VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

BEZDRÁTOVÝ SENZOROVÝ SYSTÉM

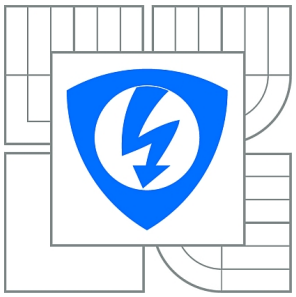
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — FILIP KOTOUČEK
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

BEZDRÁTOVÝ SENZOROVÝ SYSTÉM

WIRELESS SENSOR SYSTEM

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

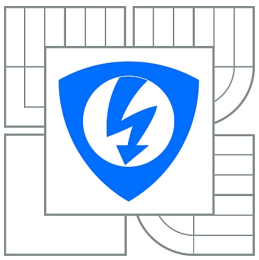
AUTOR PRÁCE — FILIP KOTOUČEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. MIROSLAV BOTTA

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Filip Kotouček	***ID:***	154775
Ročník:	3	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Bezdrátový senzorový systém

POKYNY PRO VYPRACOVÁNÍ:

Úlohou studenta v rámci řešení bakalářské práce bude nastudování problematiky bezdrátových sítí, zvolí vhodný komunikační framework pro testování funkčnosti sítě a zprovozní základní testovací síť obsahující senzory a aktuátory. Síť bude obsahovat také bránu pro zasílání nasnímaných dat pro další zpracování na server. Součástí práce bude vytvoření webového rozhraní pro ovládání a kontrolu bezdrátové sítě. Jednotlivá uložená data bude možno zobrazit pomocí grafických průběhů.

DOPORUČENÁ LITERATURA:

[1] FARAHANI, Shahin. Zigbee Wireless Networks and Transceivers. [s.l.] : Elsevier, 2008. 329 s. ISBN 978-0-7506-8393-7
[2] Atmel Lightweight Mesh, URL: http://www.atmel.com/tools/LIGHTWEIGHT_MESH.aspx
[3] Atmel BitCloud – ZigBee PRO, URL: http://www.atmel.com/tools/BITCLOUD-ZIGBEEPRO.aspx

Termín zadání:	9.2.2015	***Termín odevzdání:***	2.6.2015

Vedoucí práce: Ing. Miroslav Botta
Konzultanti bakalářské práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Bakalářská práce se zabývá návrhem bezdrátové senzorové sítě složené z modulů deRFmega128 a Raspberry Pi. Práce obsahuje popis vytvořeného protokolu pro bezdrátovou komunikaci v síti a protokolu pro sériovou linku na rozhraní brány. Bylo také navrženo webové rozhraní pro celkovou správu sítě. Dále práce obsahuje analýzu energetické náročnosti koncových jednotek a měření výkonu solárních panelů jako možných sekundárních zdrojů energie.

KLÍČOVÁ SLOVA

IEEE 802.15.4, bezdrátová senzorová síť, energetická optimalizace, Atmel Lightweight mesh, Raspberry Pi

ABSTRACT

Bachelor thesis describes the design of wireless sensor network composed of modules deRFmega128 and Raspberry Pi. It contains a description of the created protocol for wireless communication and protocol for serial interface on the gateway. It also describes web interface for general management of network. The work includes analysis of nodes energy performance and measurement of solar panels as a potential secondary sources of energy.

KEYWORDS

IEEE 802.15.4, wireless sensor network, energy optimalization, Atmel Lightweight mesh, Raspberry Pi

KOTOUČEK, Filip *Bezdrátový senzorový systém*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2015. 42 s. Vedoucí práce byl Ing. Miroslav Botta

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Bezdrátový senzorový systém“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Miroslavovi Bottovi, za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské pr byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .
(podpis autora)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Mikrokontrolery v dnešních dnech doslova zaplavují náš svět. Nachází se v mobilních telefonech, používáme je, když jedeme do práce, nebo když si vaříme kávu. Díky dostupným technologiím jsme dnes schopní vytvořit velice úsporné, levné a výkonné procesory, které mohou řídit cokoli, na co si vzpomeneme. Co se týče bezdrátových senzorových sítí jsou tyto mikroprocesory ideálním kandidátem pro realizaci. Bezdrátové sítě mají nepřeberné množství aplikací. Od monitorování zdravotních funkcí pacientů, přes sledování počasí, monitorování strojů v továrnách až po řízení „chytrých“ budov a další.

Cílem práce je vytvořit funkční senzorovou síť s obousměrnou komunikací pro měření a zobrazování jakýchkoli základních veličin (např. teplota, vlhkost vzduchu, hluk, osvětlení,...). Síť se skládá z koncových zařízení, které bezdrátově posílají naměřená data výchozí bráně. Brána obsahuje webserver, kam se může kdokoli připojit a sledovat výsledky v reálném čase. Také je možné odesílat příkazy zpět do koncových jednotek, aby sepnuly aktuátor, nebo udělali jinou definovanou akci.

Práce je dělena do čtyř hlavních částí a přílohy. První část obsahuje teoretické poznatky a informace týkající se této problematiky, definuje několik základních pojmů a popisuje běžně používané technologie. Druhá část pak obsahuje popis použitého hardwaru a analýzu dostupných technologií a frameworků, které se používají pro programování mikroprocesorů. Například z pohledu náročnosti práce, náročnosti pro paměť, délky kódu atd. Ve třetí části naleznete popis struktury celého protokolu. Síť se dělí na bezdrátovou část, kde se využívá metod frameworku, a na část sériové komunikace. V závěrečné části se nachází návod, jak tuto síť připravit pro konkrétní měření či jiné požadavky.

1 TEORETICKÁ ČÁST PRÁCE

Bezdrátové sítě se v poslední době začínají využívat stále více. Jednak díky tomu, že instalace bezdrátových sítí je daleko jednodušší a mobilnější než sítí drátových, a také pořizovací cena potřebného hardwaru je dnes již poměrně malá.

1.1 Senzorové sítě

Bezdrátové senzorové sítě [1] jsou platforma, kterou lze snadno využít na monitorování životního prostředí, analýzu místních dat nebo jiné kontrolní a ochranné funkce. Celek se skládá ze základních tří částí: koncové zařízení, brána a software. **Koncové zařízení** rozprostřené po dané oblasti přijímá data ze senzorů a odesílá pak data bráně. Většinu času však koncové zařízení spí, aby byla snížena spotřeba energie. **Brána**, která pracuje v roli koordinátora sítě, může přijatá data analyzovat sama, nebo může být propojena s dalším systémem, který se stará o výpočty a reprezentaci výsledků měření. **Firmware** je nedílnou součástí systému a jeho úkolem je spolehlivý, efektivní a bezpečný přenos dat. Musí být optimalizovaný kvůli úspoře energie, protože zařízení bývají napájené klasickými bateriemi, a samozřejmě je požadována co nejdelší bezobslužná funkce (v řádu několika let).

Jak je výše zmíněno, je nepřeberné množství aplikací těchto sítí. Zde je uvedeno pár příkladů: monitorování znečištění ovzduší, detekce požáru v lesích, kontrola kvality půdy v zemědělství, monitorování stavebních struktur, „chytré“ domy, atd. [2].

1.2 Internet of Things

Definice výrazu Internet of Things (IoT – internet věcí), je idea Kevina Ashtona z roku 1999. Ta říká, že lidé, zvířata nebo stroje budou disponovat jednoznačnými identifikátory a budou připojeni do struktury typu internet. Dnes IoT vypadá spíše jako sada nezávislých zařízení reprezentujících nasbírané informace pomocí internetu [3]. Důležitým faktorem ve vývoji IoT je zavedení IPv6 (IP adresa verze 6), které nabízí tak velké možnosti, že bychom mohli přiřadit IP adresu každému atomu na zemském povrchu a ještě by nám zbylo.

Velmi zajímavým příkladem IoT je sběr dat, který provedli odborníci při jaderné katastrofě ve Fukushimě. Sbírali informace o radioaktivitě a publikovali je na internetu. Díky tomu byli zasažení lidé vystaveni jen malé dávce ozáření [4].

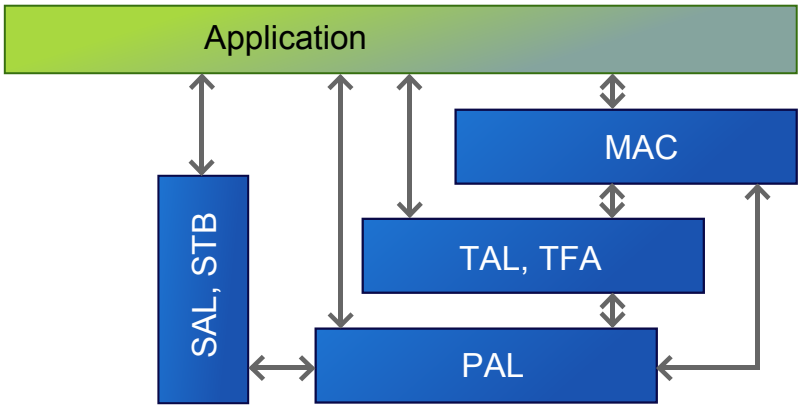
1.3 Popis bezdrátových frameworků

Existují standardizované i proprietární frameworky, které řeší bezdrátovou komunikaci. Mezi hlavní standardizované a široce používané technologie patří ZigBee protokol a další popsané níže.

1.3.1 IEEE 802.15.4 – MAC stack

Standard IEEE 802.15.4 (naposledy aktualizován v roce 2011) definuje zásady pro komunikaci na fyzické a spojové vrstvě pro nízkoenergetické zařízení. Standard definuje požadavky na MAC a PHY vrstvu pro sítě WPAN (wireless personal area network), což jsou bezdrátové sítě s dosahem cca do 30 metrů. WPAN sítě, na rozdíl od větších sítí jako WLAN (wireless local area network), mohou existovat samostatně bez nutnosti propojení do širší sítě. Toto umožňuje vznik malých, energeticky nenáročných a levných řešení, která mohou být implementována do mnoha zařízení [5].

MAC stack[1] od firmy Atmel implementuje výše popsaný standard. Vnitřní architektura je znázorněna na obrázku 1.1. Podporuje všechny rádiové čipy od Atmelu a umožnuje velkou flexibilitu v úpravě firmwaru. Je poměrně snadno portovatelný, ale podporuje pouze peer-to-peer a hvězdicovou konfiguraci sítě (viz obrázek 1.3). Hlavní vrstvy stacku jsou: PAL – Platform Abstraction layer, TAL – Transceiver Abstraction layer a MCL – MAC Core layer [6].



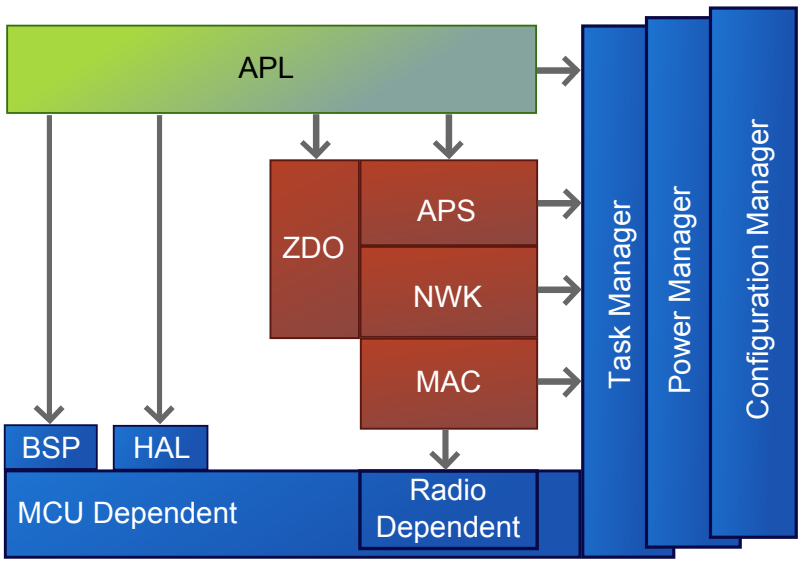
Obr. 1.1: Architektura MAC frameworku [6]

[1] Balíček připravených knihoven, které se starají například o bezdrátovou komunikaci, časovače, přerušení nebo další periferie.

1.3.2 ZigBee – BitCloud stack

ZigBee je jeden z nejpoužívanějších stacků v aplikacích pro bezdrátové senzorové sítě. Zařízení, která používají ZigBee, vysílají na frekvencích 868 MHz, 915 MHz a 2,4 GHz. Protokol implementuje standard IEEE 802.15.4, který definuje fyzickou a MAC vrstvu. ZigBee pak přidává aplikační a zabezpečovací vrstvu a specifikace, aby se zajistila přenositelnost mezi zařízeními různých výrobců. Maximální datový tok je 250 kb za sekundu. ZigBee je vhodný pro zařízení napájená z baterií, kterým postačuje velmi malý datový tok a mají malou spotřebu. U těchto zařízení nás zajímá poměr mezi dobou, kdy je aktivní a kdy je v módu spánku. Zařízení spí většinu času, aby bylo schopné fungovat z baterií i několik let [7]. Tento standard je spravován organizací tvořenou zástupci z velkých firem vyvíjejících převážně polovodičové součástky jako je Samsung, Honeywell, Philips a další [8]. Základními rysy ZigBee protokolu jsou:

- různé podporované topologie sítě, mezi které patří i point-to-multipoint a mesh sítě (viz obrázek 1.3)
- nízká spotřeba energie
- až 65 000 uzlů v jedné síti
- 128b šifrování AES pro bezpečnou komunikaci
- zabránění kolizím, opakování přenosu a potvrzení o příjmu dat

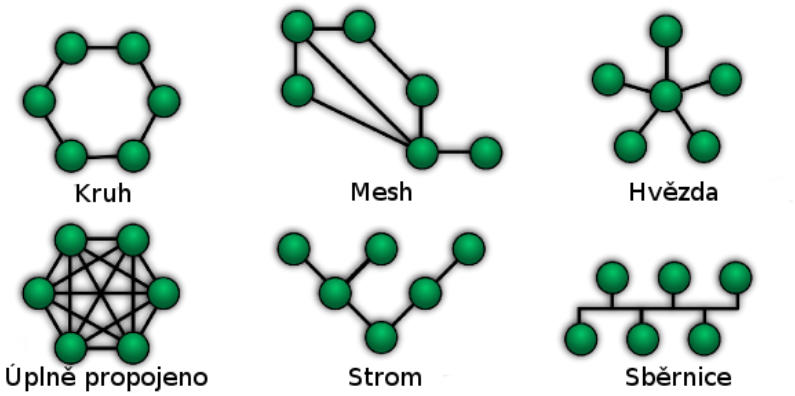


Obr. 1.2: Architektura BitCloudu [9]

BitCloud je framework pro embedded[2] systémy od firmy Atmel, který podporuje velkou řadu funkcí a mimo jiné plně implementuje zmíněný IEEE 802.15.4 standard.

[2]Zabudovaný jednoúčelový systém, kde je procesor zcela integrován do zařízení.

Velkou výhodou tohoto stacku je stabilita přenosu dat při směrování v mesh síti a optimalizace pro velice nízkou spotřebu (až 10 let). Na obrázku 1.2 je vnitřní schéma BitCloudu. Obsahuje předdefinované metody, které zahrnují všechny základní funkce procesoru, jako jsou časovače, AD převodníky, sériové komunikace atd. Hlavními částmi jsou služby jádra, uživatelské aplikace a sdílené služby na nízkých vrstvách. Nevýhodu, kterou si BitCloud nese, je však, že některé zdrojové kódy nejsou otevřené, ale používají se jejich zkompilované obrazy [9].



Obr. 1.3: Topologie sítí

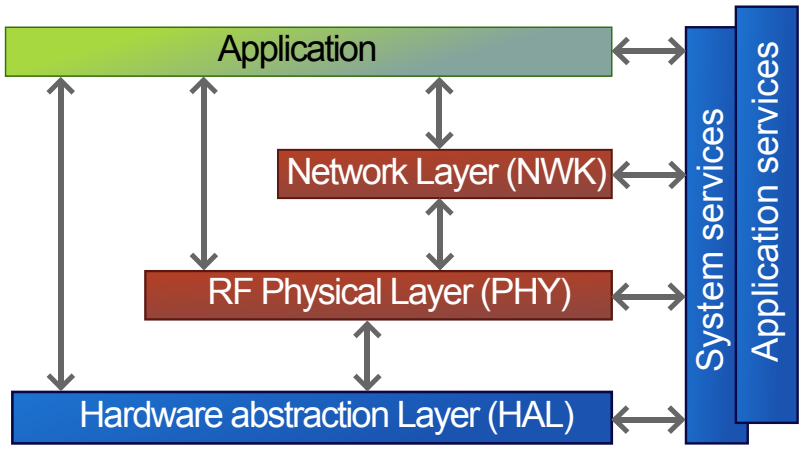
1.3.3 Lightweight mesh stack

Posledním stackem je velice odlehčený proprietární Lightweight mesh (dále pouze LwM). Ten se stará pouze o bezdrátovou komunikaci (používá datový rámec specifikovaný standardem IEEE 802.15.4), základní zabezpečení a automatické shromažďování měřených dat. Neimplementuje ale například komunikaci po SPI sběrnicích jako BitCloud nebo se nestará o přidělování adres v síti. Jak už název napovídá, tento stack podporuje mesh topologii sítě. V této topologii jsou dva hlavní typy zařízení. Koncová zařízení se starají o sběr dat a v mezičase mohou přecházet do úsporného režimu. Druhý typ jsou směrovací zařízení, která navíc přeposílají data od svých sousedů k bráně a zpět. Znamená to tedy, že budou mít daleko větší spotřebu energie, protože musí být připraveny směrovat provoz v síti. Na obrázku 1.4 je základní architektura LwM, která obsahuje tyto základní vrstvy:

- HAL – zajišťuje jednoduché hardwarové funkcionality, jako jsou časovače, kontrola spánku nebo přístup k rádiovému čipu.
- PHY – má na starosti funkce pro přístup rádiového čipu. Některé jsou přístupné pouze ze síťové vrstvy (například požadavek na odeslání dat nebo indi-

kace přijatých dat). Některé jsou přístupné z aplikační vrstvy (výběr kanálu, generování náhodných čísel, ...)

- NWK – tvoří jádro stacku. Její jednotlivé funkce uvedu později.
- System services – obsahuje běžné funkce dostupné všem vrstvám. Definuje základní datové typy a definice, softwarové časovače a jiné.
- Application services – obsahují moduly, které nejsou vyžadované stackem, ale jsou využívány většinou aplikací. Ve výchozím nastavení tato služba obsahuje jen modul Over-The-Air upgrade. Do této vrstvy bude případně vložen ovladač pro SPI komunikaci.



Obr. 1.4: Architektura Lightweight mesh stacku [10]

Hlavními vlastnostmi LwM stacku jsou:

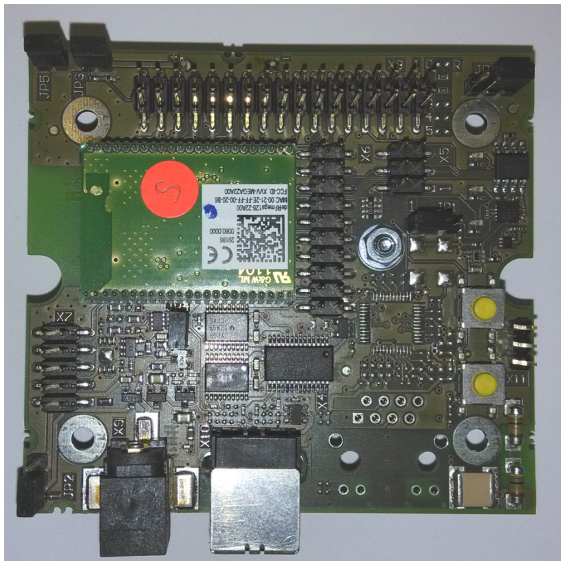
- jednoduchost konfigurace a používání
- 15 nezávislých aplikačních endpointů (portů)
- až 65 000 uzlů v jedné síti
- podpora jednoduchého routování a acknowledgment pakety
- možnost multicastového přenosu
- malé nároky na paměť (8 kB flash paměti a 4 kB RAM paměti pro typickou aplikaci)
- jednoduché zabezpečení přenosu [10]

2 PRAKTICKÁ ČÁST PRÁCE

Tato kapitola obsahuje specifikace použitých zařízení, analýzu problematiky a detailnější popis frameworků, které budou použity k realizaci sítě.

2.1 Hardware

(a) Raspberry Pi model B+ (procesor ARM BCM2835)



(b) koncová jednotka (procesor AVR ATMEGA128RFA1)

Obr. 2.1: Obrázky použitého hardwaru

2.1.1 Bezdrátová jednotka

Tento hardware bude snímat okolní veličiny a odesílat je bezdrátově bráně, případně dalším koncovým zařízením, pokud bude využito směrování v síti.

Popis architektury

Procesory s AVR architekturou se objevily poprvé v roce 1997. Dnes jich na trhu najdeme nepřeberné množství od malých 8-pinových (rodina tiny AVR) až po velké 100-pinové čipy (rodina mega AVR). Všechny tyto procesory jsou postavené na hardvardské architektuře (tj. oddělené paměti a sběrnice pro program a data) s redukovanou instrukční sadou RISC (Reduced Instruction Set Computing). Zařízení mají

standardně 1–256 KB paměti a disponují dalšími periferiemi, jako jsou hardwarové časovače, AD převodníky, sériové linky aj.

Použitý hardware

Byl použit vývojový kit od firmy Dresden-elektronik. Konkrétně byly pro koncová zařízení použity moduly deRFmega128 , ve kterých je procesor od Atmelu ATmega128RFA1 (viz obrázek 2.1b). Jedná se o 8-bitový procesor architektury AVR, který má přímo v pouzdře integrované rozhraní pro rádiový přenos. Modul obsahuje keramickou anténu a vysílá s výkonem +2,4 dBm. Výrobce udává, že lze použít pro přenos do 200 m [11]. Vybrané funkce procesoru:

- 128 kB flash paměť, 4 kB EEPROM a 16 kB SRAM
- JTAG rozhraní pro nahrávání programu a debuggování
- několik 8-bitových časovačů, 330 Ks/s AD převodník, analogový komparátor, on-chip teplotní senzor
- SPI, 2krát UART, TWI
- integrovaný rádiový čip s nízkou spotřebou (250–2000 Mb/s)
- hardwarové zabezpečení přenosu 128-bitovým AES šifrováním[12]

2.1.2 Brána

Brána je zařízení, které shromažďuje data z celé senzorové sítě a zajišťuje přístup k těmto datům přes webový server, který je součástí implementovaného softwaru.

Popis architektury

Typickým představitelem čipů s ARM (Advanced RISC Machine) architekturou jsou procesory ve smartphonech. ARM procesory byly uvedeny na trh už v roce 1985 jako výpočetní jednotky pro levné počítače [13].

Tyto procesory jsou oproti AVR komplexnější, dražší, náročnější na spotřebu (5 V, 600 mA), ale na druhou stranu pro ně lze programovat složitější aplikace. Tyto procesory jsou v mnoha variantách od 32-bitových s jedním jádrem až po několika jádrové 64-bitové. Obvykle mají více paměti na rozdíl od jednodušších AVR čipů. Dnes se používají procesory s taktem zhruba od 50 MHz až do několika GHz.

Použitý hardware

Celá brána se skládá z dvou fyzických zařízení. Hlavní část tvoří Raspberry Pi B+ (viz obrázek 2.1a), ke kterému je přes sériovou linku připojena stejná jednotka jako pro koncové zařízení. V této jednotce je program, který se chová jako obousměrný

most mezi bezdrátovou a sériovou komunikací. Raspberry Pi je asi nejrozšířenější zařízení pro domácí použití jako multimediální centrum, ale i pro vývoj embedded aplikací. Jako operační systém se používá linux v nejrůznějších distribucích [16]. Oficiální je ale odlehčená distribuce Debian (Raspbian). Několik parametrů RPi:

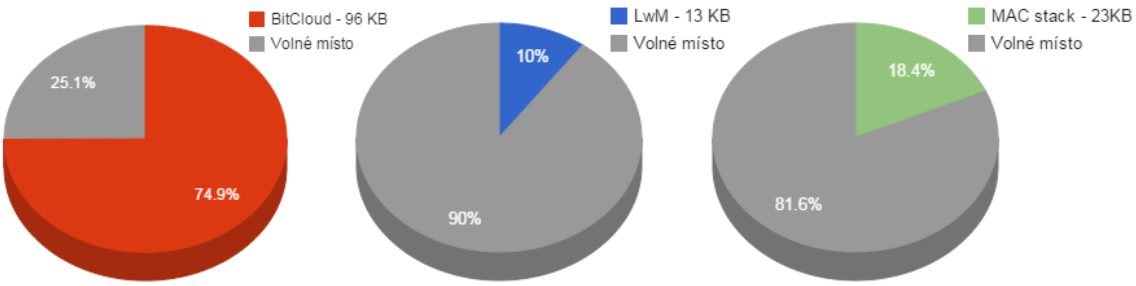
- 512 MB RAM, uložiště tvoří micro SD karta
- procesor ARM1176JZF-S 700 MHz (velikost čipu 1,9 mm^2)
- 4krát USB 2.0
- 40 GPIO pinů pro různé použití (přes tyto piny je propojeno sériovou linkou RPi s AVR modulem)
- HDMI, audio výstup (3,5mm jack)
- ethernet 100BASE-T
- ethernet 100BASE-T

2.2 Analýza řešení

Nejprve bylo provedeno srovnání dostupných knihoven pro bezdrátové jednotky a byl vybrán vhodný software pro Raspberry Pi. Dále byl navržen samotný komunikační protokol.

2.2.1 Koncové zařízení

Při rozhodování, který stack použít, bylo zohledněno několik faktorů. Za prvé, jak jsou velké jednotlivé zkompilované programy, a za druhé, co bude třeba doimplementovat. První bod byl poměrně jednoduchý. Ve všech frameworcích byl naprogramován jednoduchý program na blikání LED diodou a byly porovnány velikosti zkompilovaných programů. V grafech na obrázku 2.2 je vidět rozdíl jejich velikostí. Procesor má flash pamět o velikosti 128 KB.



Obr. 2.2: Srovnání velikosti programů pro jednotlivé balíky knihoven

BitCloud je robustní stack a implementuje velkou řadu funkcí, tudíž by do něj nemusel být implementován například ovladač SPI komunikace. Zatímco LwM je

framework pouze pro bezdrátovou komunikaci. Tudíž bude muset být implementována funkce na měření AD převodníku nebo ovladač pro I2C komunikaci. AD převodník byl použit pro měření výkonu solárních panelů. I přes tyto nedostatky byl vybrán LwM framework kvůli jeho jednoduchosti a open-source kódu.

Lightweight mesh

Z bezdrátového frameworku bylo použito několik již implementovaných funkcí pro bezdrátovou komunikaci. Konkrétně bylo využito směrování dat celou sítí. LwM podporuje topologii mesh, to znamená, že síť může mít jakoukoli strukturu a velikost. Síť musí ale splňovat tyto podmínky: dvě sousedící jednotky jsou navzájem propojeny a k přeposílání dat se používají jednotky, které nepřechází do stavu spánku.

Dále byla využita právě funkce uspávání jednotek kvůli úspoře elektrické energie. V kapitole 2.4.1 je toto téma rozvedeno.

Poslední využitou funkcí byla správa dat. Framework vytváří v každém paketu kontrolní součet CRC pro ověření integrity obsahu dat, a dále se také stará o potvrzování odesílaných dat, aby nedocházelo ke ztrátě.

Implementace aplikace

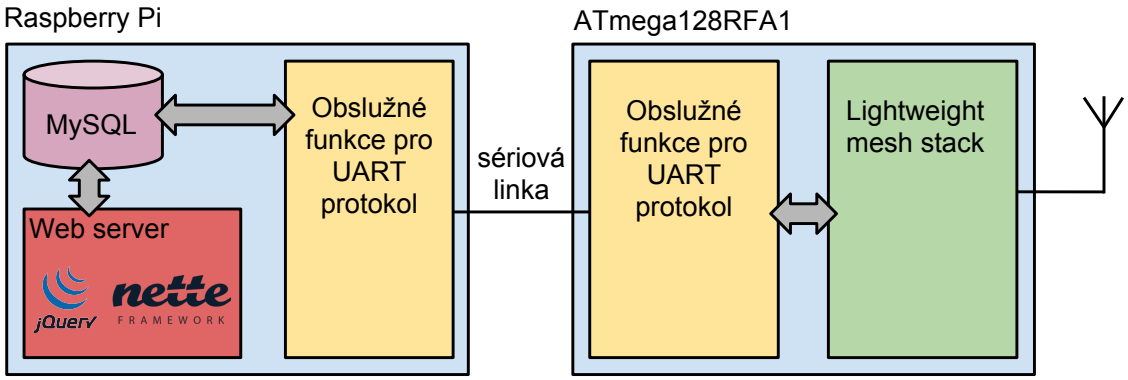
Celý stack je vlastně knihovna funkcí rozdělených do logických sekcí. Program je sestaven z dílčích *handlerů* (stavové mechanismy realizované pomocí funkce switch), které mají definované stavy, podle nichž se řídí chod aplikace. Aplikace používá ke svému řízení a neblokujícímu chodu takzvané stavové mechanismy. Hlavní aplikační mechanismus se jmenuje: `APP_TaskHandler`. Tímto handlerem se řídí celá aplikace od stavu inicializace, přes odesílání dat, potvrzení až po uspání a opětovné probuzení. Koncové jednotky mají tyto hlavní funkce:

- **Inicializace** - Nastavení bezdrátového přenosu, AD převodníků, sériové komunikace a nastavení časovače, po jehož uplynutí bude periodicky odesílána inicializační zpráva bráně.
- **Odeslání dat a potvrzení** - Po potvrzení inicializace jednotka odesílá s daným intervalem naměřená data a po odeslání čeká na potvrzení.
- **Dotaz na událost** - Než se jednotka uspí, odešle na bránu dotaz, zda nemá jednotka provést akci (sepnutí relé). Pokud ano, pak akci provede a posléze se uspí.

2.2.2 Brána

Jak je zmíněno výše, brána skládá ze dvou zařízení. Jednotka s procesorem ATmega128RFA1 má funkci mostu mezi komunikací bezdrátovou a sériovou s Raspberry Pi. Využívá tedy také LwM stack. Bezdrátová komunikace je z velké části obsluhována LwM, ale komunikace mezi ATmega128RFA1 a Raspberry Pi musela být navržena jako nový celek. Protokol provozovaný na sériové lince byl navržen, aby byl dostatečně abstraktní pro použití i v jiných systémech, ale na druhou stranu jednoduchý, aby nebyla zapotřebí velká režie provozu. Struktura protokolu je popsána v kapitole 2.3.1. Blokové schéma brány je na obrázku 2.3

V Raspberry Pi bylo nutné zprovoznit webový server, MySQL databázi pro ukládání veškerých dat a skript komunikující přes sériovou linku s celou sítí. Jako webový server byl použit standardní balík `apache2`, používaný jak v systémech Linux tak MS Windows. Dále byl doinstalován balík `php5` použitý pro tvorbu webového rozhraní a samozřejmě balíky databáze `mysql-server` a `python-mysqldb`.



Obr. 2.3: Blokové schéma brány sítě

Pro tvorbu webového výstupu aplikace byl použit český framework Nette [17], navržený pro tvorbu webových aplikací. Framework je objektový a skládá se z hlavních tří tříd:

- **Model** - Třída pro práci s daty. Ukládání a čtení z databáze.
- **Presenter (controller)** - Prostředník mezi Modely a Pohledy. Tato třída obsahuje všechny metody aplikace. Presenter pak volá podle potřeby různé pohledy a plní se daty z Modelů.
- **Pohled (view)** - Toto jsou jednotlivé HTML šablony předávané prohlížeči ke zpracování.

Nette se vyznačuje svou modularitou. Obsahuje sofistikované metody čtení dat z databáze, implementuje bezpečné formuláře a bezpečné vstupy (odolné vůči MySQL injection nebo Cross-site scripting) aplikace obecně.

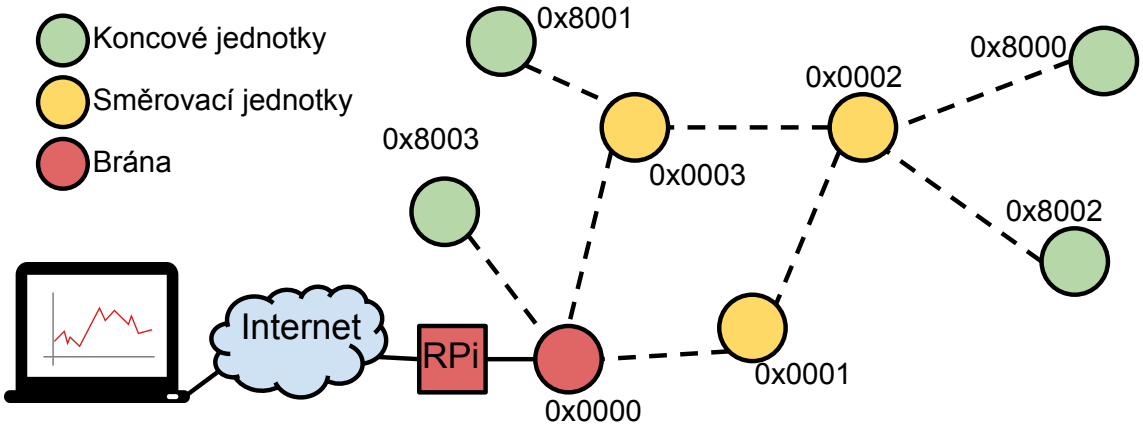
Další součástí webového interface je Javascriptová knihovna jQuery [18]. Bylo jí využito proto, že implementuje ajaxové požadavky, které jsou využity na vykreslování real-time grafů a zpětnovazebních indikací.

Pro komunikaci Raspberry s AVR byl použit programovací jazyk Python (konkrétně verze 2.7), protože obsahuje knihovny pro sériovou komunikaci, kontrolní součet crc i pro práci s databází. Bylo zapotřebí napsat tento skript, aby mohl být spuštěn v několika nezávislých vláknech, aby nebyl pro procesor blokován. Například pro sériovou komunikaci je zvolena jedna samostatná metoda, která s ostatními komunikuje pomocí *queues* (front), a obstarává obousměrnou komunikaci přes UART.

Jako databázový systém byl vybrán MySQL, protože je podporován všemi použitými nástroji a není zde potřeba instalovat robustnější řešení. Jako webová správa databáze se běžně používá *phpMyAdmin*, ale v tomto případě byla využita součást balíku Nette s názvem *Adminer*. Používá podobnou koncepci jako *phpMyAdmin*. Nemá všechny funkce jako *phpMyAdmin*, ale na druhou stranu je rychlejší a přehlednější. Struktura databáze je uvedena v kapitole 2.3.1.

2.3 Struktura sítě

Jak bylo zmíněno, síť podporuje mesh topologii sítě (vz obrázek 2.4), což je výhodné, protože není třeba se tolik zaobírat samotným rozmisťováním jednotek. Stačí, aby byly jednotky navzájem propojeny. Systém sítě je vytvořen tak, že každé jednotce se při počáteční konfiguraci nastaví, jaká data bude měřit, jakou bude mít adresu a s jakým intervalem bude data odesílat. Podrobné nastavení sítě je popsáno v příloze A.1.



Obr. 2.4: Schéma struktury sítě

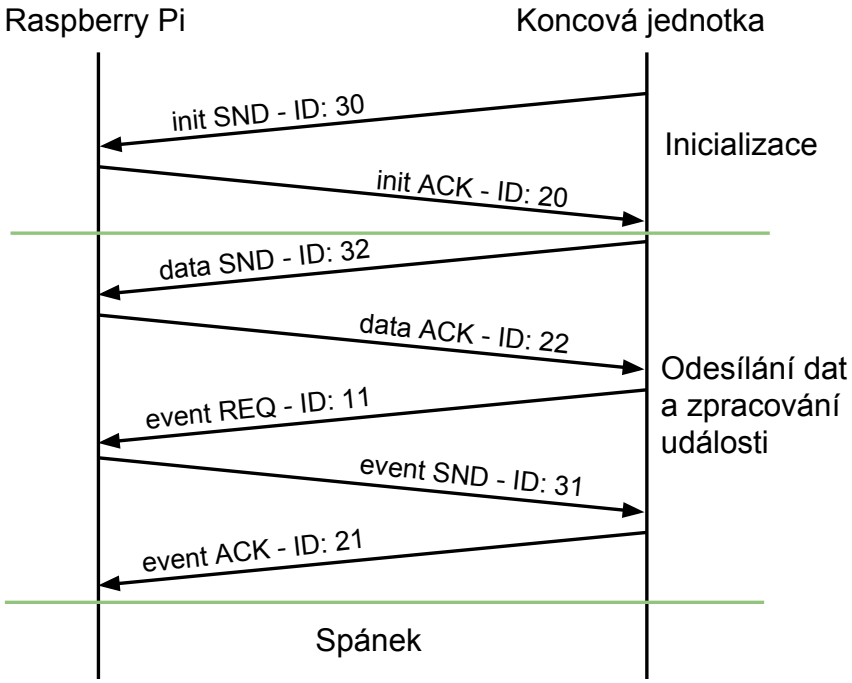
Na obrázku 2.4 jsou mimo jiné adresy jednotek. Takto definuje LwM typ jednotky. Koncové body (jednotky, které se mohou uspávat) mají adresní prostor `0x8000`

až `0xffff`. Směrovací jednotky pak musí být v aktivním režimu nepřetržitě. Jejich adresy mohou být v rozmezí `0x0000` až `0x7fff`, přičemž adresa brány pro vytvořenou síť je `0x0000` podle vzoru IP adres, kde má výchozí brána první možnou adresu v adresním prostoru.

Bylo také třeba komunikaci optimalizovat, protože brána, resp. její bezdrátová část, je stejná jednotka jako koncové body. Toto bylo jedno z úzkých hrdel celého systému a musely se v programu využít buffery pro bezdrátovou i sériovou komunikaci.

2.3.1 Popis navrženého protokolu

Tato část se konkrétně zabývá navrženým protokolem. Zvláště na straně sériové komunikace v kapitole 2.3.1. Požadavky na rychlost přenosu dat v senzorové síti nejsou tak velké jako například u přenosů videa nebo jiných multimediálních dat, ale spíše je kladen důraz na stabilitu sítě. Pokud provádíme měření několik týdnů, nebo dokonce i nepřetržitě musí být síť stabilní a to pokud možno s co nejmenším odběrem energie. Proto je síť navržena tak, aby v případě, že by došlo k výpadku jednotky a opětovnému připojení, nedošlo ke ztrátě dat, pouze by se indikovalo odpojení a opětovné přidání jednotky do sítě. Na obrázku 2.5 následuje životní cyklus jednotky:



Obr. 2.5: Diagram znázorňující životní cyklus koncové jednotky

- Jednotka nejdříve prochází inicializačním procesem, kdy nastaví parametry sítě, případně se inicializují SPI či I2C komunikace se samotnými senzory. Následuje pravidelné odesílání paketu na adresu brány se základními informacemi o jednotce. To je například počet měření, které bude jednotka odesílat, samozřejmě také unikátní identifikátory každého měření a interval, ve kterém se budou data odesílat.
- Pokud nepřijde jednotce od brány odpověď, přechází do chybového stavu, který je indikován souhlasným blikáním LED diod 2 a 3. Pokud odpověď obdrží, pak zahajuje odesílání prvních naměřených dat.
- Všechna odeslaná data jsou potvrzována. Proto jednotka čeká v aktivním režimu na odpověď. Jakmile přijme potvrzení odeslaných dat, odesílá požadavek na událost. Síť je totiž obousměrná. Nejen, že jednotky posílají bráně data, ale klient může také jednotlivé koncové body ovládat tzv. událostmi. V protokolu jsou definovány unikátní události například pro rozsvícení LED diod, ovládání relé, restart jednotky či jiné, které může klient doplnit. Pokud nastane situace, že jednotka neobdrží od brány odpověď, opět přechází do chybového stavu, kdy se snaží znovu připojit do sítě.
- Koncové zařízení obdrží od brány informace o událostech, které má potenciálně vykonat a po provedení odesílá potvrzovací zprávu bráně o proběhlé události. Poté jednotka přejde do spánku. Potvrzovací zpráva bráně se odesílá z toho důvodu, aby klient mohl vidět jestli se daná úloha provedla. Mějme například jednotku, která měří vlastnosti půdy na vinohradu, a protože se tento parametr mění pozvolna, tak vinař nastavil interval odesílání dat na 10 minut. Zároveň ale chce vinař spustit zavlažovací systém, tudíž odešle událost na příslušnou jednotku, která tuto úlohu zpracuje, až uplyne interval, po který je procesor uspaný. Aby vinař zjistil, že se zavlažovací systém spustil, jednotka zpět odesílá potvrzení o vykonané události.

Tento cyklus se pak stále opakuje kromě prvního bodu, který nastává pouze pokud nastal výpadek. Toto schéma je znázorněno na obrázku 2.5.

Síť používá definovanou sadu identifikátorů dat. Ta se dělí na identifikátory měření, příkazů a událostí. Názorný příklad příkazových identifikátorů je na obrázku 2.5.

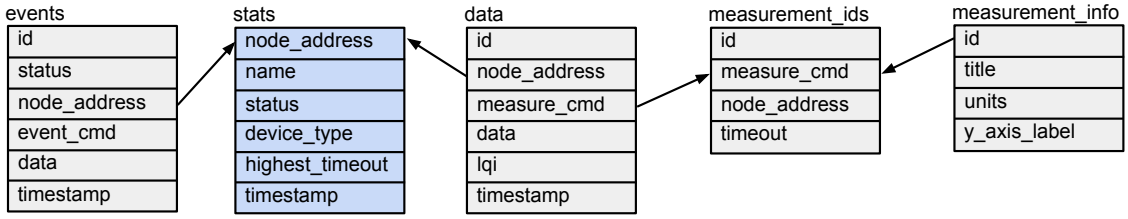
- **Command IDs** - Příkazové kódy jsou obsaženy v každém paketu a specifikují jeho obsah. Příkazy `1X` definují požadavky (požadavek zda je k dipozici v databázi událost k vykonání má kód `11`). Příkazy `2X` definují potvrzovací zprávy (init ACK, event ACK apod.). Poslední definovanou skupinou jsou posílaná data používající kódy `3X`. Do této kategorie patří například i data ohledně události, kterou má jednotka vykonat.

- **Event IDs** - Identifikátory událostí specifikují typ akce, kterou má jednotka provést, např. sepnout aktuátor, nastavit danou hodnotu na pinu atd.
- **Measurement IDs** - Poslední skupina kódů definuje dané měření. Tento kód se odesílá v paketu o měřených datech vždy před daty samotnými.

Jako příklad je uveden tento paket: `[ 170, 0, 0, 0, 128, 32, 3, 1, 242, 14, 245, 129, 240]`. Rámec odeslala jednotka s adresou `0x8000` (`0, 128`) bráně. Rámec obsahuje měřená data (`32`) o napětí na bateriích (`1, 100, 28`), kde první bajt je identifikátor měření a další dva udávají hodnotu napětí. V našem případě 3826 mV.

MySQL

Databáze zprostředkovává předávání dat mezi klientskou částí (webovým rozhraním) a částí sítě. Ukládají se do ní data o měření, stav sítě, události čekající na zpracování a další informace. Na následujícím obrázku 2.6 je uvedena struktura databáze.



Obr. 2.6: Struktura tabulek MySQL databáze

Hlavní tabulkou celé struktury je tabulka stats. Ta obsahuje údaje o každé jednotce v síti. Obsahuje sloupec pro *status* pro pojmenování, ve sloupci *status* je uchováván stav jednotky. (Možné stavy: *new*, *connected*, *reconnected*, *offline* a *out*). Důležitý je také v této tabulce sloupec *timeout*. Jeho hodnota se přepisuje pokaždé, když jsou přijata data. Python skript pak porovnává hodnotu tohoto sloupce u každé jednotky, a pokud je hodnota starší než pětinásobek intervalu odesílání, pak je jednotka považována za odpojenou a změní se její status na *offline*. Na webu se pomocí ajaxového volání vykreslí upozornění, že se jednotka odpojila, a přejde do stavu *out*.

Tabulka *events* uchovává data o čekajících, probíhajících i ukončených událostech. Když jednotka provede danou událost, změní se události stav z *processing* na *done* a webu se pak indikuje provedení události. Například když chci sepnout relé, tak po vykonané události systém indikuje na webovém rozhraní, že relé bylo sepnuto.

V tabulce *data* jsou uchovávána data o měření. Za zmínku stojí sloupec *lqi*, který uchovává kvalitu linky od brány (přes všechny routery) ke koncové jednotce. Tímto můžeme udržovat síť stabilní a případným slabým spojům může být přidána směrovací jednotka na posílení linky.

Tabulky *measurement_ids* a *measurement_info* jsou spíše doplňkové. *Measurement_ids* uchovává pro každou jednotku seznam odesílaných měření a *measurement_info* obsahuje popisky grafů ke každému měření.

Celá struktura tabulek je vždy obnovována s restartováním Raspberry Pi. Staré verze tabulek jsou archivovány přidáním časového otisku za jméno tabulky, tudíž jsou tabulky starších měření konzistentně uchovávány pro případné pozdější zpracování.

Bezdrátová část sítě

Bezdrátový přenos využívá struktury protokolu LwM. Stack má standardizovanou pouze strukturu rámce podle IEEE 802.15.4 (viz obrázek 2.7). Samotná data v rámci ale podle tohoto standardu zavedená nejsou. Jedná se tedy jen o MAC hlavičku, hlavičku s informacemi o síti, nepovinný kontrolní kód zprávy (MIC) a kontrolní kód (CRC).

16	8	16	16	16	8	8	16	16	4	4	0/16	Variable	0/32	16
Frame Control	Sequence number	PAN ID	Destination Address	Source Address	Frame Control	Sequence number	Source Address	Destination Address	Source Endpoint	Destination Endpoint	Multicast Header	Variable	MIC	CRC
MAC Header					Network Header							Payload	MIC	CRC

Obr. 2.7: Hlavní formát rámce Lightweight mesh stacku [10]

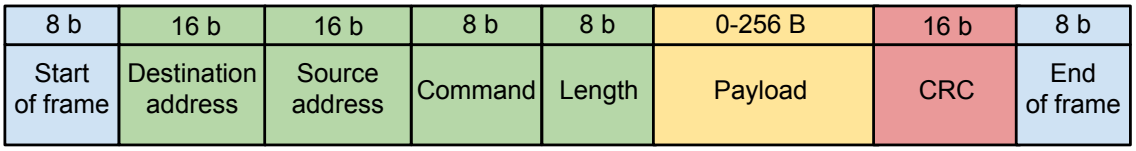
Nejdůležitější části rámce: *MAC header* obsahuje 2-bajtové pole *Frame control*, kde jsou uložené informace o potvrzení rámce, zda je použité šifrování a jestli se jedná o multicastový rámec. V části *Network header* je důležité zmínit pole *Source Address Field* (2 B), *Destination Address Field* (2 B), *Source Endpoint Field* (4 b) a *Destination Endpoint Field* (4 b), kde jsou uloženy adresy odesílatele, příjemce a koncových bodů, které si můžeme představit jako aplikační porty.

Stack samotný řídí směrování, šifrování, spolehlivý přenos dat atd.

UART rozhraní

Největší částí práce byl celkový návrh protokolu na sériové lince. Strukturu rámce znázorňuje obrázek 2.8. Každý rámec začíná binární posloupností `0xAA`. Jakmile přijímač zachytí tento signál, začíná další data ukládat a tvoří postupně celý rámec. Ve struktuře pak následují adresy zdroje a cíle. Pole *Command* obsahuje identifikátor, který se používá při zpracování dat. Například rámec s měřenými daty obsahuje v tomto poli hodnotu `32`. Následuje pole *Length*, které určuje délku posílaných dat. Paket má tedy dynamickou velikost. To je výhodné, protože se z drtivé většiny data vejdou do jednoho paketu a tak šetříme energii jednotek. Při odesílání má jednotka logicky největší spotřebu (cca 35 mA). Posledním důležitým polem je kontrolní cyklický součet *CRC*. Data se musí na cílové stanici kontrolovat, protože používáme

reálné přenosové médium, kde se projevuje rušení, a data mohou být poškozena. Pokud se tak stane, jsou data zahozena. Celý rámec ukončuje opět binární posloupnost, v tomto případě s hodnotou `0xF0`.



Obr. 2.8: Formát rámce sériové komunikace

Je samozřejmě možné, že se v paketu kdykoli objeví startovací nebo zakončovací bajt, který by znovu nastartoval čtení nového paketu, nebo by zakončil paket předčasně. Proto je v paketu důležité pole *Length*, které říká jak dlouho má jednotka data číst, ať mají jakoukoli hodnotu. Zkrátka na začátku jednotka zkontroluje, zda první přijatý bajt má hodnotu startovacího bajtu, pak vyčítá a ukládá data, až dojde nakonec a opět zkontroluje, zda poslední bajt má správnou hodnotu. Následně pak ověřuje, zda nebyla data poškozena.

2.3.2 Kritická místa

Při tvorbě sítě musela být některá místa optimalizována, aby nedocházelo k nežádoucím výpadkům a síť byla konzistentní.

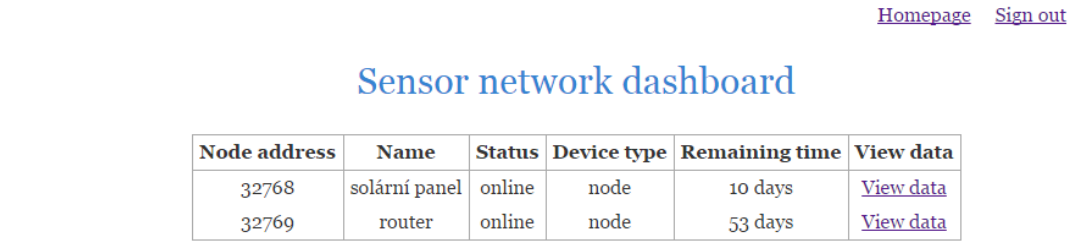
První takové místo byl obslužný Python skript v Raspberry Pi. Bylo třeba rozdělit funkce do několika logických částí, aby byly spuštěné paralelně v několika vláknech a procesor resp. operační systém je prováděl současně. K sériové lince může v jeden časový úsek přistupovat jen jeden proces, proto byla vytvořena funkce, která nezávisle na ostatních částech skriptu, obsluhuje sériovou linku. Další vlákno tvoří funkce, která periodicky kontroluje stav jednotek a případně ohlásí, že některá z jednotek je ve stavu *offline*. Dále je ve skriptu funkce pro přípravu paketu k odeslání a funkce, která zpracovává data podle pole *Command*.

Další místo bylo v samotné koncové jednotce, konkrétně bylo třeba se zaměřit na správnou funkci stavových mechanismů. Například, pokud jednotka obdržela data, pak změnila svůj stav a měla data zpracovat. Ale než se začala data zpracovávat, stav aplikace se změnil na odesílaní dat, protože vypršel časovač pro periodické odesílání. Proto bylo třeba stavové mechanismy správně nakonfigurovat.

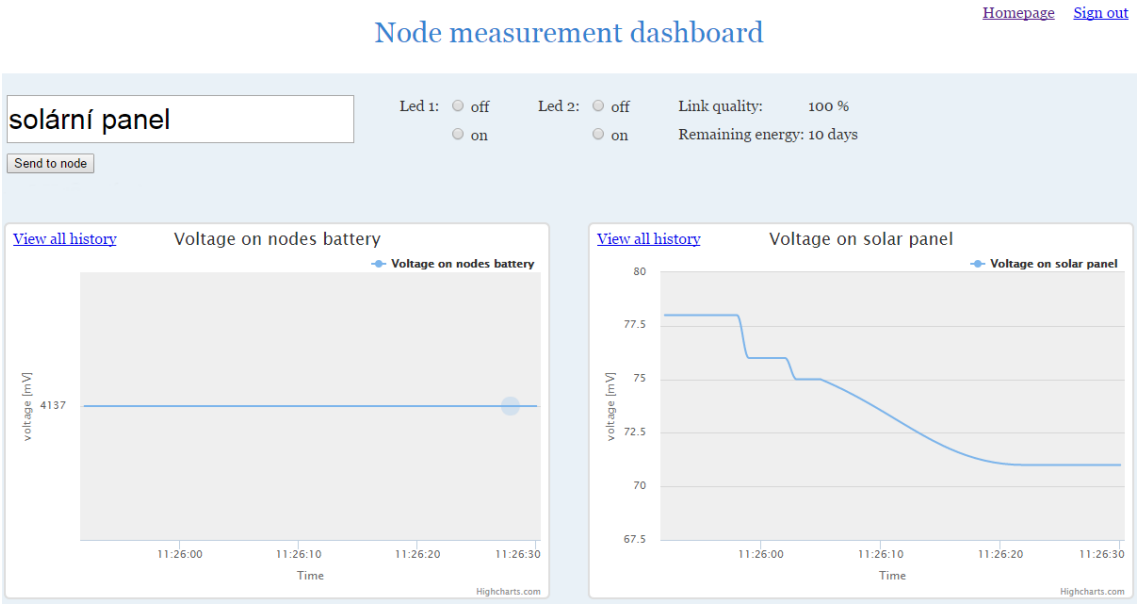
Potenciální úzké hrdlo by mohlo nastat na sériové lince při připojení velkého počtu jednotek, tudíž by se nestihla data přenést s aktuálně nastavenou rychlostí linky. Pokud by to nastalo, lze jednoduše rychlost změnit v obou zařízeních brány.

2.3.3 Obsluha sítě

K měřeným datům se přistupuje přes webové rozhraní. Po přihlášení se zobrazí tabulka viz obrázek 2.9 se všemi jednotkami v síti.



Obr. 2.9: Domovská stránka senzorové sítě



Obr. 2.10: Karta konkrétní koncové jednotky

Kliknutím na „view data“ se zobrazí karta viz obrázek 2.10 s podrobnostmi a měřením pro jednotlivé uzly. Zde můžeme jednotku libovolně pojmenovat. Dále na této kartě můžeme zpětně posílat příkazy jednotkám. Pro demonstraci této funkce můžeme ovládat LED diody jednotek. V pravé horní části jsou zobrazeny aktuální informace o kvalitě spojení s jednotkou a odhadovaný čas do vybití baterií.

2.4 Energetická náročnost

Velice důležitým parametrem u senzorových sítí je energetická *nenáročnost*. Je kladen důraz na to, aby jednotka vydržela co nejdéle bez obsluhy. Proto byla aplikace v jednotkách naprogramována tak, aby se většinu času jednotka udržovala v úsporném režimu. Použitý procesor ATmega128RFA1 má několik módů spánku podle toho, jakou část jednotky nemusíme využívat. V našem případě je použit mód *power-save*, který vypíná všechny periferie a přechází do tzv. asynchronního módu, kdy může být procesor probuzen externím přerušením nebo vypršením časovače *Timer2*. Tento mód není nejúspornější, protože musí udržovat aktivní interní oscilátor a obvody časovače. V tabulce 2.1 je uvedeno porovnání spotřeby energie uváděné výrobcem a reálně změřené.

Tab. 2.1: Srovnání udávané výrobcem a měřené energetické náročnosti jednotky

	výrobce	změřeno	průměrná doba v režimu [s]
inicializace (aktivní režim) [mA]	14,5	20	3,75
odesílání dat [mA]	22	25	0,3
úsporný režim [µA]	310	234	záleží na intervalu odesílání

Tabulka 2.2 obsahuje dobu, po kterou může jednotka fungovat s různými intervaly odesílání dat. První tři intervaly byly zvoleny pro reálné měření. Jednotka je napájena třemi články Ni-MH baterií s kapacitou 1800 mAh, proto i tyto teoretické výpočty budou vztaženy na stejnou kapacitu baterií. Reálně se však jednotka vypne, pokud je na bateriích menší napětí než 3,5 V. Napětí na vybitém Ni-MH článku je 1 V, proto při napětí 3,5 V nejsou ještě baterie zcela vybity a využije se zhruba jen 90 % jejich kapacity. U Ni-MH baterií se také velice významně podílí efekt samovybíjení, který je závislý na typu baterie, ale pohybuje se kolem 20% úbytku kapacity za 6 měsíců. Výsledky měření a porovnání se nachází v kapitole 2.4.1.

První důraz je kladen na samotný úsporný režim jednotky. Další krok je následně jednotku nabíjet z dalších zdrojů energie, jako je například vítr nebo slunce. Proto bylo v kapitole 2.4.2 provedeno měření výkonu solárních panelů za účelem zjistit, zda by mohli být některé panely použity jako plnohodnotné nebo částečné zdroje energie.

Výpočet doby, po kterou bude jednotka pracovat zjistíme takto: nejprve je potřeba zjistit poměr mezi dobou, kdy je jednotka v aktivním a kdy v úsporném režimu:

$$N_{sleep} = \frac{t_{sleep}}{t_{active} + t_{sleep}} = \frac{0,1}{0,3 + 0,1} = 0,25. \tag{2.1}$$

Tab. 2.2: Teoretická doba, po kterou zůstane jednotka aktivní v závislosti na intervalu odesílání dat

interval odesílání dat	výdrž jednotky podle výrobce	výdrž jednotky podle změřených hodnot
0,1 sekundy	4 dny	3 dny 12 hodin
1 sekunda	11 dní 16 hodin	10 dní 10 hodin
10 sekund	48 dní 12 hodin	48 dní
1 minut	78 dní	83 dní 15 hodin
10 minut	87 dní	97 dní

Dále pak již zjistíme celkovou dobu:

$$\begin{aligned} t &= \frac{C_{bat}}{I_{active} \cdot (1 - N_{sleep}) + I_{sleep} \cdot N_{sleep} + 0,45} \cdot 24 = \\ &= \frac{1620}{22 \cdot 0,75 + 0,31 \cdot 0,25 + 0,45} \cdot 24 \doteq 4 \text{ dny.} \end{aligned} \tag{2.2}$$

Rovnice 2.2 obsahuje sníženou kapacitu baterie C_{bat} zhruba o 10 %. Také se ve jmenovateli musí staticky přičíst samovybíjecí konstanta, která je cca 0,45 mAh. Nakonec se výsledek vynásobí počtem hodin za den. Proudy I_{active} a I_{sleep} jsou v tomto případě dodané výrobcem.

V další části jsou uvedena měření reálné výdrže baterií jednotek a měření výkonu solárních panelů, které by mohly být alternativním nebo přídavným zdrojem energie.

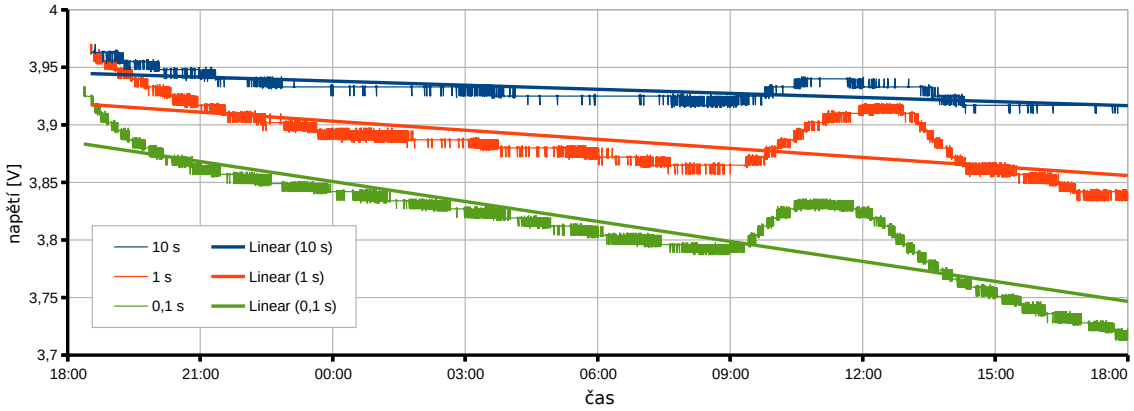
2.4.1 Měření spotřeby energie

Celý vytvořený systém byl následně využit pro praktické měření.

Bylo provedeno měření trvající 24 hodin, aby bylo možné odhadnout reálnou spotřebu zařízení. Jednotkám byly nastaveny stejné intervaly jako je uvedeno v prvních třech řádcích tabulky 2.2 tj.: 0,1, 1 a 10 sekund. Solární panely byly rozmístěny na parapet okna na jihovýchodní straně budovy. Graf na obrázku 2.11 znázorňuje průběhy napětí. Pro zjednodušení byly grafy proloženy přímkou, z níž byla odhadnuta reálná spotřeba jednotek.

Grafům byla změněna časová osa pouze na rozmezí, kde panely dodávaly energii. Nelze tudíž předpokládat, že by jednotka byla napájena pouze ze slunce. Bude tedy třeba zachovat napájení z baterií jako hlavní zdroj.

Z grafů byly vyčteny směrnice přímek, které udávají rychlost vybíjení baterií. Tabulka 2.3 uvádí výsledky reálného měření a srovnání s teoretickými výpočty. Je možné si na obrázku všimnout převýšení u všech jednotek zhruba kolem 11. hodiny dopoledne. Je to s největší pravděpodobností, proto že v tuto dobu na jednotky nejvíce svítilo slunce, a tím, jak se zahřály baterie jednotek, se na nich krátkodobě zvedlo napětí.



Obr. 2.11: Graf úbytku napětí baterií v závislosti na intervalu odesílání dat

Tab. 2.3: Reálná výdrž jednotek a srovnání s teoretickými výpočty

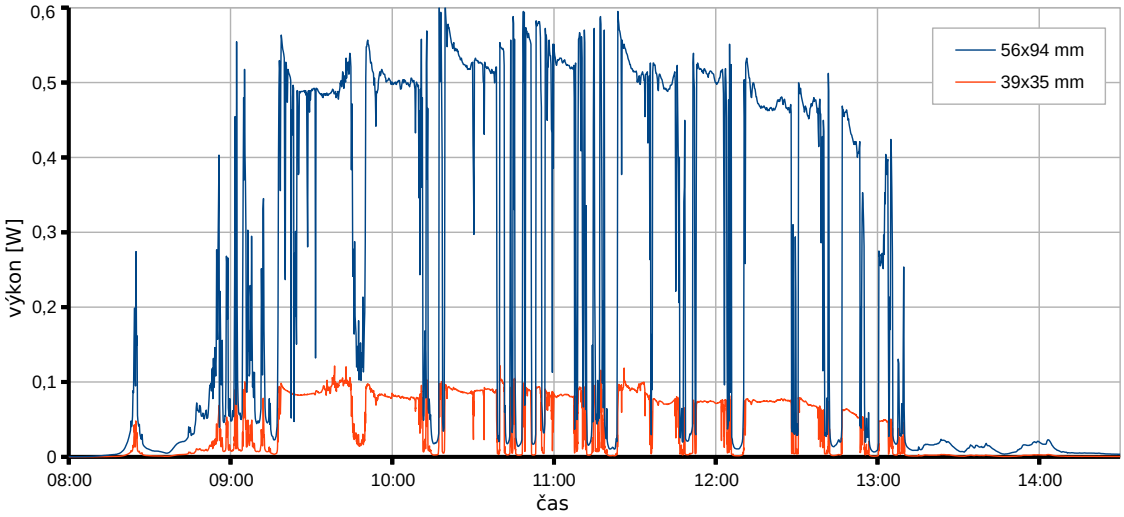
interval odesílání dat	0,1 s	1 s	10 s
pokles napětí za hodinu [mV/h]	5,56	2,5	1,1
reálná výdrž jednotky	5 dní	11 dní 16 hodin	26 dní 12 hodin
výdrž jednotky podle výrobce	4 dny	11 dní 16 hodin	48 dní 12 hodin
výdrž jednotky podle měření spotřeby	3 dny 12 hodin	10 dní 10 hodin	48 dní
procentuální odchylka [%]	25	5	80

Výsledky pro malé intervaly odesílání jsou podobné jako reálně vypočítané. Doba pro interval 10 sekund se liší od teoretického výpočtu. Je mnoho faktorů, které mohly tento výsledek zkreslit. Pro přesnější hodnoty by bylo třeba měřit delší dobu. Dále může být výsledek ovlivněn větší spotřebou jednotek kvůli indikačním LED diodám nebo starším bateriím, které již nemají deklarovanou kapacitu. Proto jsou tyto výpočty pouze orientační.

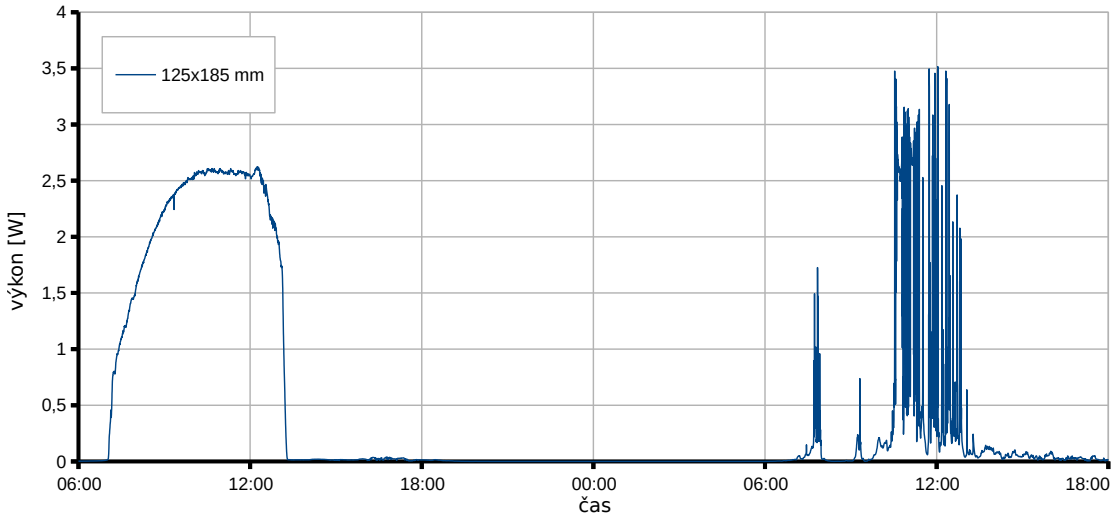
2.4.2 Měření výkonu solárních panelů

Měření výkonu proběhlo ve dvou fázích. V první proběhlo měření na dvou menších panelech (39x35 a 56x94 mm) a v druhé byl změřen velký panel (125x185 mm), viz obrázek 2.12. Menší panely byly měřeny po dobu jednoho dne a větší po dobu dvou dnů. Na obrázku 2.13 je graf prvního měření a na obrázku 2.14 se nachází graf z měření panelu 125x185 mm. Zde je jasně vidět rozdíl mezi dnem s oblačností a naprosto jasným dnem. Bylo kontrolováno, zda se nejedná o chybu měření, ale podle archivu meteorologických stanic v Brně byl první den měření naprosto bez oblačnosti.

Obr. 2.12: Použité solární panely

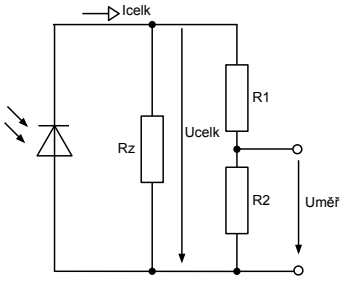


Obr. 2.13: Graf výkonu solárních panelů 39x35 a 56x94 mm



Obr. 2.14: Graf výkonu pro měření 125x185mm solárního panelu

K měření výkonu bylo nutné zjistit voltampérovou charakteristiku solárního panelu. Pokud změříme proud panelem nakrátko a poté začneme postupně zvyšovat odpor zátěže, zjistíme, že na panelu zůstává podobný proud (tato část charakteristiky se dá považovat za lineární). Když napětí přesáhne určitou mez, začne dodávaný proud poměrně prudce klesat. Tento zlom se samozřejmě liší pro různě silné osvětlení panelu. Pro profesionální měření se používají složité obvody, které přizpůsobují svůj odpor a tak panel zatěžují ideálně v závislosti na osvitu. V našem případě bylo použito jiné řešení. Bylo použito zapojení naznačené na obrázku 2.15.



Obr. 2.15: Zapojení pro měření na solárních panelech

Dělič napětí složený z $R1$ a $R2$ byl vypočítaný tak, aby na rezistoru $R2$ nevzniklo větší napětí, než by byla jednotka schopna změřit. Rezistor Rz pak tvořil zátěž pro solární panel, aby se ze změřeného napětí na $R2$ dal vypočítat reálný výkon.

Pro výpočet výkonu bylo nejdříve zjištěno celkové napětí na panelu:

$$U_{celk} = U_{R2} \cdot \frac{R1 + R2}{R2} \tag{2.3}$$

a celkový odpor zapojení:

$$R_{celk} = \frac{(R1 + R2) \cdot R_z}{(R1 + R2) + R_z}. \tag{2.4}$$

Následně byl vypočítán výkon panelu:

$$P_{panel} = \frac{U_{celk}^2}{R_{celk}}. \tag{2.5}$$

Každému panelu byl změřen proud nakrátko a napětí naprázdno a podle obecných charakteristik solárních panelů byl vybrán vhodný zatěžovací odpor, aby solární panel byl co nejefektivněji využit pro různé osvětlení. Změřené výkony tudíž nejsou maximální, které jsou panely schopné dodat. Jsou to spíše reálné hodnoty, kterých jsme schopni dosáhnout v normálních neideálních podmínkách.

Tab. 2.4: Průměrný výkon solárních panelů

rozměr panelu	průměrný výkon [W]	pokrytí příkonu jednotky pro 1s interval [%]
39x35 mm	0,01	42
56x94 mm	0,06	252,1
125x185 mm	0,3	1260

Průměrný výkon panelů je uveden v tabulce 2.4. Z těchto dat vyplývá, že pro kompletní napájení jednotky by se mohl použít panel 56x94 a 125x185 mm. Po připojení panelů by teoretická výdrž jednotek byla nekonečná. Nicméně v každém případě bude jednotka muset být napájena z baterií, které budou přes den dobíjeny panelem, jelikož z předchozích grafů je patrné, že přes noc ani největší panel nedodává žádnou energii. Baterie Ni-MH mají ale problém se samovybíjením, tudíž by se reálná výdrž jen prodloužila.

3 ZÁVĚR

Po nastudování problematiky senzorových sítí byl vybrán konkrétní software pro bezdrátové jednotky a výchozí bránu. Bezdrátová část sítě je postavena na frameworku Lightweight mesh. Raspberry Pi pak obsahuje Python skript, který tvoří rozhraní mezi webovým serverem a samotnou senzorovou sítí.

Byla sestavena testovací síť s topologií mesh, kde mohou bezdrátové jednotky mít funkci směrovačů nebo koncových měřících zařízení. Koncové jednotky přecházejí do úsporného režimu, aby šetřili energii.

Byl navržen komunikační protokol, který zajišťuje správnou inicializaci jednotek a celkový tok dat. I v případě, že dojde k výpadku, je systém připraven znovu spojení navázat a pokračovat ve sběru dat. Protokol je navržen pro obousměrnou komunikaci. Senzorové jednotky jednak odesílají data bráně, která je ukládá do MySQL databáze, ale také se koncové jednotky pravidelně dotazují na akce, které by měly vykonat. Například sepnutí aktuátoru nebo provedení jiné definované akce.

Brána sítě obsahuje webové rozhraní, které zobrazuje naměřená data a slouží pro správu sítě. V hlavním výpisu je zobrazen seznam připojených jednotek s informacemi o zbývající energii, jejich jménu a aktuálním stavu. Systém informuje uživatele o výpadcích a znovupřipojení jednotek do sítě. Karta konkrétního zařízení obsahuje grafy vykreslující měřená data v reálném čase. Dále je možné si prohlédnout celou historii měření nebo grafy exportovat například do pdf nebo svg.

Prostřednictvím vytvořené aplikace bylo provedeno měření energetické náročnosti sítě. Koncové senzorové jednotky vydrží při napájení z baterií řádově desítky dní v závislosti na intervalu odesílání dat.

Dále byly měřeny tři solární panely, z nichž by mohly být dva použity jako přídavný zdroj energie, který by dobíjel baterie a výdrž jednotky by byla teoreticky neomezená.

Tato práce může být rozšířena o knihovny k nejrůznějším senzorům pro další měření a využita komerčně i nekomerčně pro sledování libovolných veličin.

LITERATURA

[1] *Wireless Sensor Network*, National Instruments [online]. 2014, [cit. 24. 11. 2014]. Dostupné z URL: <http://www.ni.com/wsn/whatis/>.

[2] *Sensor Applications for smarter world*, Libelium [online]. 2014, [cit. 24. 11. 2014]. Dostupné z URL: <http://www.libelium.com/top_50_iot_sensor_applications_ranking/#show_infographic>.

[3] BECHYNSKÝ, Š. *Internet of Things*, Zdroják [online]. 2014, poslední aktualizace 9. 7. 2012 [cit. 24. 11. 2014]. Dostupné z URL: <http://www.zdrojak.cz/clanky/internet-of-things/>.

[4] ROUSE, M. *Internet of Things (IoT)*, WhatIs [online]. 2014, poslední aktualizace červenec 2014 [cit. 24. 11. 2014]. Dostupné z URL: <http://whatis.techtarget.com/definition/Internet-of-Things>.

[5] *IEEE 802.15™: WIRELESS PERSONAL AREA NETWORKS (PANs)*, IEEE STANDARD ASSOCIATION [online]. 2014, [cit. 24. 11. 2014]. Dostupné z URL: <http://standards.ieee.org/about/get/802/802.15.html>.

[6] *User Guide – Atmel IEEE 802.15.4 MAC Software*, Atmel Corporation [online]. 2014, poslední aktualizace květen 2012 [cit. 24. 11. 2014]. Dostupné z URL: <http://www.atmel.com/images/doc8412.pdf>.

[7] FARAHANI, S. Popis Zigbee protokolu V *Zigbee wireless networks and transcievers*, Amsterdam: Elsevier ; Newnes, 2008. ISBN 978-0750683937.

[8] POOLE, I. *ZigBee Technology Tutorial*, Radio-Electronics [online]. 2014, [cit. 24. 11. 2014]. Dostupné z URL: <http://www.radio-electronics.com/info/wireless/zigbee/zigbee.php>.

[9] *Developer Guide – Atmel BitCloud*, Atmel Corporation [online]. 2014, poslední aktualizace duben 2012 [cit. 24. 11. 2014]. Dostupné z URL: <http://www.atmel.com/Images/Atmel-8199-BitCloud-Developer-Guide_User-Guide_AVR2050.pdf>.

[10] *Developer Guide – Lightweight Mesh*, Atmel Corporation [online]. 2014, poslední aktualizace březen 2014 [cit. 24. 11. 2014]. Dostupné z URL: <http://www.atmel.com/images/atmel-42028-lightweight-mesh-developer-guide_application-note_avr2130.pdf>.

[11] *Datasheet deRFmega128*, Dresden-elektronik [online]. 2014, [cit. 24. 11. 2014]. Dostupné z URL: <http://www.dresden-elektronik.de/funktechnik/products/radio-modules/avr-single-chip-modules/description/?L=1&eID=dam_frontend_push&docID=1203>.

[12] *Datasheet ATmega128RFA1*, Atmel Corporation [online]. 2014, poslední aktualizace září 2014 [cit. 26. 11. 2014]. Dostupné z URL: <http://www.atmel.com/Images/Atmel-8266-MCU_Wireless-ATmega128RFA1_Datasheet.pdf>.

[13] *ARM Company Milestones*, ARM Holdings [online]. 2014, [cit. 1. 12. 2014]. Dostupné z URL: <http://www.arm.com/about/company-profile/milestones.php>.

[14] *Datasheet AT91sam7x512*, ARM Holdings [online]. 2014, [cit. 11. 12. 2014]. Dostupné z URL: <http://www.atmel.com/Images/Atmel_32-bit-ARM7TDMI-Flash-Microcontroller_SAM7X512-256-128_Datasheet.pdf>.

[15] *Introduction to Freertos*, RTOS & embedded software [online]. 2014, [cit. 6. 12. 2014]. Dostupné z URL: <http://www.freertos.org/RTOS.html>.

[16] *Raspberry Pi distribudions*, elinux.org [online]. 2015, [cit. 5. 5. 2015]. Dostupné z URL: <http://elinux.org/RPi_Distributions>.

[17] *Český framework Nette*, nette.org [online]. 2008, [cit. 11. 5. 2015]. Dostupné z URL: <http://nette.org/cs/>.

[18] *Javascriptová knihovna jQuery*, jquery.com [online]. 2006, [cit. 11. 5. 2015]. Dostupné z URL: <https://jquery.com/>.

[19] *Srovnávací test samovybíjení tužkových (AA) baterií*, jquery.com [online]. 2006, poslední aktualizace 31. 3. 2008 [cit. 13. 5. 2015]. Dostupné z URL: <http://www.fotonmag.cz/svitilny/baterie/srovnavaci-test-samovybijeni-tuzkovych-aa-baterii/>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ACK	potvrzení přijetí dat – acknowledgement of recieved data
AD	analogově digitální převod – analog digital conversion
AES	pokročilý šifrovací standard – advanced Encryption Standard
ALU	aritmeticko-logická jednotka – arithmetic logic unit
ARM	pokročilé RISC zařízení – advanced RISC machine
AVR	procesorová architektura firmy Atmel – processors architecture of Atmel corporation
CRC	cyklická redundantní kontrola – cycklic redundancy check
DA	digitálně analogový převod – digital analog conversion
EEPROM	elektricky mazatelná programovatelná paměť pouze pro čtení – ellectricaly erasable programmable read-only memory
GHz	giga hertz – giga hertz
GPIO	sběrnice pro všeobecné vstupně/výstupní použití – general-purpouse input/output
HDMI	multimediální rozhraní s vysokým rozlišením – high-definition multimedia interface
HTML	hypertextový značkovací jazyk – hypertext markup language
I2C	multi-masterová počítačová sériová sběrnice – inter-integrated circuit
IEEE 802.15.4	standard pro bezdrátové senzorové sítě – ZigBee standard for wireless sensor networks
IoT	internet složen z věcí – internet of things
IP	adresa internet protokolu – internet protocol address
IPv6	adresa internet protokolu verze 6 – internet protocol version 6 address
JTAG	rozhraní pro programování a testování mikročipů – joint test action group

LED	dioda vyzařující viditelné světlo – light emmiting diode
LwM	odlehčený mesh framework pro senzorové sítě – lightweight mesh stack
MAC	jedinečná fyzická adresa jednotky – media access control address
MHz	mega hertz – mega hertz
MySQL	databázový jazyk strukturovaných dotazů – my structured query language
Ni-MH	nikl-metal hydridový akumulátor – nickel-metal hydride battery
PHY	fyzická vrstva sítě – physical layer of network
RAM	paměť s přímím přístupem – random-access memory
RISC	redukovaná počítačová instrukční sada – reduced instruction set computing
SD	standard paměťových karet – secured digital memory card
SPI	sériová sběrnice periferií – serial peripheral interface
SRAM	statická paměť s přímím přístupem – static random-access memory
TWI	dvouvodičové sériové rozhraní – two wire interface
UART	univerzální asynchonní sériová komunikace – universal asynchronous receiver/transmitter
USB	univerzální sériová sběrnice – universal serial bus
WiFi	bezdrátový přenos – wireless fidelity
WLAN	lokální bezdrávoá počítačová síť – wireless local area network
WPAN	personální bezdrátová síť – wireless personal area network

SEZNAM PŘÍLOH

A PŘÍLOHA

A.1 Konfigurace a zprovoznění celého projektu

Tato část je určena pro pozdější využití této práce. Obsahuje podrobný návod jak tuto síť zprovoznit s co nejmenší námahou.

A.1.1 Potřebný software a hardware

Jako vývojové prostředí pro jednotky ATmega128RFA1 byl použito Atmel studio verze 6.2.1548 s programátorem JTAG mkII. Je možné, že po instalaci této verze nastanou potíže s ovladači pro programátor. Ty se mohou odstranit tak, že se přiinstaluje starší verze Atmel studia. V tomto případě byla použita vezre 6.1.2674, která obsahuje starší verzi ovladačů.

Jako výchozí brána bylo použito Raspberry Pi verze B+. Je doporučeno používat SD kartu class 10 kvůli rychlejšímu přenosu dat. Zvláště, když by jednotky měly intervaly odesílání dat v řádech jednotek sekund. Kvalitní program pro zálohování a obnovu obsahu SD karty je Win 32 disk Imager. Pro úpravu kódu v Raspberry Pi postačí nástroje obsažené v jeho operačním systému. K Raspberry se přistupuje vzdáleně přes SSH. Případně lze použít například PsPad a připojovat se k RPi přes FTP.

Kódy pro bezdrátové jednotky i pro RPi jsou na přiloženém CD ve složkách *node*, *Raspberry Pi* a *gateway*.

A.1.2 Konfigurace koncových jednotek

Hlavní soubor v projektu Atmel studia pro konfiguraci je `config.h`. Zde jsou důležité tyto parametry:

- `APP_ADDR` - Tento parametr nastavuje adresu jednotky. Zároveň ale také nastavuje, zda se bude chovat jako směrovač (to znamená, že bude směrovat provoz v síti, ale nebude přecházet do úsporného režimu). Adresy pro směrovače jsou v rozmezí `0x0001-0x7fff` a pro koncové jednotky `0x8000-0xffff`.
- `DATA_SEND_INTERVAL` - Zde nastavujeme jak často bude jednotka data odesílat. Mezi dvěma odesláními je ve výchozím nastavení uspaná. Čas se zde nastavuje v milisekundách!
- `WAIT_FLUSH_TIMER_INTERVAL` - Tímto parametrem nastavujeme, jak dlouho po odeslání paketu jednotka čeká na odpověď. Pokud tento čas od odeslání vyprší, pak jednotka přejde do chybového stavu a začne vysílat inicializační data.

- `SUM_OF_MEASUREMENTS` - Tento parametr nastavuje počet měření, které bude jednotka odesílat. S tímto údajem je spojena další konfigurace v souboru `template.c`. Musí se zde nastavit správná struktura paketu. Bližší návod k tomuto bodu je popsán v následujícím odstavci.

Další nastavení je potřeba provézt v hlavním souboru projektu a to v `template.c`. Zde musíme nastavit strukturu paketu inicializace a dat.

Ve funkci nazvané `initSendTimerHandler()` pod řádkem definujícím počet odesílaných měření následuje seznam identifikátorů měření. Nastavené měření pro napětí interní baterie a napětí na solárním panelu by pak vypadalo jako na obrázku A.1.

```
static void initSendTimerHandler(SYS_Timer_t *timer)
{
    ...
    packetToSend.command = 30;                                          //command 30: init data
    packetToSend.payload[packetToSend.length] = NUMBER_OF_MEASUREMENTS;  //sum of measurements
    packetToSend.length++;
    packetToSend.payload[packetToSend.length] = DATA_SEND_INTERVAL/1000; //data send interval (timeout)
    packetToSend.length++;
    packetToSend.payload[packetToSend.length] = 1;                      //internal battery voltage ID
    packetToSend.length++;
    packetToSend.payload[packetToSend.length] = 2;                      //solar panel ID
    packetToSend.length++;
    ...
}
```

Obr. A.1: Nastavení inicializačního paketu

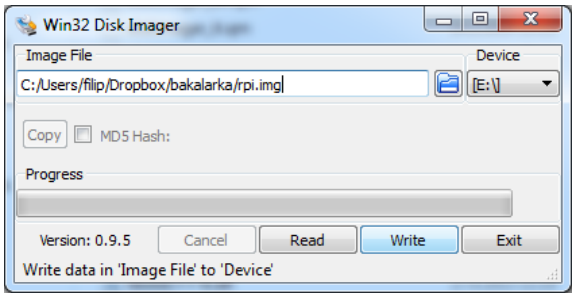
Jako poslední krok je potřeba nastavit samotný paket dat. Pokud se bude využívat AD převodníku, pak je možné využít implementovanou funkci `adcRead()`. Při použití I2C nebo SPI komunikace bude nutné ovladače pro tuto komunikaci implementovat. Samotný paket se pak konfiguruje ve funkci `appMeasuredDataSend()`, kde se nastaví podle definice nejdříve ID daného měření a poté se naplní specifikovaný počet bajtů pro dané ID. Obsah odesílaných dat může mýt až 256 bajtů.

A.1.3 Konfigurace brány

Nejprve je nutné nahrát obraz systému na SD kartu RPi. To se nejsnadněji provede programem Win 32 Disk Imager. Pouze vybereme obraz, který chceme na kartu zapsat, a potvrdíme tlačítkem write jako na obrázku A.2.

Následně se můžeme připojit k RPi vzdáleně přes SSH s loginem `pi` a heslem `raspberry` nebo můžeme použít FTP se stejnými přihlašovacími údaji.

Webové rozhraní je uložené ve složce `/var/www/nette/` a python skript s bash spouštěčem pak ve složce `/home/pi/python/`. Python skript se spouští automaticky po zapnutí nebo restartu RPi. Toto spuštění je definováno v cron tabulce systému



Obr. A.2: Nahrání OS pro RPi na SD kartu

Raspbian. Pokud je potřeba spouštění po startu zakázat, stačí zakomentovat poslední řádku v souboru crontab příkazem v příkazové řádce `sudo nano crontab -e`.

Bezdrátová část sítě může být ponechána v základním nastavení. Stačí do jednotky nahrát program ze složky *gateway* na CD. Pokud by bylo nutné implementovat nové druhy měření, je potřeba upravit Python skript pro RPi (konkrétně upravit funkci `packet_switch`) a upravit odesílací funkci `appMeasuredDataSend()` v koncových jednotkách.