART_ADDR,sizeof(XDataMsg),
&msg_buffer) != SUCCESS
```

Po změně:

```
if (call SendMsg.send(TOS_BCAST_ADDR,sizeof(XDataMsg),
&msg_buffer) != SUCCESS)
```

Příkaz *SendMsg.send* používá první parametr k rozhodnutí, kam by se paketová zpráva měla poslat. Změnou *TOS_UART_ADDR* na *TOS_BCAST_ADDR* říkáme-li komunikační komponentě, aby poslala zprávu broadcastově. mezi všechny uzly v dosahu. Jestliže se má zpráva odeslat přímo do základnové stanice, musí se změnit parametr na 0, protože jakýkoli uzel připojený k základně má ID automaticky nastavené na 0.

6.4.5 Použití *XSniffer* k zobrazení senzorové zprávy

XSniffer je součástí *MoteWorks* a slouží k zachycování zpráv, které jsou posílány bezdrátově.

a) Upravte soubor *MyApp.nc* podle předchozího bodu, připojte uzel bez akumulátorů. Stiskněte F6 (nebo *Tools – Shell*) a pomocí následujícího příkazu zkompilujte a nainstalujte aplikaci do komunikační jednotky:

```
make iris install,1 mib520,comX
```

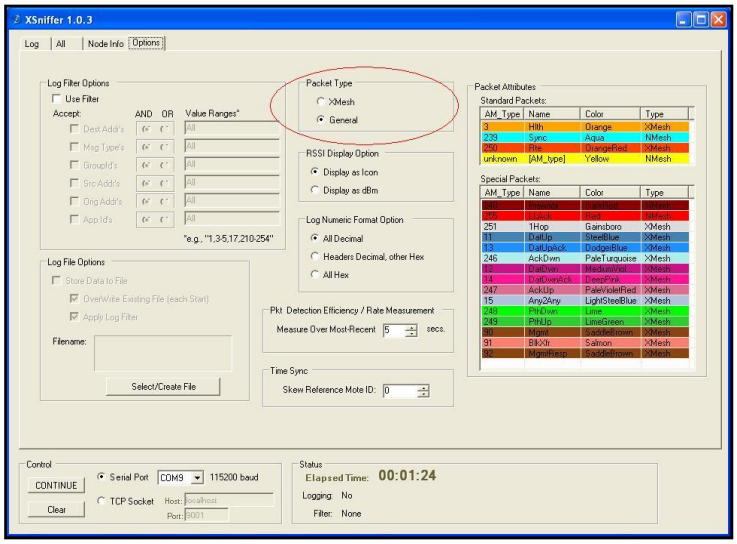
b) Odpojte uzel, vložte 2 kusy AA akumulátoru, připojte senzorovou desku *MDA100* a zapněte. Měly by se rozblikat LED diody každou vteřinu.

c) Připojte další uzel, který bude sloužit jako základna a otevřete soubor *TOSBase.nc* umístěný ve složce **\apps\general\XSniffer.**

d) Stiskněte F6 (nebo *Tools – Shell*) a pomocí následujícího příkazu zkompilujte a nainstalujte aplikaci do komunikační jednotky:

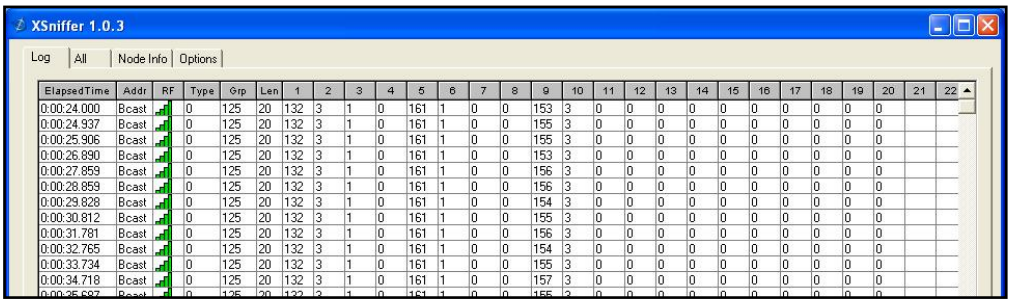
```
make iris install,2 mib520,comX
```

e) Uzel nechte připojený k bráně a spusťte aplikaci *XSniffer* umístěnou na ploše. V záložce *Option* a v *Packet Type* zvolte *General*, viz Obrázek 15.



Obrázek 15 - Nastavení programu XSniffer

V záložce *Log* vyberte komunikační port COM (komunikační port=COMX+1)a spustě zachytávání. Pozn. pokud chcete provést změnu nastavení programu, musíte zachytávání zastavit a poté spustit znovu. Pokud je vše v pořádku, je vidět podobný výsledek jako na Obrázek 16. Analyzujte možnosti analyzátoru *XSniffer*. Zkuste také zastínit senzorovou desku a pozorujte změny hodnot v přijímaných paketech. Je vidět, že data jsou zachycena zhruba každou vteřinu.



Obrázek 16 - XSniffer - výpis zachycených dat

6.5 *Rozbor zdrojového kódu*

Aby mohl světelný senzor vzorkovat data, musí se do konfiguračního souboru přiřadit komponenta *Photo*. Tato komponenta implementuje rozhraní *StdControl* pro zapnutí a vypnutí světelného senzoru a rozhraní *ADC* pro vzorkování hodnot ze senzoru přes port *ADC*.

V části zdrojového kódu souboru *MyApp.nc* je vidět propojení *MyAppM.PhotoControl* (rozhraní *StdControl*) do *Photo.PhotoStdControl* (rozhraní *StdControl* pro světelný senzor). Další propojení je *MyAppM.Light* (rozhraní *ADC*) do *Photo.ExternalPhotoADC* (rozhraní *ADC* pro světelný senzor).

Rozhraní *ADC* je specifické dvěma příkazy *getData* a *getContinousData* a jednou událostí *dataReady*. Pro základní snímání světelného senzoru použijeme příkaz getData. Ten spustí proces vzorkování skrz hardwareový procesor rozhraní *ADC*. Po dokončení se zobrazí skutečné hodnoty událostí *dataReady*. V části zdrojového kódu níže je vidět rozhraní *ADC*.

```
interface ADC {
  async command result_t getData();
  async command result_t getContinuousData();
  async event result_t dataReady(uint16_t data);
}
```

AM_XSXMSG identifikuje aktivní typ zprávy. Tato hodnota, která se má poslat se používá pro rozlišení mezi několika zprávami. Rozhraní *SendMsg* se specifikuje příkazem *send* a událostí *sendDone*. Jestliže se má zpráva poslat, volá se příkazem *send* a správnými parametry. Odeslání zprávy potvrdí událost *sendDone*. Každá zpráva, která se odesílá rozhraním *SendMsg* je definována datovou strukturou *TOS_Msg*:

```
typedef struct TOS_Msg
{
     uint16_t addr;
     uint8_t type;
     uint8_t group;
     uint8_t length;
     int8_t data[TOSH_DATA_LENGTH];
}
typedef TOS_Msg *TOS_MsgPtr;
```

addr – cílová adresa;

type – typ aktivní zprávy (AM_XSXMSG);

group – ID skupiny;

lenght – velikost zatížení;

data – proměnná délka zatížení (senzorová data)

7 Laboratorní úloha č. 3 – Multi-skoková síť a rozbor pomocí *XMesh*

7.1 *Cíl úlohy*

V této úloze se rozšíří předešlá úloha o multi-skokovou síťovou službu, použije se program *XServe* k analýze paketů na PC a programy *XSniffer* a *MoteView* k zobrazení dat ze senzorů.

7.2 *Potřebný hardware*

- 1x programovací jednotka (brána) *MIB520*
- 3x uzel *IRIS XM2110*
- 1x senzorová deska *MDA100CB*

7.3 *Zadání*

- Pomocí sítové mulit-skokové služby *XMesh* se odešlou senzorová data do základnové stanice.

7.4 *Pracovní postup*

a) V programu *Total Commander* vytvořte adresář s vaším jménem **C:\Crossbow\cygwin\opt\MoteWorks\apps\tutorials\MSSY\lab3\jmeno.**

b) Stáhněte si všechny patřičné soubory a uložte do vašeho adresáře. Všimněte si, že chybí hlavní soubor *MyApp.nc*, který je třeba vytvořit.

7.4.1 Vytvoření aplikace *MyApp.nc*

Soubor *MyApp.nc* můžete vytvořit přidáním dvou komponent, které nejsou v předchozí úloze. Komponenta *MULTIHOPROUTER* je *XMesh* multi-skoková služba. Pro výchozí nastavení využívá služba *MULTIHOPROUTER* komponentu *GenericCommPromiscuous* k poskytnutí bezdrátové komunikace. Tato komponenta je podobná jako *GenericComm.GenericCommPromiscuous* z předchozí úlohy, ale navíc poskytuje „zachytávání“ komunikace vyžadované službou *XMesh.*

7.4.2 Instalace aplikace *MyApp* do komunikační jednotky *IRIS*

a) Připojte programovací jednotku *MIB520* k počítači pomocí USB kabelu a poté ve správci hardwaru zjistěte, na kterých USB portech pracuje.

b) Nyní připojte uzel IRIS bez akumulátorů k programovací jednotce. Uzel může být vypnutý, napájí se z brány.

c) Zkompilujte program a nahrajte jej do mikrokontroleru pomocí následujících kroků. Přepněte se do nově vytvořeného souboru ***MyApp.nc***. Stiskněte F6 (nebo *Tools – Shell*) a pomocí následujícího příkazu zkompilujte a nainstalujte aplikaci do komunikační jednotky:

```
make iris install,1 mib520,comX
```

d) Po úspěšném dokončení se na posledním řádku výstupní části objeví text: ***Uploading: flash.***

e) Nyní odpojte uzel *IRIS* z programovací brány, připojte senzorovou desku *MDA100CB*, vložte 2ks AA akumulátorů a zapněte postranní vypínač.

f) Připojte jiný uzel, opět bez akumulátorů k programovací jednotce a v *PN 2* načtěte soubor ***XMeshBase.nc*** umístěný v **/apps/XMesh/XMeshBase.** Stiskněte F6 (nebo *Tools – Shell*) a napište následující příkaz:

```
make iris install,0 mib520,comX
```

g) Po úspěšném dokončení nechte uzel připojený a postupujte pomocí následujících kroků.

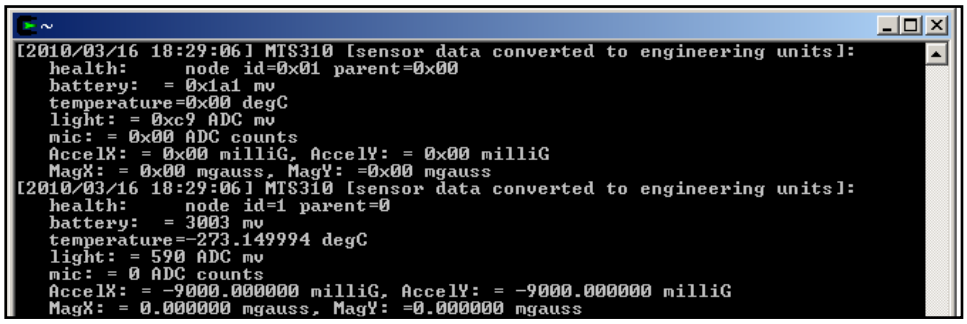
7.4.3 Rozbor senzorové zprávy pomocí XServe

K zobrazení senzorové zprávy přicházející do PC přes sériový port se použ ije nástroj *XServe*, který je součástí *MoteWorks* a běží společně v programu *CYGWIN*.

a) Na pracovní ploše spusťte program *CYGWIN* a napište příkaz:

```
xserve -device=COMX
```

Jestliže je brána správně připojena a vybrán správný komunikační port, zobrazí se Vám zpráva podobná Obrázek 17. Oproti předchozí úloze si můžete všimnout, že přibylo pole *parent*. Toto pole je součástí multi-skokové sítě a udává směrování paketů uzlu s ID 1 přímo do základnové stanice (uzel s ID 0). Základnová stanice poté směruje paketovou zprávu přes sériový port a je zpracována programem *XServe*.



Obrázek 17 - Printscreen Cygwin a běžící XSereve

b) Odpojte uzel, vypněte a odložte ho stranou, budete ho za chvíli potřebovat.

7.4.4 Použití *XSniffer* k zobrazení senzorové zprávy

a) Připojte třetí uzel, který bude sloužit jako základna a otevřete soubor *TOSBase.nc* umístěný ve složce **C:\Crossbow\cygwin\opt\MoteWorks\apps\general\XSniffer**

b) Stiskněte F6 (nebo *Tools* – *Shell*) a pomocí následujícího příkazu zkompilujte a nainstalujte aplikaci do komunikační jednotky:

```
make iris install,2 mib520,comX
```

c) Uzel nechte připojený k bráně a spusťte aplikaci *XSniffe* umístěnou na ploše. V záložce *Option* a v *Packet Type* zvolte *General*, v záložce *Log* vyberte komunikační port COM komunikační port (COMX+1) a spustě zachycování.

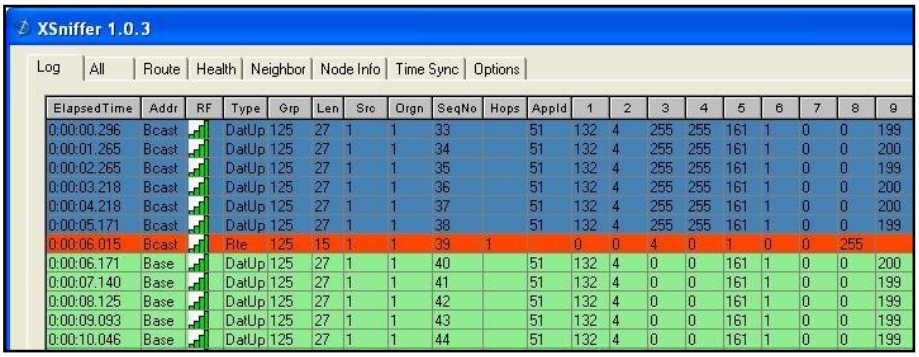
ElapsedTime	Addr	RF	Type	Grp	Len	Src	Orgn	SeqNo	Hops	AppId	1	2	3	4	5	6	7	8	9	10
0:00:43.203	Bcast		DatUp	125	27	1	1	27		51	132	4	255	255	161	1	0	0	152	3
0:00:44.156	Bcast		DatUp	125	27	1	1	28		51	132	4	255	255	161	1	0	0	152	3
0:00:45.140	Bcast		DatUp	125	27	1	1	29		51	132	4	255	255	161	1	0	0	152	3
0:00:46.125	Bcast		DatUp	125	27	1	1	30		51	132	4	255	255	161	1	0	0	152	3
0:00:47.093	Bcast		DatUp	125	27	1	1	31		51	132	4	255	255	161	1	0	0	152	3
0:00:48.062	Bcast		DatUp	125	27	1	1	32		51	132	4	255	255	161	1	0	0	152	3
0:00:49.046	Bcast		DatUp	125	27	1	1	33		51	132	4	255	255	161	1	0	0	152	3
0:00:50.031	Bcast		DatUp	125	27	1	1	34		51	132	4	255	255	161	1	0	0	153	3
0:00:51.000	Bcast		DatUp	125	27	1	1	35		51	132	4	255	255	161	1	0	0	152	3
0:00:51.984	Bcast		DatUp	125	27	1	1	36		51	132	4	255	255	161	1	0	0	153	3
0:00:52.953	Bcast		DatUp	125	27	1	1	37		51	132	4	255	255	161	1	0	0	152	3
0:00:53.937	Bcast		DatUp	125	27	1	1	38		51	132	4	255	255	161	1	0	0	153	3
0:00:54.765	Bcast		Rte	125	12	1	1	39	255		255	0	255	0	0					
0:00:54.906	Bcast		DatUp	125	27	1	1	40		51	132	4	255	255	161	1	0	0	153	3

Obrázek 18 - XSniffer - výpis zachycených dat s jedním uzlem

Zpráva je odeslána přibližně každou vteřinu. První věc, které si můžete všimnout, je ve sloupci *Addr* zpráva *Bcast*, což znamená, že senzor s ID 1 (sloupec *Src*) rozesílá jeho pakety do všech uzlů v síti. Tento stav trvá tak dlouho, dokud se síť efektivně neuspořádá.

Další důležitou věcí, která je z obrázku patrná, je ve sloupci *Type*. *DatUp* identifikuje datovou zprávu posílanou z uzlu do základnové stanice. Typ *Rte* označuje obnovení směrování. Tato zpráva se pravidelně posílá mezi všechny uzly v síti za účelem aktualizace směrových tabulek.

d) Nyní zapněte uzel s aplikací *XMeshBase* a měli byste pozorovat změnu v programu *XSniffer*.



ElapsedTime	Addr	RF	Type	Grp	Len	Src	Orgn	SeqNo	Hops	AppId	1	2	3	4	5	6	7	8	9
0:00:00.296	Bcast		DatUp	125	27	1	1	33		51	132	4	255	255	161	1	0	0	199
0:00:01.265	Bcast		DatUp	125	27	1	1	34		51	132	4	255	255	161	1	0	0	200
0:00:02.265	Bcast		DatUp	125	27	1	1	35		51	132	4	255	255	161	1	0	0	199
0:00:03.218	Bcast		DatUp	125	27	1	1	36		51	132	4	255	255	161	1	0	0	200
0:00:04.218	Bcast		DatUp	125	27	1	1	37		51	132	4	255	255	161	1	0	0	200
0:00:05.171	Bcast		DatUp	125	27	1	1	38		51	132	4	255	255	161	1	0	0	199
0:00:06.015	Bcast		Rte	125	15	1	1	39	1		0	0	4	0	1	0	0	255	
0:00:06.171	Base		DatUp	125	27	1	1	40		51	132	4	0	0	161	1	0	0	200
0:00:07.140	Base		DatUp	125	27	1	1	41		51	132	4	0	0	161	1	0	0	199
0:00:08.125	Base		DatUp	125	27	1	1	42		51	132	4	0	0	161	1	0	0	199
0:00:09.093	Base		DatUp	125	27	1	1	43		51	132	4	0	0	161	1	0	0	199
0:00:10.046	Base		DatUp	125	27	1	1	44		51	132	4	0	0	161	1	0	0	199

Obrázek 19 - XSniffer - výpis zachycených dat se dvěma uzly

Po zapnutí uzlu je ve zprávě *Rte* vidět (aktualizace směrování), že se nyní zpráva generuje pro uzel 0 – základnová stanice (označení *Base* ve sloupci *Addr*)

a přiřadí ho jako jeho rodiče. Poté začne uzel ID 1 odesílat zprávy přímo do základnové stanice namísto všesměrového vysílání.

7.5 *Rozbor zdrojového kódu*

GenericComm – jedno-skoková služba

XMesh – mulit-skoková síťová služba

MULTIHOPROUTER – komponenta implemetující *XMesh*.

Oproti předešlé úloze je vidět změna ve způsobu spojení. V souboru *MyAppM.nc* se nyní používá rozhraní *MhopSend*.

```
interface MhopSend {
     command result_t send(uint16_t dest, uint8_t mode,
TOS_MsgPtr msg,uint16_t length);
     command void* getBuffer(TOS_MsgPtr msg, uint16_t*
length);
     event result_t sendDone(TOS_MsgPtr msg, result_t
success);
}
```

Příkaz *send* přidává parametr, který specifikuje typ komunikačního přenosu. Jedná se o *MOTE_UPSTREAM*, který odesílá zprávu směrem k základnové stanici. V odesílání senzorové zprávy je rozdíl v přiřazení funkce *StdControl.init*. *XMesh* voláme příkazem *Send.getBuffer*, který vrací pointer do oblasti zatížení pomocí *msg_buffer*. Aby paket mohl obsahovat stávající směrování na rodiče, musí se volat příkazem *RouteControl.getParent*. Po odeslání zprávy příkazem Send.send se určí základnová stanice jako místo doručování zpráv a přenosový mód jako *MOTE_UPSTREAM*. Potvrzení odeslání je stejné jako v předchozí úloze a to událostí *Send.sendDone*.

8 Laboratorní úloha č. 4 – přídavné funkce sítě *XMesh*

8.1 *Cíl úlohy*

Úloha je zaměřena na přídavné funkce síťové služby *XMesh*. První část je potvrzování mezi koncovými uzly a v druhé části úlohy je popsáno použití vzdálených příkazů posílaných ze základnové stanice do jednotlivých uzlů.

8.2 *Potřebný hardware*

- 1x Programovací jednotka (brána) *MIB520*
- 2x Uzel *IRIS XM2110*

8.3 *Zadání*

- Upravte kód z předchozí úlohy tak, aby docházelo k potvrzování ze základnové stanice.
- Rozblikejte žlutou LED diodu, jako signalizaci potvrzení příchozí zprávy zpět na základnovou stanici.
- Použijte komponentu *XCommand* ke zpracování a nahrání příkazu do uzlu.

8.4 *Část první – přídavné funkce sítě XMesh*

8.4.1 Pracovní postup

a) V programu Total Commander vytvořte adresář s vaším jménem **C:\Crossbow\cygwin\opt\MoteWorks\apps\tutorials\MSSY\lab4a\jmeno.**

b) Stáhněte si všechny patřičné soubory a uložte do vašeho adresáře. Chybí hlavní soubor *MyApp.nc*, který je třeba vytvořit.

8.4.2 Vytvoření aplikace *MyApp.nc* a *MyAppM.nc*

Otevřete soubor *MyApp_Sensing_nc.html* a důkladně prostudujte vazby mezi jednotlivými komponentami. Pomocí tohoto grafického vyjádření vytvořte soubor *MyApp.nc*. Detailně prostudujte seznam rozhraní, které modul *MyAppM.nc*

používá. Důležitá jsou vyjádření, jak jsou jednotlivá rozhraní v modulu definována, abyste mohli tato rozhraní definovat v aplikaci *MyApp.nc*.

V souboru MyApp.nc je změna oproti předchozí aplikaci v přidání konfiguračního souboru do rozhraní *ReceiveAck*. Toto rozhraní je nezbytné k provedení funkce zpětného volání, které se generuje při potvrzení o doručení do základnové stanice programem *XMesh*. První změna v *MyAppM.nc* je způsob doručení do *MODE_UPSTREAM_ACK* vyžadující od *XMesh* zpětné potvrzení o doručení na základnovou stanici. Druhá změna je přidání funkce *ReceiveAck.receive* pro potvrzení, že zpráva přišla ze základnové stanice. Při správné funkci bude blikat žlutá LED dioda, jako indikace obdržení tohoto potvrzení. *MODE_UPSTREAM_ACK* je pouze pro potvrzení, nezaručí skutečné doručení zprávy!

8.4.3 Instalace aplikace *MyApp* do komunikační jednotky *IRIS*

a) Připojte programovací jednotku MIB520 k počítači pomocí USB kabelu a poté ve správci hardwaru zjistěte, na kterých USB portech pracuje.
b) Nyní připojte uzel IRIS bez akumulátorů k programovací jednotce, který bude zastávat funkci senzoru. Uzel může být vypnutý, napájí se z brány.
c) Zkompilujte program a nahrajte jej do mikrokontroleru pomocí následujících kroků. Přepněte se do nově vytvořeného souboru ***MyApp.nc*.** Stiskněte F6 (nebo *Tools – Shell*) a pomocí následujícího příkazu zkompilujte a nainstalujte aplikaci do komunikační jednotky:

```
make iris install,1 mib520,comX
```

d) Nyní odpojte uzel IRIS z programovací brány, vložte 2ks AA akumulátorů.
e) Připojte uzel bez akumulátorů k programovací jednotce, který bude základní stanice a v *PN2* načtěte soubor ***XMeshBase.nc*** umístěný v **apps/general/XMeshBase.** Stiskněte F6 (nebo Tools – Shell) a napište následující příkaz:

```
make iris install,0 mib520,comX
```

f) Po úspěšném dokončení nechte uzel připojený a první uzel s ID 1 zapněte. Než se síť stabilizuje, bude blikat červená a zelená dioda. Poté co se uzel připojí do sítě se základnovou stanicí, rozbliká se pouze žlutá dioda, což signalizuje potvrzování.

g) Pro zobrazení příchozích paketů použijte také *XServe* a *MoteView*.

8.5 *Část druhá – vzdálené příkazy pomocí XServeterm*

8.5.1 Pracovní postup

a) V programu Total Commander vytvořte adresář s vaším jménem

C:\Crossbow\cygwin\opt\MoteWorks\apps\tutorials\MSSY\lab4b\jmeno

b) Stáhněte všechny patřičné soubory a uložte do vašeho adresáře

8.5.2 Vytvoření aplikace *MyApp.nc* a *MyAppM.nc*

V *MyApp.nc* je změna oproti úloze 3 přidání komponenty *XCommandC*, která umožňuje použití vzdálených příkazů. Do modulu *MyAppM.nc* je komponenta poslána skrz rozhraní *XCommand*. Toto rozhraní poskytuje jednu událost nazvanou *received* která musí být implementována v aplikačním modulu – v tomto případě *MyAppM.nc*. Tato změna také značí příchod příkazu do uzlu. Jediná nutná změna v *MyAppM.nc* je implementace funkce *XCommand.received*. Změna stavů LED diod je řešena implicitně *XCommand. Set_rate* je pouze jiný příkaz prováděn lokálně.

a) V *Programmer's Notepad 2* otevřete soubor *MyApp.nc*, zvolte F6 (nebo *Tools – Shell*) a napište následující příkaz:

```
make iris install,1 mib520,comX
```

b) Připojte uzel bez akumulátorů k programovací jednotce, který bude základnová stanice a načtěte soubor ***XMeshBase.nc*** umístěný v **/MoteWorks/apps/general/XMeshBase.** Stiskněte F6 (nebo *Tools – Shell*) a napište následující příkaz:

```
make iris install,0 mib520,comX
```

c) Po úspěšném dokončení nechte uzel připojený k *MIB520* a první uzel s ID 1 zapněte.

8.5.3 Využití *Serveterm* k posílání příkazů do uzlu

XServeterm je instance běžící v terminálovém rozhraní *XServe* a poskytuje mnoho příkazů pro monitorování a konfiguraci senzorových sítí. Jedna z možností jsou příkazy, které je schopen *XMesh.*

a) Na pracovní ploše spusťte program *CYGWIN* a napište příkaz:

```
xserve –device=COMX
```

b) Spusťte druhé okno s programem *CYGWIN* a napište následující příkaz:

```
xserveterm –group125
```

c) První příkaz je ***get_config***. Ten zobrazí aktuální parametry uzlu (podobné jako příkaz ping). Nyní zjistěte aktuální stav základnové stanice příkazem:¨

```
get_config 0
```

d) Zkuste stejný příkaz pro uzel s ID 1. Jestli se zobrazí chyba *time-out*, uzel ještě není v síti. Vyčkejte a vyzkoušejte příkaz později. Po úspěšné

odpovědi můžete vyzkoušet další příkazy. Další z nich je ***set_rate***, kterým se nastavuje vzorkovací frekvence uzlu. Standardní doba je 1000 ms. Změňte dobu na 5000 ms příkazem s ID uzlu:

```
set_rate 1 5000
```

Rychlost vzorkování jde rozpoznat díky frekvenci blikání diod.

e) Příkazem ***actuate*** můžete nastavit stavy jednotlivých LED diod. Pro přehled slouží následující tabulka. Příkaz obecně:

```
actuate <cílová adresa – ID> <ID zařízení> <stav>
```

Příkaz pro uzel ID 1 a rozblikání žluté diody:

```
actuate 1 1 2
```

Tabulka 2 - XServeterm - přehled zařízení a jejich ID

Zařízení	ID zařízení	Stav
Zelená LED	0	OFF - 0
Žlutá LED	1	ON - 1
Červená LED	2	Přepínání - 2
Všechny LED	3	
Reproduktor	4	
Relé 1	5	
Relé 2	6	
Relé 3	7	

9 Laboratorní úloha č. 5 – Měření se senzorovou deskou MTS420/400CC

9.1 *Cíl úloh*

Cílem této úlohy je navázat komunikaci uzlu se senzorovou deskou MTS420/400CC a získat potřebná data. Deska je osazena potřebnými částmi pro měření teploty, vlhkosti, osové polohy, barometrického tlaku, okolního světla a GPS souřadnic.

9.2 *Potřebný hardware*

- 1x programovací jednotka (brána) *MIB520*
- 2x uzel *IRIS XM2110*
- 1x senzorová deska *MTS420/400CC*

9.3 *Zadání*

- Napište potřebný kód pro senzorový uzel a základnovou stanici, nahrejte ho do uzlů a pomocí XServe zobrazte data.

9.4 *Pracovní postup*

a) V programu *Total Commander* vytvořte adresář s vaším jménem **C:\Crossbow\cygwin\opt\MoteWorks\apps\tutorials\MSSY\lab5\jmeno.**

b) Stáhněte si všechny potřebné soubory a uložte do adresáře s vaším jménem. Prostudujte soubor MyApp.nc a MyAppM.nc.

9.4.1 Instalace aplikací do komunikačních jednotek *IRIS*

a) Připojte programovací jednotku *MIB520* k počítači pomocí USB kabelu a poté ve správci hardwaru zjistěte, na kterých USB portech pracuje.

b) Nyní připojte uzel *IRIS* bez akumulátorů k programovací jednotce. Uzel bude plnit funkci základnové stanice a může být vypnutý, napájí se z brány.

c) Načtěte soubor *XMeshBase.nc* a zkompilujte klávesou F6 a příkazem:

```
make iris install,mib520,comX
```

d) Nyní dejte uzel *IRIS* z programovací brány stranou a připojte druhý uzel k bráně.

e) Načtěte soubor *MyApp.nc* a zkompilujte s následujícími parametry

```
make iris install,1 mib520,comX
```

f) Po úspěšném nahrání programu do uzlu, senzor odpojte, vložte 2ks AA akumulátorů, připojte senzorovou desku MTS420/400CC a zapněte.

g) Do brány znovu připojte první uzel a vyčkejte, až se sestaví síť.

9.4.2 Rozbor senzorové zprávy pomocí XServe

Na pracovní ploše spusťte program *CYGWIN* a napište příkaz:

```
xserve –device=COMX
```

Jestliže je brána správně připojena, vybrán správný komunikační port, programy správně napsány a zkompilovány, začnou se zobrazovat data z uzlu. Aktualizace GPS souřadnic trvá přibližně 2 minuty. Pro kontrolu můžete souřadnice přepočítat i na vteřiny a na Internetu vyhledat polohu. Popřípadě porovnat polohu mobilním GPS zařízením.

Závěr

Praktická část práce obsahuje 5 laboratorních návodů. Jelikož je pro správnou kompilaci programů nezbytné používat adresářovou strukturu programového balíku od firmy Crossbow, je v návodech uvedena cesta pro uložení projektu. Nejsou však uvedeny cesty, kde jsou uloženy potřebné počáteční části programů. Na začátku každé laboratoře sdělí vyučující aktuální umístění, popřípadě může editovat jednotlivé návody s aktuální složkou, případně adresou.

V první úloze mají studenti za úkol rozblikat LED diody v libovolném čase a rozebrat kód pomocí grafického diagramu. Tímto úkolem se seznámí s programovacím prostředím, připojením hardwaru, zjištění potřebných portů v počítači a dalšími základními věcmi potřebnými pro složitější měření. Druhý návod je zaměřený na komunikaci uzlu se základnovou stanicí. Jako kontrola slouží indikace 3 různě barevných LED diod. Třetí laboratorní cvičení popisuje multi-skokovou síť a rozbor paketových zpráv programy XServe, XSniffer a MoteView. Další úkol je zaměřen na využití přídavných síťových funkcí XMesh a jejich možnému využití. Poslední pátý návod je zaměřen na problematiku GPS měření se senzorovou deskou MTS420, která je schopna signál přijímat.

Seznam literatury

1. Eric Brewer, David Culler, David GAy. nesC: A Programming Language for Deeply Networked Systems. *UC Berkley WEBS Projects.* [Online] 14. December 2004. http://nescc.sourceforge.net/.

2. TinyOS Documentation Wiki. *TinyOS Community Forum.* [Online] http://docs.tinyos.net/index.php/main_page.

3. Gay, David, a další. *nesC 1.1 Language Reference Manual.* Barkely : University of California, May 2003.

4. Gay, David, Welsh, Matt a Levis, Philip. *The nesC Language: A Holistic Approach to Networked Embedded Systems.* Berkley, California : University of California, 2003.

5. *XMesh User's Manual.* San Jose, California : Crossbow Technology, Inc., April 2007. 7430-0108-01.

6. *MIB520 Datasheet.* San Jose, California : Crossbow Technology, Inc. 6020-0091-03.

7. *IRIS Datasheet.* San Jose, California : Crossbow Technology, Inc. 6020-0124-01.

8. *MTS MDA Datasheet.* San Jose, California : Crossbow Technology, Inc. . 6020-0047-03.

9. Kohvakka M., Suhonen J., Kuorilehto M. *Ultra-Low Energy Wireless Sensor Networks in Practice.* místo neznámé : John Wiley &Sons, Ltd., 2007. ISBN:978-0-470-05786-5.

10. Bakshi B. A., Prasanna K. V. *Architecture-Independent Programming for Wireless Sensor Networks.* místo neznámé : Wiley &Sons, Ltd., 2008. ISBN: 978-0-471-77889-9.

11. *MoteWorks Getting Started Guide.* San Jose : Crossbow Technology, Inc., March 2007.

12. Levis, Philip. *TinyOS Programming.* October 2006.

Seznam použitých zkratek a veličin

ZKRATKA	VÝZNAM
COM	Communication Port
EEPROM	Electrically Erasable Programmable Read-Only Memory
FIFO	First In, First Out
GPS	Global Position System
I/O	Input/Output
I2C (I^2C)	Inter-Integrated Circuit
IEEE	The Institute of Electrical and Electronics Engineers
ISP	In System Programming
LAN	Local Area Network
LED	Light-Emitting Diode
LPT	Line Print Terminal
OTAP	Over the Air Programming
PC	Personal Computer
PN2	Programmer's Notepad
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
ADC	Analog to Digital Converter

VELIČINA	ZÁKLADNÍ JEDNOTKA
Barometrický tlak	bar/s
Frekvenční spektrum	Hz
Jednotka informace	bit
Jednotka modulační rychlosti	1 baud
Množství dat	byte
Přenosová rychlost v bitech	b/s
Přenosová rychlost v bytech	B/s
Vlnová délka	m