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 SESTAVENÁ Z KOMPONENT ARDUINO

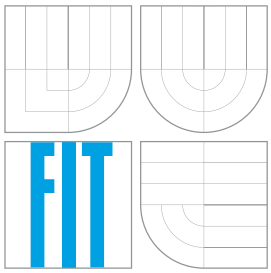
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAKUB ŠPLÍCHAL

BRNO 2012

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 SESTAVENÁ Z KOMPONENT ARDUINO

WIRELESS SENSOR NETWORK WITH ARDUINO COMPONENTS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JAKUB ŠPLÍCHAL
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN SAMEK, Ph.D.
SUPERVISOR

BRNO 2012

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2011/2012

Zadání diplomové práce

Řešitel: **Šplíchal Jakub, Bc.**
Obor: Inteligentní systémy
Téma: **Bezdrátová senzorová síť sestavená z komponent Arduino**
Wireless Sensor Network with Arduino Components
Kategorie: Počítačové sítě

Pokyny:

1. Prostudujte a popište hardwarovou platformou a vývojové prostředí Arduino. Zaměřte se především na využití platformy pro bezdrátovou senzorovou síť.
2. Navrhněte bezdrátovou senzorovou síť z komponent Arduino založenou na modulech XBee. Zaměřte se na problematiku směřování dat ze senzorových uzlů do základního uzlu (base-station) a problematiku dynamické topologie sítě.
3. Vámi navrženou síť sestavte a implementujte firmware pro jednotlivé senzorové uzly a pro base-station, která bude ze senzorových uzlů odečítat měřené veličiny. Dále implementujte jednoduchou aplikaci na PC, která bude odeslaná ze sítě do base-station načítat a uživatelsky přívětivě je zobrazovat (formou tabulky a grafu).
4. Výsledek vaší práce otestujte v budově FIT VUT, zaměřte se především na schopnost směřování dat ze senzorových uzlů do base-station.
5. Zhodnoťte výsledky vaší práce a diskutujte možnosti jejího rozšíření.

Literatura:

- Margolis, M.: Arduino Cookbook, O'Reilly Series, 2011.
- Faludi, R.: Building Wireless Sensor Networks: With ZigBee, XBee, Arduino, and Processing, O'Reilly Series, 2011.

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1 a 2.

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

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

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

Vedoucí: **Samek Jan, Ing.**, UITS FIT VUT
Datum zadání: 19. září 2011
Datum odevzdání: 23. května 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Diplomová práce se zabývá vytvořením bezdrátové senzorové sítě sestavené z komponent Arduino. Práce obsahuje seznámení s platformou Arduino a jejími možnostmi v kombinaci s bezdrátovými moduly XBee. Důležitou částí práce je návrh bezdrátové sítě z těchto komponent a aplikace pro zobrazení naměřených hodnot ze senzorových uzlů. Cílem práce je vytvoření senzorové sítě s dynamickou topologií a prozkoumání jejího chování v reálném prostředí a vytvoření aplikace pro uložení a zobrazení naměřených dat z jednotlivých senzorových uzlů.

Abstract

This thesis deals with the creation of wireless sensor networks consisting of components Arduino. The work includes introduction to the Arduino platform and its capabilities in combination with the wireless XBee modules. The important part is design a wireless network from these components and applications for the display of measured values from sensor nodes. The goal is to create sensor networks with a dynamic topology and examine its behavior in real environment and the creation of applications for saving and displaying measured data from individual sensor nodes.

Klíčová slova

Arduino, XBee, senzory, DigiMesh, WSN, Qt.

Keywords

Arduino, XBee, sensors, DigiMesh, WSN, Qt.

Citace

Jakub Šplíchal: Bezdrátová senzorová síť
sestavená z komponent Arduino, diplomová práce, Brno, FIT VUT v Brně, 2012

Bezdrátová senzorová síť sestavená z komponent Arduino

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Jana Samka, Ph.D.

. . . . . .
Jakub Šplíchal
20. května 2012

Poděkování

Chtěl bych poděkovat vedoucímu této práce Ing. Janu Samkovi, Ph.D. za jeho odbornou pomoc při vytváření této práce.

Obsah

Kapitola 1

Úvod

S bezdrátovými sítěmi se v dnešní době setkáváme stále častěji. S rozšířenější sítí, se kterou se můžeme setkat, je síť WiFi. Nalezneme ji ve školách, kancelářských budovách nebo domácnostech. Síť WiFi nejčastěji slouží k připojení osobních počítačů, notebooků, mobilních telefonů a další řady zařízení k síti Internet. Dnešní doba ale také vyžaduje sítě, které slouží pro spojení nízkovýkonových zařízení v sítích PAN[1] na malé vzdálenosti například jako sítě založené na specifikaci ZigBee. ZigBee je určeno pro vytváření malých, spolehlivých, energeticky nenáročných a v neposlední řadě cenově dostupných digitálních rádiových modulů. Využití těchto ZigBee bezdrátových modulů můžeme najít v dálkových ovladačích osvětlení, elektroměrech, spotřebičích a v mnoha jiných zařízeních.

Jednou z možností využití těchto sítí je spojení s měřením fyzikálních veličin za pomocí senzorů. Na dnešním trhu můžeme najít velké množství senzorů pro měření teplot, vlhkosti vzduchu, znečištění vzduchu, osvětlení, vibrací, radiace, spotřeby paliv a další. Tímto spojením vznikají senzorové jednotky, které vytvářejí senzorové bezdrátové sítě. Jejich uplatnění je velmi široké například při požárech lesa, měření emisí na farmách, kontrole chemických nebo mechanických procesů, bezpečnosti, kontrole zboží v logistice, zavlažovaní, měření spotřeby vody a energie.

1.1 Cíl a struktura práce

Cílem této diplomové práce je navrhnout a realizovat senzorovou bezdrátovou síť spolu s aplikací na zobrazování naměřených dat. Samotná síť bude sestavena z open source komponent Arduino a bezdrátových modulů XBee. Arduino poslouží ke snímání hodnot ze senzorů a za pomocí bezdrátových modulů XBee se jednotlivé Arduino spojí do jediné bezdrátové sítě. Takto vytvořená síť bude založena na mesh topologii. Síť s touto topologií má zabezpečenu automatickou konfiguraci struktury sítě, spolehlivé směrování mezi jednotlivými uzly a automatický přístup nových uzlů do sítě prostřednictvím dosavadních uzlů. Jeden z těchto uzlů bude tzv. výchozí uzel, který přepošle naměřené hodnoty aplikaci v počítači. Výchozí uzel bude připojený pomocí USB spojení přímo s počítačem, ve kterém poběží aplikace. Aplikace následně uloží data do lokální databáze a bude také spravovat informace o uzlech a použitých senzorech. Uložená data budou následně zobrazována pomocí vytvořené aplikace ve formě grafů a tabulek.

Na samotném začátku práce je čtenář obeznámen s open source platformou Arduino, s jejími vlastnostmi, rozšířitelností a možnostmi vývoje. V této kapitole je také uveden popis hardwarových desek Arduino využívaných v této práci a to UNO a FIO. Dále se věnuje bezdrátovým modulům XBee firmy Digi International. V poslední části kapitoly je uveden popis senzorů a jiného použitého hardwaru. Po této kapitole je prezentován návrh senzorové bezdrátové sítě, jednotlivých senzorových

[1] Personal area network

uzlů, přičemž je zde také probrán návrh aplikace a databáze. Kapitola také popisuje v menší míře použité technologie při návrhu aplikace. V kapitole 4 je probrána implementace jednotlivých senzorových uzlů a aplikace na uložení naměřených dat a jejich zobrazení. Kapitola také obsahuje testování sítě v budově VUT FIT. Jsou zde také zmíněny problémy, jež se během vývoje objevily. V předposlední kapitole 5 jsou popsána navrhovaná rozšíření do budoucna, která se týkají aplikace i samotné bezdrátové sítě.

Kapitola 2

Základní teorie a pojmy

V této kapitole budou uvedeny důležité informace o open source platformě Arduino, její specifikaci a možnostech využití. Samotná platforma Arduino je rozdělena na hardware a software část. Dále zde bude uveden popis bezdrátových modulů Xbee a jejich rozdělení dle fyzických i softwarových vlastností. Poslední část této kapitoly bude věnována popisu použitých senzorů a jiného hardwaru.

2.1 Arduino

Arduino je open source vývojová elektronická prototypová platforma založena na flexibilním, snadno použitelným hardware a software. Arduino je určeno pro studenty, návrháře, učitele a kohokoliv, kdo se zajímá a vytváření interaktivních projektů a prostředí. Samotný vývoj platformy začal v roce 2005 a jeho cílem bylo vytvořit open source, jednoduchou a cenově dostupnou platformu pro ovládání studenských projektů. Arduino vychází z open source platformy *Wiring*[1] a grafického vývojového prostředí *Processing*[2]. Programovací jazyk vychází ze samotného projektu Wiring, který je podobný jazyku C++ s některými modifikacemi.

Arduino desky jsou relativně levné ve srovnání s jinými mikrokontrolér platformami. Desky také mohou být sestaveny manuálně díky dostupnosti součástek a schémat dostupných pod *Creative Commons licencí Attribution-ShareAlike 2.5*. Samotné vývojové prostředí je multiplatformní a běží pod operačními systémy Windows, Linux a Mac OSX. Samotný projekt získal ocenění v kategorii digitálních komunit na *Prix Ars Electronica 2006*. Více o platformě Arduino v [42, 15, 34].

Hardware Arduina je založen na mikrokontrolérech ATmega od firmy Atmel. Jedná se o mikrokontroléry s architekturou AVR, což je tedy 8-bitový procesor typu RISC. Platforma Arduino využívá hlavně typy ATmega8, ATmega168 a ATmega328, jejichž srovnání lze najít v tabulce 2.1.

Tabulka 2.1: Mikrokontroléry ATmega. Převzato z [12].

	ATmega8	ATmega168	ATmega328
Flash paměť	8 kB	16 kB	32 kB
Počet pinů	32	32	32
Maximální frekvence	16 MHz	16 MHz	20 MHz
Velikost EEPROM	512 B	512 B	1 kB

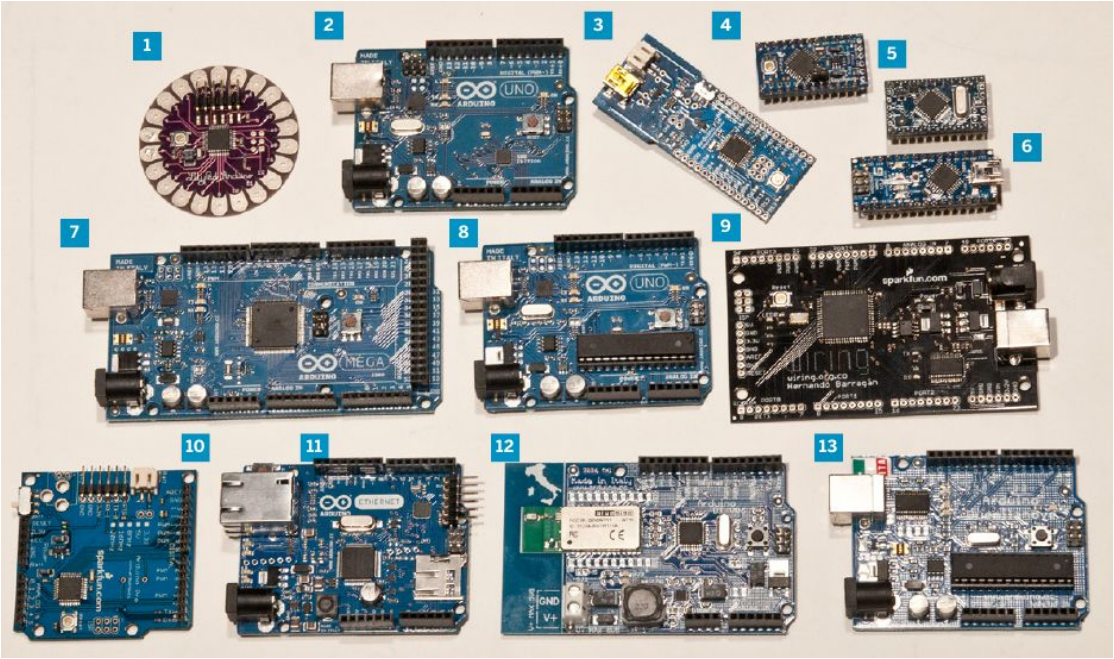
Z tabulky je vidět, že mezi hlavní rozdíly patří velikosti vnitřní paměti EEPROM a FLASH

[1] http://wiring.org.co
[2] http://processing.org/

paměti. Mikrokontroléry ATmega obsahují také tři časovače s porovnávacími módy, interní a externí přerušení, SPI sériový port, programovatelný watchdog s vnitřním oscilátorem a 5 softwarových úsporných módů. Více o mikrokontrolérech ATmega v [12].

2.1.1 Druhy desek Arduino

Platforma Arduino nemá pouze jediný druh hardwarové desky, ale má několik oficiálních i neoficiálních variant, které vznikly díky otevřenosti projektu. Liší se jak velikostí, tak periferiemi. Mezi nejznámější oficiální desky patří Arduino Uno, Mega2560, Nano, Bluetooth, LilyPad, Mini, Pro, Pro Mini, a pár dalších starších modelů. Většina těchto modelů je vyráběna firmou SmartProjects v Itálii. Arduino Pro, Pro Mini a LilyPad vyrábí firma SparkFun Electronics. Další informace o deskách Arduino v [26].



Obrázek 2.1: Desky Arduino. Obrázek převzat z [4].

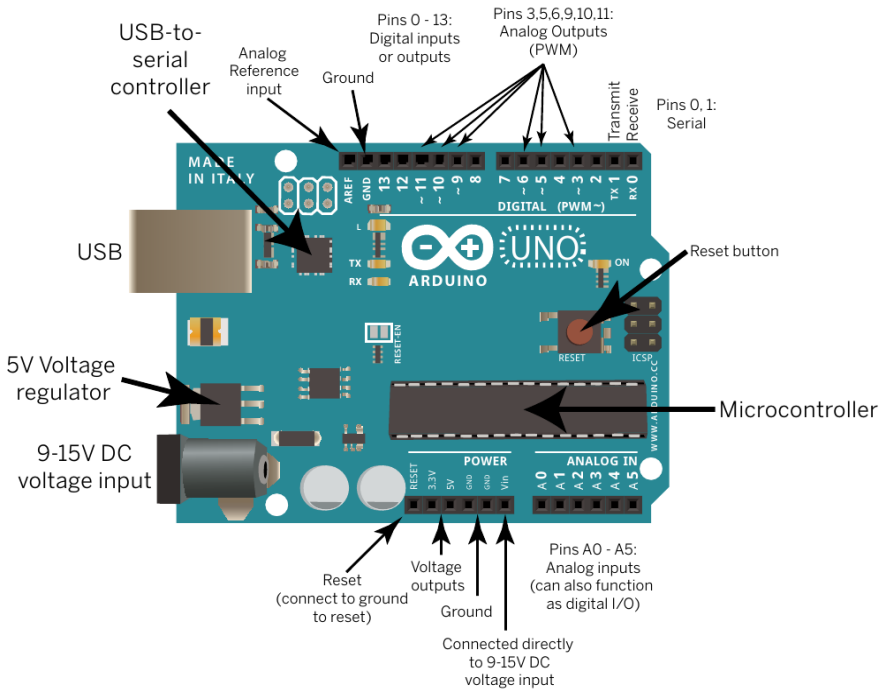
Na obrázku 2.1 jsou zobrazeny různé druhy desek Arduino a také deska Wiring, ze které platforma Arduino vychází: 1. LilyPad Arduino 2. Arduino Uno SMD 3. Arduino Fio 4. Arduino Pro Mini 5. Arduino Mini 6. Arduino Nano 7. Arduino Mega 2560 8. Arduino Uno 9. Wiring board 10. Arduino Pro 11. Arduino Ethernet 12. Arduino Bluetooth 13. Arduino Duemilanove. Více se můžete dočíst v knize [4].

V dalších částech této kapitoly budou popsány desky Arduino UNO a FIO. Tyto desky byly použity při návrhu a implementaci senzorové sítě. V kapitole je také obsažen popis rozšiřujících desek pro rozšíření funkčnosti desek například o možnost připojení XBee modulů a dalších.

Arduino Uno

Arduino Uno je hardwarová deska založena na mikrokontroléru ATmega328. Obsahuje 14 digitálních vstupů a výstupů, kde 6 může být použito pro PWM[3] výstup, 6 analogových vstupů, 16 MHz

[3] Pulse-width modulation



Obrázek 2.2: Arduino Uno. Převzato z [4].

krystalový oscilátor, USB připojení, konektor napájení, ICSP patice a tlačítko pro reset. Arduino Uno oproti minulým verzím nepoužívá čip pro FTDI USB-to-serial, ale využívá čip Atmega8U2 jako USB-to-serial převodník.

„Uno" v italštině znamená jedna a deska je tedy pojmenována k příležitosti nadcházejícího vydání Arduino 1.0. Na obrázku 2.2 je zobrazen vzhled desky a její popis.

Deska je napájena pomocí USB připojení nebo externího napájecího zdroje, přičemž si sama zvolí zdroj napájení. Externí napájení, tedy ne USB, může být z napájecího adaptéru nebo baterie. Adaptér lze připojit 2,1 mm konektorem do napájecího konektoru na desce.

Každý ze 14 digitálních pinů může být použit jako vstup nebo výstup. Zda je pin výstupní nebo vstupní se určí při běhu programu pomocí funkce `pinMode()`. Pomocí funkcí `digitalWrite()` a `digitalRead()` probíhá zápis a čtení hodnot z pinů. Každý pin operuje na 5 V a může přijímat nebo poskytovat až 40 mA. Také obsahují interní pull-up rezistor s rozmezím 20-50 kOhm. Tyto rezistory standardně nejsou připojeny. Některé výstupy mají specializované funkce:

Sériové rozhraní Pin 1 (TX) se používá k odeslání a pin 0 (RX) k příjmu TTL sériových dat. Tyto piny jsou připojeny k ATmega8U2 USB-to-Serial čipu.

Externí přerušení Piny 2 a 3 mohou být využity k přerušení při nízké hodnotě, náběžné nebo klesající hraně a při změně hodnoty. K tomuto účelu slouží funkce `attachInterrupt()`.

PWM Piny 3, 5, 6, 9, 10 a 11 umožňují 8 bitový PWM výstup za pomocí funkce `analogWrite()`.

SPI Piny 10 (SS), 11 (MOSI), 12 (MISO) a 13(SCK) podporují SPI komunikaci pomocí knihovny *SPI library*.

LED Pin 13 obsahuje vestavěnou LED diodu. Při vysoké hodnotě napětí je dioda rozsvícena a při nízké nesvítí.

TWI Piny A4 (SDA) a A5 (SCL) podporují TWI komunikaci pomocí knihovny *Wire library*.

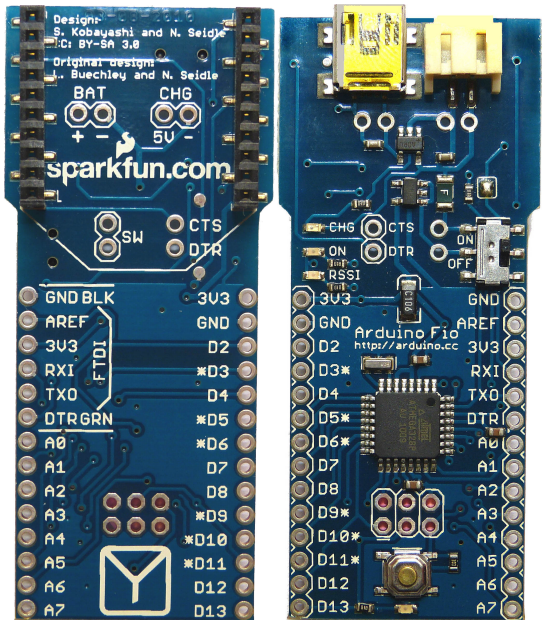
AREF Pin pro referenční napětí pro analogové vstupy.

RESET Slouží k restartování mikrokontroléru.

Na desce se také nachází 6 analogových vstupů označených jako A0–A5. Každý z pinů je 10 bitový a měří napětí od 0 do 5V. Tento rozsah je možno změnit pomocí pinu *AREF* a funkce `analogReference()`. Programování probíhá pomocí vývojového prostředí Arduino. Mikroprocesor ATmega328 na Arduino Uno obsahuje bootloader, který umožňuje nahrát nový kód bez použití externího programátoru. Další informace lze nalézt v [16].

Arduino FIO

Arduino FIO je založeno na mikrokontroléru ATmega328P. Deska obsahuje 14 digitálních vstupů a výstupů, kde 6 může být použito jako PWM výstup, 8 analogových vstupů, 8 MHz krystalový oscilátor a tlačítko pro reset. Deska neobsahuje předpájené piny a je tedy možno použít různé typy konektorů nebo přímé pájení drátů. Deska také obsahuje konektor pro Lithium Polymer baterii a XBee patici na zadní straně desky. Na obrázku 2.3 lze vidět Arduino FIO. Arduino Fio navrhl Shigeru Kobayashi a firma SparkFun Electronics, která je též výrobcem.



Obrázek 2.3: Arduino FIO.

Arduino FIO je primárně uzpůsobeno pro bezdrátové aplikace a předpokládá se, že bude napájeno pomocí baterie. Deska obsahuje USB port, který ale slouží pouze pro nabíjení připojené baterie. Ke komunikaci s počítačem musí být použit převodník z UART[4] na USB rozhraní. Jako převodník slouží FTDI[5] kabel nebo například S*parkfun FTDI breakout board*. Další možností je Arduino FIO naprogramovat bezdrátově pomocí USB-to-XBee adaptéru jako je *Sparkfun XBee Explorer USB*.

Arduino FIO může být napájeno pomocí FTDI kabelu, zdroje 3,3 V napětí připojeného na 3V3 pin a nebo pomocí Lithium Polymer baterie připojení na BAT pin.

[4] Universal Asynchronous Receiver and Transmitter

[5] Future Technology Devices International – výrobce převodníku z jiných rozhraních na USB rozhraní

Každý ze 14 digitálních pinů může být použit jako vstup nebo výstup. Zda je pin výstupní nebo vstupní se určí při běhu programu pomocí funkce `pinMode()`. Pomocí funkcí `digitalWrite()` a `digitalRead()` probíhá zápis a čtení hodnot z pinů. Každý pin operuje na 3,3 V a může přijímat nebo poskytovat 40 mA. Také obsahují interní pull-up rezistor s rozmezím 20-50 kOhm. Tyto rezistory standardně nejsou připojeny. Některé výstupy mají specializované funkce:

Sériové rozhraní Pin 1 (TX0) se používá k odeslání a pin 0 (RXI) k příjmu TTL sériových dat. Tyto piny jsou připojeny k DOUT a DIN pinům XBee modemu.

Externí přerušení Piny 2 a 3 mohou být využity k přerušení při nízké hodnotě, náběžné nebo klesající hraně a při změně hodnoty. K tomuto účelu slouží funkce `attachInterrupt()`.

PWM Piny 3, 5, 6, 9, 10 a 11 umožňují 8 bitový PWM výstup za pomocí funkce `analogWrite()`.

SPI Piny 10 (SS), 11 (MOSI), 12 (MISO) a 13 (SCK) podporují SPI komunikaci pomocí knihovny *SPI library*.

LED Pin 13 obsahuje vestavěnou LED diodu. Při vysoké hodnotě je dioda rozsvícena a při nízké hodnotě dioda nesvítí.

I^2C: Piny A4 (SDA) a A5 (SCL) podporují TWI komunikaci pomocí knihovny *Wire library*.

AREF Pin pro referenční napětí pro analogové vstupy.

DTR Slouží k restartování mikrokontroléru.

Na desce se také nachází 8 analogových vstupů označených jako A0–A7. Každý z pinů je 10 bitový a měří napětí od 0 do 3,3V. Deska také obsahuje dalších 8 nepřipájených pinů:

BAT Jedná se o piny BAT+ a BAT-, které se typicky připojí k baterii.

CHG Pin CHG 5V a CHG- pro napájecí stanice.

SW Pro připojení napájecího přepínače.

CTS a DTR Piny se využívají k ovládání úsporného režimu XBee modemu.

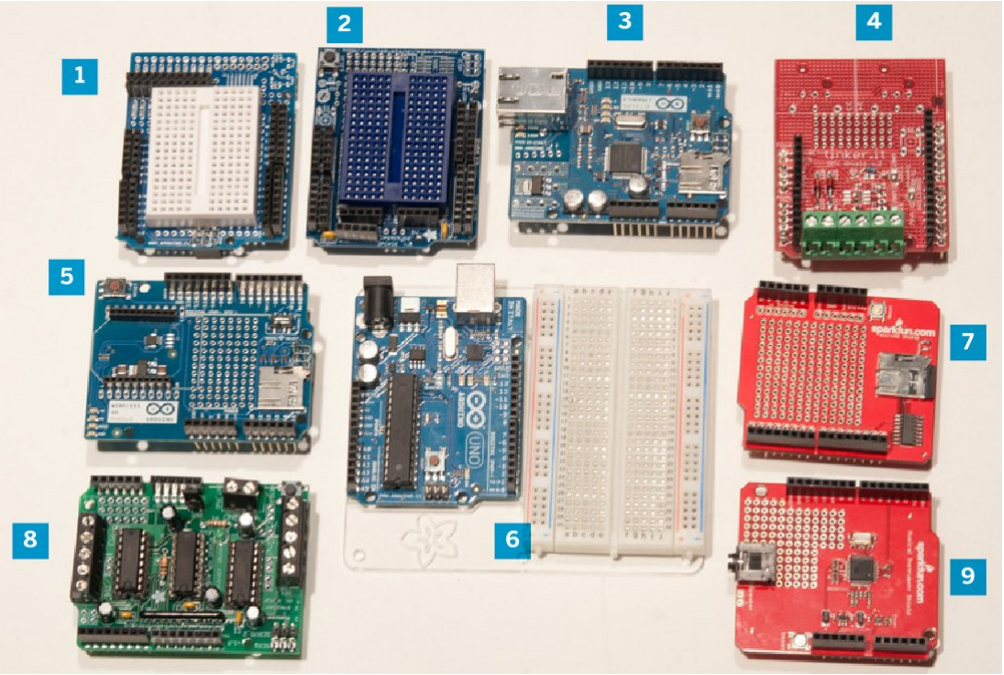
Další informace lze nalézt v [14].

2.1.2 Přídavné desky pro Arduino

Další z vlastností desek Arduina je možnost rozšíření funkčností pomocí tzv. „shields". Pomocí těchto rozšiřujících desek je možno přidat například komunikaci přes LAN, WiFi, Bluetooth, XBee, připojení microSD karty a další.

Na obrázku 2.4 jsou zobrazeny různé druhy rozšiřujících desek pro Arduino: 1. Arduino prototyping shield 2. Adafruit prototyping shield 3. Arduino Ethernet shield 4. TinkerKit DMX shield 5. Arduino wireless shield 6. Oomlout Arduino/breadboard mount vyrobeno v Adafruit 7. Spark Fun microSD card shield 8. Adafruit motor driver shield 9. Spark Fun musical instrument shield. O využití těchto rozšiřujících modulů se můžete dočíst v [4].

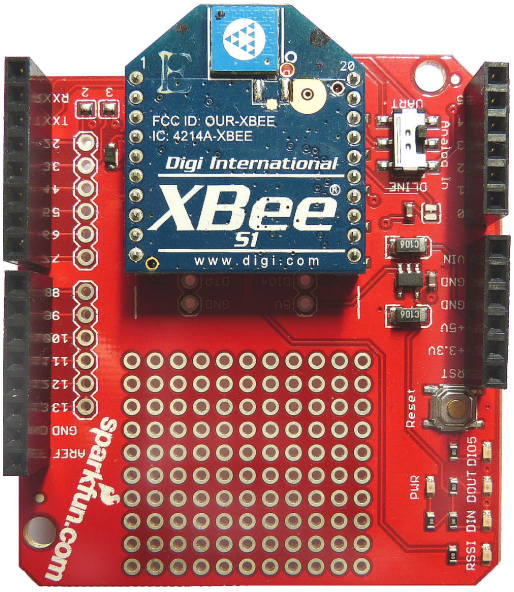
Počet těchto rozšiřujících desek přesahuje 200 a jejich seznam můžete nalézt na [8]. Některé rozšiřující desky je možno spojovat a využívat více těchto desek zároveň.



Obrázek 2.4: Rozšiřující desky pro Arduino. Převzato z [4].

SparkFun XBee Shield

Toto rozšíření zjednodušuje propojení XBee modulu s Arduino deskou. Na obrázku 2.5 je ukázka připojeného shield s XBee modulem.



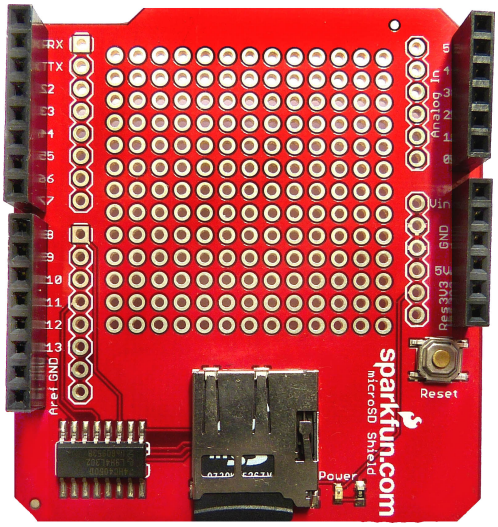
Obrázek 2.5: SparkFun XBee shield.

Tato deska je určena pro Arduino Pro a Arduino Uno. K tomuto rozšíření je možno připojit XBee moduly série 1 a 2, standardní a Pro verze. Sériové piny DIN a DOUT XBee modulu jsou propojeny přes SPDT přepínač, který umožňuje vybrat spojení pomocí pinů D0, D1 nebo D2, D3.

Shield se také stará o změnu napájení z 5 V na 3,3 V. Deska také obsahuje LED diody o indikaci aktivity XBee modulu. Další informace o SparkFun XBee Shield v [39].

MicroSD Shield

Tento shield rozšiřuje desku Arduino o možnost připojení microSD karty přes SPI rozhraní. Deska obsahuje převodník z 5V na 3.3V, aby nedošlo k poškození microSD karty. Piny microSD patice SCK, DI a DO jsou připojeny k SPI pinům. Jedná se o digitální porty 11-13 na desce Arduino. Pin CS je připojen k Arduino pinu D8. S pomocí open source FAT knihoven lze na kartě vytvářet soubory, zapisovat a číst data. Většina knihoven předpokládá, že CS pin je připojen na pin D10 a je tedy nutné jej změnit na D8. Na desce se také nachází LED dioda pro stav napájení a tlačítko na reset. Na obrázku 2.6 je lze vidět microSD shield. [30]



Obrázek 2.6: MicroSD Shield.

2.1.3 Vývoj

Samotný vývoj probíhá pomocí Arduino IDE, které je založeno na vývojovém prostředí pro programovací jazyk Processing a projektu Wiring [42]. Na obrázku 2.7 je zobrazeno samotné vývojové prostředí, které obsahuje textový editor pro psaní kódu, textovou konzoli pro zobrazení zpráv nebo chybových hlášení, panel nástrojů s tlačítky pro základní funkčnost a položky menu. Arduino IDE také slouží k nahrávání programů do Arduino hardware a k následné komunikaci.

Software napsaný v Arduino IDE se nazývá „sketch". Příklad takového jednoduché sketch můžete vidět v příkladu 2.1. Součástí každého arduino sketch jsou dvě speciální funkce *setup()* a *loop()*.

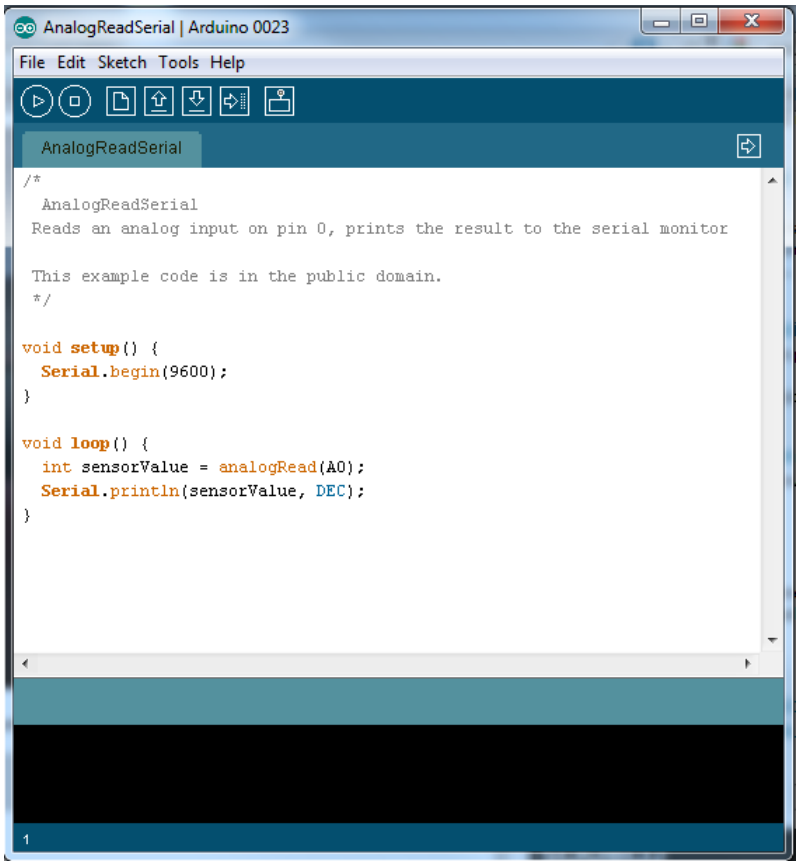
Kód 2.1: Příklad kodu pro arduino.

```
/*
   Priklad kodu
*/
 void setup() {
  Serial.begin(9600);
}

void loop() {
```

```
  Serial.println("Ahoj svete.");
}
```

Funkce *setup()* je volána při startu programu. Slouží k inicializaci proměnných, knihoven a nastavení pinů pomocí funkce *pinMode()*. Funkce se spustí pouze jednou po každém zapnutí nebo resetování desky Arduino. Funkce loop() slouží jako hlavní smyčka programu, kde probíhá volání funkcí, načítání hodnot ze vstupů atd. Tyto funkce musejí být zahrnuty do každého sketch, i když nebudou zapotřebí. [24]

Proces sestavení programu pro desku Arduino se skládá z několika kroků. Prvně je sketch předzpracován a převeden do C++ programu. Převedený kód je přeložen překladačem (avr-gcc), který převede zdrojový kód do strojového kódu. Strojový kód je následně sestaven se standardními knihovnami pro Arduino, které poskytují základní funkce. Výsledkem sestavení je hex soubor, který je následně nahrán do Arduino desky. Programování Arduina probíhá pomocí USB nebo sériového portu přes bootloader, který se nachází na čipu nebo přes externí programovací hardware. [17]



Obrázek 2.7: Arduino IDE.

2.2 XBee

XBee je produktové označení pro bezdrátové rádiové moduly firmy Digi International. Tyto moduly se zakládají na standardech 802.15.4 a ZigBee pro vytváření bezdrátových sítí. Firma Digi International nabízí více druhů produktů XBee, které se liší hardwarem, obsaženým protokolem, vysílací frekvencí, vysílacím výkonem a druhem antény. Moduly XBee se dělí na sérii 1 a 2. Každá

z těchto sérií obsahuje verze standard a PRO. V následujících částech bude následovat popis těchto rozdělení.

2.2.1 Série XBee modulů

XBee moduly série 1 využívají mikročip firmy Freescale. Tyto moduly mohou obsahovat firmware, který je založen na standardu 802.15.4, ale také proprietární protokol DigiMesh. Firmware založený na standardu 802.15.4 umožňuje point-to-point nebo point-to-multypoint komunikaci. Protokol DigiMesh je popsán v části 2.2.3. Série 1 je více zaměřena na nahrazení fyzických spojení a na projekty menší velikosti.

Moduly série 2 využívají mikročip firmy Ember Networks a obsahují firmware založený na standardu ZigBee. Moduly s tímto firmware mohou vytvářet robustní senzorové sítě s mesh topologií. Hardware série 2 má lepší vysílací dosah a využívá méně energie, ale tento rozdíl není příliš velký.

Moduly série 1 a 2 mohou být v nasazených systémech zaměněny často pouze s menšími úpravami software. Samotné moduly verze 1 a 2 nemohou mezi sebou komunikovat. Tabulka 2.2 zobrazuje vlastnosti a rozdíly mezi verzemi.

Tabulka 2.2: Vlastnosti standardních modulů XBee série 1 a 2. Převzato z [3].

Xbee	**Série 1**	**Série 2**
Typický dosah	30 m	40 m
Nejlepší dosah	100 m	120 m
Proud vysílání/příjem	45/50 mA	40/40 mA
Firmware	802.15.4 point-to-point	ZB ZigBee mesh
Digitální vstup/výstup piny	8 (plus 1 pouze vstupní)	11
Analogové vstupy	7	4
Analogové výstupy	2	0
Úspora energie, adresovatelné	ano	ano
Vytváření mesh sítí	ano s DigiMesh	ano
Point-to-point	ano	ano
Jediný firmware pro všechny uzly	ano	ne
Je potřeba koordinátora	ne	ano
Point-to-point konfigurace	jednoduchá	složitější
Chipset	Freescale	Ember
Dostupný firmware	802.15.4, DigiMesh	ZB
Stále podporovaný	ano	ano

Verze XBee modulů

Samotné série 1 a 2 se také dělí na verzi standard a PRO. Tyto verze se liší vysílacím výkonem a dosahem. Příkladem standardní verze je modul XBee 802.15.4 a pro verzi PRO je XBee-PRO 802.15.4. Verze PRO je také rozměrově větší než standardní verze. Oba tyto moduly patří do série 1. Na obrázku 2.8 je ukázka těchto modulů a v tabulce 2.3 je uveden rozdíl mezi různými moduly a verzemi.

Obrázek 2.8: XBee moduly pro a standard.

Tabulka 2.3: XBee standard a PRO.

Verze XBee	Frekvence	Vysílací výkon	Maximální dosah	Přenosová rychlost	Protokol
XBee 802.15.4	2,4 GHz	1 mW	90 m	250 Kbps	802.15.4
XBee-PRO 802.15.4	2,4 GHz	63 mW	1.6 km	250 Kbps	802.15.4
XBee DigiMesh 2.4	2.4 GHz	1 mW	90 m	250 Kbps	DigiMesh
XBee-PRO DigiMesh 2.4	2.4 GHz	63 mW	1.6 km	250 Kbps	DigiMesh
XBee ZB	2.4 GHz	1.25/2 mW	120 m	250 Kbps	ZigBee
XBee-PRO ZB	2.4 GHz	63 mW	3.2 km	250 Kbps	ZigBee

2.2.2 Druhy antén XBee modulů

XBee moduly potřebují k odesílání a příjímání signálu anténu. Firma Digi nabízí XBee moduly, které mají rozlišné antény a konektory pro připojení externích antén. Každá z těchto variant má své výhody a nevýhody. Na obrázku 2.9 jsou zobrazeny jednotlivé druhy antén a konektorů.

Drátová anténa

Jedná se o ohebnou anténu, která vystupuje z těla modulu. Ve většině případů drátová anténa plně dostačuje a nabízí všesměrové záření. Maximální přenosová vzdálenost je téměř stejná ve všech směrech, když jeho vodič je rovný a kolmý k XBee modulu.

Čipová anténa

Je plochý keramický čip, který je v jedné rovině s XBee modulem. Čip je menší a odolnější, ale tato anténa má i své nevýhody. Chip antény je cardioid (má tvar srdce) vyzařovací rádius, což znamená, že signál je oslabený v mnoha směrech. Vhodná je v systémech, kde můžeme čekat mechanické

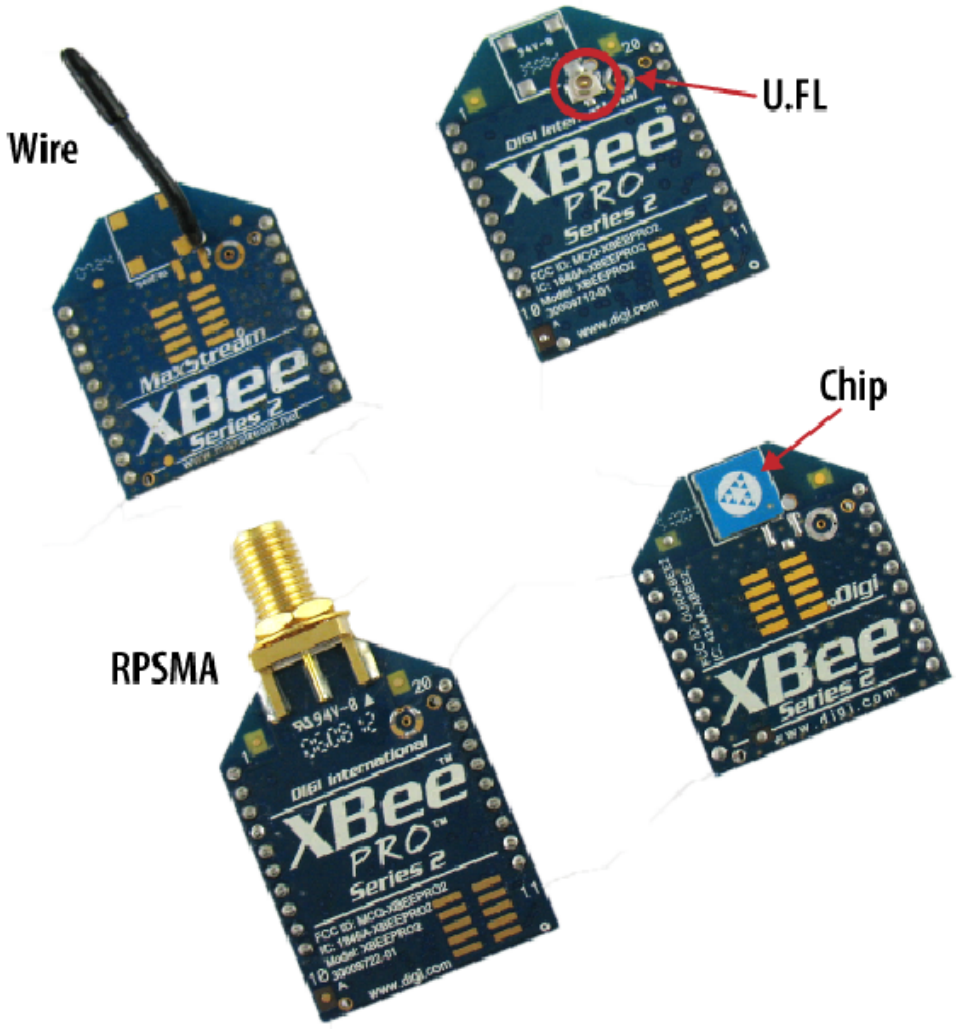
namáhání a kde by se mohl vodič antény zlomit. Díky malé velikosti může být umístěn do těsných prostor. Čipová anténa je často správnou volbou pro nositelná zařízení.

U.FL konektor

Jedná se o menší ze dvou typů konektorů pro externí antény. Tento typ antény je vhodné použít v místech, kde bude modul stíněný, například bude uložen v plechové krabičce. Také je zapotřebí orientovat anténu jinak než při použití drátěné nebo čipové antény. U.FL konektor je malý, poněkud křehký a téměř vždy s krátkým spojovací kabelem, který přenáší signál od antény.

RPSMA konektor

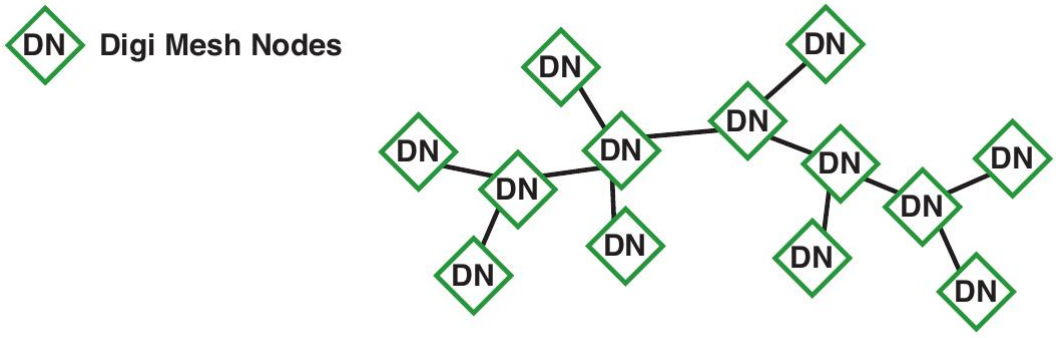
V tomto případě se jedná se o větší a objemnější druh U.FL konektoru. Externí anténa může být montována přímo na XBee modul bez propojovacího kabelu.



Obrázek 2.9: Antény a konektory modulů XBee. Převzato z [3].

2.2.3 DigiMesh

DigiMesh je proprietární komunikační standard pro vytváření mesh sítí. Tento standard je patentován a vyvinutý firmou Digi International. Oproti ZigBee přináší rozšířené funkce jako je například možnost celou síť uspat, schopnost samoopravy při výpadku v síti, má větší průchodnost dat a také větší dosah než ZigBee. Síť DigiMesh je založena na peer-to-peer síťové topologii, kde je síť složena pouze z jednoho typu uzlu. Díky homogenní struktuře sítě je pak DigiMesh výrazně jednodušší na funkčnost pomocí jednoho typu uzlu, které zároveň pracují jako směrovače i jako koncové uzly. Adresování uzlů v síti je pomocí 64 bitové MAC adresy a díky peer-to-peer komunikaci může mít síť libovolnou strukturu. Příklad DigiMesh sítě můžete vidět na obrázku 2.10.



Obrázek 2.10: Příklad DigiMesh sítě. Převzato z [37].

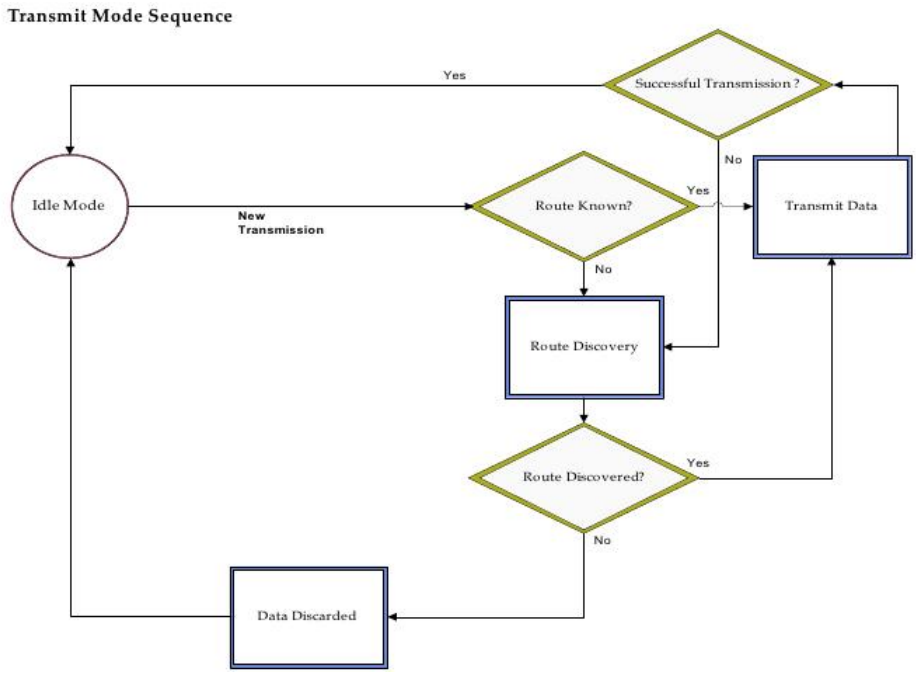
Základní vlastnosti DigiMesh zahrnují samo-opravu sítě při výpadku uzlu nebo připojení do sítě, architektura peer-to-peer, vyhledávání trasy, kdy je trasa vytvářena v případě potřeby. Selektivní potvrzování, kdy pouze cílová stanice informuje o doručení, z čehož vyplývá spolehlivost doručení zpráv. V poslední řadě je režim spánku, při kterém je možné uspat všechny uzly v síti, aniž by došlo ke ztrátě informací.

DigiMesh síť podporuje unicast a broadcast adresování. Při unicast je využito opakování přenosu s potvrzením přijetí pro spolehlivé doručení paketů. Počet opakování je určeno parametrem NR (Network Retries). Datové pakety jsou vysílány až NR+1 krát a při přijetí paketu je vyslán ACK paket. Pokud ACK paket nedorazí za dobu, kterou by paket potřeboval na průchod sítí dvakrát, dojde k opětovnému odeslání paketu. V broadcast adresování jsou pakety přijaty všemi uzly a znovu odeslány všemi uzly v síti. V tomto režimu není využito potvrzování ACK a tady vysílací uzel odesílá data čtyřikrát. To má za následek, že všechny uzly opakují přenos také čtyřikrát a zde kvůli kolizím se musí využit náhodného zpoždění vloženého před přeposláním broadcast zprávy. Sledování paketu chrání každý uzel před dvojnásobným rozesíláním broadcast zpráv. Broadcast by neměl být často využíván z důvodu snadného zahlcení sítě.

Pro směrování dat v DigMesh síti se využívá směrovací tabulky a směrovacího protokolu, který využívá reaktivní metodu odvozenou od metody AODV[6]. Asociativní směrovací tabulka je využita k mapování adresy cílového uzlu na uzel souseda, což může být cílový uzel nebo přechodný uzel, který přepošle zprávu na jiný uzel. Jestliže uzel nezná cestu k cílovému uzlu, je paket zařazen do fronty a čeká na RD[7]. RD proces také zahájí uzel, který odeslal paket a nedostal paket ACK. RD

[6] Ad-hoc On-demand Distance Vector
[7] Route Discovery – vyhledávání cesty

tedy začíná u tohoto uzlu odesláním broadcast žádosti RREQ[8]. Každý uzel, který žádost obdrží a není koncovým uzlem, se stane přechodným uzlem. Tyto uzly mohou RREQ žádost zahodit nebo přeposlat. Jestliže RREQ obsahuje lepší cestu zpět k uzlu, který ji odeslal, je cesta aktualizována spolu s RREQ, které je následně přeposláno pomocí broadcast. Jakmile cílový uzel obdrží RREQ, je odeslána odpověď RREP[9] zpět ke zdroji po stejné trase, odkud k němu přišla. Odpověď RREP je odeslána bez ohledu na kvalitu cesty a na počtu dříve zaslaných RREQ. Zdrojový uzel si na základě přijatých RREP může vybrat nejlepší cestu k cílovému uzlu. Pakety čekající ve frontě jsou následně odeslány. Na obrázku 2.11 je zobrazen diagram pro odeslání paketu a RD.



Obrázek 2.11: Diagram RD. Převzato z [40].

Další z vlastností DigiMesh je možnost synchronizovat časy spánku a probuzení pro všechny uzly v síti. Z tohoto důvodu je DigiMesh velmi vhodný i pro sítě složené z velké části bateriově napájených prvků. Všechny synchronizované uzly v síti ve stejný čas vstoupí do režimu spánku a ve stejný čas z něj vystoupí. Tento typ sítě je označován jako cyklická spící síť. Uzly se synchronizují přijmutím speciálního synchronizačního RF paketu zaslaného uzlem, který koordinuje spaní. Koordinátorem spaní se může stát uzel pomocí procesu nominace. Koordinátor spaní vyšle jednu synchronizační zprávu na začátku každého probuzení jako broadcast paket. Čas spánku a probuzení pro celou síť lze změnit lokálně změnou nastavení na jediném uzlu. Síť použije nejnovější nastavení spánku.

Uzly v síti mohou být nakonfigurovány do tří režimů provozu:

SM0 – Standardní mód, který není kompatibilní s režimem spánku.

SM7 – Mód podpory spánku, kompatibilní s režimem spánku.

SM8 – Cyklický spící nízkopříkonový režim, kompatibilní s režimem spánku.

[8] Route REQuest
[9] Route REPly

Ve většině případů by se síť měla skládat pouze z uzlů působících ve stejném režimu spánku (pouze SM0 uzly nebo jen SM8 a SM7 uzly). Směrování a hledání cesty v sítí je nekompatibilní mezi uzly se standardním módem a uzly kompatibilní s režimem spánku.

Normální režim SM0

Normální režim je výchozí režim pro nově zapnuté uzly. V tomto režimu uzel nebude spát, ale bude synchronizován se spící sítí. Uzel bude přeposílat synchronizační zprávy vytvořené uzly kompatibilní s režimem spánku, ale nebude je moci vytvářet. Jakmile je uzel synchronizován se spící sítí, může být kdykoliv uveden do kompatibilního režimu spánku.

Cyklický režim spánku SM8

V cyklickém režimu spánku uzel spí po dobu naprogramovaného času a probudí se spolu s ostatními uzly. Vymění si data a synchronizační zprávy a následně se vrátí ke spánku. Spící uzel nemůže přijímat RF zprávy nebo číst příkazy z UART portu. Časy spánku ST a probuzení SP jsou specifikovány koordinátorem spánku. Nesynchronizovaný nově připojený uzel bude čekat na synchronizační zprávu a spát po dobu SP. Tento cyklus se bude opakovat do doby, než obdrží synchronizační zprávu. Jakmile uzel obdrží synchronizační zprávu, synchronizuje se se sítí.

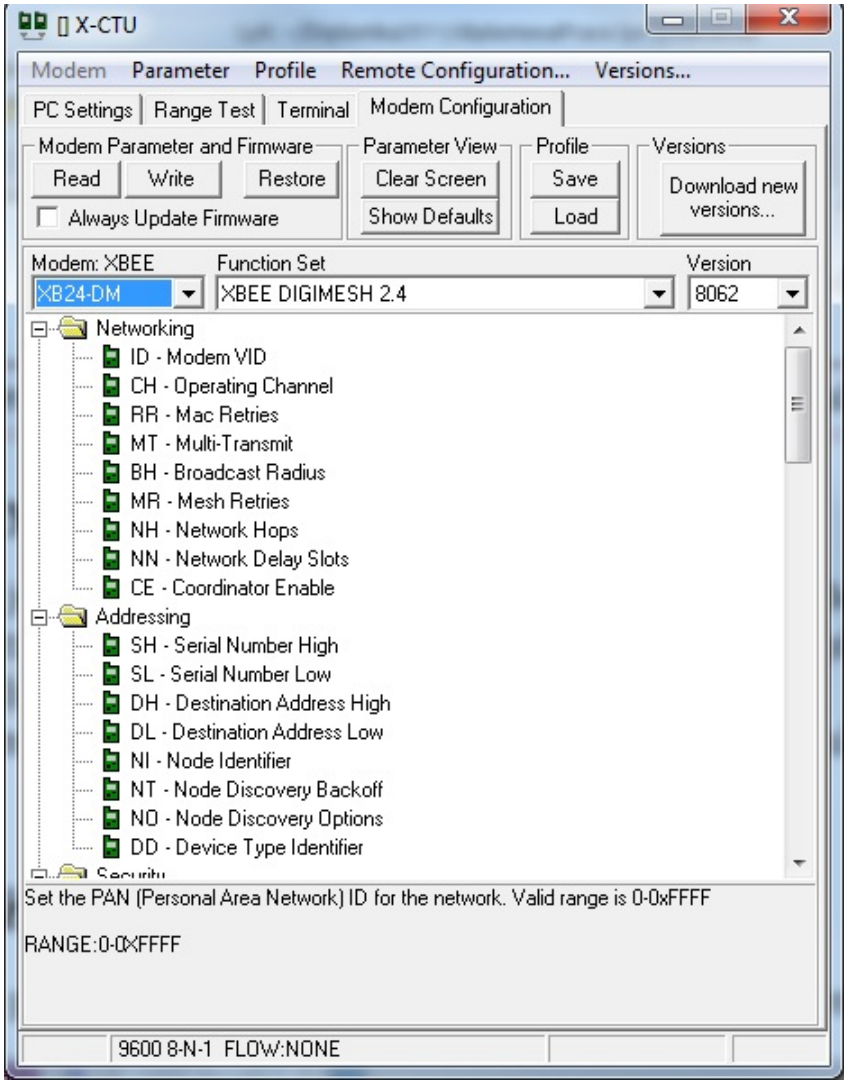
Mód podpory režimu spánku SM7

Uzel v tomto režimu se synchronizuje se spící sítí, ale nebude spát. Uzel bude odpovídat synchronizační zprávou novým uzlům v dosahu, které se snaží připojit do spící sítě. Uzly s podporou spánku jsou užitečné zejména při použití jako koordinátor spánku.

2.2.4 Programování modulů XBee

Pro nastavení XBee modulu je potřeba jej propojit s počítačem například pomocí XBee Explorer USB viz 2.3.3. Pomocí terminálového programu je nutno se připojit k portu, na kterém je modul připojen. Po odeslání znaků „+++" se modul přepne do příkazového módu. V tomto módu modul setrvá po dobu 10 sekund od posledního zadaného příkazu. [4]

Další možností je využít řešení od firmy Digi, která dodává program X-CTU pro snadnější nastavení a otestování modulů XBee. Jedná se o aplikaci běžící pod operačním systémem Windows. Program X-CTU můžete vidět na obrázku 2.12. Program automaticky detekuje typ XBee modulu a jakou verzi firmware obsahuje. Všechny parametry, které lze nastavit, jsou vypsány v záložce „Modem Configuration". V záložce „Range Test" umožňuje spustit test na určení vzdálenosti mezi dvěma moduly. X-CTU také slouží k aktualizaci a změně firmware XBee modulů. [41]



Obrázek 2.12: X-CTU.

2.2.5 Rozdíly DigiMesh a ZigBee

Tyto bezdrátové komunikační standardy jsou založeny na standardu IEEE 802.15.4 a vytvářejí mesh sítě, ale mají mezi sebou několik rozdílů, které jsou uvedeny v tabulce 2.4. Standard ZigBee je otevřený a je tedy možnost propojení se zařízeními jiných výrobců. Sítě ZigBee obsahují 3 druhy uzlů a to koordinátora, směrovače a koncové uzly. Kromě toho ZigBee nabízí zavedené profily pro běžné aplikace jako je správa energie a řízení osvětlení. Také podpora nástrojů pro diagnostiku sítě je větší.

DigiMesh je proprietární protokol, který je ale dostupný pro platformy s větším dosahem a propustností dat. Díky zjednodušenému adresování je jednoduší sestavit bezdrátovou síť, která je i více flexibilní.

Jak bylo řečeno v 2.2.3 XBee moduly série 1 mohou obsahovat firmware 802.15.4 a DigiMesh. V nejnovějších modulech může být tento firmware libovolně zaměňován. Více informací můžete najít v [33, 37, 35, 40].

Tabulka 2.4: Rozdíly DigiMesh a ZigBee. Tabulka převzata z [37].

	ZigBee	**DigiMesh**
Typy uzlů a výhody	Koordinátor, směrovače, koncové uzly. Koncová zařízení mohou být levnější.	Jeden typ uzlu. Více flexibility při rozšiřování sítě. Možnost somoopravy sítě.
Spící režim, bateriové napájení	Pouze koncové uzly	Všechny uzly lze upsat bez ztráty dat.
Bezdrátový aktualizace firmwaru	ano	ne
Zvětšení dosahu	Většina zařízení má max. dosah 3.2km	V XTend verzi až 64 km
Velikost datové části paketu	až 80 bajtů	až 256 bajtů
Podporované frekvence a dat. tok	2.4 GHz (250 kb/s). Dále pak 900 MHz (40 kb/s) a 868 MHz (20 kb/s)	900 MHz (10, 125, 150 kb/s) nebo 2.4 GHz (250 kb/s)
Bezpečnost	AES šifrování. Lze zamknout síť proti přidání dalších uzlů.	AES šifrování.
Součinnost	Otevřený standard. Možná komunikace zařízení různých výrobců.	Je patentováno a není otevřeno
Odolnost proti interferencím	Direct-Sequence Spread Spectrum (DSSS)	Pásmo 900 MHz: Frequency-Hopping Spread Spectrum (FHSS) Pásmo 2.4GHz: Direct-Sequence Spread Spectrum (DSSS).
Adresování	MAC adresa (64 bitů) a síťová adresa (16 bitů)	MAC adresa (64 bitů)
Údržba	Diagnostické nástroje.	Jednodušší adresování může pomoct při diagnostice problému a nastavené sítě.

2.3 Senzory a další použitý hardware

V této části bude obsažen popis dalších použitých hardware využitých při návrhu a konstrukci senzorové bezdrátové sítě.

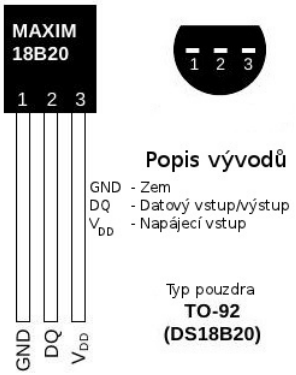
2.3.1 DS18B20

DS18B20 je polovodičové čidlo od firmy MAXIM, které je koncipováno jako monolitický senzor, který umožňuje jednoduché měření teplot s nastavením rozlišení. Senzor komunikuje pomocí 1-Wire sběrnice, která ze své podstaty vyžaduje pouze jednu datovou linku a zem pro komunikaci s mikroprocesorem. Každý senzor vlastní unikátní 64-bitové sériové číslo uložené v paměti ROM, jenž slouží jako adresa. Lze tedy připojit velké množství senzorů k jediné sběrnici a připojený mikroprocesor může ovládat více teploměrů rozmístěných na větší ploše. K aplikacím, které mohou těžit z této funkce patří například systémy monitorování teploty uvnitř domů, skleníků, sklepů,

skladů a jiných objektů a nebo mohou být součásti nejrůznějších zařízení. Na obrázku 2.13 je zobrazen fyzický vzhled a přehled vývodů. [22]

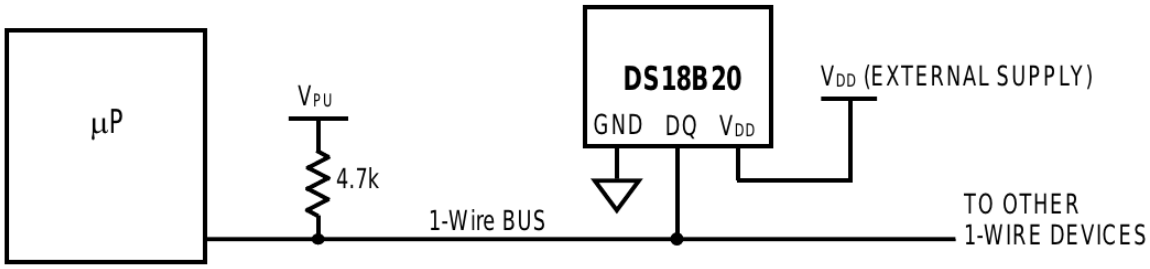
Základní vlastnosti čidla:

- Komunikace pomocí sběrnice 1-Wire, která vyžaduje pouze jediný konektor.
- Nevyžaduje žádné externí komponenty.
- Teplotní rozsah senzoru -55 °C do +125 °C.
- Chyba měření ±0.5 °C v rozsahu -10 °C do +85 °C.
- Napájecí napětí v rozsahu 3 – 5,5 V.
- Převod teploty na 12-bitové digitální slovo za maximálně 750 ms.
- Rozlišení měření lze nastavit od 9-ti do 12-ti bitů.



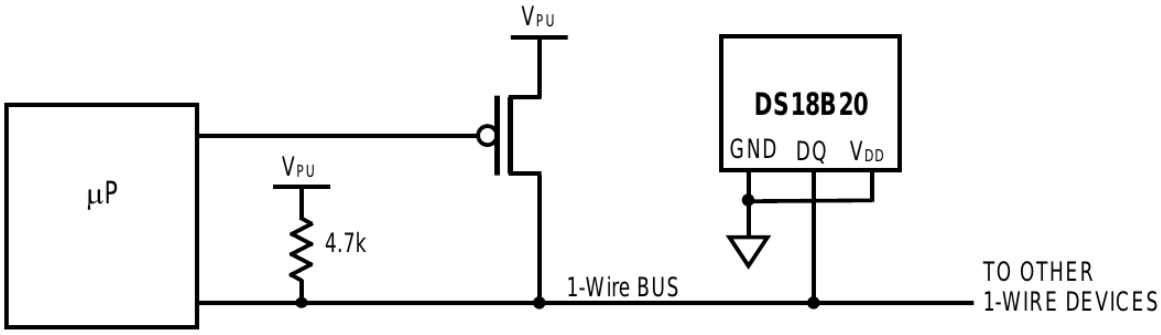
Obrázek 2.13: Popis vývodů DS18B20. Převzato z [22].

Napájení DS18B20 je možné řešit dvěma způsoby. Prvním ze způsobů je napájení z externího zdroje přivedeného na vývod V_{DD} viz. obrázek 2.14.



Obrázek 2.14: Schéma zapojení při externím napájení.Převzato z [22].

Další možnost je použít tzv. parazitní napájení, kdy spojíme-li vývody GND a V_{DD} a energie bude pro provoz senzoru odebírána z 1-wire sběrnice přes zdvihací rezistor viz obrázek 2.15. Při napětí na sběrnici se nabíjí kondenzátor C_{pp}(je umístěn v DS18B20) a při nulovém napětí se tedy stane zdrojem pro senzor. Výhodou tohoto napájení je to, že se při zapojení velkého počtu senzorů použijí pouze dva vodiče. Nevýhodou tohoto zapojení je snížení teplotního rozsahu na hodnotu maximálně 100 °C .



Obrázek 2.15: Schéma zapojení při parazitním napájení.Převzato z [22].

2.3.2 RHT03

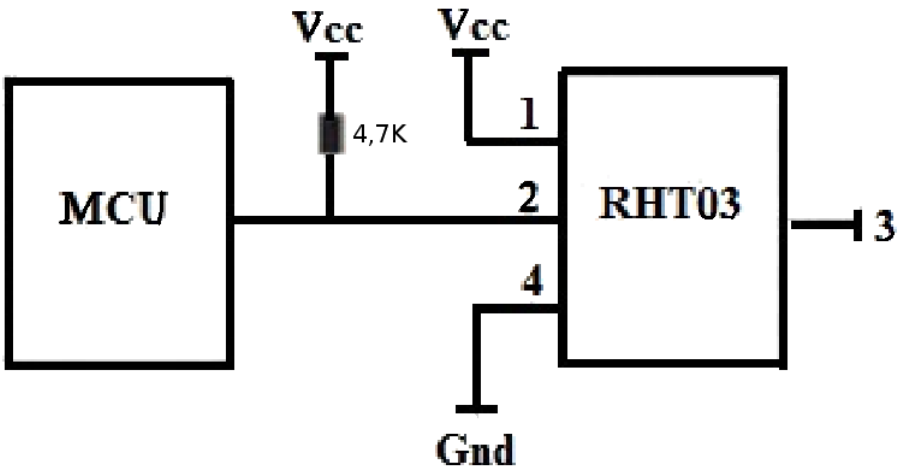
Tento senzor je vyráběn firmou MaxDetect a obsahuje teplotní a vlhkostní senzor v jediném pouzdře. Tento senzor používá vlastní implementaci sběrnice 1-Wire a není tedy kompatibilní se sběrnicí 1-Wire od firmy Maxin/Dallas. Každý senzor je kalibrován a není tedy potřeba žádných externích součástek. Kalibrační koeficient je uložen v paměti senzoru a při měření fyzikálních veličin se využije tohoto koeficientu. Na obrázku 2.16 můžete vidět senzor RHT03 a jeho přehled vývodů. [6] Tento senzor má také starší označení pod názvem DHT-22. [21]

Základní vlastnosti čidla:

- Komunikace pomocí sběrnice 1-Wire, která vyžaduje pouze jediný konektor.
- Nevyžaduje žádné externí komponenty.
- Teplotní rozsah senzoru -40 °C do +80 °C.
- Vlhkostní rozsah senzoru 0-100%.
- Chyba měření ±0.5°C a ±2% vlhkosti.
- Napájecí napětí v rozsahu 3,3 – 6 V.

Pin	Funkce
1	VDD - napájecí vstup
2	DATA
3	NULL
4	GND

Obrázek 2.16: RHT03.

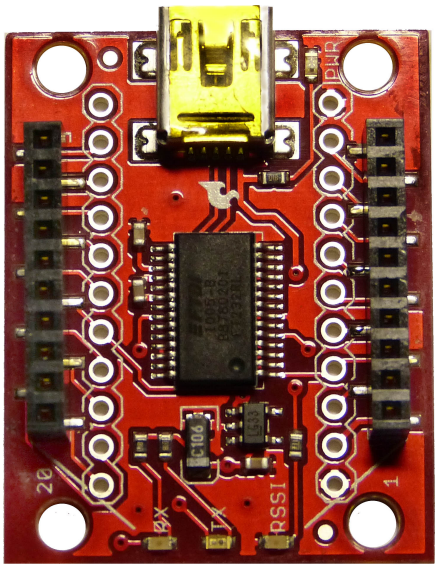


Obrázek 2.17: Schéma zapojení RHT03. Převzato z [6].

Napájení RHT03 je řešeno pomocí napájení z externího zdroje přivedeného na vývod V_{DD} viz. obrázek 2.17.

2.3.3 XBee Explorer USB

Zařízení XBee Explorer slouží k přímému propojení sériových a programovacích pinů pomocí USB k počítači. Zařízení podporuje všechny typy XBee modulů včetně série 1 a 2, standardní a Pro verze. Zařízení vyrábí firma Sparkfun a je zobrazeno na obrázku 2.18. [38]

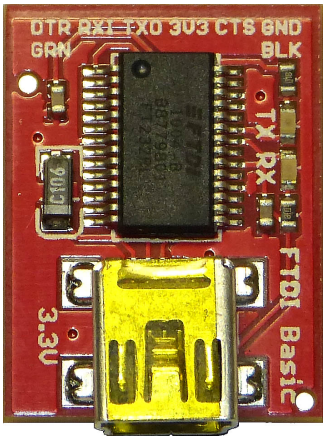


Obrázek 2.18: XBee Explorer USB.

2.3.4 FTDI Basic

Jedná se o plošnou desku obsahující FTDI FT232RL čip, který umožňuje převod USB na UART rozhraní. Deska se tedy dá využít k naprogramování Arduino FIO viz 2.1.1. Hlavní rozdíl u této

desky je, že spojí DTR pin s pinem RTS na kabelu FTDI. DTR pin slouží k automatickému resetování desky při nahrání nového programu. Tato deska byla navržena tak, aby snížila náklady na vývoj a zvýšila jednoduchost použití desky Arduino. Plošná deska je vyráběna firmou Sparkfun a je zobrazena na obrázku 2.19. [27]



Obrázek 2.19: FTDI Basic Breakout.

Kapitola 3

Návrh a použité technologie

V této kapitole bude popsán návrh zapojení jednotlivých senzorových uzlů a aplikace pro zobrazení naměřených dat. Správný návrh je při tvorbě každého projektu velmi důležitý, protože s případnými chybami se budeme potýkat po celou dobu vývoje projektu. Správný návrh je tedy klíčový pro zdárný vývoj celé aplikace. Tato kapitola se zaměří na další použité technologie, které jsem využil při návrhu a implementaci aplikace.

V sekci 3.1 je popsán návrh samotné sítě, její konfigurace a obsahuje popis zapojení jednotlivých uzlů v síti. V další sekci 3.2 je rozebrána hlavní funkčnost aplikace a návrh základního rozhraní a použité implementační technologie. Z tohoto návrhu vychází sekce 3.3, ve které je popsán návrh databáze a zvolení vhodného databázového systému. Následující sekce 3.4 se zabývá návrhem komunikačního protokolu. Poslední sekce 3.5 obsahuje volbu technologií pro zobrazení naměřených hodnot.

3.1 Senzorová sít

Návrh senzorové bezdrátové sítě vychází ze zadání této práce s využitím platformy Arduino a modulů XBee se zaměřením na vytváření dynamických mesh sítí. Moduly XBee podporují vytváření mesh sítí v sérii 1 s firmware DigiMesh a sérii 2 s firmware ZigBee mesh viz 2.2. Z důvodu nekompatibility jednotlivých sérií musí být vybrána pouze jediná verze a firmware. V tabulce 2.4 jsou zobrazeny rozdíly mezi DigiMesh a ZigBee. Jako nejvýhodnější se tedy jeví využít standard DigiMesh, který lépe podporuje vytváření dynamických mesh sítí. Jeho další výhodou je jednodušší adresování uzlů v síti a možnost samo-opravy při výpadku nebo připojení nového uzlu.

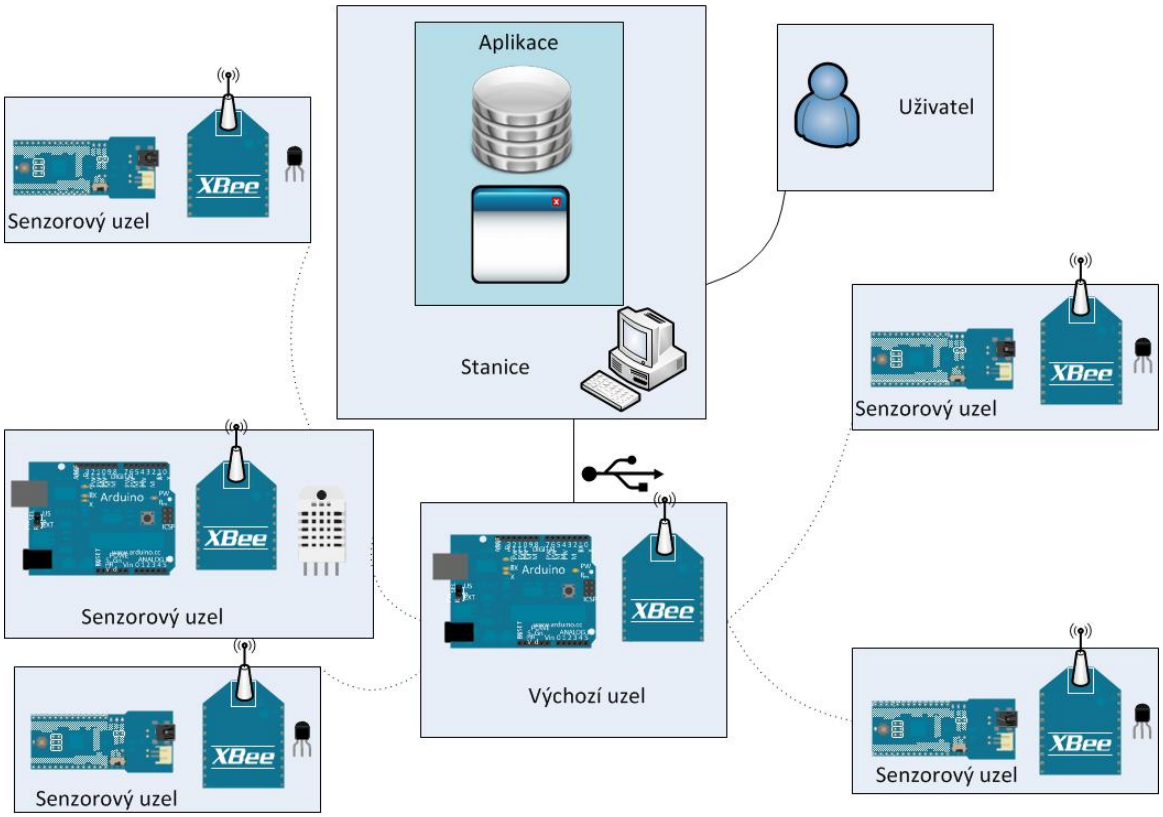
Na obrázku 3.1 můžete vidět obecný návrh sítě. Síť je složena z jednotlivých senzorových uzlů a výchozího uzlu. Jednotlivé uzly tedy posílají své data na adresu výchozího uzlu, který předává tyto data aplikaci. Výchozí uzel může být přímo propojen k počítači, kde běží aplikace pomocí rozhraní USB. Tento uzel může využít rozšiřujících Arduino desek pro spojení s ostatními sítěmi jako LAN, WiFi, Bluetooth a dalších. Takto rozšířený výchozí uzel nemusí být přímo spojený se zařízením, kde běží aplikace, ale může odesílat svá data vzdálenému serveru. Aplikace následně příchozí data uloží do databáze.

Jak již bylo řečeno, síť bude využívat standard DigiMesh. Jednotlivé XBee moduly tedy budou typu XB24-DM s firmwarem XBEE DIGIMESH 2.4 viz obrázek 2.12. Jednotlivé uzly musí být zařazeny do stejné sítě, aby mohly spolu komunikovat. Každý XBee modul musí mít tedy nastaveno „PAN ID" na stejnou hodnotu. V programu X-CTU je tato hodnota označena pod položkou ID. Jednotlivé uzly v síti mají nastavenu svoji jedinečnou 64-bitovou adresu, která je pevně stanovena a nelze ji změnit. Tato adresa se skládá ze dvou části, a to SH a SL. V síti je potřeba potřeba určit

adresu uzlu, který bude sloužit jako výchozí uzel. Tato adresa je také složena ze dvou části, a to DH a DL. V tabulce 3.1 můžete vidět příklad nastavení výchozího uzlu.

Tabulka 3.1: Nastavení XBee modulu výchozího uzlu.

Výchozí uzel	Nastavení
ID	1111
SH	13A200
SL	40657F71



Obrázek 3.1: Návrh sítě.

Adresa výchozího uzlu (SH a SL) bude sloužit jako cílová adresa (DH a DL) u ostatních uzlů v síti. V tabulce 3.2 lze vidět nastavení jednoho ze senzorových uzlů. Pro přidání dalších senzorových uzlů se musí nastavit hodnoty ID, DH a DL, které jsou uvedeny v tabulce 3.2. Pouze hodnoty SH a SL budou jedinečné pro každý uzel.

Tabulka 3.2: Nastavení senzorového uzlu.

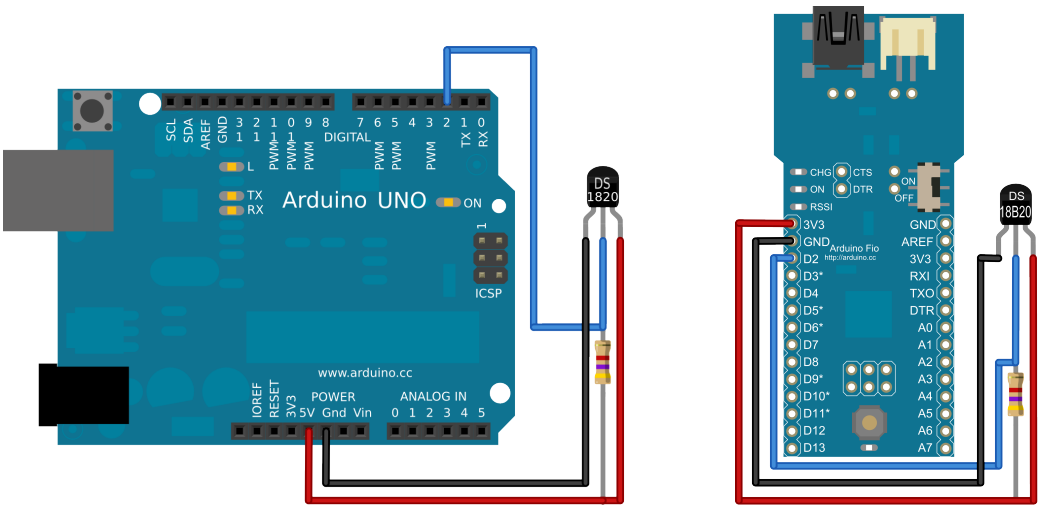
Senzorový uzel	Nastavení
ID	1111
SH	13A200
SL	40657B64
DH	13A200
DL	40657F71

Samotné propojení desek Arduino UNO a XBee modulu bude řešeno pomocí SparkFun XBee Shield pro snadnější propojení XBee modulu a samotné desky. Arduino FIO XBee patici již obsahuje a není nutné desku jinak upravovat.

Díky využití standardu DigiMesh je zajištěno dynamické připojení uzlů do sítě a vytváření mesh topologie sítě. Připojení nového senzorového uzlu, případně odpojení uzlu, bude možno kdykoliv, aniž by bylo potřeba bezdrátovou síť vypínat. Fyzickou polohu jednotlivých uzlů lze měnit, ale uzel musí zůstat v dosahu jiných uzlů, aby nedošlo ke ztrátě informace. Tento problém lze řešit přepnutím z transparentního režimu do API režimu. Transparentní režim je podobný sériovému spojení, kdy data, která přečte modul na svém vstupu, odešle na adresu, kterou má nastavenu v DH a DL. V režimu API je nutné vytvářet rámce, pomocí kterých můžeme odesílat příkazy jiným uzlům a nebo datové pakety. V tomto režimu se při odeslání paketu odesílatel dozví, zda zpráva dorazila nebo došlo k chybě a může si uchovat data na pozdější odeslání. Pro jednoduší testování bezdrátové sítě bude zvolen transparentní režim.

3.1.1 Uzel se senzorem DS18B20

Jak bylo řečeno v sekci 2.3.1, teplotní senzor komunikuje pomocí 1-Wire sběrnice. Digitální signálový vodič je zapotřebí propojit s digitálním pinem na desce Arduino. K tomuto účelu jsem zvolil pin 2, ale možno zvolit jakýkoliv jiný digitální port. Pouze digitální porty D0 a D1 slouží ke komunikaci s připojeným XBee modulem. Při návrhu zapojení jsem se rozhodl použít externí napájení a díky rozsahu napájecího napětí senzoru lze využít napájecí konektory +5V desky Arduino UNO nebo 3V3 desky Arduino FIO. Dále je zapotřebí propojit konektor země s pinem GND na deskách Arduino. Pro správnou funkčnost je nutné připojit digitální a napájecí pin pomocí 4,7k rezistoru. Na obrázku 3.2 můžete vidět návrh zapojení teplotního senzoru DS18B20 s deskou arduino UNO a FIO. Pro komunikaci se senzorem slouží knihovna OneWire [31].

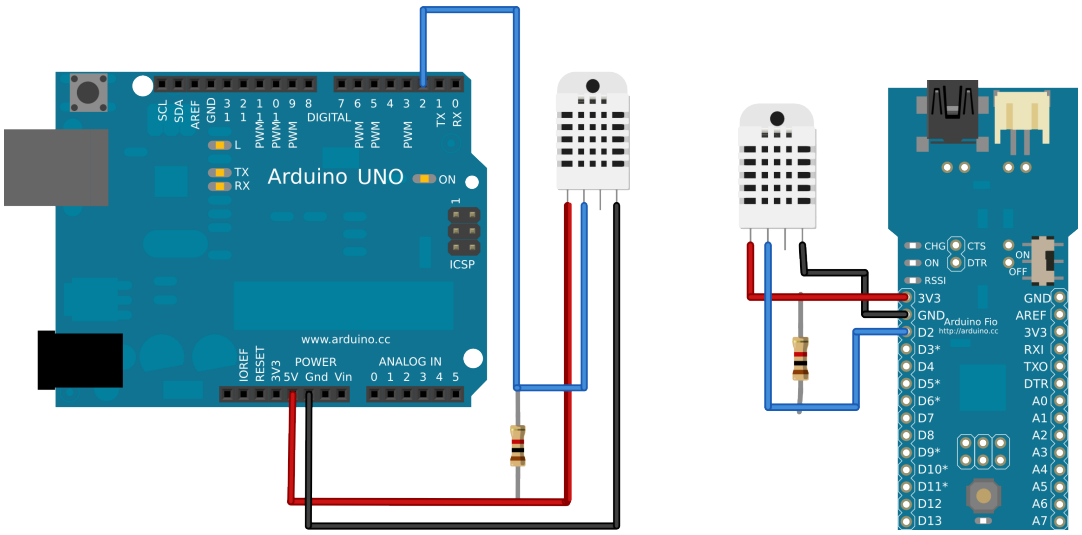


Obrázek 3.2: Teplotní senzorový uzel.

3.1.2 Uzel se senzorem RHT03

RHT03 vyžaduje podobné zapojení jako teplotní senzor DS18B20. Ke komunikaci využívá také 1-Wire sběrnici a je zapotřebí propojit vodič pro data s digitálním pinem na desce Arduino. Tento senzor má napájecí napětí v rozsahu 3,3 – 6 V a lze jej připojit na napájecí konektory 5V desky Arduino UNO nebo 3V3 desky Arduino FIO, aniž by bylo potřeba jiného externího napájení.

V poslední řadě je nutno propojit konektor země s pinem GND na deskách Arduino. Na obrázku 3.3 můžete vidět návrh zapojení teplotního senzoru RHT03 s deskou arduino UNO a FIO. Ke komunikaci se senzorem existuje knihovna od firmy Adafruit Industries [20].



Obrázek 3.3: Návrh zapojení RHT03 s deskami Arduino.

3.1.3 XBee a režim spánku

Pro nastavení režimu spánku je potřeba vybrat uzel pro podporu spánku. Parametr SM XBee modulu tohoto uzlu je potřeba nastavit na číslo 7. Také je nezbytné nastavit parametr SO na hodnotu 1 pro výběr preferovaného koordinátora spánku. Jako nejvhodnější uzel se jeví výchozí uzel, který je připojen ke zdroji napájení a to portu USB. Dále je potřeba nastavit parametr SP, který určuje dobu spánku a ST pro dobu probuzení. Pro ostatní uzly v síti je potřeba nastavit parametr SM na číslo 8 pro aktivaci cyklického režimu spánku a nastavit parametr SO na hodnotu 2, aby se tento uzel nestal koordinátorem spánku. Při přechodu do režimu spánku se pin CTS XBee modulu nastaví na hodnotu logické jedna a při probuzení na hodnotu logické nuly. Změnu CTS lze využít pro externí přerušení u desek Arduino. Další informace o režimu spánku se můžete dočíst v [40].

3.1.4 Arduino a úspora energie

Z důvodu využití desek Arduino FIO, které jsou primárně napájeny pomocí baterií, jsem se rozhodl prozkoumat možnosti uspání mikrokontroléru ATmega. Mikrokontrolér nabízí různé režimy spánku, které umožňují uživateli nastavit individuální spotřebu energie na požadavcích aplikace. Tyto režimy spánku vypínají nepoužívané moduly v mikrokontroléru, čímž šetří energii. Existuje 6 těchto módů.

Prvním z těchto módů je tzv. IDLE mode, což by se dalo přeložit jako klidový režim. Jedná se o nejlehčí formu spánku, při kterém procesor zastaví provádění instrukcí a tímto se ušetří velká část energie. Všechny ostatní periferie jsou ale nadále aktivní, a tedy spotřebovávají energii. Další v pořadí jsou ADC Noise Reduction, Power-save, Standby, Extenden Standby a Power Down, kde poslední režim přináší největší úsporu energie. Při vstupu do režimu spánku je následně potřeba přerušení, aby se spánek přerušil a mikrokontrolér mohl vykonávat instrukce. Pro přerušení můžeme použít pin mikrokontroléru ATmega nebo tzv. watchdog časovač (WDT).

Watchdog časovač (WDT), který může být aktivní v každém režimu spánku. WDT umožňuje dva režimy, přerušení nebo reset systému, pokud systém nerestartuje počítadlo, než čítač dosáhne daného

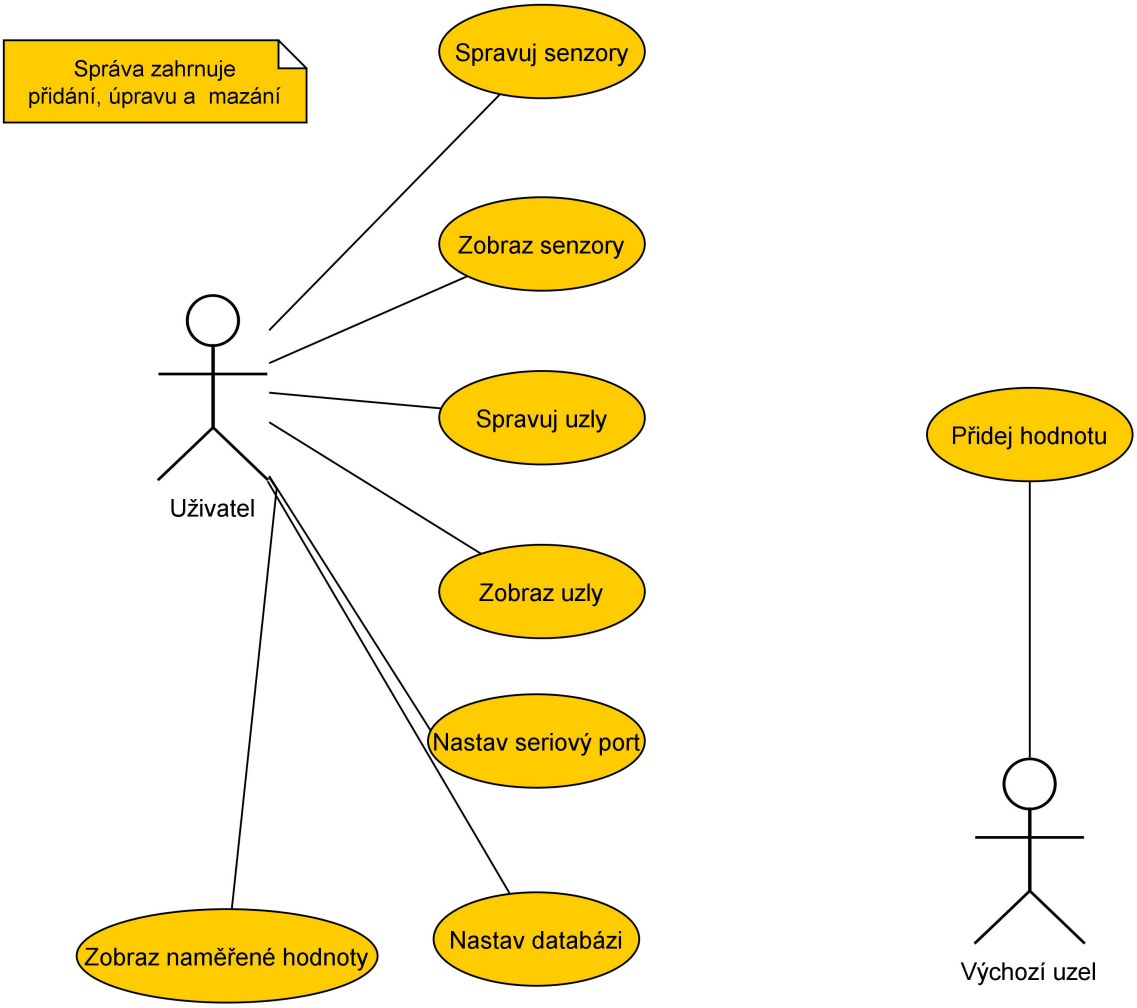
časového limitu. Pro nastavení WDT je potřeba prvně zapsat logickou hodnotu jedna do WDCE (Watchdog change enable bit) a WDE (Watchdog System Reset Enable) do registru WDTCSR. Logická hodnota jedna musí být zapsána do WDE, ať už byla jeho hodnota jakákoliv. V dalším kroku je potřeba nastavit WDP[3:0] (Watchdog Timer Prescaler 3, 2, 1 a 0) pro nastavení času. Při nastavení WDP3 a WDP0 se nastaví na maximální časový limit 8 sekund. Pro aktivaci režimu přerušení slouží WDIE (Watchdog Interrupt Enable).

Při použití přerušení pomocí pin musíme použít funkci *attachInterrupt*, které se předají parametry s číslem portu, funkcí a režimem, při kterém se má vyvolat přerušení. Pro přerušení můžeme zvolit porty 2 a 3 viz popis desek Arduino UNO a FIO. Existují 4 režimy a to LOW při vyvolání přerušení při hodnotě logické nuly, CHANGE při jakékoliv změně hodnoty, RISING při změně z hodnoty logické nula do logické jedna a FALLING z hodnoty logické jedna do logické nula.

Více o režimech spánku, přerušeních a watchdog v [11, 10].

3.2 Návrh aplikace

Aplikace se musí zaměřit na uložení a zpracování dat a jejich zobrazení pomocí grafu a tabulek. Základní funkce aplikace jsou zobrazeny pomocí diagramu případu užití na obrázku 3.4.



Obrázek 3.4: Diagram případu užití.

Jak lze vidět v tomto diagramu případu užití, existují dva aktéři.

- Uživatel
- Výchozí uzel

Aktér Uživatel představuje reálného uživatel aplikace. Uživatel může spravovat jednotlivé uzly a zobrazit si jejich seznam. Pod pojmem „správa" se myslí vytvoření nového senzorového uzlu, jeho následná editace a případné smazání. Uživatel dále může spravovat jednotlivé senzory a následně si je i zobrazit. Uživatel má také možnost nastavit databázi, která bude použita na ukládaní dat. Uživatel si může zobrazit naměřené hodnoty ve formě grafů a tabulek.

Aplikace také musí umožnit uživateli nastavit sériový port pro komunikaci s připojeným výchozím uzlem ke stanici, kde aplikace poběží. V tomto případě použití aplikace zobrazí dostupné porty a uživatel si zvolí port a přenosovou rychlost. Na zvoleném portu bude aplikace přijímat naměřená data. Dalším aktérem je Výchozí uzel. Tento aktér reprezentuje výchozí uzel senzorové bezdrátové sítě, který přeposílá naměřená data aplikaci.

Rozhraní aplikace by mělo být jednoduché na ovládání, aby uživatel měl přehled o připojené databázi, uzlech, senzorech a informacích o sériovém portu. Samotná aplikace by měla být multiplatformní a zobrazovat naměřená data pomocí grafů a tabulek.

3.2.1 Implementační technologie

Jak již bylo řečeno, aplikace si klade za podmínku, aby podporovala více platforem. Při splnění této podmínky lze aplikaci provozovat na běžném stolním počítači s operačními systémy Windows i Linux, ale také na vestavěných systémech s architekturou ARM. Je tady několik možností ohledně vhodné volby implementační technologie. Po prozkoumání možností a úvaze se jako vhodní kandidáti jeví tyto možnosti:

- Apache
- Qt framework

Apache je webový server, který je šířen jako open source a je dostupný pro GNU/Linux, BSD, Solaris, Mac OS X a Microsoft Windows. Díky své otevřenosti se jedná o velmi často nasazovaný webový server. Tento server podporuje různé rozšíření a nejrůznější programovací jazyky, jako je například PHP. [1]

Qt je multiplatformní framework pro tvorbu aplikací s grafickým rozhraním, které je možné spustit na systémech Windows, Linux/Unix, Mac OS X, Symbian a Meego. Tyto knihovny jsou postaveny na jazyce C++, ale také hojně využívají vlastní speciální Meta-Object překladač. Qt je rozdělen do specializovaných modulů, které obsahují třídy pro zobrazení grafických uživatelských prvků, práci s multimédii, databázovými systémy, XML a další. Poskytují téměř úplnou náhradu pro STL třídy a podporují některé speciální funkce bez použití moderního kompilátoru. [13]

Obě tyto možnosti mají svoje přednosti. Apache a PHP by umožnila uživateli zobrazit aplikaci pomocí webového prohlížeče. Ale nevýhodou je nutnost instalovat a nastavit Apache server. S pomocí framworku Qt lze vytvořit aplikaci, která by byla multiplatformní a také by odpadla nutnost instalace serveru, což je z uživatelského hlediska velmi pozitivní. Proto jsem se tady rozhodl využít Qt framework pro tvorbu aplikace. V dalších odstavcích se budu snažit popsat specifické technologie z frameworku Qt, které budu využívat při implementaci aplikace.

3.2.2 Qt framework

Jak bylo řečeno Qt framework využívá Meta-Object překladač, který vytváří meta objekty. Meta objekt je objekt, který popisuje strukturu jiného objektu. Třída, která má podporu meta objektu podporuje reflexi, kterou můžeme nalézt v mnoha objektově orientovaných jazycích, ale v C++ se tato vlastnost nenalézá. Systém meta objektů také poskytuje mechanismus „signals-slots" pro mezi objektovou komunikaci. Tento mechanismus je jedním z hlavních prvků Qt a také se tímto liší od většiny funkcí poskytovaných jinými frameworky. Starší frameworky obsahují tzv. callback. Jedná se o ukazatel na funkci, který se bude volat při nějaké události. Nevýhodou je, že tento ukazatel musí být předán objektu, který jej bude následně volat. Qt využívá signály pro oznámení událostí a slotů, které jsou volány jako odpověď na konkrétní signál. Sloty je možné použít pro příjem signálu, ale také fungují jako normální členské funkce. Stejně jako objekt neví, jestli jiný objekt obdrží jeho signály, slot neví, jestli jsou jakékoliv signály s ním spojené. Tím je zajištěno, že lze skutečně vytvářet nezávislé komponenty pomocí Qt. Aplikace se bude snažit využít tohoto přístupu pro vytvoření nezávislých komponent.

Qt resource system

Jedna z dalších vlastností Qt je „Qt resource system", který slouží pro ukládání binárních souborů spolu s aplikací. Tento mechanismus je užitečný, pokud aplikace používá určité soubory a nechceme riskovat jejich ztrátu na disku. Těmito soubory mohou být obrázky, soubory s překladem a jiné. Výhodou začlenění binárních souborů do aplikace je, že se lze na ně odkazovat pomocí cest, které nejsou závislé na lokálním souborovém systému. Nevýhodou je zvětšení velikostí samotné aplikace.

Překlad aplikace

Qt poskytuje vynikající podporu pro překlad aplikací do různých jazyků. Pro přípravu textu na překlad je potřeba použít metodu QObject::tr() pro obklopení jakéhokoliv řetězce v aplikaci. Při použití jako nestatickou funkci používá název třídy, dostupného z QMetaObject, který se používá jako kontext pro seskupení řetězců. K vytvoření samotného překladu slouží tři nástroje. Prvním je *lupdate* a používá se k synchronizaci zdrojových kódů a překladů. Tento nástroj extrahuje přeložitelné řetězce, které jsou obsaženy ve volání metody *tr()* a uloží je do XML souboru. Druhým nástrojem je *Qt Linguist* pro samotný překlad textů. Posledním v řadě je nástroj *lrelease*, který se používá k vytvoření run-time překladových souborů pro použití v aplikace. Rozhodl jsem se tedy využít těchto nástrojů k překladu aplikace do českého a anglického jazyka. Pro další informace lze nahlédnout do referenční literatury [2, 13].

3.2.3 Sériový port

Z důvodu připojení výchozího uzlu na USB port stanice, kde aplikace poběží, je potřeba číst hodnoty z tohoto portu. Framework Qt neobsahuje podporu pro práci se sériovým portem neobsahuje. Tuto situaci se snaží řešit několik knihoven, které obsahují třídy, které pracují se sériový portem. Nalezl jsem tyto knihovny:

- QSerialDevice [29].
- QextSerialPort [32].

Obě tyto knihovny podporují práci se sériovými porty na platformách Windows, Linux a Mac OS X. Tyto knihovny obsahují třídy pro zápis a čtení dat ze sériového portu, které jsou založeny na třídě

QIODevice. *QIOD*evice obsahuje jak společnou implementaci a abstraktní rozhraní pro zařízení, která podporují čtení a zápis bloků dat, jako jsou například soubory a nebo sokety. Příklad použití knihovny *QextSerialPort* ilustruje ukázka výpisu 3.1.

Kód 3.1: Ukáka použití knihovny QextSerialPort

```
#include "qextserialport.h"
...
MyClass::MyClass()
{
  port = new QextSerialPort("COM1");
  connect(port, SIGNAL(readyRead()), this, SLOT(readFromSerialPort()));
  port->open();
}

void MyClass::readFromSerialPort()
{
  QByteArray data = port->readAll();
  ...
}
```

Při využití knihovny *QSerialDevice*, by se místo třídy *QextSerialPort* využila třída *SerialPort*.

3.3 Databáze

Samotná aplikace potřebuje ukládat data ze senzorových uzlů. Aplikace také využívá databázi pro uložení informací o jednotlivých senzorových uzlech a použitých senzorech. Qt framework obsahuje QtSql modul pro práci s databázemi a podporuje několik druhů databází. Je několik možností ohledně volby správné databáze. Po prozkoumání podpory databází a jejich dostupnosti pro uživatele se jako vhodné jeví tyto databázové systémy:

- MySQL
- PostgreSQL
- SQLite

MySQL je vysoce výkonný, vícevláknový, mnohouživatelský RDBMS[1] postavený na architektuře klient-server. Za posledních pár let se tento databázový systém stal nejvyhledávanější volbou pro komerční nebo domácí použití díky celosvětové podpoře komunity uživatelů. Zdrojové kódy MySQL jsou volně dostupné pod licencí GNU GPL. MySQL server je dostupný pro operační systémy typu UNIX jako Linux, Solaris, FreeBSD, OS/2, MacOS a také systémy Windows. [9]

PostgreSQL je objektově relační databáze šířená pod otevřenou licencí. Tento databázový systém má za sebou více než 15 let vývoje. Podporuje několik operačních systému jako Linux, BSD, Mac OS X, Solaris a Windows. [7]

SQLite je RDBMS, které neběží ve vlastním procesu nebo systému, ale za pomocí SQLite knihovny aplikace přímo přistupují k datům. Celá instance databáze sídlí v jednom multiplatformním souboru a nevyžaduje žádnou správu a celý databázový systém je integrován do aplikace pomocí knihovny. SQLite je šířen pod licencí GNU GPL a je dostupný pod operačními systémy Linux, Mac OS X a Windows.

Všechny zmíněné databáze mají své přednosti. Databáze MySQL a PostgreSQL nejsou pro účely této práce zcela vhodné a to z důvodu, že databáze je potřeba instalovat a nastavit. SQLite

[1] relational database management system

databáze se jeví jako nejvhodnější volba, kde se databáze bude distribuovat spolu s aplikací, uživatel nebude nucen nastavovat server a samotný Qt framework tuto databázi podporuje. [5]

Pokud potřebujete přesunout nebo zálohovat SQLite databázi, můžete jednoduše zkopírovat soubor. Tento přístup zjednodušuje softwarové komponenty a téměř eliminuje potřebu podpory pokročilého operačního systému. Na rozdíl od tradičního serveru RDBMS, který vyžaduje pokročilý multitasking a vysoce výkonnou komunikaci mezi procesy, SQLite vyžaduje schopnost číst a zapisovat do nějakého typu média. SQLite transakce jsou plně ACID kompatibilní, což umožňuje bezpečný přístup z více procesů nebo vláken a podporuje většinu dotazů jazykových prvků nalezených v SQL92 standardu.

ER diagram na obrázku 3.5 zobrazuje navrženou strukturu databáze pro aplikaci. Databáze obsahuje tři tabulky, které aplikace využívá ke správě uzlů, senzorů a naměřených dat.

Tabulka Nodes

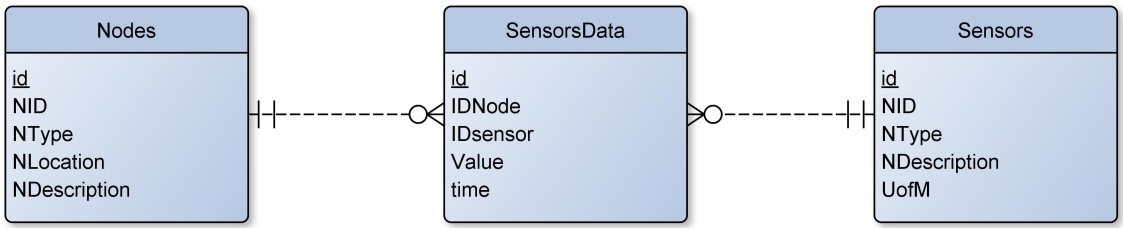
Tato tabulka obsahuje informace o jednotlivých uzlech v síti. Pro přidání nového uzlu musí uživatel vyplnit NID uzlu, zbylé ostatní položky jsou nepovinné, ale je vhodné tyto položky vyplnit. Uživatel může vyplnit o jaký typ uzlu se jedná, jeho umístění a popis. V diagramu je vidět vazba 1–N s tabulkou *SensorsData*. Sloupec id slouží jako primární klíč tabulky a sloupec NID jako id uzlu, který volí uživatel a je unikátní. Slouží aplikaci k rozpoznání, který uzel poslal naměřená data aplikaci. NType slouží pro typ označení typu uzlu například Arduino FIO, UNO nebo jakýkoliv jiný typ uzlu. Sloupec NLocation, jehož hodnota bude sloužit pro určení místa, kde se bude nacházet uzel například místnost C209-okno a sloupec NDescription pro popis uzlu a další informace.

Tabulka Sensors

Tabulka obsahuje informace o senzorech, které využívají uzly v síti. Uživatel musí vyplnit NID, další informace nejsou vyžadovány. Sloupce NType, NDescription a UofM slouží aplikaci pro lepší přehlednost a zobrazení informací. V případě použití senzoru, který měří dvě hodnoty viz podkapitola 2.3.2, musí uživatel rozdělit tento senzor na dva samostatné senzory. Sloupce id a NID plní podobnou funkci jako u tabulky Nodes. Sloupec NType slouží k určení typu senzoru například DS18B20-teplota. NDescription sloupec slouží pro popis senzoru a sloupec Uofm, jehož hodnota bude sloužit pro zobrazení veličiny, kterou senzor měří (například °C).

Tabulka SensorsData

Tato tabulka uchovává naměřené hodnoty z jednotlivých uzlů a jejich senzorů. Samotný uživatel s touto tabulkou nepracuje. Sloupec IDNode slouží jako cizí klíč do databázové tabulky Nodes a určuje, kterému uzlu náleží naměřená hodnota. IDSensor sloupec obsahuje cizí klíč do databázové tabulky Sensors a určení, který senzor hodnotu naměřil. Value slouží k uchování hodnoty, která byla naměřena senzorem a sloupec time pro čas, ve kterém byla hodnota přidána do databáze.



Obrázek 3.5: ER diagram.

3.4 Komunikační protokol

Jednotlivé senzorové uzly potřebují odesílat naměřené data aplikaci. Pro přenos informací ze senzorových uzlů jsem se rozhodl použít mnou navržený jednoduchý textový protokol. Uzel potřebuje odeslat své ID, ID senzorů a hodnoty naměřených veličin. Při vytvoření formátu zprávy, kterou odešle senzorový uzel jsem se inspiroval souborovým formátem CSV[2], ale jednotlivé položky jsou odděleny znakem středníku. Příklad odeslané zprávy pro aplikace od senzorového uzlu můžete vidět ve výpisu 3.2.

Kód 3.2: Ukázka odeslané zprávy ze senzorového uzlu

```
1;1;25.30;2;40
```

První hodnota v odeslané zprávě je ID uzlu, který ji odeslal. Dále následují dvojce ID senzoru a neměřená hodnota. Jednotlivé dvojce pro senzor a hodnotu lze libovolně přidávat a v jediné zprávě může být více naměřených hodnot. Na konci každé odeslané zprávy se očekává znak pro konec řádku.

3.5 Zobrazení naměřených dat

Aplikace musí zobrazovat data, které naměřily jednotlivé senzorové uzly pomocí grafů a tabulek. V této části se prvně zaměřím na zobrazení grafů pomocí frameworku Qt.

3.5.1 Grafy

Samotný Qt framework neobsahuje žádné knihovny pro vykreslování grafů, ale existuje několik možností řešení tohoto problému. Po jejich prozkoumání spolu s možnostmi frameworku Qt se nabízejí tyto varianty:

- Vytvoření vlastního vykreslování grafů
- Knihovna Qwt
- QtWebKit

První možnost je vytvořit třídy, které by využívaly Arthur Paint System pro vykreslování grafů. Tento přístup by mohl být velmi časově náročný.

Další možností je použití knihovny Qwt, která obsahuje třídy na tvorbu grafů pro více technické aplikace. Knihovna umožňuje vytvářet jak grafy, spektrogramy, tak také obsahuje ovládací prvky jako číselníky, kompas a posuvníky.

Poslední možností je QtWebKit, jenž poskytuje webový engine WebKit pro vykreslování webových stránek a most k propojení jeho JavaScriptového prostředí s objekty třídy QObject. QtWebKit poskytuje zázemí pro vykreslování HyperText Markup Language (HTML), Extensible HyperText Markup Language (XHTML) a Scalable Vector Graphics (SVG) dokumentů, kaskádových stylů (CSS) a skripty pomocí jazyka JavaScript. Díky meta-objekt systému Qt frameworku, lze zpřístupnit QObject v JavaScriptovém prostředí QtWebKit viz kód 3.3. Toto lze využít například při stisku tlačítka v prohlížeči, při kterém se zavolá metoda C++ objektu.

Kód 3.3: Zpřístupnění C++ objektu do JavaScript prostředí

```
QWebFrame *frame = myWebPage->mainFrame();
frame->addToJavaScriptWindowObject("jmenoProObjekt", myObject);
```

[2]Comma-separated values

Webové stránky mohou být transparentně načítány z webových severů, lokálního souborového systému nebo Qt resource systému.

Jako nejvhodnější se jeví využití QtWebKit a JavaScriptových knihoven pro vykreslování grafů, které mají lepší možnosti zobrazení dat a také je možný případný export grafů pro zobrazení na webových stránkách. Webové stránky mohou být načteny z Qt resource spolu s kaskádovými styly a JavaScriptovými knihovnami. QtWebKit také umožňuje provádět JavaScriptový kód přímo z C++ prostředí viz kód 3.4.

Kód 3.4: Provedení JavaScriptového kódu z C++.

```
QWebFrame *frame = myWebPage->mainFrame();
frame->evaluateJavaScript("alert('Pozor');");
```

Je tedy potřeba vybrat vhodnou knihovnu pro tvorbu grafů. Požadovaná vlastnost těchto knihoven je zobrazení času na ose X. Knihoven pro vytváření grafů ve webové prostředí existuje velké množství a po prozkoumání možností jsem vybral tyto volně dostupné kandidáty:

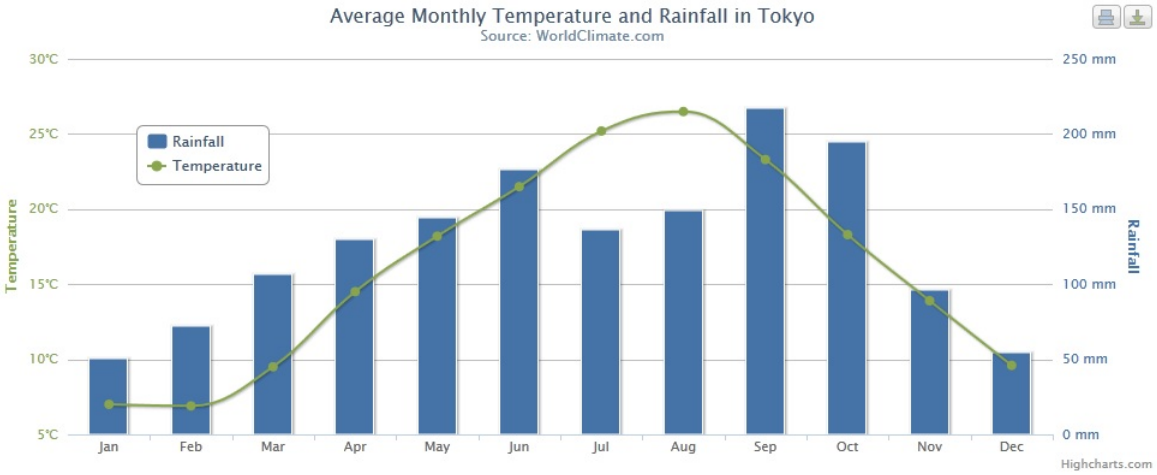
- Highcharts [36].
- dygraphs [23].
- Flotr2 [25].

Highcharts je knihovna pro vytváření grafů napsaná v čistém JavaScriptu, která nabízí intuitivní a interaktivní grafy pro webové stránky a webové aplikace. Highcharts v současné době podporuje řadu typů grafů od čárových, sloupcových, plošných, koláčových a bublinových. Tato knihovna je volně dostupná pro nekomerční použití a její zdrojový kód je otevřený. Ke správné funkčnosti vyžaduje jeden z frameworku jQuery, MooTools nebo Prototype.

Dygraphs je open source JavaScriptová knihovna, která vytváří interaktivní grafy hodnot v závislosti na čase. Je určena k zobrazení velkého množství dat a umožňuje uživatelům je prozkoumat pomocí funkce přiblížení.

Flotr2 je knihovna pro kreslení HTML5 grafů. Je založena na knihovně flotr a odstraňuje závislost na frameworku Prototype a obsahuje mnoho dalších vylepšení. Podporuje řadu grafů od čárových, sloupcových, koláčových a bublinových.

Všechny tyto knihovny obsahují podporu pro vykreslování hodnot závislých na čase a splňují podmínku pro zobrazení času na ose X. Knihovna Highcharts ale obsahuje navíc podporu pro více os, které by umožňovaly zobrazit například teplotu, tlak a vlhkost v jediném grafu. Každá osa může být umístěna na pravé nebo levé, horní nebo dolní části grafu. Všechny možnosti lze nastavit individuálně, včetně stylu a umístění. Nastavení tvorby grafu se uvádí v JavaScriptové objektové notaci, která je v podstatě sada klíčů a hodnot spojených dvojtečkami, oddělené čárkami a seskupené do složených závorek. Další výhodou je zobrazení textové nápovědy s informacemi o každém bodu v sérii. Na obrázku 3.6 můžete vidět příklad grafu vytvořeného pomocí knihovny Highcharts, jenž obsahuje dvě osy y, kde levá osa je pro teplotu a pravá pro vodní srážky. Pro správnou funkčnost knihovny jsem zvolil framework jQuery.



Obrázek 3.6: Graf v Highcharts. Převzato z [28].

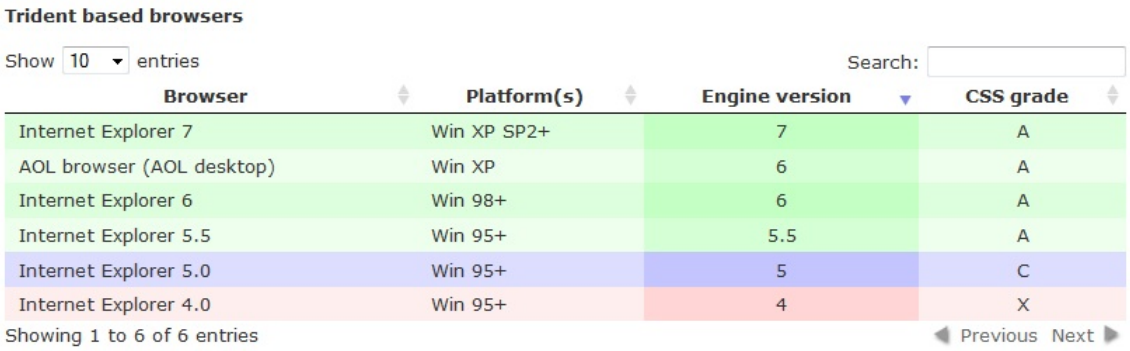
3.5.2 Tabulky

Pro zobrazení tabulek existuje také několik variant možných řešení. Aplikace potřebuje zobrazit informace o jednotlivých uzlech, senzorech a naměřených hodnotách. Toto jsou možnosti, které se jeví jako vhodní kandidáti:

- Qt Interview Framework
- QtWebKit

Qt framework obsahuje třídy pro model/view architekturu. QtSQL využívá také tuto architekturu a obsahuje třídy pro vytvoření modelů přímo z databázových tabulek. Pro zobrazení informací o uzlech a senzorech ve formě tabulek se jeví vhodné využít tyto třídy.

Pro zobrazení neměřených hodnot jsem zvolil přístup pomocí QtWebKit z důvodu sjednocení zobrazení grafů a tabulek s naměřenými hodnotami. Při větším počtu dat ale bude tabulka velmi velká a nepřehledná, proto jsem se rozhodl využít volně dostupného rozšíření pro framework jQuery, který je potřebný pro správnou funkčnost knihovny Highcharts jménem DataTables [18]. Toto rozšíření přidá ovládací prvky k tabulkám pro možnosti řazení dle sloupců, stránkování dat a filtrování dat. Příklad takové tabulky můžete vidět na obrázku 3.7.



Obrázek 3.7: Tabulka v DataTables. Převzato z [19].

Kapitola 4

Implementace

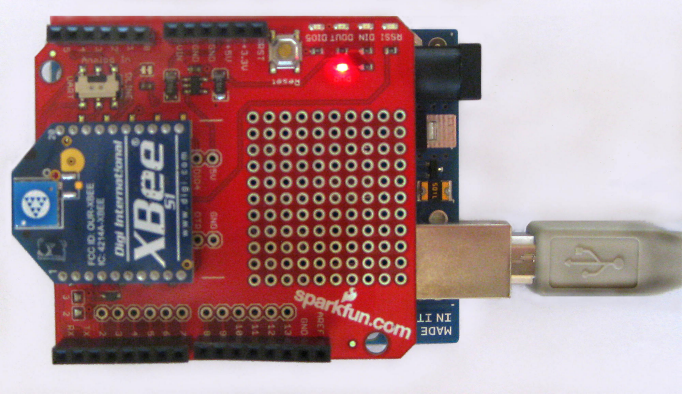
V této kapitole bude popsána samotná implementace jednotlivých senzorových uzlů. Dalším popsaným tématem je úspora energie. V následujících částech kapitoly je popsáno testování sítě v budově VUT FIT a popis implementace aplikace pro zobrazení naměřených dat.

4.1 Implementace bezdrátové sítě

V této části se nachází popis uzlů, které byly implementovány. Jedná se o koordinátora a uzly se senzory DS18B20 a RHT03. Pro vývoj software bylo použito vývojové prostředí Arduino verze 1.0. Jednotlivé XBee moduly byly nakonfigurovány dle sekce 3.1.

4.1.1 Výchozí uzel

Je připojen pomocí USB konektoru přímo se stanicí, kde běží aplikace pro ukládání naměřených hodnot. Tento uzel je složen z desky Arduino UNO, která je rozšířena o SparkFun XBee Shield s připojeným XBee modulem. Na obrázku 4.1 můžete vidět samotný uzel.



Obrázek 4.1: Výchozí uzel.

Při implementaci výchozího uzlu vznikl problém, protože Arduino UNO obsahuje pouze jediné fyzické sériové rozhraní, které využívá porty D0 a D1. Toto rozhraní je potřebné pro komunikaci s aplikací přes USB a XBee modulem. Tento problém lze řešit pomocí knihovny `SoftwareSerial`, která simuluje sériový port pomocí libovolných dvou digitálních portů na

desce Arduino UNO. Rozšiřující deska obsahuje přepínač, jak již bylo řečeno v 2.1.2, pomocí kterého bude XBee modul komunikovat místo portů D0 a D1 na portech D2 a D3. Přepínač tedy bude přepnut na porty D2 a D3, přes které bude deska Arduino komunikovat s modulem XBee pomocí knihovny `SoftwareSerial`. Ve výpisu 4.1 můžete vidět využití knihovny a kód pro výchozí uzel.

Kód 4.1: Kód pro koodinátora

```
#include <SoftwareSerial.h>
// RX je digitalni pin 2 (pripojeny na  TX jineho zarizeni)
// TX je digitalni pin 3 (pripojeny na  RX jineho zarizeni)

SoftwareSerial   xbeeSer(2, 3);//xbee RX, TX

void setup() {
  Serial.begin(9600);  // rychlost pro USB pripojeni
  xbeeSer.begin(9600); // rychlost pro  SoftSerial port
  delay(500);
}

void loop() {
  while(xbeeSer.available() == 0);
  while(xbeeSer.available() != 0){
    Serial.write(xbeeSer.read());
  }
}
```

Objektu třídy `SoftwareSerial` se musí v konstruktoru předat porty, na kterých bude komunikovat s připojeným XBee modulem. Před samotnou komunikací je potřeba nastavit rychlost přenosu pomocí metody `begin`. Ve funkci loop program aktivně čeká na přijetí dat z XBee modulu. Po přijetí dat z modulu jsou následně data odeslána pomocí objektu `Serial` a metody `write` do připojené stanice.

4.1.2 Odeslání hodnoty ze senzorového uzlu

Senzorové uzly sestavené z desky Arduino UNO jsou rozšířeny o SparkFun XBee Shield s připojeným XBee modulem obdobně jako u výchozího uzlu. Deska nepotřebuje komunikovat přes USB spojení, a tedy není potřeba přepínat přepínač pro změnu portů z D0 a D1 na porty D2 a D3. Uzly z desky Arduino FIO již obsahují patici pro XBee modul a není potřeba desku modifikovat nebo rozšiřovat. Ve výpisu kódu 4.2 je zobrazeno odeslání naměřených hodnot ze senzoru.

Kód 4.2: Odeslání naměřených hodnot.

```
#define IDNODE "1"
#define DS18S20Teplota "1"

void setup()
{
Serial.begin(9600);
...
}

void loop()
{
float temperature = getTemp();
...
  Serial.print(IDNODE);
```

```
  Serial.print(";");
  Serial.print(DS18S20Teplota);
  Serial.print(";");
  Serial.print(temperature);
  Serial.print(";");
  Serial.print("\n");
...
}
```

Odeslaná zpráva je ve formátu, který byl navrhnut v sekci 3.4. Nejprve je odesláno ID senzorového uzlu, ID senzoru a naměřená hodnota.

4.1.3 Uspání senzorové bezdrátové sítě

Jednotlivé moduly XBee byly nastaveny dle návrhu v sekci 3.1.3. Jako koordinátor spánku byl vybrán modul připojený k výchozímu uzlu. Ostatní moduly byly přepnuty do režimu cyklického spánku a byl propojen pin CTS s pinem D3 na desce Arduino pro přerušení při probuzení modulu z režimu spánku. Při testování sítě v režimu spánku docházelo po probuzení k nezesynchronizovaní uzlů a trvalo delší dobu než se uzel znovu připojil do sítě. Toto chování může být způsobeno špatným nastavením parametrů SP a ST. Od režimu spánku pomocí modulů XBee jsem při dalším testování opustil a využil dále popsané možnosti úspory energie.

4.1.4 Spánek mikrokontroléru ATmega

Jak bylo řečeno v 3.1.4, mikrokontrolér ATmega obsahuje režimy spánku. Na výpisu 4.3 je uvedena funkce sleep pro spánek. Pro aktivaci režimu spánku je potřeba přiložit soubor `sleep.h`. Makro `set_sleep_mode` nastaví vybraný režim spánku a k aktivaci spánku slouží makro `sleep_mode`. Pro probuzení pokračuje vykonávání programu dále od tohoto makra.

Kód 4.3: Funkce pro aktivaci spánku

```
#include <avr/sleep.h> //pro spanek

//funkce pro samotny spanek
void sleep(void)
{
  //nastavi se mod spanku
  // V souboru avt/sleep.h file je 5 modu pro spanek:
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);

  // zde je spanek - pri preruseni se pokracuje zde !!
   sleep_mode();
}
```

Pro probuzení z režimu spánku je potřeba přerušení. K vyvolání přerušení jsem využil integrovaný watchdog, který každých 8 sekund vyvolá přerušení. Samotný watchdog je potřeba nastavit, aby místo resetování mikrokontroléru vyvolal přerušení a také je potřeba nastavit dobu, za kterou se má přerušení vyvolat. Na výpisu 4.4, který je v příloze zobrazeno nastavení watchdog.

Kód 4.4: Nastavení watchdog

```
//WDT_vect - vektor preruseni pro watchdog - kniha
//muze byt prazdne
ISR(WDT_vect)
{
}
```

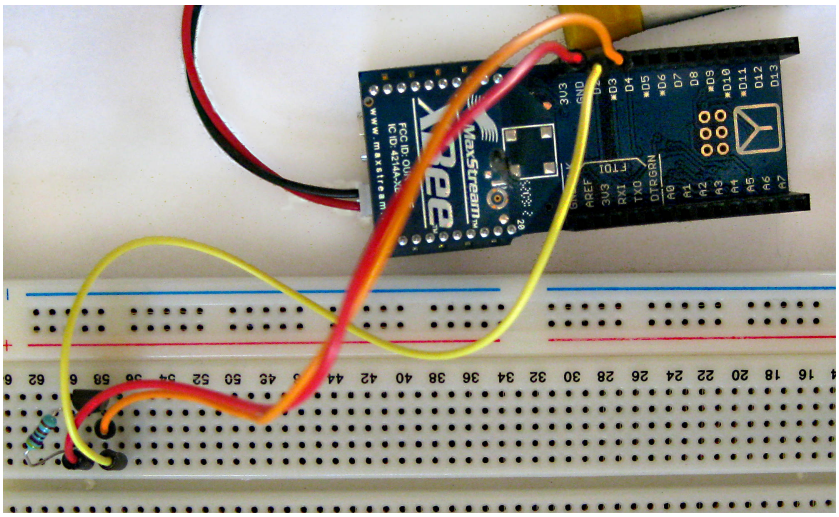
```
void setup()
{
...
  //je potrebe resetovat watch dog - avr manual
  bitClear(MCUSR,WDRF);

  //aby se mohla zmenit nasobicka je potreba nastavit WDCE
  //zmeny je mozno provadet po 4 hodinove cykly
  WDTCSR |= (1<<WDCE) | (1<<WDE);

  //nastaveni nasobicky
  //povoleni preruseni watchdog
  WDTCSR = (1<<WDIE) | (1<<WDP0) | (1<<WDP3); // 8.0 sekund
}
```

Jak již bylo řečeno v sekci 3.1.4 je potřeba resetovat watchdog pomocí smazání bitu `WDRF` v registru `MCUSR`. Ve výpisu kódu je k tomuto použito makro `bitClear`. Dále je nutné nastavit bity `WDCE` a `WDE` registru `WDTCSR` na hodnotu logické jedna. Následně se nastaví bity `WDP0`, `WDP3` a `WDIE` na hodnotu logické jedna pro nastavení času na 8 sekund a aktivaci režimu přerušení. Tuto dobu lze libovolně opakovat k dosáhnutí delší doby spánku, kdy se mikrokontrolér na kratší dobu probudí a posléze znovu uspí.

4.1.5 Uzel se senzorem DS18B20

Senzor DS18B20 byl zapojen dle návrhu v 3.1.1. Pro komunikaci desky Arduino a senzoru byla použita knihovna `OneWire`. Tato knihovna obsahuje třídu `OneWire`, která vlastní metody pro vyhledávání a komunikaci se zařízeními na sběrnici. Na výpisu B.1, který je umístěn v příloze, můžete vidět funkci `getTemp` pro přečtení naměřené hodnoty.

Pro práci se senzorem se musí vytvořit objekt třídy `OneWire` a v konstruktoru objektu předat číslo pinu, na kterém je připojený senzor. Metoda `search` vyhledá zařízení na sběrnici a naplní 8 bitové pole adresou zařízení. Třída také obsahuje statické metody pro kontrolu kontrolního součtu. Metoda `reset` slouží resetování 1-wire sběrnice a je nutné je zavolat před komunikaci se zařízením.



Obrázek 4.2: Arduino UNO se senzorem DS18B20.

Pro výběr zařízení, se kterým chceme komunikovat, slouží metoda `select` s parametrem

adresy. Metod `write` a `read` slouží k zápisu a čtení hodnot na sběrnici. Na obrázku 4.2 můžete vidět zapojení desky Arduino FIO se senzorem DS18B20. Zapojení pro desku Arduino UNO je obdobné.

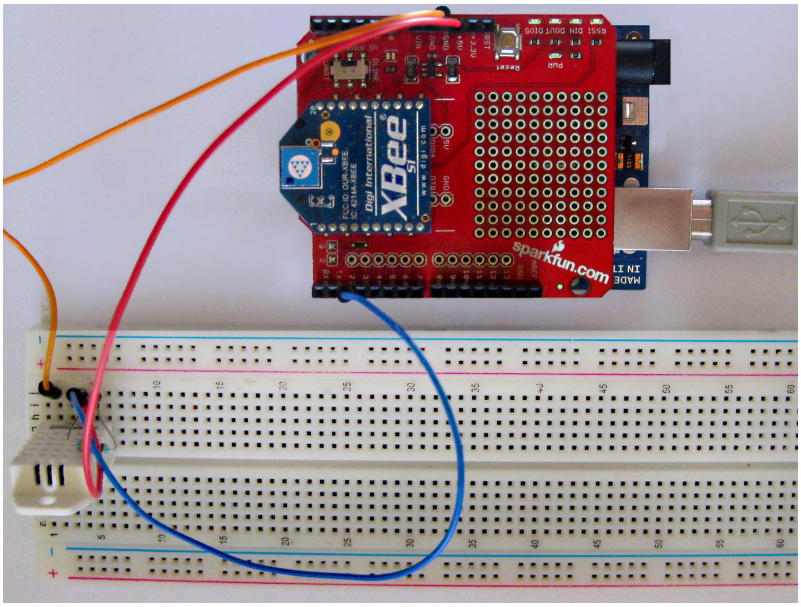
4.1.6 Uzel se senzorem RHT03

RHT03 byl zapojen také dle návrhu uvedeného v 3.1.2. Knihovna pro komunikaci se senzorem obsahuje třídu `DHT`. Tato třída obsahuje metody pro získání vlhkosti a teploty ze senzoru. Na výpisu 4.5 je zobrazena práce s knihovnou a získání teploty a vlhkosti ze senzoru.

Kód 4.5: Kod pro senzor RHT03

```
#include "DHT.h"
#define DHTPIN 2     // bude na pinu 2
#define DHTTYPE DHT22   // DHT 22  (AM2302)

DHT dht(DHTPIN, DHTTYPE);

void setup() {
...
    dht.begin();
...
}

void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();
...
}
```

V konstruktoru objektu třídy `DHT` je přidán pin, ke kterému je připojen senzor a typ senzoru, se kterým pracujeme. Před získáním hodnot ze senzoru je nutné zavolat metodu `begin`. Metoda `readHumidity` slouží k přečtení vlhkosti ze senzoru a metoda `readTemperature` k získání teploty. Na obrázku 4.3 je zobrazeno zapojení senzoru k desce Arduino UNO.



Obrázek 4.3: Arduino UNO se zapojeným senzorem RHT03.

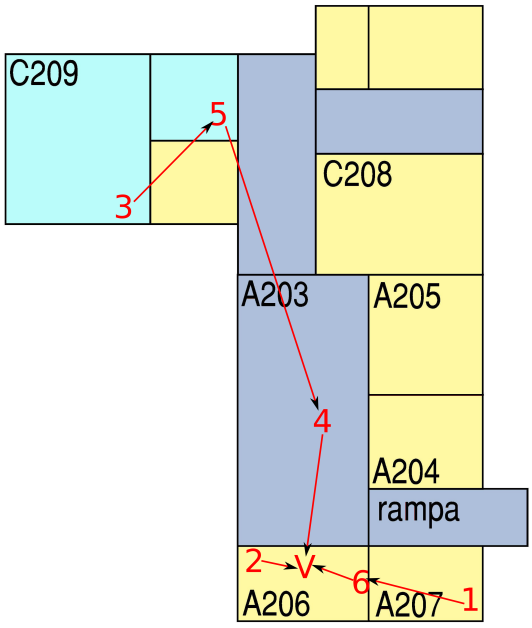
4.2 Testování sítě

Samotná implementovaná síť byla otestována v budově VUT FIT. Zaměřil jsem se také na schopnost směrování dat ze senzorových uzlů do výchozího uzlu. Síť byla sestavena z celkového počtu sedmi uzlů. V tabulce 4.1 můžete vidět konfiguraci jednotlivých uzlů. Měření bylo provedeno dvakrát pro ověření funkčnosti celé sítě a jejího chování. V prvním měření používaly všechny senzorové uzly senzor DS18B20 pro měření teploty. Uzel s ID 1 odesílal hodnoty přibližně každých pět minut, ostatní uzly odesílaly své hodnoty každých deset minut a do dalšího měření přešly uzly do režimu spánku.

Tabulka 4.1: Konfigurace uzlů.

Druh desky Arduino	ID uzlu	Druh uzlu	Druh antény XBee modulu
UNO	-	Výchozí uzel	Čipová anténa
UNO	1	Senzorový uzel	Čipová anténa
FIO	2	Senzorový uzel	Drátěná anténa
FIO	3	Senzorový uzel	Drátěná anténa
FIO	4	Senzorový uzel	Drátěná anténa
FIO	5	Senzorový uzel	Drátěná anténa
FIO	6	Senzorový uzel	Čipová anténa

Senzorové uzly sestavené z desky Arduino FIO byly napájeny pomocí připojené baterie. První část testování byla rozdělena do dvou částí. V první části bylo navrhnuto rozmístění jednotlivých senzorů po místnostech tak, aby všechny senzorové uzly komunikovaly s výchozím uzlem. Na obrázku 4.4 můžete vidět prvotní rozložení senzorů.



Obrázek 4.4: Rozmístění uzlů v první částí měření.

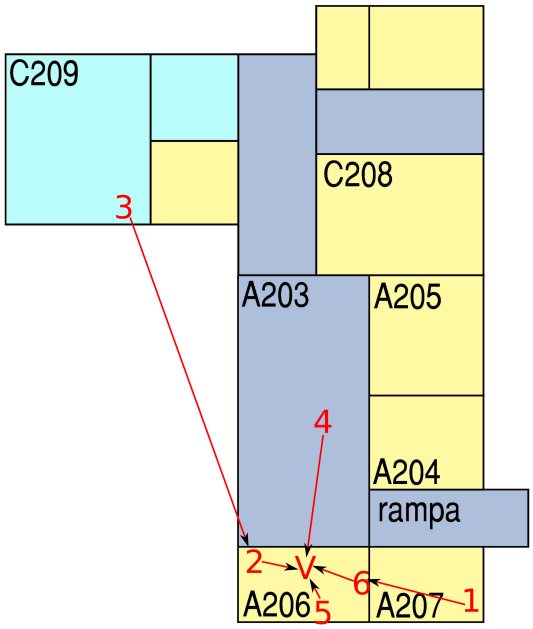
Toto rozložení senzorových uzlů bylo vybráno po řadě menších měření k určení dosahu jednot-

livých senzorových uzlů. Při těchto testech se projevil rozdíl v anténách modulů XBee a senzorové uzly, které byly více vzdáleny od výchozího uzlu se k tomuto uzlu nepřipojily. Takovým příkladem může být senzorový uzel s ID 4, který obsahoval čipovou anténu. Tento uzel komunikoval s výchozím uzlem, ale uzel s ID 5 nebyl v dosahu tohoto uzlu. Při změně modulu XBee za modul s drátěnou anténu,se uzel s ID 5 propojil s tímto uzlem a jeho zprávy byly přeposlány koordinátorovi.

Výchozí uzel pod označením V byl umístěn do místnosti A206 a byl připojen k počítači, kde se nacházela aplikace na uložení naměřených dat. V této místnosti se také nacházel senzorový uzel s ID 2 umístěný na okenním parapetu a uzel s ID 6 umístěný u vchodu do místnosti A207. V místnosti A207 se nacházel uzel s ID 1, který byl připojen k počítači a umístěn na okenním parapetu. Uzel číslo ID 4 se nacházel v zásuvce na chodbě A203. Tento uzel sloužil pro přenos zpráv z uzlů ID 5 a ID 3. Uzel s ID 5 byl umístěn ke vstupu do kuchyňky místnosti C209 z důvodu rušení signálu v jiných částech této místnosti. Uzel s ID 3 se nacházel na okenním parapetu místnosti C209. Takto rozestavené senzorové uzly byly ponechány po dobu jednoho dne.

V druhé části testování byl senzorový uzel ID 5 přesunut do místnosti A206. Na obrázku 4.5 můžete vidět nové rozmístění senzorových uzlů. Při přemístění došlo ke změně cesty k výchozímu uzlu pro uzel s ID 3, který využil uzel s ID 2 umístěný na okenním parapetu místnosti A206 ke spojení s výchozím uzlem. Při této změně musel být uzel s ID 3 umístěn na vyvýšenější místo, aby došlo ke spojení. Měření opět probíhalo po dobu jednoho dne.

Výsledkem tohoto měření bylo otestování dynamické změny topologie sítě a otestování dosahu XBee modulů. Při změně umístění uzlu s ID 5 se uzel s ID 3 dokázal propojit s uzlem ID 2. Pro rozmístění uzlů bylo také nutno vzít v úvahu typ antény modulů XBee.



Obrázek 4.5: Rozmístění uzlů v druhé části měření.

Druhé testování bylo více zaměřeno na celkovou výdrž baterií připojených k Arduino FIO. Jednotlivé senzory byly rozestaveny dle obrázku 4.4. Senzorový uzel s ID 1 používal senzor RHT03 a ostatní uzly senzor DS18B20. V tabulce 4.2 můžete vidět celkovou výdrž jednotlivých uzlů. Senzorový uzel s ID 1 byl připojen ke zdroji napájení po celou dobu měření. Uzel s ID 1 posílal své hodnoty každých 5 minut, uzel s ID 2 po 24 sekundách a ostatní uzly každých 10 minut. Z celkového

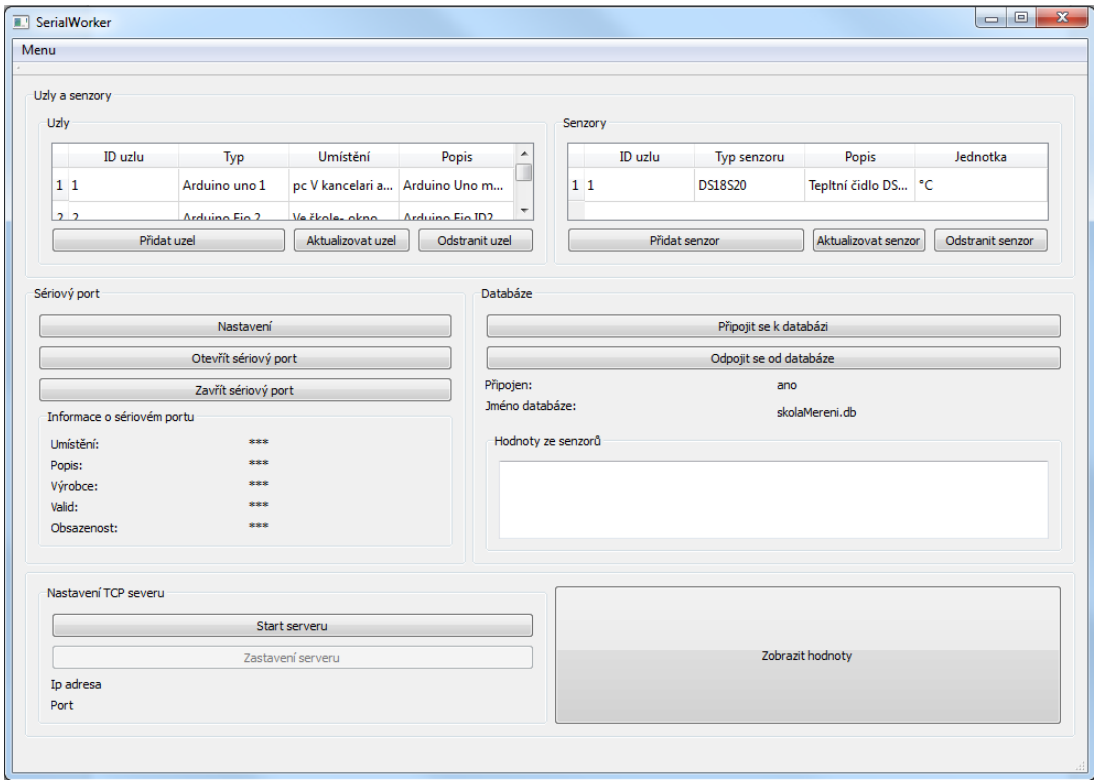
Tabulka 4.2: Testování výdrže baterií.

Druh desky Arduino	ID uzlu	Začátek měření	Konec měření	Celkový čas
UNO	1	10:42:40 04.05.2012	14:49:15 07.05.2012	76:06:35
FIO	2	10:37:38 04.05.2012	02:56:57 06.05.2012	40:19:19
FIO	3	10:50:48 04.05.2012	06:38:33 05.05.2012	19:47:45
FIO	4	10:41:22 04.05.2012	07:31:22 05.05.2012	20:50:00
FIO	5	10:46:48 04.05.2012	07:35:48 05.05.2012	20:49:00
FIO	6	10:39:54 04.05.2012	00:57:47 06.05.2012	38:17:53

času můžete vidět přerušení u uzlů s ID 3, 4 a 5, kdy při výpadku uzlu ID 4 se uzly nedokázaly spojit s výchozím uzlem. Úspora energie při uspání mikrokontroléru ATmega nedosahuje příliš velkých úspor a bylo by vhodné prozkoumat uspávání pomocí XBee modulů. Přibližná maximální doba funkčnosti byla okolo 40 hodin provozu, ale část uzlů se odpojila od sítě již po 20 hodinách provozu. Obrázek všech uzlů a grafů naměřených hodnot můžete vidět v příloze C.

4.3 Aplikace

Při implementaci aplikace bylo postupováno dle návrhu uvedeného v sekci 3.2. Hlavní obrazovku aplikace můžete vidět na obrázku 4.6. Hlavní obrazovka aplikace je rozdělena do částí pro správu uzlů a senzorů, nastavení sériového portu, připojení k databázi, nastavení TCP serveru a zobrazení naměřených hodnot. V dalších částech této sekce budou rozepsány implementační detaily a problémy, které bylo nutno vyřešit.



Obrázek 4.6: Hlavní okno aplikace.

4.3.1 Databáze

Pro připojení databáze slouží tlačítko s názvem „Připojit se k databázi". Při výběru prázdné SQLite databáze si aplikace vytvoří potřebné tabulky. Po připojení databáze se v hlavním okně zobrazí jméno databáze a zda je připojena.

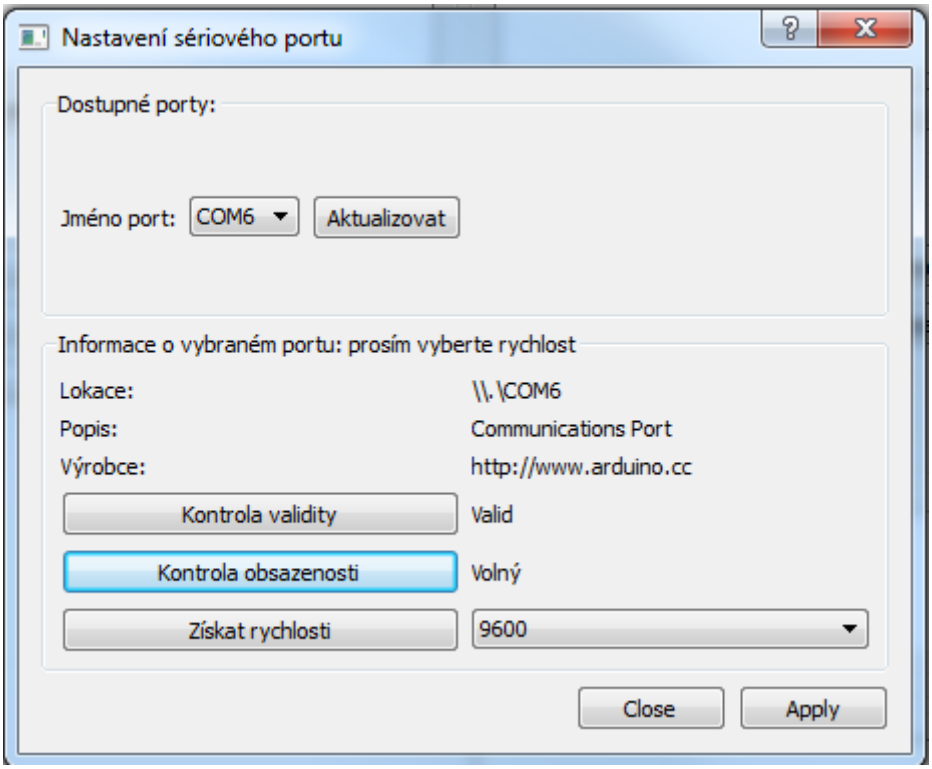
4.3.2 Správa uzlů a senzorů

Uživatel může vytvořit nové uzly a senzory, odebírat a upravovat je. Pro manipulaci s daty je nutné, aby byla připojena databáze.

4.3.3 Nastavení sériového portu

Pro nastavení sériového portu slouží tlačítko „Nastavení". Dialog pro výběr portu můžete vidět na obrázku 4.7. Pro výběr portu slouží rozbalovací seznam, který obsahuje jména dostupných portů. Při výběru portu se aktualizují údaje v dialogu a to lokace, popis a výrobce. Tyto údaje obsahuje třída `SerialPortInfo` knihovny *QSerialDevice*. Dále je nutné vybrat rychlost komunikace, se kterou komunikuje deska Arduino. Po stisknutí tlačítka „Získat rychlosti" se naplní rozbalovací seznam s dostupnými rychlostmi. Po výběru portu a rychlosti a potvrzení volby je nutné otevřít port. Hodnoty načtené z portu se zobrazí v části hlavní obrazovky „Hodnoty ze senzorů".

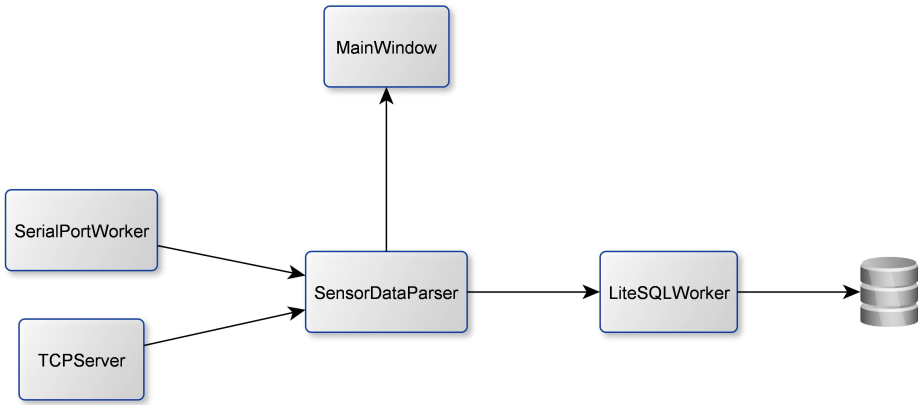
Pro práci se sériovým portem byla vybraná knihovna *QSerialDevice* pro její lepší zobrazování informací o dostupných portech, ale při testování docházelo k zamrznutí rozhraní aplikace. Tento problém způsobovala třída `SerialPort`, a proto byla nahrazena třídou `QextSerialPort` z knihovny s obdobným názvem.



Obrázek 4.7: Dialog pro výběr portu.

4.3.4 TCP server

Aplikace byla také rozšířena o možnost přijímat zprávy ze senzoru pomocí TCP komunikace. Po stisknu tlačítka „Start serveru" je nutné vyplnit port, na kterém bude server naslouchat. Toto rozšíření lze využít například při rozšíření výchozí uzlu o desku Arduino Ethernet shield. Na obrázku 4.8 můžete vidět příjem, zpracování a uložení zprávy od výchozího uzlu. Přijatá zpráva je rozdělena na jednotlivé hodnoty, které se zobrazí v hlavním okně aplikace a uloží do databáze.



Obrázek 4.8: Příjem, zpracování a uložení zprávy.

4.3.5 Zobrazení naměřených hodnot

Pro zobrazení naměřených hodnot byl použit QtWebkit, tři webové stránky a JavaScriptové knihovny Highcharts a JQuery. Po stisku tlačítka „Zobrazit hodnoty" se zobrazí první stránka se seznamem dostupných uzlů. Uživatel si musí zvolit, který uzel chce zobrazit. Pro zjednodušení zobrazení hodnot jsem se rozhodl vložit další stránku pro výběr dne, ve kterém probíhalo měření. Po výběru dne se zobrazí stránka s grafem a tabulkou s naměřenými hodnotami. Na obrázku 4.10 můžete vidět graf a na obrázku 4.9 tabulku s naměřenými hodnotami.

dht22 vlhkost

Zobraz záznamů 10 Hledat:

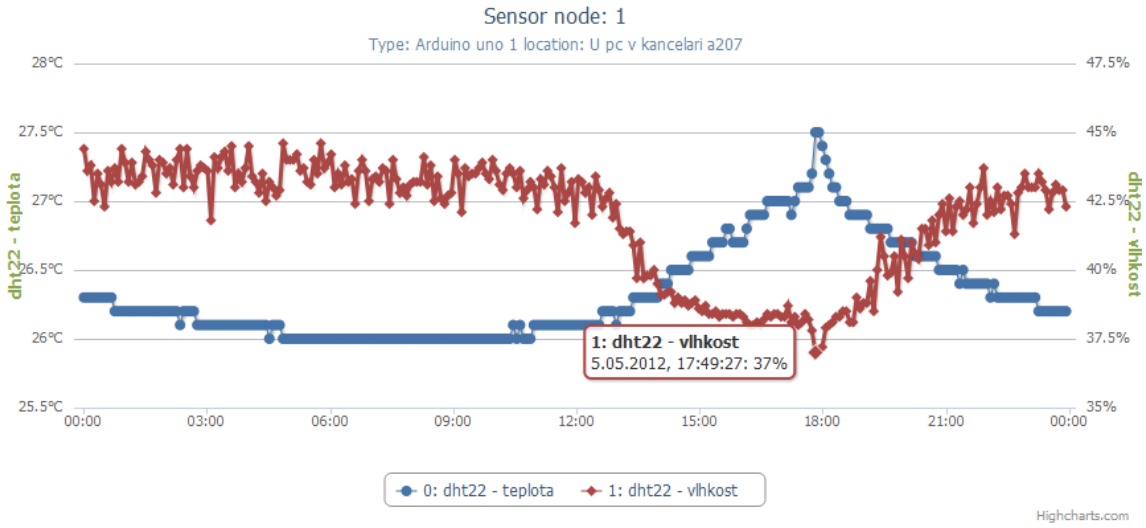
Datum a čas	Hodnota: %
00:00:34 05.05.2012	44.4 %
00:05:35 05.05.2012	43.6 %
00:10:36 05.05.2012	43.8 %
00:15:38 05.05.2012	42.5 %
00:20:39 05.05.2012	43.5 %
00:25:40 05.05.2012	43.1 %
00:30:41 05.05.2012	42.3 %
00:35:42 05.05.2012	43.6 %
00:40:43 05.05.2012	43.2 %
00:45:44 05.05.2012	43.7 %
Datum a čas	Hodnota: %

Zobrazuji 1 až 10 z celkem 287 záznamů Return Předchozí Další

Obrázek 4.9: Tabulka s hodnotami vlhkosti.

Tabulky obsahují dva sloupce pro čas a naměřenou hodnotu. Jednotlivé sloupce lze řadit vzestupně nebo sestupně při stisku malých šipek v pravém horním rohu sloupce. Pomocí rozbalovacího seznamu lze změnit počet záznamů, které tabulka zobrazuje. Tabulka také podporuje vyhledávání, které vyhledá zadaný řetězec ve sloupcích. Ve spodní části stránky se nachází tlačítko „Return" pro návrat na první stranu.

Graf umožňuje výběr plochy v grafu pro přiblížení a uživatel tak může zobrazit pouze specifický úsek grafu. Tlačítko v „Reset zoom" v pravém horním rohu grafu slouží pro návrat do původního zobrazení. Při najetí myši na bod v grafu se zobrazí malé okénko s časem a hodnotou bodu. Při kliknutí na název senzoru v legendě, která je umístěna ve spodní části grafu, lze vypnout zobrazení hodnot tohoto čidla. Při implementaci grafů a využití knihoven bylo nutno vyřešit několik problémů. Knihovna Highcharts v nejnovější verzi nebyla zcela funkční a možnost přiblížení nefungovala. Proto jsem využil dostupnosti starší verze, kde přiblížení funguje bez problémů. Další problém, který byl potřeba vyřešit, byl formát datumu pro data grafu, kdy metoda `Date.parse` nerozpoznala datum ve formátu ISO 8601. Tento problém jsem vyřešil přidáním knihovny `js-iso8601` pro rozpoznání tohoto formátu[1].



Obrázek 4.10: Graf s naměřenými hodnotami.

[1] https://github.com/csnover/js-iso8601

Kapitola 5

Možnosti rozšíření

V této kapitole jsou popsána možná budoucí rozšíření bezdrátové sítě z komponent Arduino a XBee, ale také na straně aplikace na uložení a zobrazení dat.

5.1 Bezdrátová síť

Bezdrátová síť by se do budoucna mohla rozšířit o cyklický režim spánku modulů XBee. Při testování této varianty nebyla síť stabilní a bylo by vhodné prozkoumat, jak je nutné nastavit moduly XBee pro správnou funkčnost cyklického režimu spánku. Kombinace časovače watchdog a režimu spánku tak nepřinesla velkou úsporu energie.

Další možnosti rozšíření je otestování API režimu namísto transparentního režimu komunikace mezi uzly. V API režimu je nutné vytvářet pakety pro odesílání dat, ale je možné specifikovat, kterému uzlu zprávu zaslat. Odesílatel zprávy se také dozví, zda bylo doručení zprávy úspěšné nebo zda došlo k chybě. Pomocí tohoto režimu by mohla být řešena ztráta dat při nepřipojení se k síti.

Jako rozšíření se také nabízí prozkoumání projektu Amarino, který spojuje operační systém Android pomocí Bluetooth s deskou Arduino a jeho možné úpravy pro jiné systémy a využití spolu s API režimem komunikace.

5.2 Aplikace

Aplikace by mohla obsahovat ovládaní uzlů v kombinaci s API režimem komunikace. Tato kombinace by umožnila ovládat zařízení připojené k desce Arduino nebo požádat desku o zaslání hodnoty ze senzoru.

Jako vhodné rozšíření se také jeví přidání mapy pro umístění uzlů, která by sloužila jako pomůcka pro lepší orientaci v rozmístění uzlů. V kombinaci s API režimem by se uživatel dozvěděl, zda se některý uzel neodpojil od sítě. Jednotlivé uzly by také mohly zobrazovat aktuální naměřené hodnoty a další informace o uzlu.

Kapitola 6

Závěr a zhodnocení

Tato diplomová práce se zabývá implementací bezdrátové senzorové sítě sestavené z komponent Arduino a modulů XBee. Při tvorbě návrhu bylo nutno nastudovat rozsáhlé množství literatury a materiálů zabývajících se open source platformou Arduino a o bezdrátových modulech XBee. Větší část materiálů je dostupná pouze v anglickém jazyce díky malému rozšíření těchto platforem v České republice.

Platforma Arduino je především určena na vytváření prototypů a jejich testování v reálných podmínkách. Díky otevřenosti platformy vzniklo mnoho rozšíření a další druhy desek. Takovou deskou je například deska Arduino FIO. Seznámení se s platformou Arduino bylo pro mne velmi přínosné díky její nenáročnosti a snadnému použití. V rámci návrhu bezdrátové sítě jsem se seznámil s bezdrátovými moduly XBee. Těchto modulů existuje více druhů a liší se podle obsaženého firmware, antény a verze. V rámci prozkoumání možností těchto modulů jsem se seznámil s proprietárním komunikačním standardem DigiMesh, který je dostupný pro moduly verze 1. Standard DigiMesh umožňuje snadné vytváření dynamických mesh sítí spolu s jednoduchým adresováním i snazším nasazením. V rámci návrhu byla navržena struktura bezdrátové sítě a prozkoumáno zapojení uzlů se senzory a možnost úspory energie. Pro aplikaci na zobrazení dat byl zvolen framework Qt a databáze SQLite. Kombinace těchto dvou prvků se jevila jako nejrozumnější volba. Pro zobrazení tabulek a grafů naměřených hodnot byly vybrány webové technologie v kombinaci s QtWebKit.

Všechny navržené uzly sítě byly úspěšně implementovány, pouze cyklický režim spánku modulů XBee nebyl zcela funkční a byl nahrazen uspáním v kombinaci watchdog a mikrokontroléru ATmega. Otestování bezdrátové sítě v budově VUT FIT bylo rozděleno do dvou částí, kdy první část se zaměřila na schopnost směrování dat ze senzorových uzlů do výchozího uzlu. V této části byla testována změna topologie sítě a bylo testováno, zda se některý uzel neodpojí. Druhá část byla zaměřena na výdrž baterií připojených k Arduino FIO. Přibližná maximální doba funkčnosti byla okolo 40 hodin provozu, ale část uzlů se odpojila od sítě již po 20 hodinách provozu.

Aplikace dosáhla všech na ni kladených požadavků, ať už v oblasti ovládání, tak v oblasti zobrazení naměřených dat. Uživatel má jednoduchý výběr, jaký uzel, a v který den chce zobrazit. Data jsou zobrazena pomocí interaktivního grafu s možností přiblížení a tabulky, která umožňuje vyhledávat a řadit data dle sloupců. Aplikace také obsahuje menší rozšíření o možnost spuštění TCP serveru pro příjem dat ze sítě. Samotná aplikace může být použita také v kombinaci s jinými zařízeními při zachování navrženého komunikačního protokolu.

V rámci budoucího rozšíření by bylo vhodné prozkoumat cyklický režim spánku v kombinaci s API režimem komunikace. Aplikace na zobrazení dat by mohla obsahovat mapu s rozmístěním jednotlivých uzlů.

Literatura

[1] Boronczyk, T.: *Beginning PHP 6, Apache, MySQL 6 Web Development*. Programmer to Programmer Series, Wiley Pub., 2009, ISBN 9780470391143, 840 s.

[2] Ezust, A.; Ezust, P.: *Introduction to Design Patterns in C++ with Qt*. Prentice Hall Open Source Software Development Series, Pearson Education, 2011, ISBN 9780132851633, 768 s.

[3] Faludi, R.: *Building Wireless Sensor Networks: With ZigBee, XBee, Arduino, and Processing*. O'Reilly Series, O'Reilly Media, 2010, ISBN 9780596807733, 318 s.

[4] Igoe, T.: *Making Things Talk: Physical Computing with Sensors, Networks, and Arduino*. O'Reilly Series, O'Reilly Media, 2011, ISBN 9781449392437, 496 s.

[5] Jay A. Kreibich: *Using SQLite*. O'Reilly Series, O'Reilly Media, 2010, ISBN 9780596521189, 528 s.

[6] Liu, T.: *Digital relative humidity and temperature sensor RHT03*. MaxDetect Technology Co., Ltd., 2010.

[7] Matthew, N.; Stones, R.: *Beginning Databases With Postgresql: From Novice To Professional*. Expert's voice in Open Source, Apress, 2005, ISBN 9781590594780, 664 s.

[8] Oxer, J.: *Arduino Shield List* [online]. Poslední modifikace: 2011 [cit. 2011-11-14]. URL http://shieldlist.org/

[9] Vaswani, V.: *MySQL Database Usage & Administration* . Advanced skills from the experts, McGraw-Hill, 2009, ISBN 9780071605496, 368 s.

[10] Wheat, D.: *ArduinoInternals*. Apress, 2011, ISBN 9781430238829, 392 s.

[11] *8-bit Atmel Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash* [online]. Poslední modifikace: 2012 [cit. 2012-04-17]. URL http://www.atmel.com/devices/ATMEGA328.aspx?tab=documents

[12] *megaAVR* [online]. Poslední modifikace: 2011 [cit. 2011-11-12]. URL http://www.atmel.com/products/microcontrollers/avr/megaAVR.aspx

[13] *Online Reference Documentation* [online]. Poslední modifikace: 2012 [cit. 2012-04-19]. URL http://doc.qt.nokia.com/

[14] *Arduino FIO* [online]. Poslední modifikace: 2011 [cit. 2011-11-13]. URL http://arduino.cc/en/Main/ArduinoBoardFio

[15] *Arduino Introduction* [online]. Poslední modifikace: 2011 [cit. 2011-11-12].
URL http://www.arduino.cc/en/Guide/Introduction

[16] *Arduino UNO* [online]. Poslední modifikace: 2011 [cit. 2011-11-13].
URL http://arduino.cc/en/Main/ArduinoBoardUno

[17] *BuildProcess* [online]. Poslední modifikace: 2011 [cit. 2011-12-09].
URL https://code.google.com/p/arduino/wiki/BuildProcess

[18] *DataTables* [online]. Poslední modifikace: 2012 [cit. 2012-04-25].
URL http://datatables.net/index

[19] *DataTables multiple tables example* [online]. Poslední modifikace: 2012 [cit. 2012-04-25].
URL http://datatables.net/release-datatables/examples/basic_init/multiple_tables.html

[20] *DHT-sensor-library* [online]. Poslední modifikace: 2012 [cit. 2012-04-28].
URL https://github.com/adafruit/DHT-sensor-library

[21] *DHTxx Sensors* [online]. Poslední modifikace: 2012 [cit. 2012-04-28].
URL http://www.ladyada.net/learn/sensors/dht.html

[22] *DS18B20Programmable Resolution1-Wire Digital Thermometer* [online]. Poslední modifikace: 2008 [cit. 2011-12-10].
URL http://datasheets.maxim-ic.com/en/ds/DS18B20.pdf

[23] *dygraphs JavaScript Visualization Library* [online]. Poslední modifikace: 2012 [cit. 2012-04-25].
URL http://dygraphs.com/

[24] *First Sketch* [online]. Poslední modifikace: 2011 [cit. 2011-12-08].
URL http://www.arduino.cc/en/Tutorial/Sketch

[25] *Flotr2* [online]. Poslední modifikace: 2012 [cit. 2012-04-25].
URL http://www.humblesoftware.com/flotr2/documentation

[26] *Frequently Asked Questions* [online]. Poslední modifikace: 2011 [cit. 2011-11-13].
URL http://arduino.cc/en/Main/FAQ

[27] *FTDI Basic Breakout* [online]. Poslední modifikace: 2010 [cit. 2011-12-10].
URL https://www.sparkfun.com/products/9873

[28] *Highcharts Demo Gallery* [online]. Poslední modifikace: 2012 [cit. 2012-04-25].
URL http://www.highcharts.com/demo/combo-dual-axes

[29] *https://gitorious.org/qserialdevice* [online]. Poslední modifikace: 2012 [cit. 2012-04-20].
URL https://gitorious.org/qserialdevice

[30] *microSD Shield* [online]. Poslední modifikace: 2011 [cit. 2011-12-09].
URL https://www.sparkfun.com/products/9802

[31] *OneWire Library* [online]. Poslední modifikace: 2012 [cit. 2012-04-28].
URL http://www.pjrc.com/teensy/td_libs_OneWire.html

[32] *qextserialport* [online]. Poslední modifikace: 2012 [cit. 2012-04-20].
URL https://code.google.com/p/qextserialport/

[33] *Series 1 XBee / XBee-PRO 802.15.4 and DigiMesh 2.4 Compatibility Factsheet* [online].
Poslední modifikace: 2011 [cit. 2011-12-10].
URL
http://www.digi.com/support/kbase/kbaseresultdetl.jsp?id=3150

[34] *The Arduino project: open-source electronics prototyping introduced by Massimo Banzi [online]*. Poslední modifikace: 2011 [cit. 2011-11-12].
URL http://www.transmediale.de/
arduino-project-open-source-electronics-prototypin

[35] *The DigiMesh Networking Protocol* [online]. Poslední modifikace: 2011 [cit. 2011-12-10].
URL http://www.digi.com/technology/digimesh/

[36] *What is Highcharts?* [online]. Poslední modifikace: 2012 [cit. 2012-04-25].
URL http://www.highcharts.com/products/highcharts

[37] *Wireless Mesh Networking ZigBee vs. DigiMesh* [online]. Digi International, 2008 [cit. 2011-12-09]. Dostupné z.
URL http://www.digi.com/pdf/wp_zigbeevsdigimesh.pdf

[38] *XBee Explorer USB* [online]. Poslední modifikace: 2010 [cit. 2011-12-10].
URL https://www.sparkfun.com/products/8687

[39] *XBee Shield* [online]. Poslední modifikace: 2011 [cit. 2011-11-14].
URL https://www.sparkfun.com/products/9976

[40] *XBee/XBee-PRO DigiMesh 2.4 RF Modules* [online]. Digi International, 2010 [cit. 2011-12-09]. Dostupné z.
URL ftp://ftp1.digi.com/support/documentation/90000991_B.pdf

[41] *X-CTU Configuration and Test Utility Software* [online]. Poslední modifikace: 2008 [cit. 2011-12-30].
URL http://ftp1.digi.com/support/documentation/90001003_A.pdf

[42] *Arduino* [online]. 2011, Poslední modifikace: 2011 [cit. 2011-11-12].
URL http://arduino.cc/en

Příloha A

Obsah CD

Součástí této práce je CD-ROM medium, na kterém se nachází následující obsah:

- Technická zpráva.
- Zdrojové kódy aplikace pro zobrazování dat.
- Přeložená aplikace pro platformu Windows.
- Zdrojové kódy jednotlivých senzorových uzlů.
- Databázové soubory s naměřenými hodnotami.

Příloha B

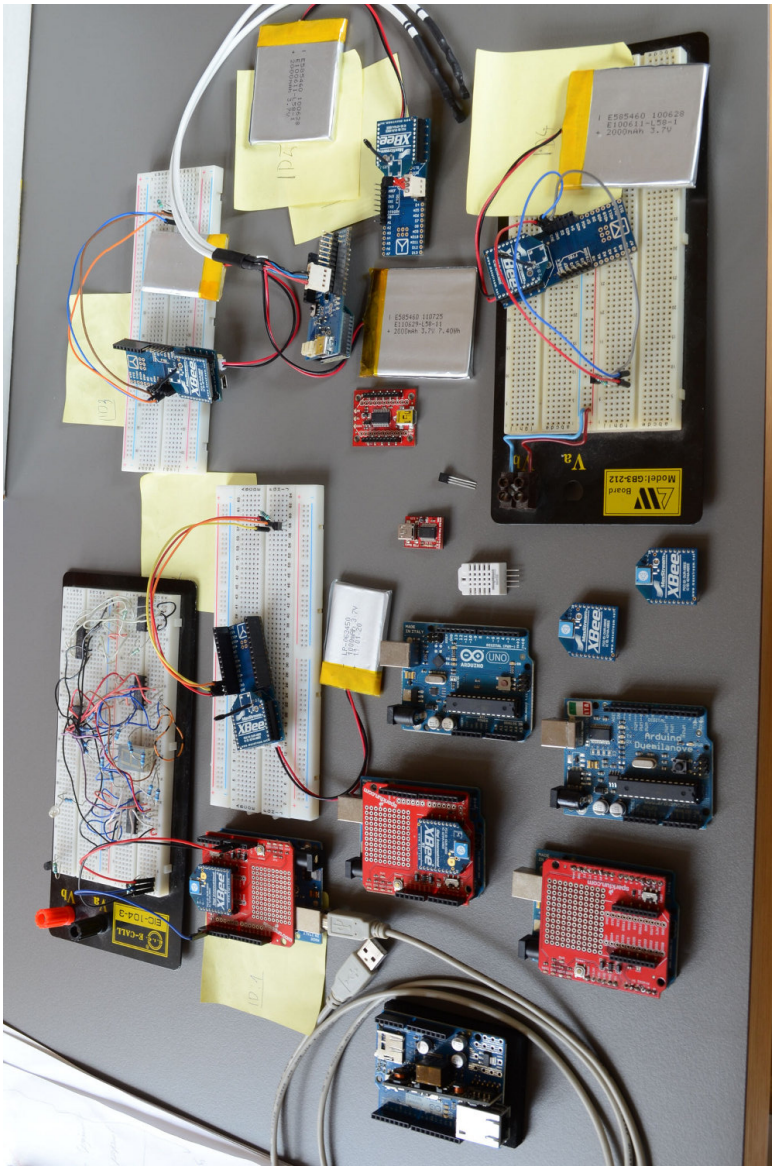
Zdrojové kódy

Kód B.1: Přečtení teploty ze senzoru DS18B20

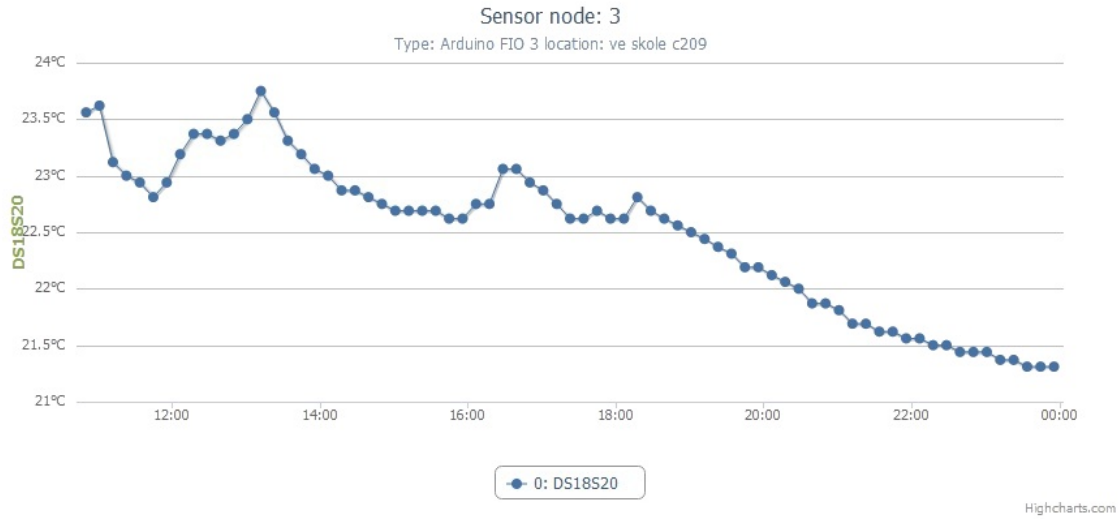
```
#include <OneWire.h>
int DS18S20_Pin = 2; //DS18S20 D2 na desce Arduino
//inicializace knihovny
OneWire ds(DS18S20_Pin);
...
float getTemp(){
  byte data[12];
  byte addr[8];
  //vyhledani senzoru
  if ( !ds.search(addr)) {
    ds.reset_search();//reset vyhledavani
    return NAN;
  }
  //v manualu adresa - 64-bit lasered rom code
  if ( OneWire::crc8( addr, 7) != addr[7]) {
    return NAN;
  }
  //kontrola zda jde o zarizeni DS18B20 - obsazeno v manualu
  if (addr[0] != 0x28) {
    return NAN ;
  }
  ds.reset();//reset, pred komunikaci se zarizenim
  ds.select(addr);//vyber zarizeni
  ds.write(0x44,0); // 0x44 prikaz pro konverzi teploty
  delay(850);
  ds.reset();//reset, pred komunikaci se zarizenim
  ds.select(addr);//vyber zarizeni
  ds.write(0xBE); //precteni scratchpad

  for (int i = 0; i < 9; i++) { // 9 bytu precist
    data[i] = ds.read();
  }
  ds.reset_search();/
  byte MSB = data[1];
  byte LSB = data[0];
  //0.0625 kooficient konverze mezi teplotou
  //a hodnotou v senzoru
  float TemperatureSum = ((MSB << 8) | LSB) / 16;
  return TemperatureSum;
}
```

Příloha C

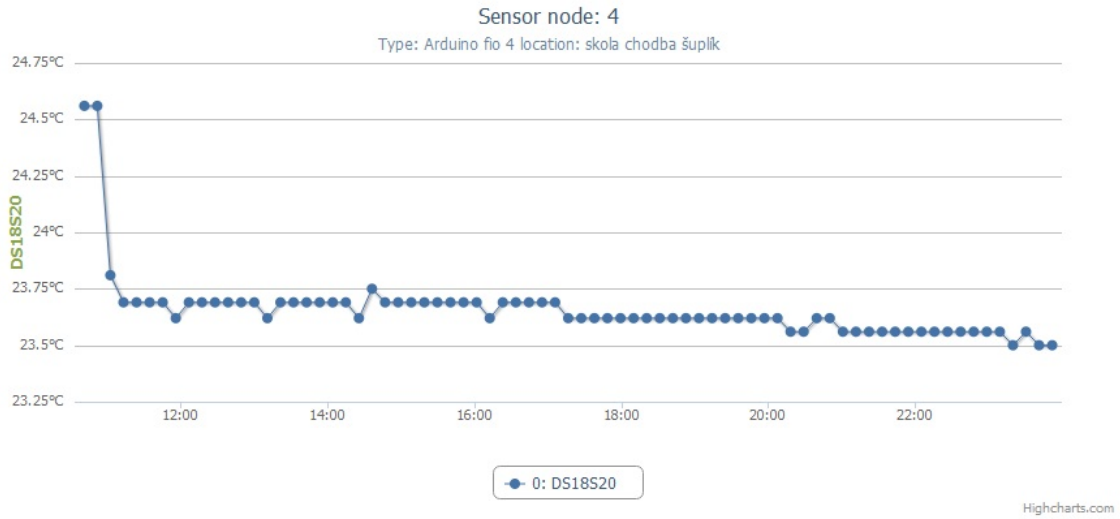
Obrázky



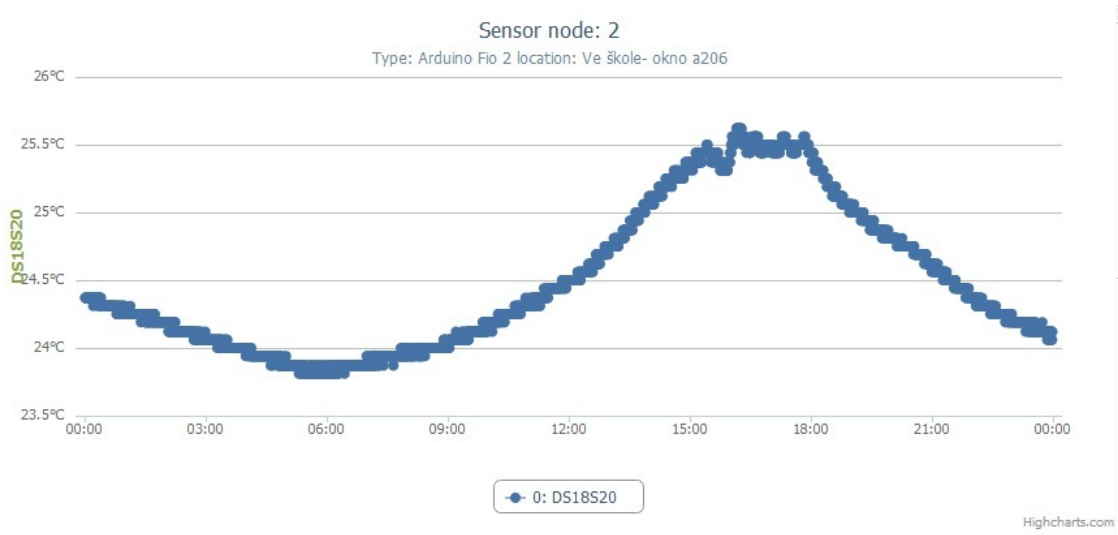
Obrázek C.1: Použité uzly pro testování sítě..



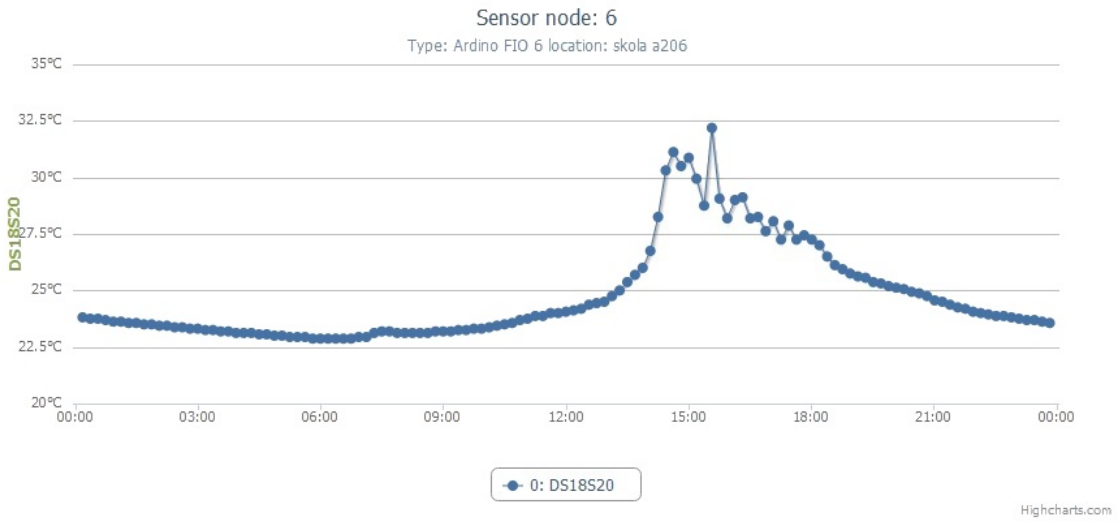
Obrázek C.2: Uzel ID 3 ze dne 4.5.2012.



Obrázek C.3: Uzel ID 4 ze dne 4.5.2012.



Obrázek C.4: Uzel ID 2 ze dne 5.5.2012.



Obrázek C.5: Uzel ID 6 ze dne 5.5.2012.