Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Senzorové sítě za využití platformy Arduino a komunikace pomocí ZigBee a XBee

Bakalářská práce

Autor: Martin Šustr
Studijní obor: Aplikovaná informatika

Vedoucí práce: Mgr. Josef Horálek, Ph.D.

Hradec Králové Leden 2018

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 22.4.2018 Martin Šustr

Poděkování:

Děkuji vedoucímu bakalářské práce Mgr. Josefu Horálkovi, Ph.D. za metodické vedení práce, cenné rady a odborný dohled.

Anotace

Bakalářská práce se zabývá možnostmi využití platformy Arduino v kombinaci s XBee moduly pro stavbu bezdrátových senzorových sítí. Práce popisuje vlastnosti senzorových sítí a možnosti jejich použití, dále popisuje platformu Arduino a způsoby bezdrátové komunikace, které nabízejí moduly XBee. Informace získané v teoretické části byly využity pro stavbu jednoduché senzorové sítě založené na deskách Arduino, které pro komunikaci využívají modulů XBee. Vytvořená senzorová síť byla podrobena testování základních přenosových parametrů jako je síla signálu nebo informace o počtu přijatých či odeslaných paketů. Data poskytnutá senzorovou sítí byla následně zpracována v aplikaci psané v Processingu, která graficky zajistila jejich reprezentaci. Práce prokázala, že je možné vystavět senzorovou síť pomocí kombinace Arduina a XBee modulů.

Abstract

Title: Sensor networks using the Arduino platform and communication via ZigBee and XBee

The bachelor thesis deals with possibilities of using Arduino platform in combination with XBee modules for building of wireless sensor network. This thesis describes the behavior of sensor networks and the possibilities of their using, next describes the Arduino platform and the wireless communication methods offered by XBee modules. Information obtained in the theoretical part was used to build a simple sensor network based on Arduino boards, which using XBee modules for communication. Built sensor network was tested for basic transmission parameters such as signal strength or information about received or sent packets. Data given by the sensor network was processed in an application written in Processing, which graphically ensured their representation. The work demonstrated that it is possible to build a sensor network based on a combination of Arduino and XBee modules.

Obsah

Úvod ... 1

1 Bezdrátové senzorové sítě ... 3

1.1 Historie ... 3

1.2 Definice ... 4

1.3 Vlastnosti ... 4

1.3.1 Výhody a nevýhody ... 4

1.3.2 Tolerance proti chybám ... 5

1.3.3 Škálovatelnost a redundance ... 5

1.3.4 Výrobní náklady ... 5

1.3.5 Prostředí ... 6

1.3.6 Spotřeba elektrické energie ... 6

1.4 Topologie ... 6

1.5 Možnosti využití ... 7

1.5.1 Zdravotnictví ... 7

1.5.2 Zemědělství ... 8

1.5.3 Kontrola ovzduší uvnitř budov ... 8

1.5.4 Rozsáhlé sítě ... 8

2 Arduino ... 10

2.1 Představení projektu Arduino ... 10

2.1.1 Počátky projektu Arduino ... 10

2.1.2 Další desky ... 12

2.2 Shieldy ... 18

2.3 Arduino IDE ... 19

2.3.1 Stažení a instalace ... 19

2.3.2 Nastavení ... 19

2.3.3 Psaní kódu 20
2.4 Wiring 21
2.4.1 Základní struktura jazyka 21
2.4.2 Práce s digitálními a analogovými vstupy a výstupy 21
2.4.3 Speciální funkce 22
2.4.4 Sériová komunikace 23
2.4.5 Myš či klávesnice pomocí Arduina 25
2.5 Processing 26
2.5.1 Základní informace 26
2.5.2 Arduino a Processing 27
2.6 Arduino vs. Raspberry Pi 27
3 XBee 28
3.1 Protokoly 28
3.1.1 IEEE 802.15.4 28
3.1.2 ZigBee 29
3.2 Moduly 31
3.2.1 Rozdělení dle Series 31
3.2.2 XBee nebo XBee PRO 31
3.2.3 Druhy antén 32
3.2.4 Rozdělení podle frekvence 33
3.3 X-CTU 34
3.4 Komunikace s XBee v režimu AT 34
3.4.1 Transparency-mode 35
3.4.2 Command-mode 35
3.4.3 AT příkazy 35
3.5 API-mode 36

3.5.1 API protokol a jeho struktura 36
3.5.2 Typy rámců 37
4 Praktická část 44
4.1 Stanovení cílů 44
4.2 Návrh senzorové sítě 44
4.2.1 Použité komponenty 44
4.2.2 Schéma senzorové sítě 46
4.2.3 Nastavení modulů XBee 50
4.2.4 Přenos dat 50
4.2.5 Měření teploty 57
4.3 Ověření funkcionality 58
4.4 Zhodnocení aplikace 60
4.5 Možnosti rozšíření 61
5 Shrnutí výsledků 62
6 Závěry a doporučení 64
Seznam použité literatury 65
Příloha 1: Struktura souboru se zdrojovými kódy 69
Příloha 2: Zadání práce 69

Seznam obrázků

Obr. 1 – Druhy topologií 7
Obr. 2 – Arduino Serial 11
Obr. 3 – Arduino NG 12
Obr. 4 – Arduino Extreme 12
Obr. 5 – Arduino Lillypad 13
Obr. 6 – Arduino Mini 13
Obr. 7 – Arduino Nano 13
Obr. 8 – Arduino UNO R3 14
Obr. 9 – Arduino Leonardo 14
Obr. 10 – Arduino Mega 15
Obr. 11 – Arduino Fio 15
Obr. 12 – Arduino Due 16
Obr. 13 – Arduino Esplora 16
Obr. 14 – Arduino Robot 16
Obr. 15 – Arduino Yun 17
Obr. 16 – XBee shield 18
Obr. 17 – Blikání LED na desce – příklad 19
Obr. 18 – Nalezení desky v IDE 20
Obr. 19 – Bloky setup() a loop() 20
Obr. 20 – Ukázka sériové komunikace Wiring 24
Obr. 21 – Processing – ukázka kódu 26
Obr. 22 – IEEE 802.15.4 topologie 29
Obr. 23 – ZigBee podporované topologie 30
Obr. 24 – XBee vs. XBee PRO 32
Obr. 25 – Druhy antén modulů XBee 33
Obr. 26 – Aplikace X-CTU pro práci s XBee 34
Obr. 27 – Ukázka konfigurace pomocí AT 36
Obr. 28 – Schéma senzorové sítě 46
Obr. 29 – Schéma zapojení koordinátoru 47
Obr. 30 – Schéma zapojení end node 1 48

Obr. 31 – Schéma zapojení end node 2 49
Obr. 32 – Metoda getAndProcessFrame() 52
Obr. 33 – Metoda processRxResponse() 53
Obr. 34 – Metoda processTxStatus 54
Obr. 35 – Metody checkNodes() a sendResetFrame() 55
Obr. 36 – Metoda getFrame() 56
Obr. 37 – Metoda rightTemp() 57
Obr. 38 – Ukázka reprezentace dat v aplikaci 58
Obr. 39 – Ukázka selhání senzorového uzlu v aplikaci 59

Seznam tabulek

Tabulka 1 – Struktura rámce typu AT Command 38
Tabulka 2 – Struktura rámce typu AT Response 38
Tabulka 3 – Struktura rámce typu ZigBee Transmit Request 39
Tabulka 4 – Struktura rámce typu ZigBee Receive Packet 40
Tabulka 5 – Struktura rámce pro práci s I/O 41
Tabulka 6 – Struktura rámce TX Request 16 bit address 42
Tabulka 7 – Struktura rámce TX Status 42
Tabulka 8 – Struktura rámce RX (Receive) packet 16 bit address 43
Tabulka 9 – Nastavení modulů XBee 50

Úvod

Člověk má tendenci neustále vyvíjet nové věci, stále se zdokonalovat a zároveň najít prostředky k tomu, aby se jeho život stal snazším, pohodlnějším. Své fantazie se snaží realizovat. To, co bylo dříve nepředstavitelné, nereálné se dnes stává skutečností a nedílnou součástí běžného života. Lidstvo se neustále posouvá kupředu díky novým technologiím, čím dál tím rychleji.

V dnešní době se stále více dostává do povědomí pojem Internet věcí neboli IoT[1]. Téměř každý člověk v ekonomicky vyspělých státech vlastní jedno či více tzv. chytrých zařízení jako jsou smartphony, chytré hodinky, či náramky a další zařízení. Všechny tyto chytré přístroje mají jednu společnou věc – dokáží spolu komunikovat pomocí internetu.

Není už žádnou novinkou si pořídit právě takový smartphone, který při připojení s notebookem bez pomoci kabelu, ale právě internetu, sdílí fotografie, videa, hudbu, cokoliv, co je uložené na mobilním zařízení. Nebo chytré hodinky, komunikující s telefonem, které upozorní na příchozí hovor, zprávu nebo spočítají tepovou frekvenci.

Do budoucna se předpokládá, že lidé nebudou používat jen jedno či dvě chytrá zařízení, ale že bude každý využívat mnoho různých zařízení, které spolu budou komunikovat, jde například o technologii inteligentního domu. Nejdůležitější částí inteligentního domu je centrální systém, který umožňuje snadné a pohodlné ovládání celého domu. Centrální systém řídí klimatizaci, topení, osvětlení, ale i například pouští hudbu podle toho, kde se někdo nachází. Aby systém rozpoznal, kde se nachází nějaký objekt, musí mít síť senzorů, kterými „poznává“ svět.

Senzorové sítě mohou být postaveny na různých zařízeních, mezi nimiž probíhá komunikace. Může se jednat o propojení několika PC skrze LAN, ale cílem této bakalářské práce bude seznámení se senzorovými sítěmi založené na platformě Arduino s využitím protokolů IEEE 802.15.4, ZigBee a modulů XBee, komunikujících bezdrátově.

---

[1] Zkratka z anglického „Internet of Things“

V teoretické části se bude bakalářská práce zaměřovat na podrobnějším seznámením s možnostmi, které jsou nabízeny jednoduchým zařízením Arduino, dále se bude zabývat novými standardy pro komunikaci v bezdrátových sítích za použití modulů XBee.

Praktická část se bude zabývat vytvořením jednoduché senzorové sítě za využití těchto nových technologií a zhodnocením použitelnosti tohoto způsobu komunikace.

1 Bezdrátové senzorové sítě

1.1 Historie

Podle Chonga a Kumara[1] jsou pro rozvoj bezdrátových senzorových sítí potřebné technologie ze tří oblastí výzkumu: snímání, komunikace a výpočetní technologie, které zahrnují hardware, software i použité algoritmy.

Podobně jako i u jiných technologií se začaly senzorové sítě využívat pro potřeby armády, která měla dostatečné zdroje na výzkum nových technologií. Během studené války vznikl systém SOSUS. Šlo o systém senzorů zvuku, které byly umístěny na dně oceánu, a měly za úkol odhalit tiché sovětské ponorky. Dnes tento systém využívá NOAA, která pomocí něho monitoruje seismickou aktivitu u dna oceánu [2].

Moderní výzkum senzorových sítí odstartovala kolem roku 1980 agentura DARPA se svým program distribuované senzorové sítě (DSN). Tehdy byla senzorová síť chápána jako kombinace mnoha nízkonákladových snímacích uzlů, které spolu navzájem komunikují, ale přitom pracují samostatně a data jsou směřována do uzlu, který tyto informace nejlépe využije [3].

Největší rozmach nastal v 80. a 90. letech dvacátého století, kdy si americká armáda začala naplno uvědomovat potenciál senzorových sítí [4], díky kterým mohli vojáci zlepšit svůj přehled o okolí a lépe směřovat i útok. Kromě armády se začaly senzorové sítě využívat i ke komerčním účelům, což snížilo náklady na jejich pořízení [3].

1.2 Definice

Definice Sohraby, Minoli, a Znati [5] říká, že senzorová síť je infrastruktura, která se skládá ze senzorů, výpočetní techniky a komunikačních prvků, které dávají uživateli nástroje pro pozorování a reagování na události a jevy ve specifikovaném prostředí. Uživatel je obvykle občanský, vládní, komerční nebo průmyslový subjekt. Prostředí může být fyzický svět, biologický systém nebo informační systém. Systémy senzorových sítí jsou chápány jako důležitá technologie, která se v příštích několika letech setká s velkým nasazením pro řadu aplikací.

1.3 Vlastnosti

Senzorové sítě se skládají ze senzorových uzlů, což jsou velmi malá zařízení, která se svými rozměry blíží kubickému milimetru [6]. Malá zařízení jsou limitována množstvím energie, které mohou uchovávat, pro svůj provoz. Z tohoto důvodu, jsou také omezena v použití energeticky nenáročného hardwaru [7]. Vlivy prostředí mohou způsobit, že uzly nebudou fungovat.

Spousta těchto speciálních vlastností definuje faktory, na které je potřeba brát ohled při stavbě senzorových sítí. Mezi sledované faktory patří tolerance proti chybám, škálovatelnost, výrobní náklady, prostředí nasazení a zejména spotřeba elektrické energie [8].

1.3.1 Výhody a nevýhody

Podle Tiwariho a kol. [9] lze výhody shrnout takto:

1. Nastavení sítě lze provádět bez pevné infrastruktury.
2. Vhodné pro nepřístupná místa, jako jsou moře, hory, nebo hluboké lesy.
3. Flexibilní, pokud může nastat situace, kdy je potřeba další senzorové uzly.
4. Cena za výstavbu sítě je nízká.
5. Není potřeba mnoho kabelů.
6. Síť lze spravovat pomocí centralizovaného monitoringu.

Nevýhodami jsou [9]:

1. Méně bezpečné, protože hackeři mohou snáze získat přístup do přístupového bodu a získat všechny informace.
2. Nižší rychlost v porovnání s drátovou sítí.
3. Složitější konfigurace ve srovnání s drátovou sítí.
4. Ovlivnění přenosu okolím (stěny, mikrovlnná trouba, velké vzdálenosti a další).
5. Pro hackery je snazší odchytávat pakety, protože není možné ovlivnit šíření vln.
6. Stále nákladné (což je nejdůležitější).

1.3.2 Tolerance proti chybám

Senzory mohou selhat, nebo se vypnout v důsledku nedostatečné energie, mohou být fyzicky poškozeny nebo je může poškodit vliv prostředí. V každém případě by selhání jednoho uzlu nemělo mít vliv na funkčnost celého systému [8]. Při návrhu je důležité rozhodnout, jak moc mohou být chyby tolerovány. Určitě to bude jiné v případě teplotních senzorů v rodinném domě nebo v případě pohybových senzorů na bitevní poli.

1.3.3 Škálovatelnost a redundance

Počet uzlů se může pohybovat v řádu desítek či stovek, záleží na účelu senzorové sítě. V extrémních případech může počet uzlů dosáhnout řádu miliónů. Systémy musí být schopny s tímto počtem uzlů pracovat. Dále musí systém umět pracovat s různou hustotou senzorů [8]. Hustota se může pohybovat od několika senzorových uzlů až po několik set senzorových uzlů v oblastech, které mohou mít průměr menší než deset metrů [10].

1.3.4 Výrobní náklady

Jelikož senzorové sítě obvykle skládají z velkého množství jednotlivých senzorových uzlů, je velmi důležité brát v potaz i náklady za jeden uzel, protože v celkovém součtu to bude mít výrazný vliv na cenu senzorové sítě. Pokud jsou tyto náklady vyšší než při použití tradičních senzorů, pak nemá cenu vytvářet

bezdrátovou senzorovou síť [8]. Ovšem důležité jsou i náklady na provoz a ty mohou být u senzorových sítí nižší.

1.3.5 Prostředí

Senzorové uzly jsou obvykle nasazeny, tam kde je to pro člověka nebezpečné nebo pozorují jevy, při kterých by člověk neměl být přítomen. Mohou pracovat na dně oceánu, na bitevním poli, v kontaminovaném prostředí, uvnitř tornáda, ve velkém skladu atd. Tento výpis ukazuje, za jakých podmínek musí senzorové uzly pracovat.

1.3.6 Spotřeba elektrické energie

Jednou z nejdůležitějších aspektů je spotřeba elektrické energie. Senzorové uzly si obvykle musí vystačit pouze s omezeným množstvím energie, které jim dodává třeba baterie, nebo si musejí energii umět získat třeba pomocí solárních článků, ovšem ani ty nejsou nekonečným zdrojem energie nebo je nejde použít vůbec (dno oceánu). Spotřebu lze ovlivnit už výběrem platformy, na které je síť postavena, nebo efektivní topologií, kde je minimalizován počet přeskoků mezi uzly [11].

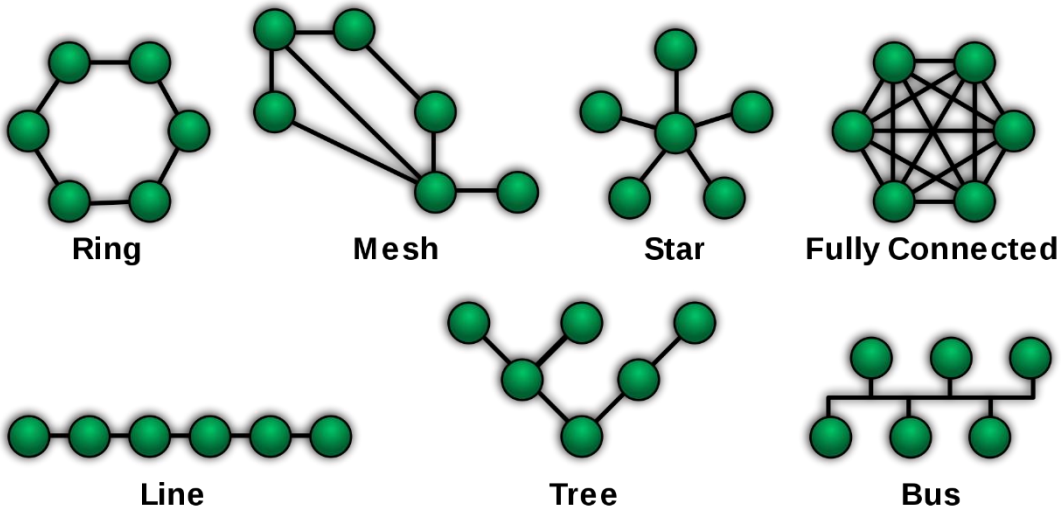
1.4 Topologie

Pro bezdrátové senzorové sítě je vytvoření a údržba správné síťové topologie velice důležitý úkol [12]. Důležité je vytvoření efektivní topologie, která bude odolná proti chybám a výpadkům.

V oblasti bezdrátových sítí se nejčastěji využívá topologie mesh. Mesh je velmi odolná proti výpadkům, a to díky redundantnímu propojení uzlů. Výhodou je i dobrá škálovatelnost. Nevýhodou této topologie je nutnost ochrany proti zacyklení, například pomocí Spanning Tree Protocol (STP), který odpojí redundantní propojení na softwarové úrovní [13], [14].

Obecně mesh topologie nepotřebuje ke svému fungování centrální prvek jako je tomu například u hvězdicové topologie, ale v senzorových bezdrátových sítích figurují jednotlivé uzly, které jsou propojeny s přístupovým bodem, který přijímá data od jednotlivých uzlů a rovnou je odesílá dále, obvykle už pomocí jiných

protokolů jako je standart Wi-Fi. V rozsáhlejších sítích je přístupový bod nahrazen kontrolérem, který zpracovává data a řídí jejich tok.



Obr. 1 – Druhy topologií

1.5 Možnosti využití

Senzorovou síť lze vybudovat na rozličných systémech využívající různý hardware i typy komunikace. Pokud má být cílem vytvořit snadno konfigurovatelnou senzorovou síť s nízkými provozními náklady, tak se s výhodou využívá platforma Arduino a komunikační moduly XBee. Možnosti použití jsou velmi rozsáhlé, jelikož Arduino je open-source platforma, která umožňuje snadné programování [15].

1.5.1 Zdravotnictví

Zulkifli, Harun a Azahar [5] využili kombinace k vytvoření senzorové sítě pro kontrolu srdečního tepu při sportovním tréninku. Srdeční tep byl měřen pomocí hrudního pásu, který posílá data přes protokol ANT+, což je protokol navržený přímo pro sportovní aktivity [6]. Data jsou přijímána modulem podporující ANT+ technologii, tyto data následně zpracovává Arduino Nano, která dále posílá data do XBee modulu. Jednotlivé XBee moduly data přeposílají centrálnímu modulu připojenému k počítači, kde může lékař sledovat naměřená data svých pacientů. Tento systém využívá hvězdicové topologie. Bezdrátová technologie umožňuje měření, která sportovci umožňují volnější pohyb.

1.5.2 Zemědělství

Očekává se, že populace dosáhne téměř deseti miliard lidí v roce 2050, to znamená, že už v roce 2035 bude potřeba vyprodukovat dvojnásobek jídla než dnes [16]. Ploch pro pěstování je stále méně, a tak je potřeba zefektivnit pěstování.

Baviskar [17] navrhl řešení v podobě vytvoření systému, který řídí a kontroluje skleník v reálném čase, jehož základem je komunikace pomocí protokolu IEEE 802.15.4. Systém využívá dvou typů koncových uzlů. Prvním typem je senzorový uzel, jež obsahuje senzory vlhkosti vzduchu, teploty, světla a půdní vlhkosti. Data ze senzorů zpracuje Arduino a pomocí protokolu odesílá data do centrálního systému, který rozhoduje, zda provést nějakou akci. Systém využívá hvězdicovou topologii. Pokud je potřeba provést nějakou akci, tak centrální systém pošle potřebné informace druhému typu uzlu, jež neobsahuje senzory, ale umožňuje ovládat klimatizaci, osvětlení a zavlažovače. Systém díky tomu udržuje ideální podmínky pro pěstování plodin.

1.5.3 Kontrola ovzduší uvnitř budov

Na podobném principu může fungovat i kontrola ovzduší uvnitř budov. Abraham [18] se snažil vytvořit nízkonákladový systém bezdrátové senzorové sítě, který bude měřit teplotu, vlhkost, ale i další vlastnosti vzduchu. Podle naměřených dat, podobně jako tomu bylo v případě Baviskarova skleníku, systém řídí ventilaci a proudění vzduchu v interiéru budovy. Čím se ovšem tento projekt liší, je topologie sítě. Tato síť využívá tzv. „ZigBee mesh“ topologii.

1.5.4 Rozsáhlé sítě

Možnosti využití jsou opravdu rozsáhle od vojenství přes zdravotnictví až po zemědělství. Díky flexibilitě lze bezdrátové senzorové sítě využít v rozličných oblastech a díky škálovatelnosti se může jednat o malé senzorové sítě s jednotkami až desítkami uzlů, nebo rozsáhlé se stovkami až tisícovkami uzlů.

Badescu [19] navrhl bezdrátovou senzorovou síť pro monitorování a ochranu tygrů ve volné přírodě. Tento systém se má rozkládat na dvou tisících čtverečních kilometrech. Plocha takových rozměrů vyžaduje tisíce senzorů, aby ji pokryly. To přináší řadu problémů, které se musí řešit.

V první řadě je to přísun elektrické energie v džungli. Elektrická síť v džungli neexistuje, a tak musejí být jednotlivé uzly napájeny z baterie. Právě proto Badescu zvolil moduly XBee, které spotřebovávají mnohem méně energie než zařízení standardu Wi-Fi. Pro nabíjení baterií by mohli sloužit sluneční kolektory.

Dalším problémem je zajistit, aby měli jednotlivé uzly co největší dosah. V Badescuově návrhu jsou umístěny senzory s moduly přibližně ve dvou metrech nad zemí, ale antény jsou umístěny až v korunách stromů. Tím je docíleno výrazného zvýšení dosahu, jelikož signál nemusí procházet hustým lesním porostem.

2 Arduino

Arduino je open-source elektronická platforma založená na jednoduchém softwaru a hardwaru. Jednotlivé desky Arduino fungují jako mikro kontrolér, který dokáže číst vstupy, zpracovávat je a odesílat data na výstupech. Původním záměrem projektu bylo vytvořit levnou výukovou pomůcku pro studenty.

Kapitola 2 byla zpracována zejména z těchto zdrojů: [15], [20].

2.1 Představení projektu Arduino

Kapitola 2.1 byla zpracována zejména z těchto zdrojů: [21], [22], [23].

2.1.1 Počátky projektu Arduino

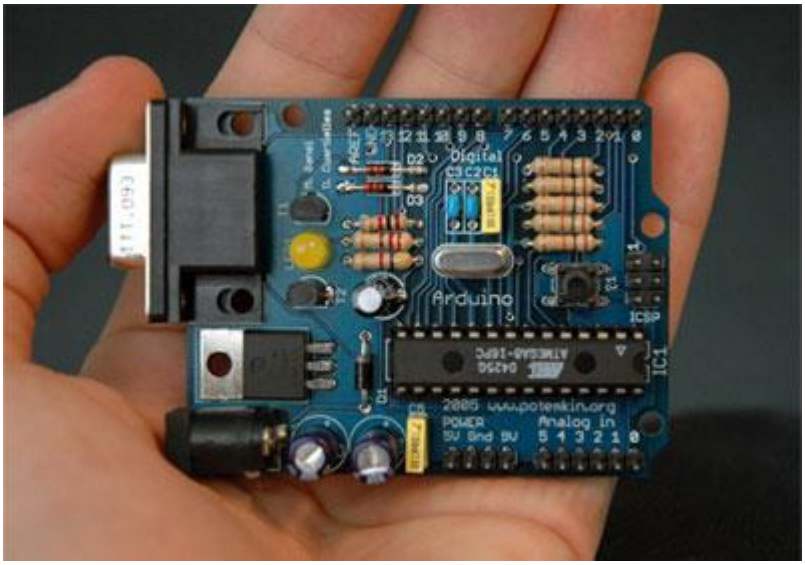
Arduino začalo jako projekt v roce 2005 v Interactive Design Institute in Ivrea (IDII) [23]. Arduino vycházelo i z některých předchozích projektů, na kterých v IDII pracovali. Cílem bylo vytvořit modernější nástroj pro studenty, než bylo možné tehdy sehnat. V té tobě většina studentů pracovala s nástrojem BASIC Stamp, který stál kolem 100$, což bylo pro studenty velmi nákladné. Díky tomu neudělali tolik práce, jelikož nechtěli utrácet spousty peněz na nákup mnoha desek. A tak začal Massimo Banzi a jeho tým zkoumat možné alternativy.

V IDII pracovali s jazykem Processing, jelikož jeden z jeho vynálezců byl učitelem v IDII. Tak se rozhodli tento jazyk využít a začali přemýšlet, jak upravit Processing pro hardware. V tomto ohledu sehrál klíčovou roli kolumbijský student Hernando Barragan.

Hernando Barragan již od roku 2003 pracoval na své diplomové práci [24]. Vedoucím diplomové práce byl právě Massimo Banzi. Cílem diplomové práce bylo usnadnit designerům práci s elektronikou. Barragan vytvořil nové vývojové prostředí a jazyk Wiring, který vycházel z jazyka Processing. Když už měl software, tak ještě vytvořil desku Wiring.

Deska Wiring se začala prodávat v roce 2004 za cenu okolo 50$. Což bylo o polovinu levnější než BASIC Stamp, ale stále byla cena vysoká pro některé lidi. Banzi a jeho tým využili Wiring a snažili se celou platformu zjednodušit, zlevnit a začali celý projekt vytvářet jako open-source, aby mohl kdokoliv přijít pomoct a spolupracovat.

Banzi a jeho tým přidali do Wiringu podporu levnějšímu mikroprocesoru ATmega8 od firmy Atmel a vytvořili tak separátní projekt nazvaný Arduino. První produkční verzí byla deska Arduino Serial, jak už název napovídá, jednalo se o desku, která pro připojení k PC využívala RS232. Byla vytvořena tak, aby si ji i každý trochu zručnější člověk mohl sestavit sám. Ještě ten samý rok byla vydána i verze s USB portem.



Obr. 2 – Arduino Serial

2.1.2 Další desky

Všechny Arduino desky mají mikroprocesor od firmy Atmel, kterému sekundují další elektronické součástky. Jelikož jde o open-source projekt, je možné si podle schémat, jež jsou volně k dispozici, postavit vlastní Arduino desku. Takto vyrobené desky jsou označovány za klony a nesmějí se jmenovat Arduino, jelikož jde o ochrannou značku. Někteří výrobci tak tyto klony prodávají pod názvy jako Funduino nebo XDRduino. Jde obvykle o plně kompatibilní zařízení. Originální desky lze poznat podle loga, názvu a také podle barvy desky, kde převažuje modrá, v novějších verzích se barva blíží více zelené.

Arduino Extreme, NG

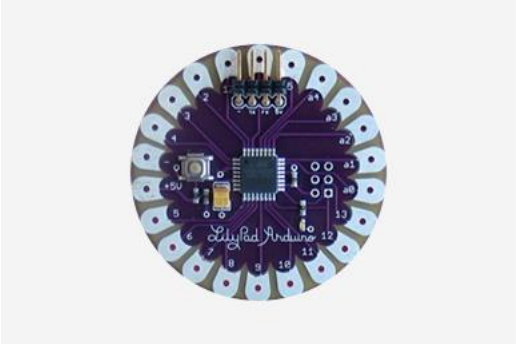
V roce 2006 byly vydány nové desky Arduino Extreme a Arduino NG. Oproti prvním deskám, které měly procesor ATmega8, tyto mají vylepšený procesor ATmega168 a dvojnásobnou kapacitu paměti. V Arduino Extreme se poprvé objevily LED diody indikující čtení nebo zápis dat a na rozdíl od Arduina USB a Serial má female piny. Arduinu NG byla navíc přidána vestavěná LED dioda spojená s 13 pinem.

Obr. 3 – Arduino NG

Obr. 4 – Arduino Extreme

Arduino Lillypad

V roce 2007 byla vydána deska Arduino Lillypad. Tato deska byla úplně jiné koncepce a měla za úkol rozšířit platformu o nositelná zařízení. Hlavními vlastnostmi jsou kompaktní velikost, tenké provedení, napájení přes baterii a snadné použití s vodivými nitěmi. Lillypad byla vyráběna v mnoha verzích, např. s USB portem nebo s jinými procesory Atmel.

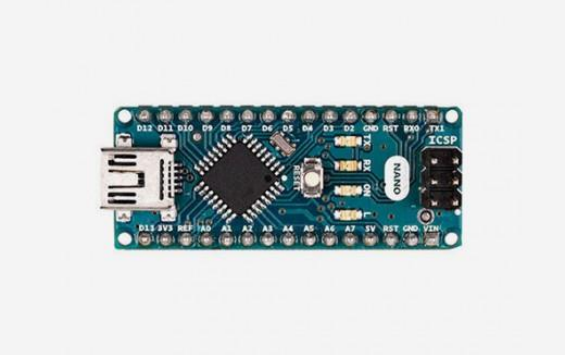


Obr. 5 – Arduino Lillypad

Arduino Nano, Mini

Arduino Nano a Mini jsou nejmenšími oficiálními deskami. Osazeny jsou novějším procesorem ATmega328. Arduino Nano se od Mini liší přítomností převodníku a USB portu.

Obr. 6 – Arduino Mini



Obr. 7 – Arduino Nano

Arduino Uno, Mega, Leonardo

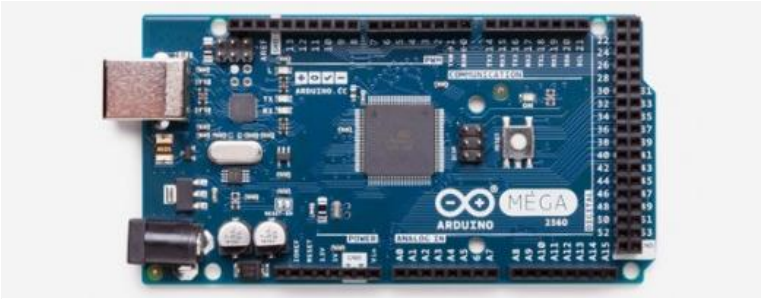
Arduino Uno je nejrozšířenější typ desky. První verze vznikla v roce 2010 a dnes se používá už Uno R3. Arduino Uno používá procesor ATmega328. Na desce je umístěno 6 analogových portů a 13 digitálních portů, pro připojení k počítači slouží klasický USB port.

Obr. 8 – Arduino UNO R3

Ardunino Leonardo je velmi podobný Arduinu Uno, používá ovšem procesor Atmega32u4, jehož výhodou je, že se může počítači hlásit jako myš či klávesnice. Dále se liší jiným type USB konektoru, pro Leonardo byl použit USB micro.

Obr. 9 – Arduino Leonardo

Arduino Mega je prodlouženou verzí Arduina Uno. Díky tomu je deska osazena větším počtem pinů pro řízení složitějších věcí. Navíc je zde prostor pro použití výkonnějších a tím pádem i větších čipů.



Obr. 10 – Arduino Mega

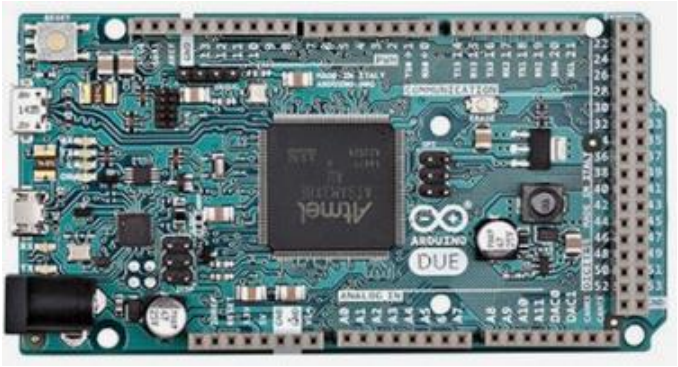
Arduino Fio

Deska Fio je první z rodiny Arduino, která byla vytvořena pro potřeby IoT. Použit byl opět procesor ATmega328. Používané napětí bylo sníženo na 3,3V a je tedy připravena k přímému připojení s komunikačními moduly jako jsou moduly XBee, které pracují se stejným napětím. Desky jako je Fio a jeho následovníci se vyznačují nízkou spotřebou, konektivitou a snadnou komunikací.

Obr. 11 – Arduino Fio

Arduino Due

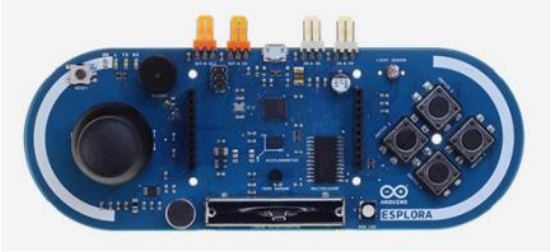
Arduino Due je prvním deskou, která je plně 32 bitová, díky použití procesoru Atmel SAM3X8E. Due vychází z Arduina Mega, liší použitým procesorem a přítomností dvou USB portů místo jednoho.



Obr. 12 – Arduino Due

Arduino Esplora, Robot

Arduino Esplora a Robot jsou speciálními typy platformy Arduino. Jejich cílem je výuka hrou. Srdcem obou desek je ATmega32u4. Nejedná se jen o samotnou desku, kde je potřeba ještě připojit jiné součástky, ale již o vybavené zařízení. Na deskách se nachází joystick, tlačítka, teploměr nebo piezzo bzučák.



Obr. 13 – Arduino Esplora

Obr. 14 – Arduino Robot

Arduino Yun

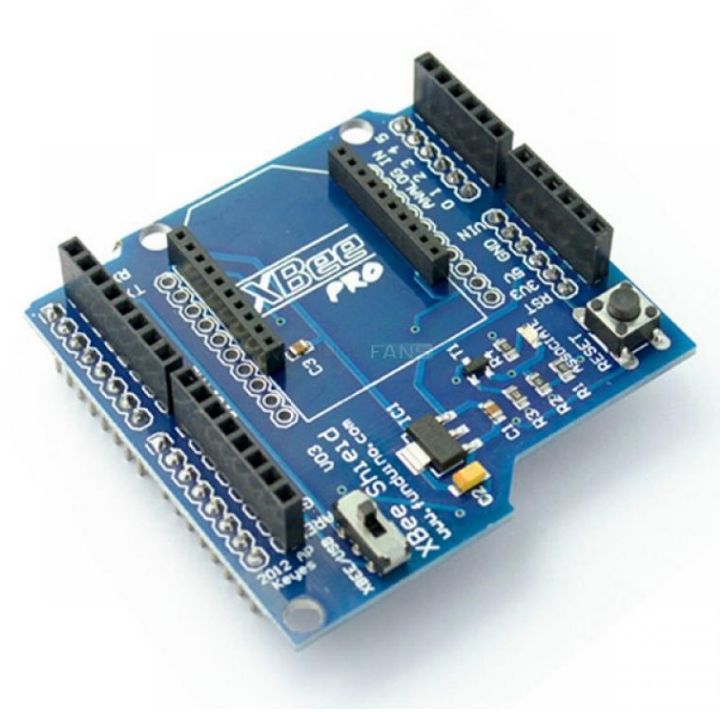
Zařízení Arduino Yun vzniklo v roce 2013 a otevřelo cestu platformě Arduino i do světa průmyslu. Jedná se o velmi výkonnou desku osazenou procesorem ATmega 32u4, kterému sekunduje Atheros AR9331, díky němuž je možné na Yun naistalovat i operační systém Linux. Další výhodou je i osazení desky dvěma USB porty a jedním Ethernet portem, to vše při zachování velikosti Arduina UNO.

Obr. 15 – Arduino Yun

2.2 Shieldy

Arduino se dá pořídit v mnoha různých provedeních, některá jsou menší, některá větší. To ovlivňuje také množství dostupných portů a konektorů. V případě, že je nutné rozšířit Arduino a další vlastnost jsou dvě možnosti. První je pořízení většího a vybavenějšího modelu, druhou jsou shieldy, tedy jakési rozšiřující karty, které se snadno připojí na piny stávajícího Arduina.

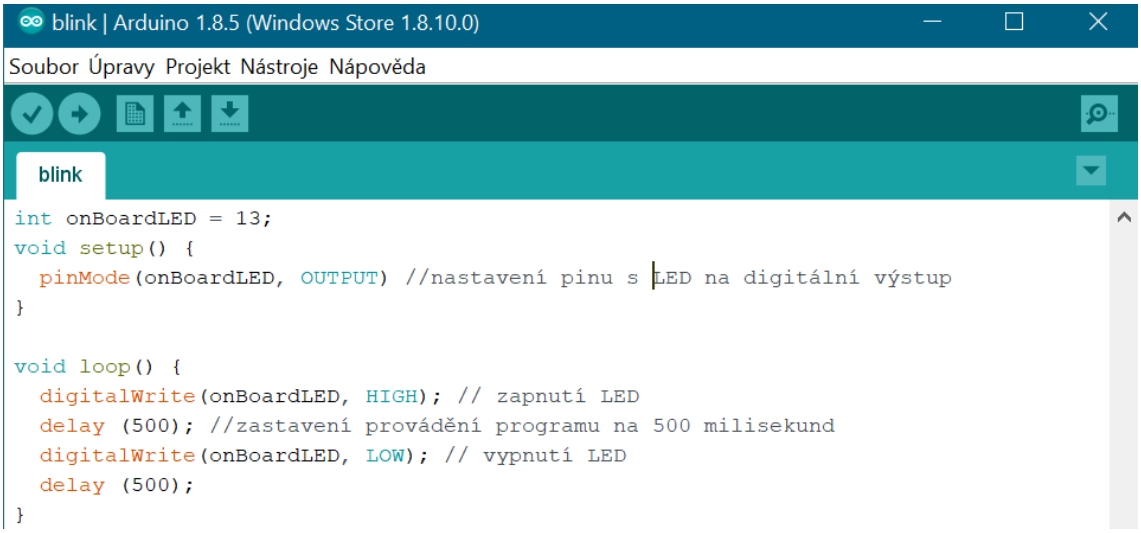
Shieldy mohou být různé, mezi základní patří Ethernet shield nebo Wi-Fi shield, které umožní Arduino pohodlně připojit k internetu. Dalším shieldem je Motor shield, který umožní připojit motory k Arduinu, je tedy možné si pomocí Arduina postavit vlastního robota. Pro tuto bakalářskou práci je nejzajímavějším shieldem XBee shield, který umožňuje připojit moduly XBee a vytvořit tak senzorové sítě pomocí Arduina.



Obr. 16 – XBee shield

2.3 Arduino IDE

Arduino IDE [25] je základní open-source software pro psaní a nahrávání kódu pro desky Arduino. IDE vychází z prostředí Processing, které bylo upraveno pro použití s Arduinem, a to zejména podporou jazyka Wiring.



Obr. 17 – Blikání LED na desce – příklad

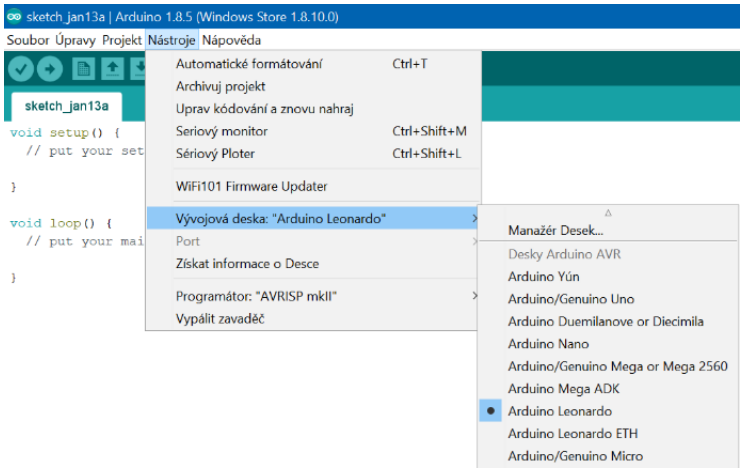
2.3.1 Stažení a instalace

Software byl napsán v jazyce Java, díky čemuž ho lze nainstalovat na různých operačních systémech jako je Windows, MacOS nebo na Linux a jeho distribuce, včetně Linux ARM. V době psaní této práce byla vydána verze 1.8.5. Na Windows lze software stáhnout buď přímo z oficiálních stránek http://arduino.cc, kde je k dispozici verze pro instalaci (.exe) či ve verze portable (.zip), nebo přímo ze Store pro Windows 8.1 nebo Windows 10.

2.3.2 Nastavení

Po instalaci je potřeba provést několik kroků, než je možné software používat. V první řadě je nutné připojit Arduino desku a naistalovat pro ni ovladače. Ve většině případů se ovladače naistalují správně, ale je možné, že systém danou desku nerozpozná, pak je nutné ovladače naistalovat ručně. Ovladače se nachází v místě, kde je naistalován Arduino IDE. Zde se nachází složka Drivers, v níž je soubor „arduino.inf“.

Jakmile jsou ovladače desky naistalovány, je ještě nutné v Arduino IDE vybrat o jakou desku se jedná, viz Obr. 18 – Nalezení desky v IDE. Následně je třeba zvolit ještě port, na kterém je Arduino připojeno.



Obr. 18 – Nalezení desky v IDE

Občas je ještě nutné nahrát různé knihovny jako je knihovna pro práci s moduly XBee a Arduino IDE je připraveno k použití.

2.3.3 Psaní kódu

Programovat Arduino lze v jazyce C nebo C++, ale nejlepší je použít knihovnu Wiring, kterou vytvořil Hernando Barragan, jak bylo zmíněno v kapitole 2.1. Existuje mnoho způsobů, jak programovat Arduino, ale všechny programy musejí obsahovat povinně bloky setup() a loop().

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Obr. 19 – Bloky setup() a loop()

Blok setup() se spouští pouze jednou, a to právě na začátku celého programu, tedy v okamžiku nahrání programu do Arduina. Pokud již je program nahraný v Arduinu, pak se blok setup() spustí, jakmile se zapne Arduino. V tomto bloku se obvykle provádí inicializace sériové komunikace, nebo nastavení režimu jednotlivých pinů [26].

Jakmile je dokončen blok setup(), pak program přechází do bloku loop(), tedy smyčky. Kód obsažený v tomto bloku se bude neustále opakovat, dokud nebude Arduino odpojeno od napájení [26].

2.4 Wiring

2.4.1 Základní struktura jazyka

Jazyk Wiring [27] je velmi podobný Javě a programování v něm je také velmi podobné. Ve Wiringu mohou být definovány proměnné podobně jako v Javě. Mohou být definovány pole, podmínky, cykly, metody. Vše podobně jako v Javě. Rozdíl lze najít v tom, že se nejedná primárně o objektový jazyk.

Wiring má číselné datové typy jako je byte, integer, long, float nebo double. Kromě číselných má i logický datový typ boolean, nabývající hodnot true nebo false a také znakový datový typ char či string, tedy opět podobně jako Java nebo C#.

2.4.2 Práce s digitálními a analogovými vstupy a výstupy

V čem se ale Wiring liší a kvůli čemu byl vlastně stvořen, je práce s hardwarem a jeho vstupy a výstupy. Arduino obsahuje vstupy a výstupy neboli piny, které slouží k dalšímu rozšiřování o čipy, diody, senzory a další. Vstupy a výstupy lze rozdělit na digitální a analogové.

Digitální vstupy a výstupy

Pro inicializaci digitálních vstupů a výstupů slouží funkce pinMode(). Tato funkce umožňuje nastavit pin pomocí dvou parametrů pinMode(pin, vstup/výstup). Tedy pokud například má být pin 2 nastaven na výstup, pak se v bloku setup() nastaví pinMode(2, OUTPUT), pro vstup by to bylo pinMode(2, INPUT). INPUT/OUTPUT jsou tedy konstanty, které určují vstup či výstup.

Toto je ale inicializace, pro práci s digitálními vstupy a výstupy slouží funkce digitalRead(vstupni_pin) a digitalWrite(vystupni_pin, hodnota). Funkce digitalRead(2) vrací hodnotu na pinu 2. Pokud proud protéká daným pinem, pak metoda vrátí hodnotu HIGH, pokud jím proud neproudí pak hodnotu LOW. Podobně se pracuje i s funkcí digitalWrite(). Funkce digitalWrite(3, HIGH) nastaví na pinu číslo 3 hodnotu HIGH, tedy pinem 3 protéká proud.

Analogové vstupy a výstupy

Pro čtení analogových vstupů se používá funkce analogRead(pin), kde pin je pouze analogový, ten je označován písmenem A a číslem, například A1. Funkce analogRead() vrací obvykle hodnotu v rozmezí 0 až 1023 v závislosti na rozlišení, které deska poskytuje.

Funkce analogWrite(pin, hodnota) slouží k nastavení hodnoty „analogového“ pinu. Její použití je omezeno pouze na piny PWM. Hodnota může nabývat rozsahu 0 až 255. Nejedná se o nastavení opravdových analogových hodnot. Pro nastavení skutečných hodnot by bylo nutné použití DA převodníku. Funkce analogWrite() tedy vysílá PWM signál, kdy se rychle střídají maximální a nulové hodnoty napětí (obvykle 0–5 V). Například pro simulaci 2,5 V na pinu 11 se použije analogWrite(11, 127).

2.4.3 Speciální funkce

Časové funkce

Arduino využívá čtyř funkcí, které pracují s časem. Jedná se o millis(), micros(), delay() a delayMicroseconds().

Funkce millis() je bezparametrická funkce, která vrací počet milisekund od spuštění Arduina. Ovšem maximální hodnota je $2^{\wedge 32}$. Poté dojde k přetečení a časovač běží od začátku.

Funkce micros() je funkcionalitou stejná jako millis(), ale vrací a počítá čas v mikrosekundách. K přetečení tedy dojde mnohem rychleji.

Funkce delay() a delayMicroseconds() jsou opět podobné funkce, jedná se o jednoparametrické funkce, kde parametrem je doba, po kterou se má pozastavit provádění programu. Funkce delay() pracuje s milisekundami, zatímco delayMicroseconds() pracuje s mikrosekundami.

Matematické funkce

Wiring nabízí mnoho předdefinovaných matematických funkcí. Funkce min(cislo1, cislo2) slouží k výběru menšího čísla. Oproti tomu je zde i funkce max(cislo1, cislo2) sloužící k výběru čísla většího. Funkce abs(cislo) slouží ke zjištění absolutní hodnoty čísla.

O něco zajímavější je funkce constrain(). Funkce přijímá tři parametry constrain(hodnota, dolniMez, horniMez). Funkce vrací vloženou hodnotu, pokud je mezi horní a dolní mezí, pokud je ovšem nižší než hodnota dolní meze, pak je vrácena právě dolní mez a naopak.

Funkce map() podobně jako constrain() upravuje rozsah. Ovšem map() upravuje hodnotu v daném rozsahu do rozsahu jiného. Funkce přijímá parametry map(hodnota, dolniMezPuvodnihoRozsahu, horniMezPuvodnihoRozsahu, dolniMezNovehoRozsahu, horniMezNovehoRozsahu)

Dále se hodí funkce pro práci s mocninami a odmocninami. Funkce pow(číslo, mocnina) vrací n-tou mocninu čísla vloženého do funkce. Funkce sqrt(číslo) vrací druhou odmocninu čísla.

Nezapomnělo se ani na goniometrické funkce jako sin(), cos() a tan(), které přijímají jako parametr hodnotu úhlu v radiánech a vrací hodnotu vybrané funkce pro daný úhel.

Posledními zmíněními funkcemi jsou random() a randomSeed(). Funkce random() může mít jeden nebo dva parametry – random(max) vrací pseudonáhodné číslo v rozsahu 0 až max a random(min, max) vrací číslo v rozsahu min až max. Aby byl generátor náhodných čísel ještě více náhodný, tak se pomocí randomSeed() nastavuje náhodná vstupní hodnota, která je obvykle získána pomocí analogRead().

2.4.4 Sériová komunikace

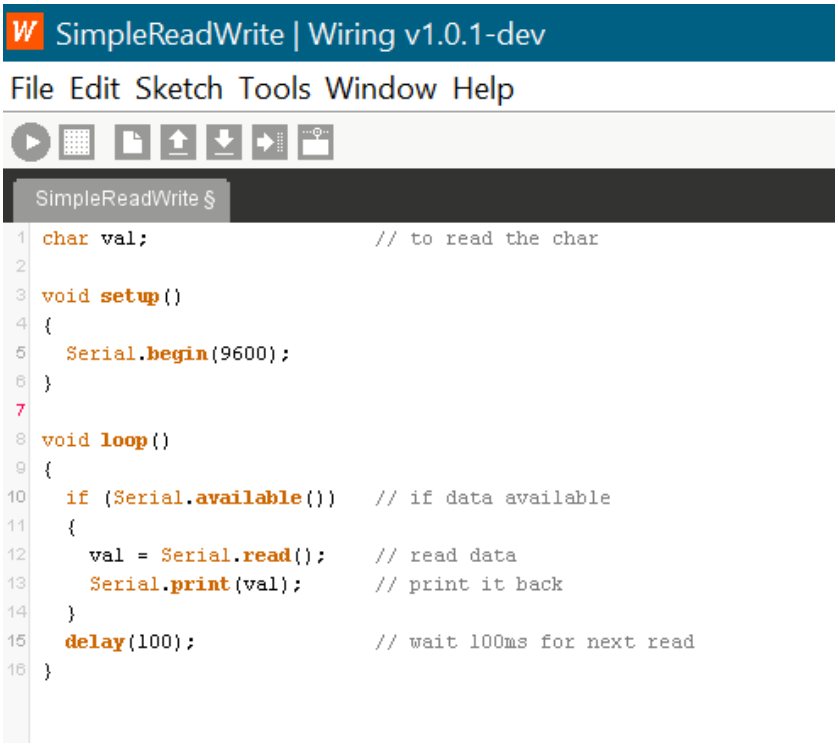
Arduino může pracovat samostatně s připojenými zařízeními, ale pokud mají být data dále zpracována nebo zobrazena na PC, pak je k tomu určena sériová komunikace. Samozřejmě existují i jiné způsoby jako je Ethernet, Wi-Fi nebo Bluetooth, ale ty nejsou součástí každé desky. Proto se nejčastěji používá sériová komunikace.

Sériová komunikace se zahájí zavoláním Serial.begin() s parametrem rychlosti komunikace. Obvykle se volí Serial.begin(9600).

Pro odesílání dat skrze sériovou linku slouží Serial.print() a Serial.println(). Funkce Serial.println() zobrazí data a odřádkuje, zatímco Serial.print() neodřádkuje, je tedy možné pokračovat v jednom řádku. Obě funkce mají jeden povinný parametr a tím jsou data, tedy co má být odesláno. Nepovinnými parametry jsou pak datový typ a délka čísel.

Pro čtení dat slouží funkce Serial.available() a Serial.read(). Serial.available() vrací počet bytů v bufferu, ze kterého se data čtou pomocí Serial.read().

Pokud je na desce více sériových linek pak první je označena Serial a druhá Serial1 atd. Pro ukončení sériové komunikace lze použít Serial.end().



Obr. 20 – Ukázka sériové komunikace Wiring

2.4.5 Myš či klávesnice pomocí Arduina

Arduino umožňuje tvářit se počítači jako myš či klávesnici, ale k tomu potřebuje Serial-USB převodník. Výhodou některých desek je použití procesoru ATmega32u4, který tento převodník již obsahuje. Mezi tyto desky patří Arduino Leonardo, Robot, Esplora a další.

Myš

Prvními důležitými funkcemi pro ovládání myši jsou Mouse.begin() a Mouse.end(). Mouse.begin() oznamuje počítači, že bude odesílat data pro ovládání myši a Mouse.end() oznamuje jeho ukončení. Všechny ostatní funkce, které myš skutečně ovládají, se musí nacházet mezi těmito funkcemi.

Funkce Mouse.click() umožňuje, jak název napovídá, kliknout tlačítkem myši. Jako parametr přijímá konstanty MOUSE_LEFT, MOUSE_RIGHT a MOUSE_MIDDLE. Konstanty určují tlačítko myši, jehož stisk má být simulován. S funkcí Mouse.click() souvisejí i Mouse.press(), Mouse.release(), Mouse.isPressed(). Všechny přijímají stejný parametr v podobě konstant reprezentujících tlačítka myši. Mouse.press() simuluje stisknutí a držení tlačítka, Mouse.realse() uvolňuje držené tlačítko a Mouse.isPressed() zjišťuje, zda je dané tlačítko stisknuté.

Poslední funkcí je Mouse.move(), která umožňuje pohyb kurzoru a také zastupuje funkci kolečka. Parametry jsou Mouse.move(pohybX, pohybY, kolecko). Hodnoty parametrů vyjadřují relativní posun od aktuální pozice.

Klávesnice

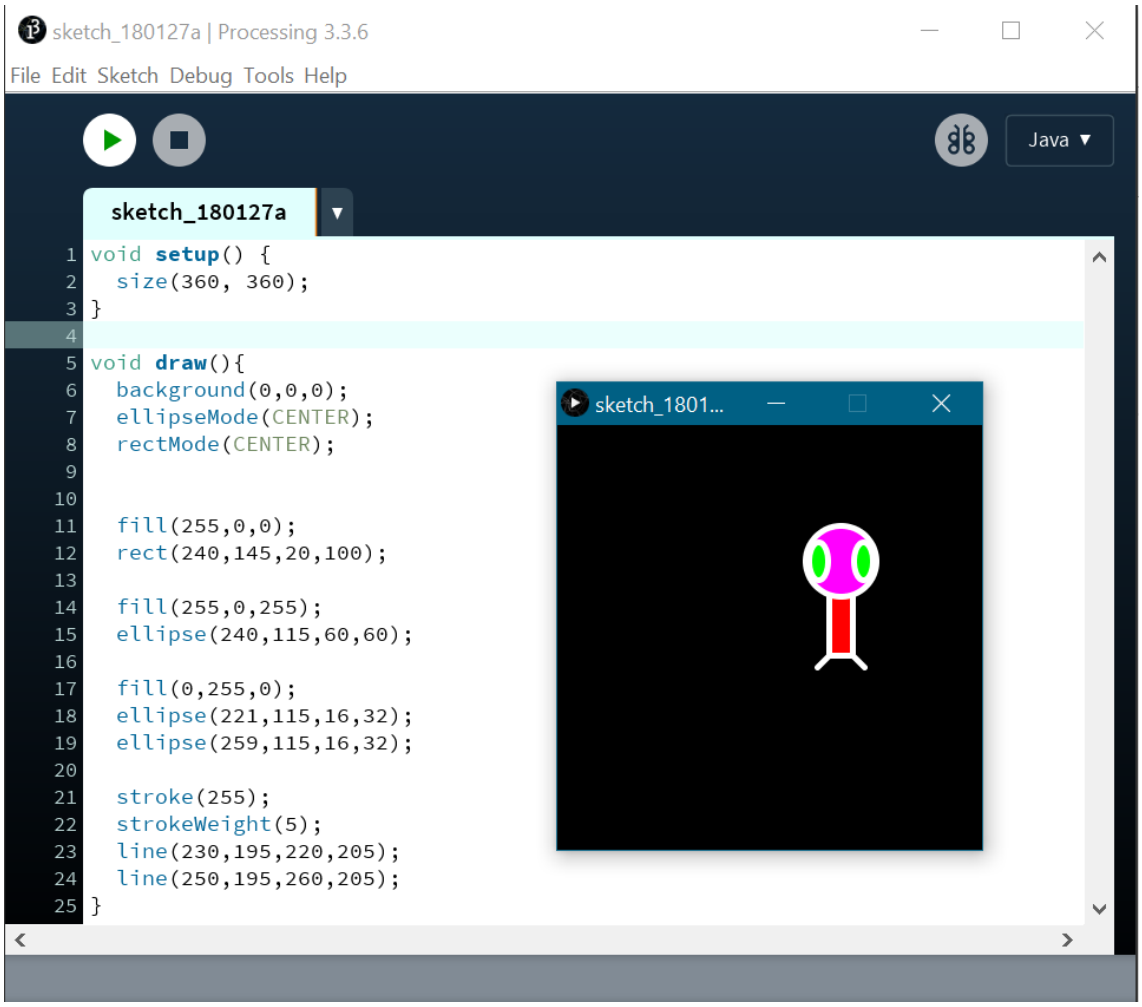
Podobně jako u myši je i ovládání klávesnice zahájeno Keyboard.begin() a ukončeno Keyboard.end().

Keybord.write() odesílá stisknutí jedné klávesy dané parametrem. Keyboard.press() odesílá zmáčknutí a držení jedné klávesy. Pro uvolnění zmáčknutého tlačítka slouží Keyboard.release(), pokud mají být uvolněna všechna tlačítka, pak se používá Keyboard.releaseAll().

2.5 Processing

2.5.1 Základní informace

Processing byl vystavěn na Javě a rozšiřuje tento jazyk o zajímavé předdefinované funkce. Processing neslouží k programování hardwaru, ale využívá se zejména pro grafické zpracování dat. Základní Processing skript obsahuje dva bloky setup() a draw(). Zde je vidět analogie s Wiring. V setup() se provádí počáteční nastavení a v draw() již probíhá vlastní kreslení. Processing je vybaven mnoha funkcemi, které umožňují například vykreslit kruh pomocí jednoho řádku kódu.



Obr. 21 – Processing – ukázka kódu

2.5.2 Arduino a Processing

Pro komunikaci mezi Arduinem a Processingem se používá sériová linka. Existuje firmware Firmata, který se stará o ovládání Arduina pomocí Processing. Stačí pouze tento firmware nahrát do Arduina, v Processingu přidat knihovnu Arduina a dále již vše ovládat v Processingu, ale tím je funkcionalita Arduina značně omezena. Proto je lepší Arduino naprogramovat, tak jak je potřeba a data posílat skrze sériovou linku a pomocí Processingu je dále zpracovávat.

2.6 Arduino vs. Raspberry Pi

Arduino je mikro-kontrolér, zatímco Raspberry Pi je mini-počítač. Arduino sice umožňuje vytvářet skripty, které pak Arduino provádí, ale v zásadě není vytvořeno proto, aby na něm běžel operační systém, jako je tomu právě u Raspberry Pi.

Arduino slouží zejména pro přijímání a předzpracování dat, které jsou následně posílány do počítače, který s nimi dále pracuje.

3 XBee

XBee je malý čip, který umožňuje bezdrátovou komunikaci, mezi ostatními XBee čipy. XBee vyrábí společnost Digi International [28]. Moduly využívají standardizovaných protokolů IEEE 802.15.4 a ZigBee, jež budou popsány podrobněji v dalších podkapitolách. XBee jsou nabízeny v mnoha provedeních, které definují jejich specifické vlastnosti, ale společnou vlastností je snaha o co největší spolehlivost přenosu s využitím minimálního množství energie.

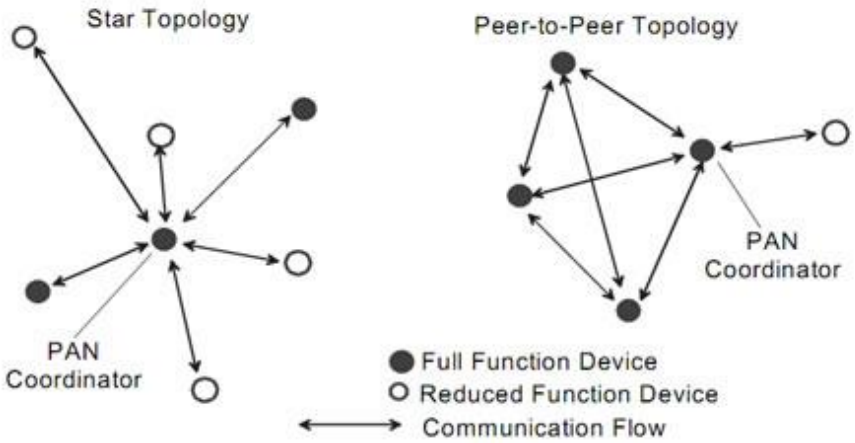
3.1 Protokoly

3.1.1 IEEE 802.15.4

Protokol 802.15.4 vznikl v roce 2003 [29] a byl koncipován, tak aby zkoumal řešení s malým datovým přenosem, které nebude složité a umožní několika měsíční až několika letou výdrž baterie. Pracuje v nelicencovaných frekvenčních pásmech, v České republice se jedná zejména o pásma 2,4GHz a 868MHz. Pásmo 900MHz, které je v některých státech také možné využívat, je v České republice vyhrazeno mobilním operátorům [30]. Tento protokol byl navržen pro použití v senzorových sítích a aplikacích IoT, například pro řízení chytré domácnosti. Rychlost přenosu je až 250 kB/s při použití 2,4GHz [31].

Topologie a zařízení

Protokol definuje dva druhy zařízení. Prvním je plně funkční zařízení (FFD) a druhým je zařízení s redukovanou funkčností (RFD) [32]. FFD může fungovat jako koordinátor celé sítě, nebo jako společný uzel. FFD může komunikovat s jakýmikoli jinými zařízeními. Oproti tomu RFD může komunikovat jen s FFD a jedná se o mnohem jednodušší zařízení, které má mnohem menší nároky na energii a komunikaci, proto nemůže být koordinátorem.



Obr. 22 – IEEE 802.15.4 topologie

Protokol umožňuje stavbu sítí typy peer-to-peer nebo star [33]. V každé síti musí být právě jeden koordinátor tedy FFD. Každé zařízení v síti je jednoznačně identifikováno pomocí 64 bitového identifikátoru. Pro komunikaci uvnitř jedné osobní sítě lze využít i zkráceného 16 bitového identifikátoru.

3.1.2 ZigBee

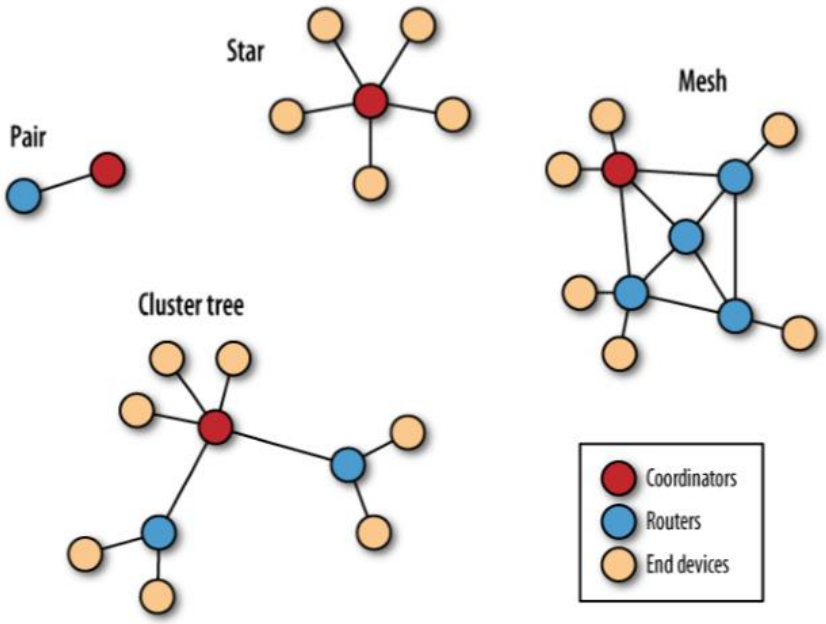
ZigBee je vystavěný na protokolu IEEE 802.15.4. ZigBee vznikl v roce 2004 [34] a za jeho vznikem stojí nezisková organizace ZigBee Alliance, jež byla založena, aby vytvořila standart, který bude možné použít i v průmyslu. Do té doby byla k dispozici pouze technologie Bluetooth, která ovšem nebyla pro použití v průmyslu vhodná. Podobně jako IEEE 802.15.04 pracuje v pásmech 868MHz, 900MHz nebo 2,4GHz. Přenosová rychlost je opět až 250 kB/s.

Druhy zařízení v ZigBee

Typy zařízení jsou opět velmi podobná tomu, jak je definuje IEEE 802.15.4. Prvním je koordinátor, který se v každé síti nachází právě jednou. Jeho úkolem je vytváření sítě, přidělování adres a celkové řízení sítě. Druhým typem je router. Router je plnohodnotné zařízení umožňující připojení k ostatním sítím, odesílání a přijímání informací. Routery musí být neustále v provozu, protože přeposílají i data od ostatních uzlů. Posledním typem je koncové zařízení, které mohou data přijímat i odesílat, neslouží však přeposílání dat od jiných routerů, díky tomu mohou být jednodušší a mohou více šetřit energii. Mohou být dokonce ve stavu sleep, tedy dočasného spánku. [35]

Topologie

Velkou výhodou standardu ZigBee oproti IEEE 802.15.4 jsou topologie, které ZigBee podporuje. Podobně jako IEEE 802.15.4 umožňuje vytvořit topologie typu peer-to-peer a star. Ovšem umožňuje i síťovou topologii neboli mesh.



Obr. 23 – ZigBee podporované topologie

Síťová topologie kromě koordinátora využívá i routery, které mohou posílat zprávy jiným routerům a koncovým zařízením. Koordinátor se sice stará o správu sítě, ale také může směrovat zprávy. Koncová zařízení nemohou komunikovat mezi sebou přímo, ale mohou využít služby koordinátora nebo routeru. Výhodou je, že tato topologie umožňuje redundantní zapojení routerů, což zvyšuje spolehlivost, a může tak být i zvýšena propustnost sítě.

Dalším podporovaným typem sítě je stromová struktura, ta je velmi podobná síťové topologii, postrádá ovšem možnost redundantních spojení. [36]

3.2 Moduly

Druhů XBee modulů jsou vyráběny desítky, mezi nimiž jsou malé ale i velké rozdíly. Základní rozdělení modulů je na Series 1 a Series 2, ale mohou být děleny dle použité antény, či frekvencí na které modul pracuje. Kapitola 3.2 byla zpracována ze zdrojů: [36],[37].

3.2.1 Rozdělení dle Series

Jak bylo zmíněno, základní rozdělení může být dle Series. Series 1, která je často nazývána XBee 802.15.4, podporuje pouze protokol IEEE 802.15.4. Series 1 je vhodná pro jednoduché nahrazení drátové komunikace peer-to-peer. Není nutné moduly nijak zvlášť konfigurovat a jsou tedy ideální pro první prozkoumání modulů. Bohužel Series 1 není kompatibilní se Series 2.

Series 2 už není, tak snadné popsat. Existuje více verzí, které jsou často označovány jako Series 2. Prvními moduly označené Series 2 jsou moduly Znet 2.5. Tyto moduly se dnes prakticky nepoužívají, jelikož se museli před použitím nakonfigurovat nahráním určitého firmwaru, nešlo je tedy konfigurovat za běhu. Znet 2.5 již plně podporuje ZigBee.

Dalšími moduly jsou XBee ZB, které jsou nejčastěji označovány za Series 2, jedná se o moduly hardwarově stejné jako Znet 2.5, ale s novým firmwarem, který již může být konfigurován.

Posledním modulem, který lze zařadit do Series 2 je XBee 2B, které mají oproti ZB nový hardware, který snižuje spotřebu energie. Fungují sice s firmwarem shodným jako u ZB, ale již nejsou kompatibilní se Znet 2.5.

3.2.2 XBee nebo XBee PRO

Moduly XBee jsou nabízeny ve standartní verzi označované XBee anebo ve vylepšené verzi XBee PRO. XBee PRO je o něco delší a nabízí výkonnější hardware, který výrazně zvyšuje dosah modulu, ale je také mnohem dražší.

Obr. 24 – XBee vs. XBee PRO

3.2.3 Druhy antén

Chip anténa

Jedná se o malý čip, který je umístěn přímo na těle modulu. Výhodu této antény je kompaktnost modulu a nízká cena. Nevýhodou může být nižší síla signálu.

PCB anténa

PCB anténa je vytištěna přímo na samotném těle. Výhody a nevýhody jsou stejné jako u chip antény.

Drátová anténa

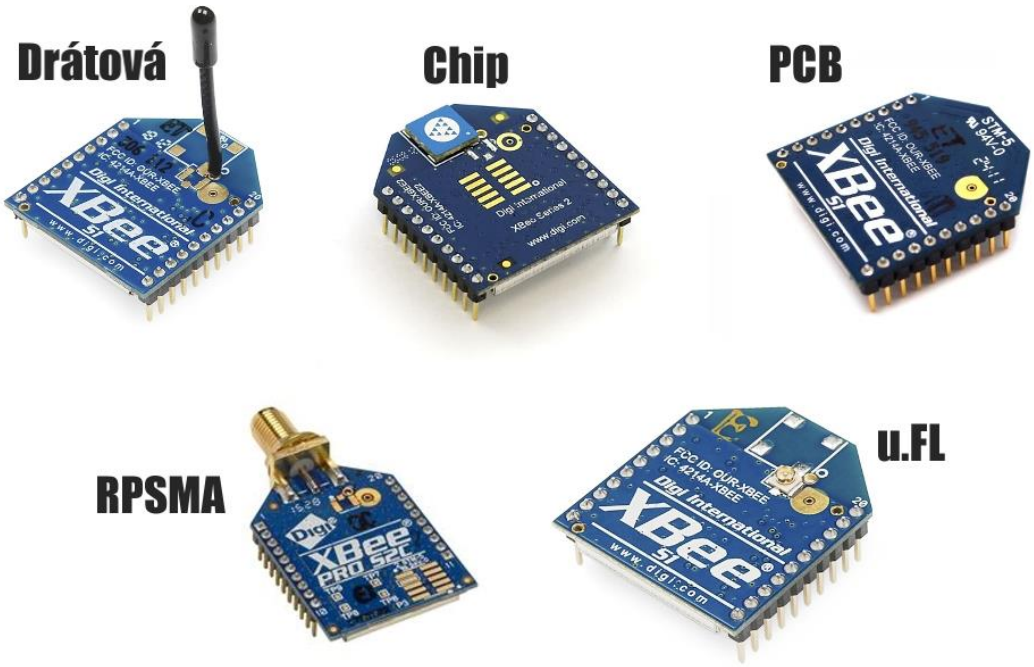
Anténa je tvořena jedním kusem drátu obaleným do platové bužírky. Jedná se o anténu, která nabízí nejlepší poměr cena/výkon. Tato anténa není nákladná na výrobu, takže moduly stojí podobně jako s PCB a chip anténou. Je jednoduchá a nabízí všesměrové vysílání signálu, díky čemuž dochází k maximalizaci přenosové vzdálenosti do mnoha směrů. Jediná drobná nevýhoda může plynout z menší kompaktnosti.

u.FL anténa

u.FL anténa je tvořena malým konektorem, jež je součástí těla modulu, do kterého je nutné umístit vlastní anténu. Výhodou je zisk vlastní antény, který může být mnohem větší, ale nevýhodou jsou zvýšené náklady na pořízení samotné antény.

RPSMA anténa

RPSMA anténa je podobná jako u.FL. Jedná se také o konektor, který je ovšem mnohem větší a je na ně nutné připojit svoji anténu. Výhodu může být, že se jedná o konektor, který využívá i mnoho Wi-Fi routerů.



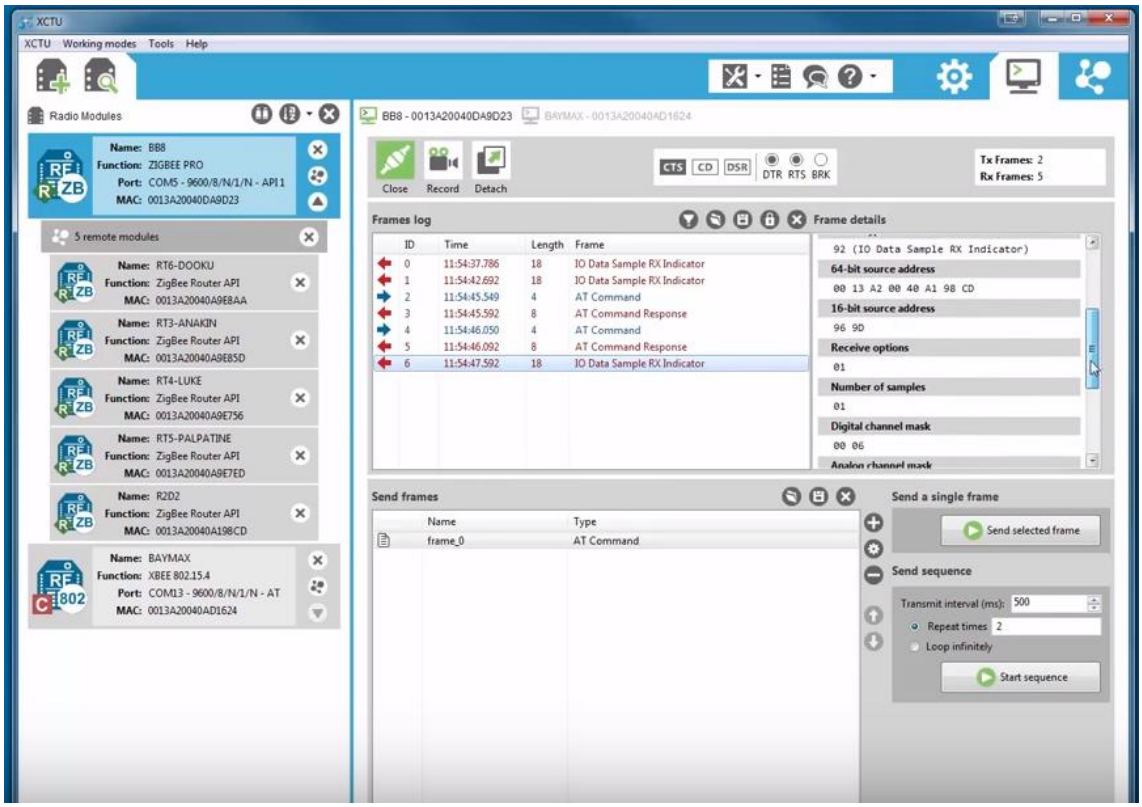
Obr. 25 – Druhy antén modulů XBee

3.2.4 Rozdělení podle frekvence

Posledním možným dělením je frekvence, na které probíhá přenos dat. Nejčastěji jsou používány moduly s frekvencí 2.4GHz, které jsou povolené v mnoha zemích. Dále jsou vyráběny moduly s frekvencí 900MHz, které nabízejí přenos na mnohem větší vzdálenosti, a také tato frekvence lépe přechází přes překážky. Bohužel 900MHz je v mnoha zemích včetně České republiky licencována. Naštěstí je i pár modulů fungující na frekvenci 868MHz, který nabízí vlastnosti shodné s 900MHz, ale je v mnoha zemích v bezlicenčním pásmu. Moduly s rozdílnou frekvencí nejsou vzájemně kompatibilní.

3.3 X-CTU

X-CTU je aplikace sloužící ke konfiguraci modulů XBee. Aplikace je multiplatformní a je ji tedy možné spustit a naistalovat jak na Windows, Linuxu tak i na MacOS. Nabízí snadnou konfiguraci XBee díky grafickému prostředí. Skrze aplikaci je možné snadno nahrávat firmware do XBee. Dále je vybavena mnoha užitečnými funkcionalitami. První takovou funkcionalitou je Frame builder, tedy generátor paketů pro XBee. S tím souvisí i Frames interpreter, který umožňuje snadné čtení paketů. Další věcí, která může přijít vhod je tovární nastavení modulů nebo Range test, tedy testování dosahu mezi jednotlivými moduly. [38]



Obr. 26 – Aplikace X-CTU pro práci s XBee

3.4 Komunikace s XBee v režimu AT

Komunikace s XBee může probíhat buď v API-módu, který obaluje informace do rámců, což bude detailněji popsáno později, nebo v režimu příkazů AT. Režim AT je mnohem lépe čitelný pro člověka a takové to moduly můžou být ve dvou režimech – transparency-mode nebo command-mode.

Kapitola 3.4 byla zpracována z [36].

3.4.1 Transparency-mode

Transparentní mód je výchozí mód pro moduly, které používají režim AT. Tento režim funguje na velmi jednoduchém principu – XBee příjme data a ty odešle dalšímu XBee skrze bezdrátovou komunikaci přesně tak, jak je dostalo. Jakmile druhé XBee přijme data, odesílá je skrze sériovou linku, tak jak je dostalo od prvního XBee.

3.4.2 Command-mode

Pokud není cílem odesílat data skrze XBee, ale komunikovat přímo s modulem XBee, například za účelem zjištění aktuální konfigurace, nebo změnit chování modulu, pak se jedná o příkazový režim.

Jak bylo řečeno defaultním nastavením je transparency-mode, pro přepnutí do režimu command-mode je nutné vstoupit do tohoto režimu pomocí třech znamének plus, tedy pomocí +++. Po přepnutí modul XBee přestane odesílat data, a začne poslouchat příkazy. Pokud uživatel deset vteřin nedodává žádné příkazy, pak se modul přepne zpátky do režimu transparency.

3.4.3 AT příkazy

Příkazy AT, které slouží k nastavení modulu XBee byly původně vytvořeno pro telefonní modemy. Každý příkaz zažíná písmeny „AT“, což značí „attention“, tedy pozor. Znamená to tedy něco ve smyslu – pozor, chci s tebou komunikovat. Po písmenech AT následují další dva znaky, které určují, o jaký příkaz se jedná. Po konkrétním příkazu může ještě následovat konfigurační hodnota.

Základní příkazy:

- AT – Slouží k ověření připojení, pokud je modul dostupný vypíše pouze „OK“
- ATID – Slouží k zjištění ID sítě, do které je modul přiřazený, pokud je příkaz doplněn hodnotou, pak se změní přidělená síť podle této hodnoty
- ATDH/ATDL – Slouží k nastavení horní nebo dolní části cílové adresy.
- ATCN – Slouží k okamžitému opuštění příkazového módu.
- ATMY – Slouží k zobrazení přidělené 16 bitové adresy.
- ATWR – Slouží k zapsání změn do firmwaru.

Obr. 27 – Ukázka konfigurace pomocí AT

3.5 API-mode

API je zkratka pro aplikační programové rozhraní. Jedná se o sadu standartních rozhraní, které umožňují komunikaci mezi dvěma softwary. Na rozdíl od režimu transparency, kde jsou data odesílána tak, jak byla přijata bez jakýkoliv úprav, režim API odesílá data prostřednictvím sériového rozhraní v organizovaných rámcích. Což umožňuje vytvářet komplexní komunikaci bez nutnosti vytvoření vlastního protokolu. Výhodami používání API jsou možnost konfigurovat lokální i vzdálená zařízení, identifikace zdrojové adresy přijatého paketu, možnost zjistit spolehlivost přenosu, díky měření počtu paketů, které selžou, získání síly signálu z každého paketu.

Kapitola 3.5 byla zpracována z [36], [39].

3.5.1 API protokol a jeho struktura

Cílem komunikace v režimu API je rychle, předvídatelně a spolehlivě předávat vysoce strukturované údaje. Podobně jako ostatní protokoly používá i režim API jednotlivé bajty k rozlišení začátku, délky odesílaných dat, kontrolního součtu a typu odesílaných dat.

Start delimiter

Start delimiter je první bajt v rámci, jedná se o speciální číslo určující samotný začátek rámce. XBee API používá hodnotu 0x7E jako první bajt.

Bajty délky

Následující dva bajty po start delimiteru v rámci slouží k zjištění délky celého rámce. První bajt se značí MSB (nejvýznamnější bajt) a je obvykle nulový. Druhý bajt se značí LSB (nejméně významný bajt), který obvykle obsahuje celou délku, ukazuje tedy na konec rámce.

Bajty reprezentující data

Bajty reprezentující samotná data mohou mít proměnlivou délku, vše záleží na typu rámce. Některé rámce mohou mít pouze dva bajty dat, jiné jich mohou mít mnohem více. Tyto bajty reprezentují konkrétní odesílané hodnoty. Prvním bajtem je vždy typ rámce.

Checksum

Poslední bajt rámce je vždy kontrolní součet neboli checksum. Kontrolní součet je vypočítán pro každý rámec před odesláním a uložen do rámce, znovu je pak kontrolní součet počítán po přijetí rámce. Pokud jsou hodnoty kontrolního součtu obsaženého v rámci a hodnoty součtu vypočítaného po jeho přijetí shodné, pak je rámec konzistentní. V opačném případě došlo k jeho narušení, například vlivem šumu.

3.5.2 Typy rámců

Uvnitř hlavní části rámce jsou podstruktury, které pokrývají různé druhy dat, jež jsou odesílány nebo přijímány jednotlivými moduly. Různé typy rámců obsahují různé typy datových struktur. Druhy rámců, které jsou definovány pro XBee Series 1 a 2 jsou desítky. V podkapitolách budou uvedeny některé základní z nich.

AT Commands / Remote AT Commands

Příkazy pro konfiguraci lokálního nebo vzdáleného XBee modulu mohou být odesílány i pomocí API paketů. Pro konfiguraci se používají stejné dvojice znaků jako v případě transparency módu, pouze jsou reprezentovány ASCII hodnotou. Strukturu rámce pro komunikaci s lokálním modulem XBee popisuje Tabulka 1 – Struktura rámce typu AT Command.

Pole rámce		Offset	Příklad hodnoty	Popis
Start delimeter		0	0x7E	
Délka		MSB 1 LSB 2	0x00 0x04	Počet bajtů mezi bajty délky a checksumem
Data specifická pro typ rámce	Typ rámce	3	0x08	
	ID rámce	4	0x52	
	AT příkaz	5	0x4E (N)	Samotný příkaz reprezentovaný ASCII
		6	0x4A (J)	
	Hodnota příkazu (volitelný)			Pokud je nastaven, pak se nastaví hodnota v něm obsažená.
Checksum		7	0x0D	

Tabulka 1 – Struktura rámce typu AT Command

Rámec pro komunikaci s vzdáleným XBee modulem obsahuje navíc adresy pro jeho identifikaci a mód vzdáleného přenosu.

AT Response / Remote AT Response

Když už je příkaz odeslán, tak je také přijata odpověď na něj. V tom případě se jedná o paket typu AT Response pro lokální moduly a Remote AT Response pro vzdálené modly. Strukturu pro lokální modul popisuje Tabulka 2 – Struktura rámce typu AT Response.

Pole rámce		Offset	Příklad hodnoty	Popis
Start delimeter		0	0x7E	
Délka		MSB 1 LSB 2	0x00 0x05	Počet bajtů mezi bajty délky a checksumem
Data specifická pro typ rámce	Typ rámce	3	0x88	
	ID rámce	4	0x01	
	AT příkaz	5	0x42 (B)	Samotný příkaz reprezentovaný ASCII
		6	0x44 (D)	
	Stav příkazu	7	0x00	0=OK 1=ERROR 2=Neplatný příkaz 3=Neplatná parametr 4=Chyba spojení
	Hodnota příkazu (volitelný)			Pokud byl v žádosti nastaven, pak nebude obsažen v rámci.
Checksum		8	0x0D	

Tabulka 2 – Struktura rámce typu AT Response

ZigBee Transmit Request

Rámec ZigBee Transmit Request se využívá k samotnému odesílání dat z lokálního XBee na jiné vzdálené XBee. Tento rámec odesílá užitečná data, která jsou doplněna o cílovou adresu a možnosti odesílaní. Tabulka 3 – Struktura rámce typu ZigBee Transmit Request popisuje rámec typu ZigBee Transmit Request.

Pole rámce		Offset	Příklad	Popis
Start delimeter		0	0x7E	
Délka		MSB 1 LSB 2	0x00 0x16	Počet bajtů mezi bajty délky a checksumem
Data specifická pro typ rámce	Typ rámce	3	0x10	
	ID rámce	4	0x01	
	64 bitová adresa cíle	5	0x00	0x0000000000000000 – je adresa rezervovaná pro koordinátora 0x000000000000FFFF – je rezervována pro broadcast
		6	0x11	
		7	0x22	
		8	0x34	
		9	0xAA	
		10	0xDA	
		11	0x02	
		12	0x44	
	16 bitová adresa cílové sítě	13	0xFF	Pokud je adresa cílového zařízení neznámá nebo se jedná o broadcast, pak se nastavuje 0xFFFE
		14	0xFE	
	Počet hopů	15	0x00	Maximální počet přeskoků k cíli
	Nastavení přenosu	16	0x00	0x01 – Zakázat ACK 0x20 – Povolit APS zabezpečení 0x40 – použití zvýšeného timeoutu pro dosažení cíle 0x00 – v ostatních případech
	Data	17	0x62	
		18	0x82	
		19	0x99	
		20	0x28	
		21	0x72	
		22	0x62	
		23	0x77	
		24	0x97	
Checksum		25	0x0D	

Tabulka 3 – Struktura rámce typu ZigBee Transmit Request

ZigBee Receive Packet

Dalším typem rámce je ZigBee Receive Packet, který nese informaci o odesílateli rámce a informace o datech samých. Tabulka 4 – Struktura rámce typu ZigBee Receive Packet ukazuje strukturu rámce.

Pole rámce		Offset	Příklad	Popis
Start delimeter		0	0x7E	
Délka		MSB 1 LSB 2	0x00 0x11	Počet bajtů mezi bajty délky a checksumem
Data specifická pro typ rámce	Typ rámce	3	0x90	
	64 bitová zdrojová adresa	4	0x00	0xFFFFFFFFFFFFFFFF – pokud je odesílatelova 64 bitová adresa neznámá
		5	0x11	
		6	0x22	
		7	0x34	
		8	0xAA	
		9	0xDA	
		10	0x02	
		11	0x44	
	16 bitová adresa zdrojové sítě	12	0xFF	16 bitová adresa odesílatele
		13	0xFE	
	Informace o přenosu	14	0x01	0x01 – Potvrzení přijetí 0x02 – Paket byl typu broadcast 0x20 – Paket byl zakódován pomocí APS 0x40 – Paket byl odeslán z koncového zařízení
	Data	15	0x62	
		16	0x82	
		17	0x99	
		18	0x28	
		19	0x72	
		20	0x62	
Checksum		21	0x0D	

Tabulka 4 – Struktura rámce typu ZigBee Receive Packet

I/O Data Sample Rx Indicator

Pro senzorové sítě je tento rámec jeden z nejdůležitějších, umožňuje totiž přenášet vstupy a výstupy z a do jednotlivých XBee. Strukturu rámce popisuje Tabulka 5 – Struktura rámce pro práci s I/O.

Pole rámce		Offset	Příklad	Popis
Start delimeter		0	0x7E	
Délka		MSB 1 LSB 2	0x00 0x14	Počet bajtů mezi bajty délky a checksumem
Data specifická pro typ rámce	Typ rámce	3	0x92	
	64 bitová zdrojová adresa	4	0x00	64 bitová adresa odesílatele
		5	0x11	
		6	0x22	
		7	0x34	
		8	0xAA	
		9	0xDA	
		10	0x02	
		11	0x44	
	16 bitová adresa zdrojové sítě	12	0xFF	16 bitová adresa odesílatele
		13	0xFE	
	Informace o přenosu	14	0x01	0x01 – Potvrzení přijetí 0x02 – Paket byl typu broadcast
	Počet vzorků dat	15	0x01	Obvykle se nastavuje 0x01
	Digitální maska kanálu	16	0x00	Bitová maska označující, které digitální I/O linky jsou povolené pro vzorkování
		17	0x1C	
	Analogová maska kanálu	18	0x02	Bitová maska označující, které analogové I/O linky jsou povolené pro vzorkování
	Digitální vzorky dat	19	0x00	Linky, které mají zakázané vzorkování, vrací 0
		20	0x14	
	Analogové vzorky dat	21	0x02	Analogová data jsou dodávána postupně od AD0 až po AD3.
		22	0x25	
Checksum		23	0xF5	

Tabulka 5 – Struktura rámce pro práci s I/O

TX Request: 16 bit address

Rámec typu TX request 16 bit address používají výhradně XBee moduly S1. Tento typ rámce funguje podobně jako rámec typu ZigBee Transmit Request. Výhodou typu rámce pro moduly S1 je možnost adresace pouze pomocí 16 bitové adresy, což umožňuje posílat podstatně méně bajtů.

Pole rámce		Offset	Příklad	Popis
Start delimeter		0	0x7E	
Délka		MSB 1 LSB 2	0x00 0x14	Počet bajtů mezi bajty délky a checksumem
Data specifická pro typ rámce	Typ rámce	3	0x01	
	ID rámce	4	0x01	
	16 bitová adresa cíle	5	0x12	16 bitová adresa příjemce
		6	0x34	
	Možnosti	7	0x00	0x01 – ACK disabled
	Data	8-n	0x66	Užitečná data, která mají být odeslána
Checksum		n+1	0xF5	

Tabulka 6 – Struktura rámce TX Request 16 bit address

TX Status

Po dokončení požadavku TX odešle XBee modul rámec TX Status. Tato zpráva označuje, zda rámec byl úspěšně přenesen nebo došlo k selhání.

Pole rámce		Offset	Příklad	Popis
Start delimeter		0	0x7E	
Délka		MSB 1 LSB 2	0x00 0x14	Počet bajtů mezi bajty délky a checksumem
Data specifická pro typ rámce	Typ rámce	3	0x89	
	ID rámce	4	0x01	
		5	0x00	0x00 – úspěch 0x01 – nepotvrzeno
Checksum		6	0xF5	

Tabulka 7 – Struktura rámce TX Status

RX (Receive) packet 16 bit address

Tento rámec generuje zařízení XBee po obdržení rámce typu TX Request. Opět se jedná o rámec používaný moduly XBee S1 umožňující 16 bitovou adresaci.

Pole rámce		Offset	Příklad	Popis
Start delimeter		0	0x7E	
Délka		MSB 1 LSB 2	0x00 0x14	Počet bajtů mezi bajty délky a checksumem
Data specifická pro typ rámce	Typ rámce	3	0x01	
	ID rámce	4	0x01	
	16 bitová adresa zdroje	5	0x12	16 bitová adresa odesílatele
		6	0x34	
	Možnosti	7	0x00	0x00 - výchozí hodnota
	Data	8-n	0x66	Užitečná data, která mají být přijata
Checksum		n+1	0xF5	

Tabulka 8 – Struktura rámce RX (Receive) packet 16 bit address

4 Praktická část

4.1 Stanovení cílů

Cílem praktické části bakalářské práce bylo vytvořit jednoduchou senzorovou síť, která bude využívat platformy Arduino a pro komunikaci poslouží moduly XBee. Výsledky této komunikace budou vizualizovány pomocí prostředí a jazyka Processing. Dalším cílem bylo ověřit testování základních přenosových parametrů. Dále bylo potřeba ověřit, jaké jsou možnosti využití platformy Arduino v kombinaci s XBee moduly, například řízení vzdálených modulů pomocí centrálního prvku.

4.2 Návrh senzorové sítě

4.2.1 Použité komponenty

Pro stavbu byly využity následující komponenty:

- 1x Robotale Leonardo (kompatibilní klon)
- 1x Arduino UNO R3
- 1x NHduino UNO R3 (kompatibilní klon)
- 1x Funduino XBee Shield
- 3x Series 1 802.15.4 XBee 2.4 GHz (anténa na PCB)
- 2x XBee adaptér do nepájivého pole
- 3x Nepájivé pole
- 4x LED dioda (různé barvy)
- 4x 330Ω rezistor
- 3x MCP 9700 (teplotní senzor)
- 1x XBee USB adapter
- Propojovací/napájecí kabely
- 2x 9 V baterie (volitelné)
- 2x klip na 9 V baterii (volitelné)

Při stavbě bezdrátové senzorové sítě byly použity různé desky platformy Arduino. Byla použita originální deska Arduino, ale i její klony s různými typy procesorů od různých výrobců. Cílem bylo ověřit nezávislost při použití desek různých výrobců.

Pro bezdrátovou komunikaci byly vybrány moduly XBee S1, které podporují protokol 802.15.4. Moduly XBee S1 byly vybrány, protože jsou méně nákladné než novější moduly XBee S2. Další výhodou modulů Series 1 je snazší konfigurace, díky čemuž jsou vhodnější pro jednoduché projekty.

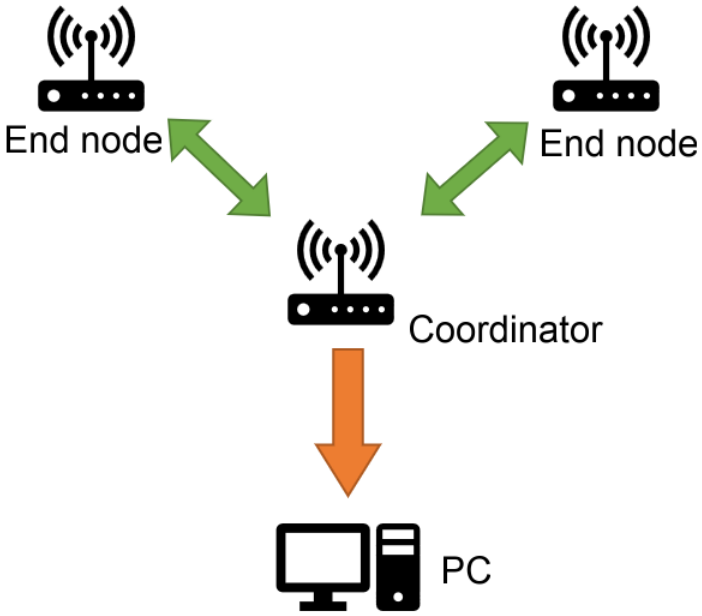
Spojení Arduina a XBee bylo realizováno více způsoby. Prvním bylo využití XBee shieldu, který se snadno připojí na desky Arduino UNO nebo Leonardo. Další možností bylo využít XBee adaptérů pro nepájivé pole. Tento adaptér musí být použit, jelikož moduly XBee mají rozteč mezi konektory 2 mm, zatímco použité nepájivé pole má rozteč 2,54 mm [40].

Teplota je snímána pomocí senzorů MCP 9700. Jedná se o velmi levné senzory, které dokáží snímat teplotu od -40 °C až do 150 °C s odchylkou +- 4 °C [41]. Tyto senzory byly použity za účelem demonstrace přenosu reálných dat. Poskytují pouze přibližnou informaci o okolní teplotě.

Moduly XBee byly konfigurovány pomocí XBee USB adaptéru a aplikace X-CTU. Pro informace o stavu bylo využito několika LED diod.

4.2.2 Schéma senzorové sítě

Navržena byla senzorová síť o hvězdicové topologii, kde budou dva senzorové uzly a jeden koordinátor. Koordinátor řídí jednotlivé senzorové uzly a přijímá od nich data, která následně odesílá do počítače skrze sériovou linku. Hvězdicová topologie byla zvolena díky své jednoduchosti. Při návrhu větších sítí by bylo vhodnější použití síťové topologie.



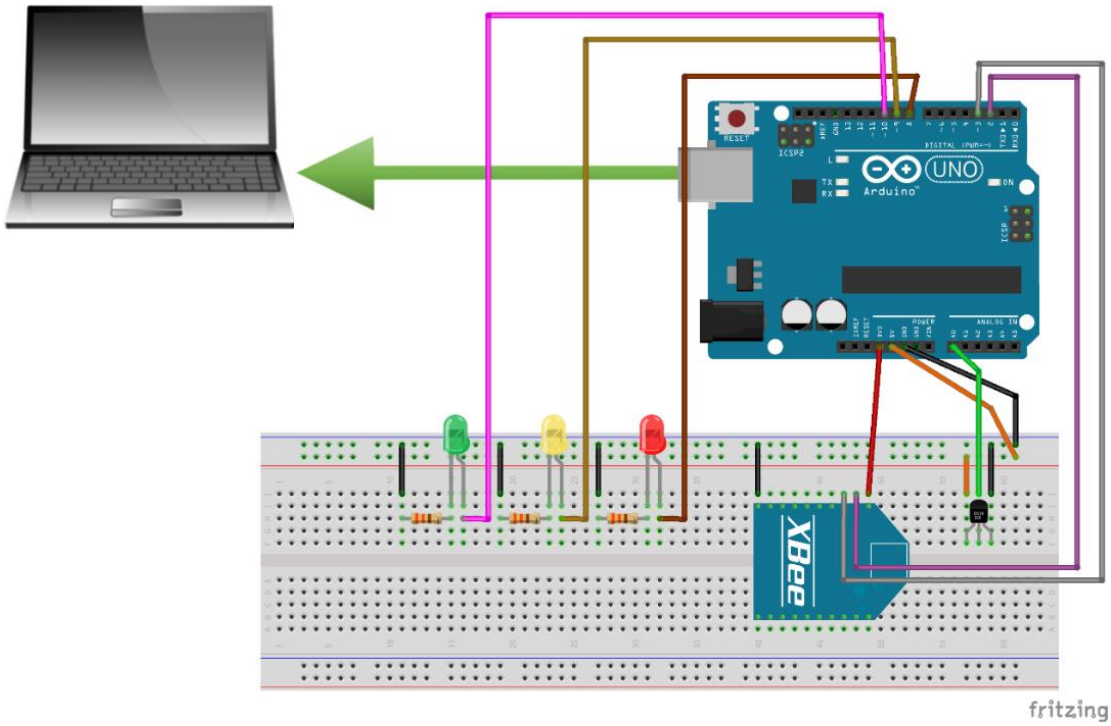
Obr. 28 – Schéma senzorové sítě

Koordinátor

Koordinátor je postaven na desce Arduino Uno R3. Arduino je skrze porty 2 a 3 propojen s modulem XBee. Přenos dat mezi XBee a Arduinem funguje na bázi sériové linky. K desce je připojen senzor teploty, jehož hodnota je čtena z portu A0.

Pro rychlou indikaci teploty naměřené senzorem slouží tři diody. Zelená dioda indukuje teplotu nižší než 25 °C, žlutá dioda indikuje teplotu mezi 25 °C a 30 °C. Červená dioda indikuje teplotu vyšší než 30 °C. Tyto teploty byly vybrány, protože jsou blízko běžné pokojové teplotě, ve které byla senzorová síť nejčastěji testována.

Koordinátor je připojen skrze sériovou linku propojen s počítačem, na němž běží program v Processing.

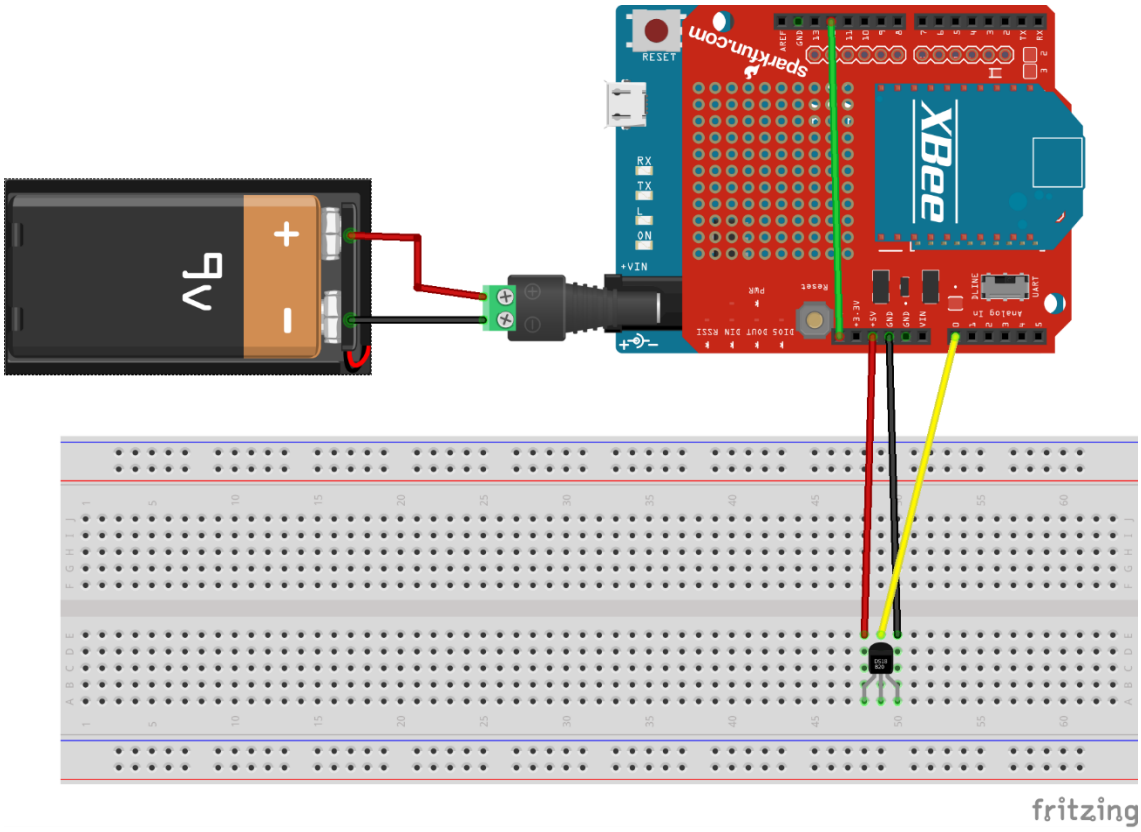


Obr. 29 – Schéma zapojení koordinátoru

Úkolem koordinátora je přijímat data od jednotlivých senzorových uzlů, které dále odesílá do počítače. Dalším úkolem je kontrola stavu jednotlivých senzorů. V případě, že některý ze senzorových uzlů delší dobu neposlal data, pak se ho koordinátor na dálku pokusí restartovat. K datům z jednotlivých senzorů koordinátor odesílá i další informace o jejich stavu. Kromě těchto dat odesílá i informace o sobě samém, včetně teploty ze senzoru připojeného ke koordinátoru.

End Node 1

První senzorový uzel využívá kombinace desky Robotale Leonardo, která je klonem desky Arduino Leonardo. Pro tento uzel byl využit XBee shield pro připojení modulu XBee. Opět je k desce připojen teplotní senzor přes port A0. Port RESET je propojen s digitálním portem 12. Toto propojení umožňuje řídit restart senzorového uzlu. Pro napájení senzorového uzlu slouží 9 V baterie.

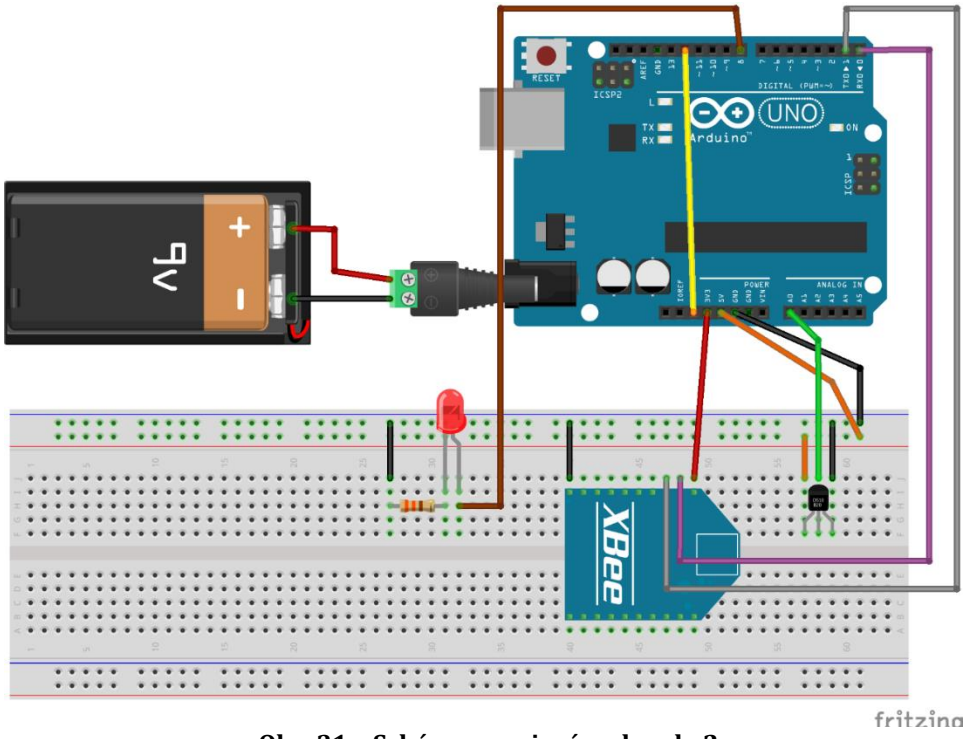


Obr. 30 – Schéma zapojení end node 1

End Node 2

Pro druhý senzorový uzel byla použit klon desky Arduino Uno od výrobce NHduino. Tento klon není plně kompatibilní s originální deskou z důvodu použití čipu CH340G sloužící jako USB driver. Po připojení toho klonu je třeba nainstalovat správný ovladač [42].

XBee modul je s deskou propojen prostřednictvím portů 0 a 1. Data z teplotního senzoru jsou čtena na portu A0. Pro možnost restartovaní je opět připojen port RESET s digitálním portem 12. LED dioda slouží k informaci o tom, zda uzel odesílá data. Tato dioda se využívá pouze, pokud je na desce nahrán testovací program, tento program po deseti odeslaných paketech přestane data odesílat, právě tehdy se rozsvítí kontrolní dioda. Pro napájení senzorového uzlu slouží opět 9 V baterie.



Obr. 31 – Schéma zapojení end node 2

Úkoly senzorových uzlů

Úkolem jednotlivých uzlů je pravidelné odesílání dat ze senzorů, doplněné dalšími informacemi o senzorovém uzlu. Senzorové uzly také čekají, zda je neosloví koordinátor.

4.2.3 Nastavení modulů XBee

Nastavení modulů XBee bylo provedeno pomocí USB adaptéru a programu X-CTU. Následující tabulka zobrazuje parametry, které byly upraveny oproti továrnímu nastavení (číselné hodnoty uvedené v tabulce jsou v hexadecimální soustavě):

Parametry	Koordinátor	Senzorový uzel 1	Senzorový uzel 2
ID	2244	2244	2244
MY	1234	0002	0003
DL	0	1234	1234
CE	Coordinator	End device	End device
AP	API 2	API 2	API 2

Tabulka 9 – Nastavení modulů XBee

Parametr ID určuje síť, ve které se daný modul nachází. Zařízení mohou mezi sebou komunikovat pouze, pokud jsou ve stejné síti.

Parametr MY umožňuje zadat 16 bitovou adresu zařízení.

S parametrem MY úzce souvisí i parametr DL, který určuje cílovou adresu pro komunikaci. Koordinátor komunikuje se všemi, proto má tuto hodnotu ponechanou na výchozí hodnotě. Ovšem senzorové uzly komunikují pouze s koordinátorem, jehož adresa je 0x1234.

Parametr CE určuje, zda je zařízení senzorovým uzlem nebo koordinátorem. Koordinátor může být v síti pouze jeden.

Posledním parametrem je AP, který určuje, jaký režim pro komunikaci bude použit. Režim API 2 nabízí nejvyšší spolehlivost pro odesílání a přijímaní dat, jelikož odesílá data v rámcích, které mají určené bajty se speciálním významem.

4.2.4 Přenos dat

V bakalářské práci byly využity tři typy rámců protokolu IEEE 802.15.4. Jedná se o rámce TX Request: 16 bit address, TX Status a RX packet 16 bit address, více informací v kapitole 3.5.2. Pomocí těchto rámců je realizována komunikace mezi XBee moduly v praktické části bakalářské práce.

Prvním druhem je rámec, který je odesílán jednotlivými senzorovými uzly. Tento rámec obsahuje základní přenosové informace – zdrojová adresa, síla signálu a checksum. Dále rámec obsahuje i užitečná data, která byla poskytnuta jednotlivými senzorovými uzly.

Jedná se o teplotu, která je odesílána pomocí šestice bajtů. Teplota je odesílána v přesném formátu – první bajt určuje znaménko, další dva bajty obsahují informaci o hodnotě před desetinnou čárkou, čtvrtým bajtem hodnota ASCII kódu desetinné čárky, poslední dva bajty obsahují informaci o hodnotě za desetinnou čárkou.

Dalšími daty jsou informace o počtu přijatých a odeslaných rámců s přesností 32 bitů, informace o času od spuštění senzorového uzlu s přesností 64 bitů.

Druhým druhem je rámec, který je odesílán koordinátorem v případě, že některý ze senzorových uzlů neodpovídá. Tento rámec v sekci užitečných dat obsahuje pouze jeden bajt 0x66, pokud senzorový uzel takový bajt obdrží, pak se restartuje.

V této kapitole budou detailně popsány funkce sloužící pro odesílání a přijímání dat. Pro snadnou reprezentaci dat z datových rámců byla v mnoha případech využita knihovna xbee-arduino [43].

getAndProcessFrame() - koordinátor

Tato metoda je použita v programu koordinátora. Metoda je volána v hlavní metodě loop(). Nejdříve se na instanci třídy XBee, jež je obsažena v knihovně xbee-arduino, zavolá metoda readPaket(). Poté se ověří, zda je získaný paket některým ze standartních typů rámců XBee. Pokud je rámec dostupný, pak se zjistí, o jaký typ rámce se jedná. Dle typu rámce se následně volají specializované metody, které zpracují přijatý paket.

```
void getAndProcessFrame(){
  xbee.readPacket(1000);
  Rx16Response rx16 = Rx16Response();
  TxStatusResponse txStatus = TxStatusResponse();
    if (xbee.getResponse().isAvailable()) {
        if (xbee.getResponse().getApiId() == RX_16_RESPONSE) {
                xbee.getResponse().getRx16Response(rx16);
                processRx16Response(rx16);
        }else if(xbee.getResponse().getApiId() == TX_STATUS_RESPONSE){
                xbee.getResponse().getTxStatusResponse(txStatus);
                processTxStatus(txStatus);
        }
    }
}
```

Obr. 32 – Metoda getAndProcessFrame()

processRx16Response() - koordinátor

Metoda processRx16Response() slouží ke zpracování paketu který je uložen v instanci třídy Rx16Response. Tato třída poskytuje metody pro přístup k základním přenosovým informacím – zdrojová adresa, síla signálu a checksum. Dále obsahuje i užitečná data od jednotlivých senzorových uzlů.

Jedná se o teplotu, informaci o počtu přijatých a odeslaných paketů, informaci o době od spuštění uzlu. Tyto informace jsou uloženy do lokálních proměnných.

Poté se v této metodě provede aktualizace proměnné lastCall pro daný uzel, která slouží pro kontrolu, zda senzor komunikuje. Informace o signálu je konvertována na procentuální hodnoty. Následně jsou všechny informace pomocí metody sentFrameToPC() odeslány skrze sériovou linku do počítače.

```
void processRx16Response(Rx16Response rx16){

      uint16_t source = rx16.getRemoteAddress16();
      uint8_t rssi = rx16.getRssi();
      char data[6];
      String tempS = "";
      int receivedPackets[]={0x00,0x00,0x00,0x00};
      int sendPackets[]={0x00,0x00,0x00,0x00};
      int timeNode[]={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
      for (int i = 0; i < 6; i++) {
        data[i] = (char)rx16.getData(i);
        tempS += data[i];
      }
      for (int i=0;i<4;i++){
        receivedPackets[i]=rx16.getData(i+6);
      }
      for (int i=0;i<4;i++){
        sendPackets[i]=rx16.getData(i+10);
      }
      for (int i=0;i<8;i++){
        timeNode[i]=rx16.getData(i+14);
      }
      char sourceCH=' ';
      if(source==0x0002){
        sourceCH='2';
        lastCallS2=millis();
      }else if(source==0x0003){
        sourceCH='3';
        lastCallS3=millis();
      }
      rssi=map(rssi,23,92,0,100);
      rssi=100-rssi;
      sendFrameToPC(sourceCH,rssi,tempS, receivedPackets, sendPackets,timeNode);

      delay(500);
}
```

Obr. 33 – Metoda processRxResponse()

processTXStatus() - koordinátor

Metoda procesTXStatus() zjišťuje, zda odeslaný paket, který měl provést restart senzorového uzlu, dorazil. Pokud je tedy stav potvrzení úspěšný, pak pouze zvýší informací o počtu provedených restartů jednotlivých uzlů. Pokud potvrzení nebylo úspěšné, pak nastaví proměnou restartFailed na hodnotu true. Proměnná lastRestartedDevice společně s proměnou restartFailed, dávají informaci, které zařízení bylo restartováno a zda se restart povedl.

```
void processTxStatus(TxStatusResponse txStatus){
    if(txStatus.isSuccess()){
       if(lastRestartedDevice==0x0002){
         restartsS2++;
       }else if(lastRestartedDevice==0x0003){
         restartsS3++;
       }
    }else{
       restartFailed=true;
    }
}
```

Obr. 34 – Metoda processTxStatus

checkNodes() a sentResetFrame() – koordinátor

Metoda checkNodes() je pravidelně volána z metody loop a slouží ke kontrole, zda jsou všechny senzorové uzly aktivní. Nejdříve se zjistí aktuální čas od spuštění koordinátora a porovná se s proměnou lastCall, která nese informaci o čase, kdy se naposled senzorový uzel ozval. Pokud je tento rozdíl větší než šedesát sekund, pak je odeslána žádost o restart. A hodnota lastCall se zaktualizuje, aby nedocházelo k neustálým pokusům o restart.

Metoda sentResetFrame() přijímá jako parametr 16 bitovou adresu senzorového uzlu. Adresa se nastaví do proměnné lastRestaredDevice a pomocí objektů knihovny xbee-arduino je odeslán paket obsahující bajt 0x66.

```
void checkNodes(){
  unsigned long now = millis();
  if(now-lastCallS2>60000UL){
    sendResetFrame(0x0002);
    lastCallS2=millis();
  }
  if(now-lastCallS3>60000UL){
    sendResetFrame(0x0003);
    lastCallS3=millis();
  }
}
void sendResetFrame(uint16_t destination){
  uint8_t payload[] = {0x66};
  //Vytvoření žádosti v Series 1
  lastRestartedDevice=destination;
  Tx16Request tx = Tx16Request(destination, payload, sizeof(payload));

  //Odeslání rámce
  xbee.send(tx);
}
```

Obr. 35 – Metody checkNodes() a sendResetFrame()

getFrame() – Senzorový uzel

Metoda getFrame() je použita na senzorových uzlech pro získání rámce. Tato metoda byla záměrně implementována bez použití knihovny xbee-arduino, pro demonstraci získání dat z rámce.

Nejdříve je nutné zjistit, zda jsou nějaká data v bufferu sériové linky. Pokud jsou data dostupná, pak se zjišťuje, zda se jedná o začátek XBee paketu. Mezi jednotlivými čteními je zpoždění 50ms, aby se data stihla načíst do bufferu. Poté se postupuje čtením jednotlivých bajtů dle struktury rámců popsaných v kapitole 3.5.2. Nejdříve se zjistí délka rámce, pak typ rámce. Může se jednat buď o typ 0x89 nebo 0x81.

Typ 0x89 informuje, zda rámec odeslaný senzorovým uzlem dorazil do koordinátoru. Pokud rámec dorazil pak se přičte počet potvrzených paketů. Tento

počet je vždy minimálně o jedna menší než, počet odeslaných paketů, jelikož v odeslaném paketu je informace o stavu před přijetím potvrzovacího rámce.

Typ 0x81 je restartovací rámec odeslaný koordinátorem, pokud se tedy v proměnné reset nachází bajt 0x66, pak se provede hardwarový restart.

```
void getFrame(){
    if (Serial.available() > 0) {
    if (Serial.read() == 0x7E) {
      delay(50);
      int lenght = Serial.read();
      delay(50);
      lenght = lenght *0x100 + Serial.read();
      delay(50);
      int frameType = Serial.read();
      if(frameType==0x89)
      {
        int discartByte=Serial.read();
        delay(50);
        if(Serial.read()==0x00){ //pokud je success
          //confirmedPackets pakety budou vždy o 1 menší
          confirmedPackets[0]++;
          checkConfirmed();
        }
        discartByte=Serial.read();
      }else{
      delay(50);
      int source = Serial.read();
      source = source * 0x100 + Serial.read();
      delay(50);
      int rssi = Serial.read();
      delay(50);//options
      int discartByte = Serial.read();
      delay(50);
      int reset = Serial.read();
      delay(50);
      int checksum = Serial.read();
        if(reset==0x66){
          digitalWrite(12,LOW);
        }
      }
    }
  }
}
```

Obr. 36 – Metoda getFrame()

4.2.5 Měření teploty

Pro získání informace o teplotě slouží metoda rightTemp() viz. Obr. 37 – Metoda rightTemp(). Jak již bylo zmíněno, pro měření teploty se používají senzory MCP 9700. Z důvodu snížení možné odchylky se provede deset měření v intervalech 100ms, z nichž se následně vypočítá průměrná teplota s přesností na dvě desetinná místa.

Teplotu je nutné vypočítat dle změřeného hodnoty na analogovém pinu. Tuto hodnotu je nejprve nutné převést na napětí. A/D převodník na Arduinu má přesnost 10 bitů, tedy 1024 hodnot. Jelikož je teplotní senzor napájen napětím 5 V, pak stačí vynásobit zjištěnou hodnotu vstupním napětím senzoru, tedy 5 a následně vydělit 1024, tímto lze zjistit skutečné napětí, které přichází do analogového pinu. 500mV znamená 0 °C, po odečtení 0,5 V se tedy provede kalibrace. Zbytek stačí již pouze vydělit 0.01, jelikož citlivost použitého senzoru je 10mV/°C. [41]

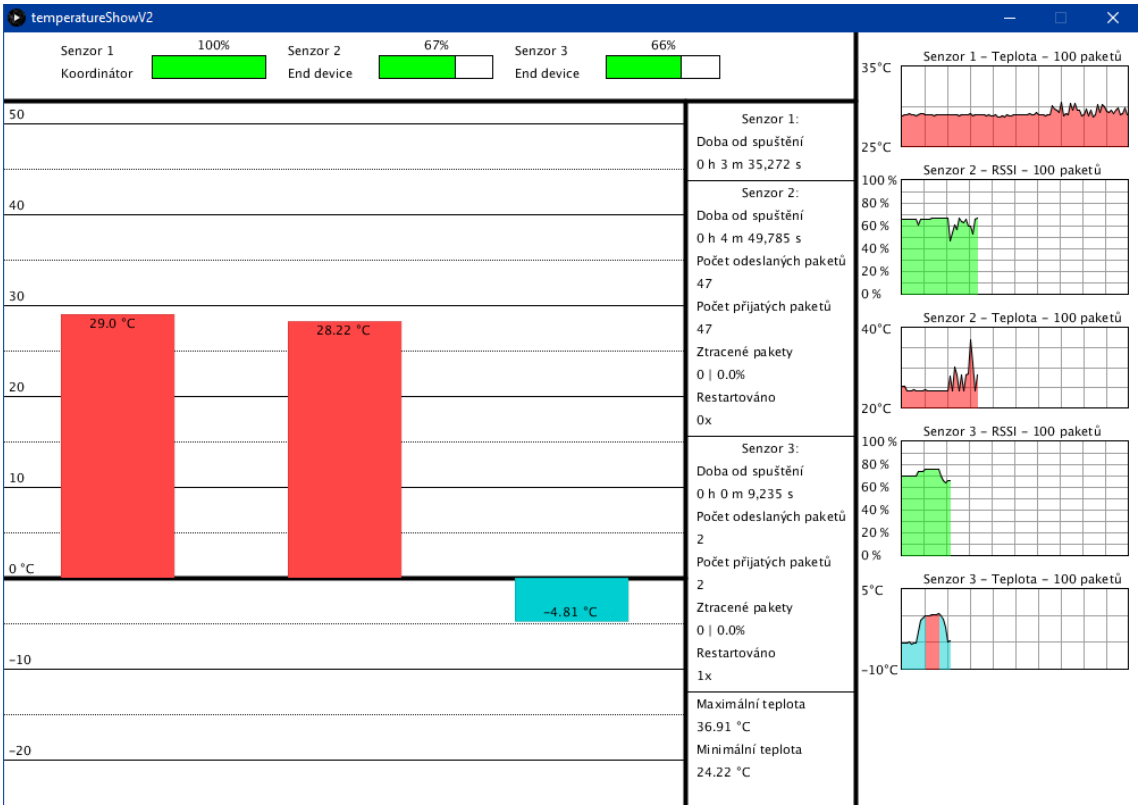
```
//vypočet průměrné teploty z 10 měření
float rightTemp(int pin){
   float temperature = 0.0;
    int sample;
    float tenSamples = 0.0;

    for (sample = 0; sample < 10; sample++) {
        temperature = ((float)analogRead(pin) * 5.0 / 1024.0) - 0.5;
        temperature = temperature / 0.01;
        // měření každých 100ms
        delay(100);
        tenSamples = tenSamples + temperature;
    }
    temperature = tenSamples / 10.0;
    return temperature;
  }
```

Obr. 37 – Metoda rightTemp()

4.3 Ověření funkcionality

Pro ověření funkcionality slouží aplikace vytvořená v Processingu. Aplikace zpracovává data poskytnutá koordinátorem skrze sériovou linku. Výstupem aplikace je okno, které se pravidelně aktualizuje a graficky reprezentuje získaná data.



Obr. 38 – Ukázka reprezentace dat v aplikaci

Základní a největší částí aplikace je zobrazení aktuálních teplot ze senzorů jednotlivých zařízení. Teplota je zobrazována na ose od -20 °C až do 50 °C.

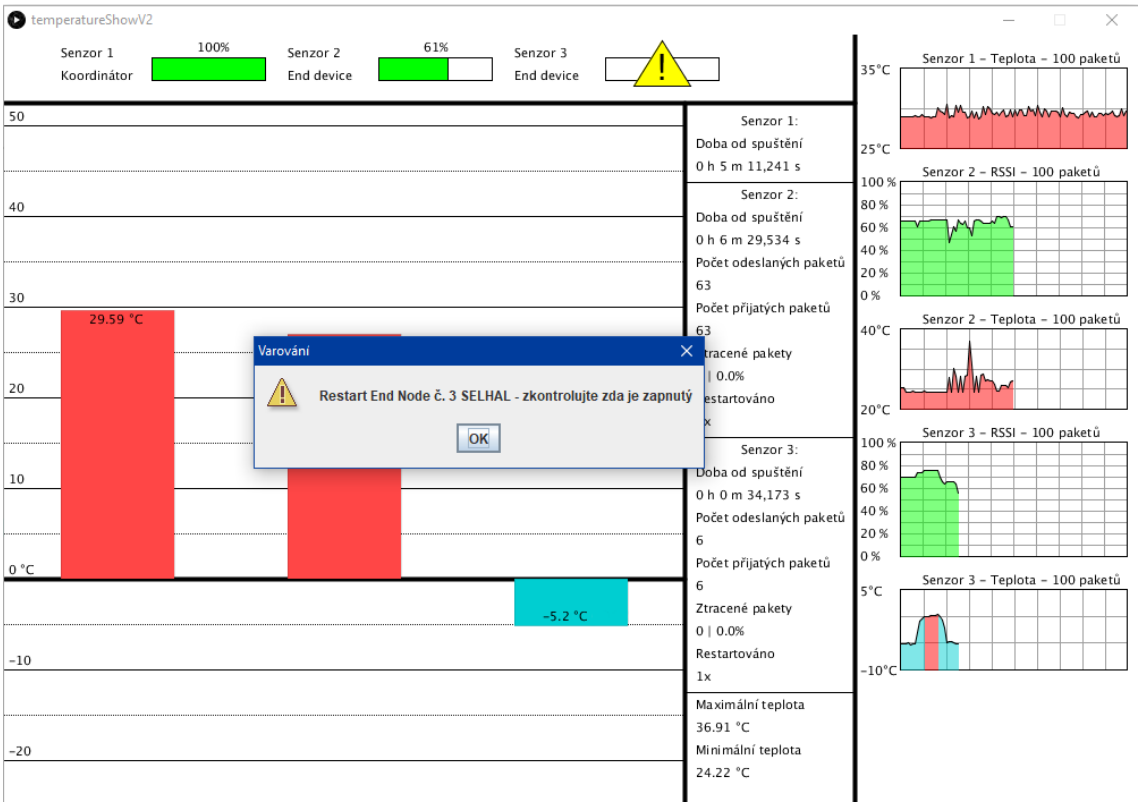
V horní části okna aplikace je informace o síle signálu jednotlivých zařízení. Jelikož je koordinátor připojen USB kabelem, pak je síla signálu zobrazena jako 100 %.

Nejdůležitější částí aplikace je sloupec s detailními informace o jednotlivých zařízeních, které informují o jejich stavu. Všechna zařízení poskytují informaci o době od spuštění. Senzorové uzly navíc poskytují informaci o počtu přijatých a odeslaných paketů, z nichž se vypočítávají ztracené pakety. Poslední informací je

počet restartování jednotlivých uzlů. Ve sloupci je navíc zobrazena informace o maximální a minimální teplotě naměřené od spuštění aplikace.

Grafy v pravé části aplikace slouží pro reprezentaci historických teplot a hodnot RSSI. Tyto grafy zobrazují posledních sto měření. Grafy pro zobrazení RSSI mají pevnou osu od 0 % až po 100 %. Grafy pro zobrazení teplot mají tuto osu dynamickou podle teplot naměřených v posledních stech měření.

V případě, že restart některého z uzlů selhal, je tato skutečnost oznámena vyskakovacím oknem s informací o uzlu, který selhal. Vyskakovací okno je navíc doplněno varovným vykřičníkem přes RSSI ukazatel daného uzlu.



Obr. 39 – Ukázka selhání senzorového uzlu v aplikaci

4.4 Zhodnocení aplikace

Hlavní cíl, kterým bylo postavit jednoduchou senzorovou síť na platformě Arduino v kombinaci s komunikačními moduly XBee byl splněn. Senzorové uzly komunikují s koordinátorem a prostřednictvím datových paketů mu odesílají data. Koordinátor tyto informace předzpracovává a dále je odesílá do počítače, na kterém běží aplikace v jazyku Processing.

Aplikace v jazyku Processing graficky reprezentuje naměřená data a zobrazuje informace o stavu jednotlivých modulů. Jazyk Processing je výborným nástrojem pro snadnou grafickou reprezentaci dat. Výborně se hodí na malé projekty, které jsou velice rychle implementovány. Prostředí Processing ovšem není vytvořeno pro vytváření větších projektů, a to zejména díky absenci objektů. Pokud by měla být aplikace dále rozšiřována, bylo by vhodné psát aplikaci objektově pomocí jiného prostředí.

Testování základních přenosových parametrů bylo docíleno zejména pomocí síly signálu RSSI, počtu odeslaných a přijatých paketů. Testování bylo poměrně snadno implementováno díky struktuře datových rámců modulů XBee. Dalším možným testováním by bylo například zjištění zbývající kapacity baterie, jelikož spotřeba energie je pro senzorové sítě důležitá. Ovšem zjištění kapacity baterie by vyžadovalo vytvoření speciálního obvodu, jelikož Arduino takovými informacemi nedisponuje.

V rámci senzorové sítě bylo vyzkoušeno i vzdálené řízení senzorových uzlů, které potvrdilo, že koordinátor nemusí být jen pasivní příjemce dat od senzorových uzlů, ale může senzorové uzly ovládat pomocí specifických paketů.

Při stavbě senzorové sítě byly záměrně použity různé druhy desek Arduino a různé možnosti propojení, které prokázaly, že platforma Arduino je multiplatformní a změny, které je nutné implementovat v programech pro různé desky, jsou minimální. Co se týče propojení desky Arduino a modulu XBee je nejjednodušší použití shieldu, ale jsou zde některá omezení. Hlavním omezením je obsazení sériové linky Arduina, pokud deska disponuje pouze jednou linkou. To komplikuje například nahrávání programů, či čtení dat skrze sériový monitor. Při použití adaptéru nepájivého pole je možné sériovou linku ušetřit a data z modulu

XBee přenášet na jiné digitální porty, na kterých lze snadno simulovat sériový přenos dat pomocí knihovny SoftwareSerial.

4.5 Možnosti rozšíření

Platforma Arduino, nabízí mnoho různých typů desek, které se hodí na různé typy projektů. Tyto desky lze snadno konfigurovat a kombinovat. Ve spojení s moduly XBee se desky Arduino stávají velmi vhodným nástrojem pro tvoření senzorových sítí. Ovšem jen u senzorových sítí možnosti Arduina a XBee nekončí.

Jednoduchá senzorová síť, která byla vytvořena v rámci této bakalářské práce, ukázala, že nezáleží na posílaných či měřených datech, a to díky možnosti odesílání paket, které můžou obsahovat až 100 bajtů užitečných dat [39], což bohatě stačí na přenos mnoha informací o vysoké přesnosti. První možností rozšíření je využití více zařízení a vytvořit, tak mnohem větší a komplexnější senzorovou síť.

Mnohem zajímavější by mohlo být vyžití Arduina a XBee modulů pro ovládání chytrého domu. Opět by se mohlo jednat o senzorovou síť, která by mohla obsahovat různé druhy uzlů. Například některé uzly by mohly sloužit jako dveřní senzory, jiné by mohly mít teplotní čidla, či senzory pohybu. Všechna poskytnutá data by mohla být odesílána skrze internet do mobilní či webové aplikace, v níž by majitel viděl, co se v jeho domě děje. Dále by mohl pomocí aplikace ovládat některá zařízení uvnitř domu. Například by mohl pomocí aplikace zapnout či vypnout topení, nebo otevřít okna. Kromě ovládání na dálku by mohlo být i možné využít různá schémata nastavení domu. Systém by následně vyhodnocoval získaná data a podle schémat by prováděl automatické akce.

Možnosti, jak využít platformu Arduino jsou široké a společně s moduly XBee se mohou stát vhodným nástrojem pro mnoho aplikací, zejména pak v oblasti internetu věcí.

5 Shrnutí výsledků

V rámci bakalářské práce bylo zjištěno, že realizace senzorové sítě na platformě Arduino s využitím protokolů modulů XBee je možná. Realizovaná síť využívá několika různých desek Arduino. Na jednotlivé desky Arduino byly připojeny moduly XBee. V jednom případě byl využit i XBee shield. Použití různých hardwarových komponent nemělo vliv na funkcionalitu sítě a prokázalo tak univerzálnost platformy Arduino.

Použité desky v kombinaci s nepájivým polem umožňovaly snadnou a rychlou změnu zapojení dalších elektronických součástek, což bylo velmi užitečné v počátečních fázích stavby senzorové sítě.

Pro konfiguraci modulů XBee byl využit USB adaptér a aplikace X-CTU. X-CTU je velice povedeným nástrojem pro konfiguraci a testování komunikace XBee modulů. Pro testování je vhodné mít více adaptérů, aby šlo operativně provádět změny v konfiguraci.

Použité moduly XBee S1 byly snadno konfigurovatelné a umožnily vzájemnou komunikaci řízenou jednotlivými Arduiny. Velkou výhodou použitých modulů byly i nízké pořizovací náklady v porovnání s jinými moduly XBee, oproti tomu nevýhodou byla anténa, jednalo se o anténu typu PCB. Tato anténa neumožňuje všesměrové vysílání a její zisk je poměrně nízký, bylo by vhodnější využít modulů s drátovou anténou, která vysílá signál do všech směrů při zachování nízkých nákladů na pořízení.

Protokol IEEE 802.15.4 potažmo ZigBee poskytuje odesílání definovaných rámců skrze API. API může být ve dvou režimech API1 a API2. API2 na rozdíl od API1 umožňuje „escapování“ tedy způsob zápisu hodnot se speciálním významem. Díky režimu API2 se zvyšuje počet doručených paketů, jelikož moduly XBee dokáží správně interpretovat odeslané pakety. Jelikož byl v praktické části použit režim API2, bylo nutné ošetřit možné „escapování“. Toho bylo dosáhnuto využitím knihovny xbee-arduino, která má implementované funkcionality pro správnou tvorbu a reprezentaci takovýchto paketů.

Pro reprezentaci dat poskytnutých senzorovou sítí byla vytvořena aplikace v Processingu. Processing umožňuje snadnou implementaci grafických elementů,

díky čemuž bylo vytvoření aplikace velice snadné. Bohužel prostředí Processing není navrženo pro tvorbu složitějších programů, které vyžadují objektové programování, tento handicap se projevil s přibývajícími funkcionalitami.

6 Závěry a doporučení

Bakalářská práce potvrdila, že potřeba malých zařízení, která řeší jednoduché úlohy a poskytují datové informace skrze síť, roste s tím, jak se postupně stává Internet věcí běžnou součástí života.

V první kapitole teoretické části byla popsána historie vzniku a potřeba senzorových sítí. Dále byly uvedeny charakteristické vlastnosti a možnosti využití bezdrátových senzorových sítí v dnešní době.

V dalších kapitolách byla popsána platforma Arduino, vysvětlen byl i důvod vzniku této open-source platformy a její přínos ve vzdělání. Arduino by nikdy nevzniklo bez jazyka Wiring, který snadno umožňuje programovat hardware, tedy desky Arduino, proto zde byly detailně popsány jeho funkcionality.

Přestože Arduino může požívat k bezdrátové komunikaci moduly Wi-Fi nebo Bluetooth je vhodné využít komunikační protokoly a zařízení, která vznikla zejména pro vytváření bezdrátových senzorových sítí. Moduly XBee se svými protokoly IEEE 802.15.4 a ZigBee jsou ideální zařízení pro stavbu senzorových sítí, jelikož poskytují nízkou spotřebu, která je stěžejní a také vysokou spolehlivost.

Informace získané v teoretické části byly využity pro realizaci vlastní jednoduché bezdrátové senzorové sítě. Tato síť byla vystavěna na deskách platformy Arduino s využitím komunikačních modulů XBee a protokolu IEEE 802.15.4. Data poskytnutá senzorovou sítí byla následně zpracována aplikací napsané pomocí jazyka Processing.

Vytvořená senzorová síť prokázala, že lze poměrně dobře vystavět senzorovou síť na platformě Arduino v kombinaci s moduly XBee. Pomocí paketů protokolů XBee lze přenést mnoho užitečných dat. Síť je schopná do jisté míry samostatně řešit vzniklé problémy díky vzájemné komunikaci.

Při použití menších desek Arduino, které jsou méně energeticky náročné, by mohla realizovaná síť být dobrým základem pro aplikaci v inteligentním domě či nasazení v průmyslu.

Seznam použité literatury

[1] CHONG, Chee-Yee; KUMAR, Srikanta P. Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE*, 2003, 91.8: 1247-1256.

[2] C. E. Nishimura and D. M. Conlon, "IUSS dual use: Monitoring whales and earthquakes using SOSUS," Mar. Technol. Soc. J., vol. 27, no. 4, pp. 13–21, 1994.

[3] NAYAK, Mamalisa; AGRAWAL, Nitin. *Security in Body Sensor Networks for Healthcare applications*. e-ISSN: 2278-0661, p- ISSN: 2278-8727, vol. 14, Issue 2 (Sep. - Oct. 2013), pp. 41-46.

[4] T. Gao, D. Greenspan, M. Welsh, "Vital sign monitoring and patient tracking over a wireless netowrk", Proceeding of 27th annual international conference of the IEEE EMBS, September 2005.

[5] SOHRABY, Kazem; MINOLI, Daniel; ZNATI, Taieb. *Wireless sensor networks: technology, protocols, and applications*. John Wiley & Sons, 2007.

[6] KAHN, Joseph M.; KATZ, Randy H.; PISTER, Kristofer SJ. Next century challenges: mobile networking for "Smart Dust". In: *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. ACM, 1999. p. 271-278.

[7] RÖMER, Kay; KASTEN, Oliver; MATTERN, Friedemann. Middleware challenges for wireless sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 2002, 6.4: 59-61.

[8] AKYILDIZ, Ian F., et al. Wireless sensor networks: a survey. *Computer networks*, 2002, 38.4: 393-422.

[9] TIWARI, Prashant, et al. Wireless Sensor Networks: Introduction. *Advantages, Applications and Research Challenges, HCTL Open International Journal of Technology Innovations and Research (IJTIR)*, 2015, 14: 2321-1814.

[10] S. Cho, A. Chandrakasan, Energy-efficient protocols for low duty cycle wireless microsensor, Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, Maui, HI Vol. 2 (2000), p. 10.

[11] PIYARE, Rajeev; LEE, Seong-ro. Performance analysis of XBee ZB module based wireless sensor networks. International Journal of Scientific & Engineering Research, 2013, 4.4: 1615-1621.

[12] MARTINCIC, Fernando; SCHWIEBERT, Loren. *Introduction to wireless sensor networking*. chapter, 2005.

[13] Spanning Tree Protocol. *Cisco* [online]. San Jose: Cisco, 2017 [cit. 28-12-2017]. Dostupné z: https://www.cisco.com/c/en/us/tech/lan-switching/spanning-tree-protocol/index.html

[14] D. Sharma, S. Verma, and K. Sharma, "Network topologies in wireless sensor networks: A review," *Int. J. Electron. Commun. Technol*., vol. 4, no. 3, pp. 93–97, Jun. 2013.

[15] Arduino – Introduction. Arduino – Home [online]. [cit. 26-10-2017] Dostupné z: https://www.arduino.cc/en/Guide/Introduction

[16] EIGENBROD, Christine; GRUDA, Nazim. Urban vegetable for food security in cities. A review. Agronomy for Sustainable Development, 2015, 35.2: 483-498.

[17] BAVISKAR, Jaypal, et al. Real time monitoring and control system for green house based on 802.15. 4 wireless sensor network. In: Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference on. IEEE, 2014. p. 98-103.

[18] ABRAHAM, Sherin; LI, Xinrong. A cost-effective wireless sensor network system for indoor air quality monitoring applications. Procedia Computer Science, 2014, 34: 165-171.

[19] BADESCU, Alina-Mihaela; COTOFANA, Lucian. A wireless sensor network to monitor and protect tigers in the wild. Ecological Indicators, 2015, 57: 447-451.

[20] VODA, Zbyšek. *Průvodce světem Arduina*. Bučovice: Martin Stříž, 2015. ISBN 978-80-87106-90-7.

[21] *Arduino The Documentary* [film]. Režie Rodrigo CALVO, Raúl ALAEJOS. USA, 2010. V digitalizované podobě dostupný prostřednictvím YouTube z: https://www.youtube.com/watch?v=kQXeMCLJtwQ

[22] BARRAGÁN, Hernando. *The Untold History of Arduino* [online]. © 2016 [cit. 07-01-2018]. Dostupné z: https://arduinohistory.github.io/

[23] *Story and History of Development of Arduino* [online]. Circuits Today: © 2014 [cit. 07-01-2018]. Dostupné z: http://www.circuitstoday.com/story-and-history-of-development-of-arduino

[24] BARRAGÁN, Hernando. (2004). Wiring: Prototyping Physical Interaction Design. [online]. [cit. 07-01-2018]. Dostupné z: http://people.interactionivrea.org/h.barragan/thesis/thesis_low_res.pdf

[25] Arduino – Software. Arduino – Home [online]. [cit. 08-01-2018] Dostupné z: https://www.arduino.cc/en/main/software

[26] *The setup() and loop() blocks* [online]. Olympia Circuits © 2014 [cit. 13-01-2018]. Dostupné z: http://learn.olympiacircuits.com/setup-and-loop-blocks.html

[27] Wiring [online]. [cit. 13-01-2018]. Dostupné z: http://wiring.org.co/

[28] Digi XBee/RF solutions [online]. Digi International Inc. © 2018 [cit. 26-01-2018]. Dostupné z: https://www.digi.com/products/xbee-rf-solutions

[29] IEEE 802.15 WPAN™ Task Group 4 (TG4) [online]. IEEE © 2003 [cit 26-01-2018]. Dostupné z: http://www.ieee802.org/15/pub/TG4.html

[30] Využívání vymezených rádiových kmitočtů [online]. Český telekomunikační úřad © 2018 [cit. 26-01-2018]. Dostupné z: https://www.ctu.cz/vyuzivani-vymezenych-radiovych-kmitoctu

[31] CALLAWAY, Ed, et al. Home networking with IEEE 802.15. 4: a developing standard for low-rate wireless personal area networks. *IEEE Communications magazine*, 2002, 40.8: 70-77.

[32] BARONTI, Paolo, et al. Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and ZigBee standards. *Computer communications*, 2007, 30.7: 1655-1695.

[33] TIMMONS, Nick F.; SCANLON, William G. Analysis of the performance of IEEE 802.15. 4 for medical sensor body area networking. In: *Sensor and ad hoc communications and networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*. IEEE, 2004. p. 16-24.

[34] *ZigBee Document 053474r06, Version 1.0, ZigBee Specification*. Zigbee Alliance. 2004.

[35] FARAHANI, Shahin. *ZigBee wireless networks and transceivers*. Boston: Newnes/Elsevier, c2008. ISBN 0750683937.

[36] FALUDI, Robert. *Building wireless sensor networks: with ZigBee, XBee, arduino, and processing*. " O'Reilly Media, Inc.", 2010.

[37] XBee Buying Guide [online]. SparkFun Electronics ® 2018 [cit 26-01-2018]. Dostupné z: https://www.sparkfun.com/pages/xbee_guide

[38] XCTU [online]. Digi International Inc. © 2018 [cit. 26-01-2018] Dostupné z: https://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu

[39] XBee/XBee-PRO S1 802.15.4 (Legacy) [online]. Digi International Inc. © 2017 [cit. 26-01-2018] Dostupné z: https://www.digi.com/resources/documentation/digidocs/pdfs/90000982.pdf

[40] Mounting considerations [online]. Digi International Inc. © 2018 [cit. 02-04-2018] Dostupné z:

https://www.digi.com/resources/documentation/Digidocs/90001500/Default.htm#Concepts/c_mounting_considerations.htm%3FTocPath%3DHardware%7C_____3

[41] MCP 9700 [online]. © 1998-2018 Microchip Technology Inc. [cit. 02-04-2018] Dostupné z: http://www.microchip.com/wwwproducts/en/en022289

[42] Chinese Arduino [online]. Dancetale Electronics © 2014 [cit. 10-04-2018] Dostupné z: https://dancetale.wordpress.com/2014/12/30/chinese-arduino-nhduino/

[43] XBee-Arduino [online]. © 2018 GitHub, Inc. [cit. 10-04-2017] Dostupné z: https://github.com/andrewrapp/xbee-arduino

Příloha 1: Struktura souboru se zdrojovými kódy

Níže je uvedena struktura adresářů, souborů a jejich popisu v archivu zrojove_kody.zip, který je přiložen k elektronické podobě bakalářské práce.

- **zdrojove_kody** – hlavní adresář
    - **arduino_source** – adresář obsahující jednotlivé projekty pro Arduino desky
        - **Coordinator**
            - **Coordinator.ino** – soubor aplikace Arduino IDE obsahující zdrojový kód koordinátora psaný pro desku Arduino Uno
        - **EndNodeLeo**
            - **EndNodeLeo.ino** – soubor aplikace Arduino IDE obsahující zdrojový kód prvního senzorového uzlu psaný pro desku Arduino Leonardo
        - **EndNodeUno_test**
            - **EndNodeUno_test.ino** – soubor aplikace Arduino IDE obsahující zdrojový kód druhého senzorového uzlu psaný pro desku Arduino Uno.
    - **mainApp** – adresář obsahující projekt aplikace Processing
        - **application.windows32** – adresář s knihovnami
        - **mainApp.pde** – soubor aplikace Processing obsahující zdrojový kód spouštěný na počítači.
    - **readme.txt** – soubor obsahující detailnější informace o jednotlivých souborech

Příloha 2: Zadání práce

Zadaní práce je v elektronické podobě obsaženo v souboru zadaniBP.pdf