TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Bezdrátová síť pro sběr geofyzikálních dat

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Matěj Řehák**
Vedoucí práce: Ing. Zbyněk Mader, Ph.D.

Liberec 2017

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Wireless network for the collection of geophysical data

Bachelor thesis

Study programme: B2646 – Information technology
Study branch: 1802R007 – Information technology

Author: **Matěj Řehák**
Supervisor: Ing. Zbyněk Mader, Ph.D.

Liberec 2017

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Matěj Řehák**

Osobní číslo: **M13000134**

Studijní program: **B2646 Informační technologie**

Studijní obor: **Informační technologie**

Název tématu: **Bezdrátová síť pro sběr geofyzikálních dat**

Zadávající katedra: **Ústav informačních technologií a elektroniky**

Zásady pro vypracování:

1. Proveďte rešerži bezdrátových prvků, které je možné použít pro vzájemné propojení měřicích uzlů snímačů geofyzikálních veličin, postavených na procesorech Microchip PIC 32MX795.
2. Navrhněte vhodná HW a SW řešení s použitím dostupných technologií, která by dovolila přenášet naměřená data z jednotlivých uzlů do koncového bodu na vzdálenosti stovek metrů až jednotek kilometrů s možností volné konfigurace topologie (sítě typu MESH). Dbejte přitom na specifika aplikace - požadavek na minimální spotřebu, dlouhá perioda vzorkování, specifické podmínky šíření signálu.
3. Vybraná řešení zrealizujte, ověřte v reálných podmínkách a zdokumentujte výstupy, včetně SW knihoven pro procesory Microchip PIC.

Rozsah grafických prací: **Dle potřeby dokumentace**

Rozsah pracovní zprávy: **cca 30-40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **Lucio Di Jasio: Programming 32-bit Microcontrollers in C - Exploring the PIC32, 2008, ISBN: 978-0-7506-8709-6**

[2] **IBRAHIM, Dogan. PIC32 microcontrollers and the digilent chipKIT: introductory to advanced projects. First edition. Kidlington, Oxford: Newnes is an imprint of Elsevier, 2015. ISBN 9780080999340**

Vedoucí bakalářské práce: **Ing. Zbyněk Mader, Ph.D.**
Ústav informačních technologií a elektroniky

Konzultant bakalářské práce: **Ing. Miloš Hernych**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **12. září 2016**

Termín odevzdání bakalářské práce: **15. května 2017**

Technická univerzita v Liberci - 14 -

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

prof. Ing. Ondřej Novák, CSc.
vedoucí ústavu

V Liberci dne 12. září 2016

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 4.5.2017

Podpis: Řehák

Poděkování

Chtěl bych poděkovat především vedoucímu této práce, Ing. Zbyňku Maderovi, Ph.D a konzultantovi práce, Ing. Miloši Hernychovi, za odborné rady. Dále bych rád poděkoval společnosti ČEZ, a. s. za poskytnutí financí ve výši 8 924 Kč, které byly použity na nákup vývojových prostředků, bez nichž by tuto práci nebylo možné dokončit.

Abstrakt

Tato práce řeší problematiku přenosu naměřených dat na vzdálenost stovek metrů až jednotek kilometrů pomocí bezdrátových technologií. Tyto technologie poskytují mnoho výhod oproti technologiím využívající drátové médium. Měření a sběr geofyzikálních dat totiž klade na kabelové médium vysoké požadavky na odolnost vůči vnějším vlivům. Navíc je nutné řešit nežádoucí jevy související s přenosem elektrického signálu po dlouhých vodičích, což ve výsledku zvyšuje pořizovací náklady na sběr dat.

Z úvodní rešerže vzešly dvě vhodné technologie - MiWi od společnosti Microchip Technology Inc. a IQRF od české společnosti Microrisc s. r. o. Obě technologie byly implementovány pro práci s mikrokontrolérem PIC32MX795F512H a podrobeny testům síly signálu v závislosti na vzdálenosti.

Proběhlé testy odhalily, že se závislost síly signálu na vzdálenosti mění v různých typech prostředí. Hlavním přínosem této práce bude možnost centralizovaného sběru a uchovávání naměřených dat v reálném čase.

Klíčová slova: PIC32, MiWi, IQRF, MESH sítě, bezdátové sítě

Abstract

This report solves the issue of wireless data transmission over distances of hundreds of meters to kilometers using wireless technologies. Wireless technologies provide many advantages compared to methods of wired data transfer. Acquisition of geophysical data puts on cabling high requirements for resistance to external influences. Moreover, it is necessary to solve the undesirable effects associated with signal transmission over long conductors resulting in increased initial cost of data collection.

From the initial literature search came two useful technologies - MiWi from Microchip Technology Inc. and IQRF from czech company Microrisc s.r.o. Both technologies have been implemented to work with microcontroller PIC32MX795F512H and subjected to tests of signal strength depending on the distance.

Past tests revealed that the dependence of signal strength on the distance varies in different types of environments. The main contribution of this work will be centralized collection and storage of measured data in real time.

Keywords: PIC32, MiWi, IQRF, MESH networking, wireless networking

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

1 Úvod

Účelem této práce je provést analýzu dostupných bezdrátových technologií a zhodnotit jejich využitelnost pro sběr geofyzikálních dat, vybrané technologie implementovat tak, aby spolupracovaly s mikrokontrolérem PIC32MX795F512H s ohledem na nízkou spotřebu a celou sestavu otestovat v reálných podmínkách.

Práce se mimo jiné zabývá i porovnáním dosahu signálu vybraných technologií v různých prostředích. Porovnání probíhalo ve dvou lokalitách - na volném rovném prostranství a ve vodovodní štole o celkové délce dvou kilometrů v obci Bedřichov.

2 Rešerše dostupných bezdrátových technologií

Vybraná technologie, která bude podrobena detailnějšímu studiu by měla splňovat následující podmínky:

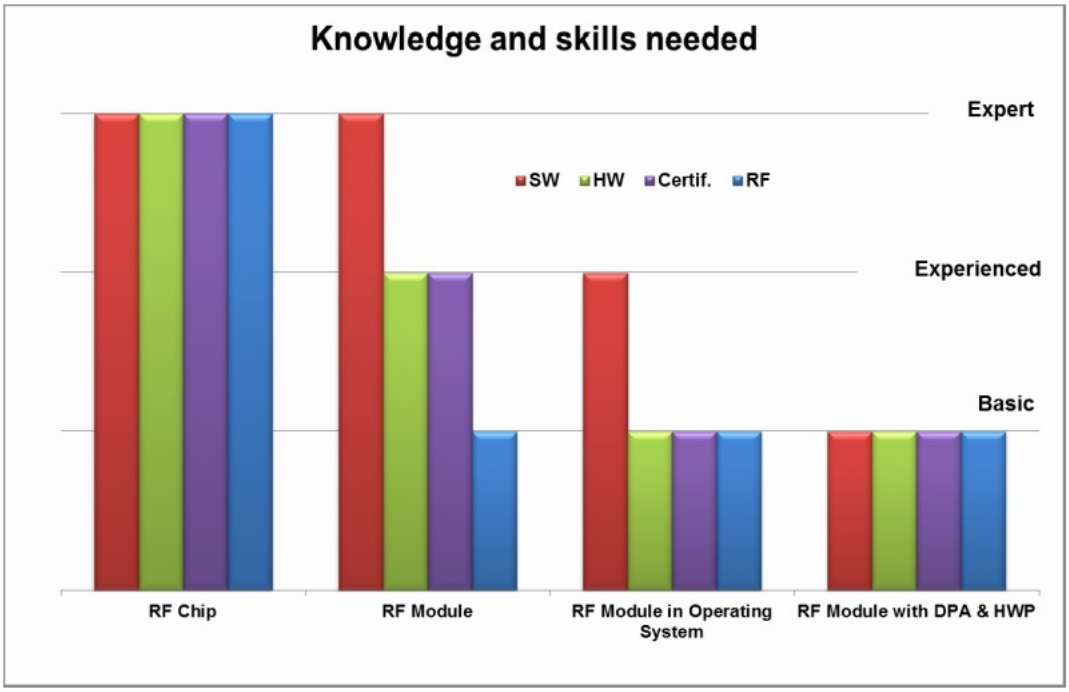
- Vytvoření sítě typu MESH [1]
- Nízká spotřeba (možnost režimu spánku)
- Dosah v řádu stovek metrů až jednotek kilometrů
- Schopnost obnovy sítě po výpadku napájení
- Rychlost implementace
- Dostatečně podrobná dokumentace

Elektroniku potřebnou pro bezdrátový přenos dat lze rozdělit do čtyř skupin:

- RF čip
- RF modul (RF čip a elektronika potřebná pro provoz umístěná na jednom modulu)
- RF modul s operačním systémem (operační systém realizuje logiku sítě)
- RF modul s operačním systémem a integrovanými protokoly pro přenos dat a řízení periferií jiných modulů

Úroveň znalostí v potřebných oblastech vývoje bezdrátové aplikace vystihuje Obr. 2.1, z něhož vyplívá, že nejvhodnější elektronikou pro návrh bezdrátové aplikace bude samostatný RF modul, RF modul s operačním systémem a modul s operačním systémem a integrovaným protokolem pro přenos dat a řízení sběrnic ostatních modulů.

[1] Síť typu mesh je bezdrátová síť, kde je zabezpečena automatická konfigurace struktury sítě, spolehlivé směrování mezi jednotlivými uzly a automatický přístup nových uzlů do sítě prostřednictvím dosavadních uzlů.[1]



Obrázek 2.1: Úroveň potřebných znalostí pro jednotlivé typy elektroniky[2]

2.1 Tinymesh

Technologie Tinymesh distribuována společností Radiocrafts je založena na modulech RC11xx-RC232 s integrovaným operačním systémem a protokolem pro přenos dat. Modul je pomocí hostitelského mikrokontroléru ovládán skrze rozhraní UART. Přenos dat je realizován paketově orientovaným protokolem RC232^{TM}. K dispozici jsou moduly, které pracují na frekvenci 433 MHz, 868 MHz a 2,4 GHz. Samotný operační systém disponuje všemi funkcionalitami pro úspěšné vytvoření a správu mesh sítě včetně režimu spánku.

Hlavní nevýhodou této technologie je poměrně vysoká cena. Vývojová sestava obsahující dva moduly a příslušenství stojí u distributora RS Components 10 902 Kč bez DPH (k 14. 12. 2016). Samotný komunikační modul stojí u stejného distributora 1075 Kč bez DPH (k 14. 12. 2016).

2.2 Texas Instruments Sub-1GHz Low Cost Mesh Network

Toto řešení je postavené na integrovaném obvodu CC1101 od společnosti Texas Instruments. Obvod realizuje pouze přístup k fyzické vrstvě a samotná logika mesh sítě musí být implementována nadřazeným systémem. K hostitelskému mikrokontroléru se připojuje pomocí sběrnice SPI. Obvod je schopen přenášet data ve frekvenčních

pásmech 315/433/868/915 MHz. Výrobce poskytuje k tomuto obvodu softwarové knihovny (tzv. stack), které realizují logiku sítě pomocí nadřazeného mikrokontroléru. Knihovny jsou ale dostupné pouze pro mikrokontrolér MSP430G2533 a vývoj vlastních knihoven pro platformu PIC32MX by byl zdlouhavý a neefektivní.

2.3 LTP5901 SmartMesh

LTP5901 je modul s integrovaným operačním systémem založený na standardu IEEE 802.15.4e. Operační systém zajišťuje všechny funkcionality potřebné k vytvoření a správě mesh sítě, data jsou od nadřazeného systému přenášena skrze rozhraní UART. Všechny uvedené vlastnosti dovolují rychlou implementaci na platformu PIC32MX.

Jedinou nevýhodou je vysoká cena, která u distributora Farnell Element14 začíná na částce 3 506 Kč bez DPH (k 14. 12. 2016).

2.4 ESP8266 - WiFi mikrokontrolér

Mikrokontrolér ESP8266 je založen na architektuře Tensilica Xtensa LX106 a obsahuje integrovaný TCP/IP stack. Samotný TCP/IP stack zajišťuje pouze standardní funkcionality v rámci TCP/IP (mikrokontrolér může být nastaven jako koncové zařízení nebo jako přístupový bod) a logika mesh sítě je zajištěna pomocí programu nahraného do mikrokontroléru. ESP8266 se k nadřazenému systému připojuje pomocí rozhraní UART a lze jej konfigurovat tzv. AT příkazy.

Hlavní výhodou tohoto mikrokontroléru, resp. modulů, je velmi nízká cena, která se pohybuje okolo 50 Kč za jeden kus. Naopak nevýhodou může být skutečnost, že první verze dokumentací byly dostupné pouze v čínském jazyce a je tedy možné, že i budoucí verze dostupné v anglickém jazyce budou k dispozici se zpožděním.

2.5 MiWi - Microchip Wireless

Technologie MiWi je soubor hardwarových a softwarových prostředků, které slouží vývojáři k vytváření bezdrátových sítí typu mesh. Mezi hardwarové prostředky patří bezdrátové transceivery, resp. moduly, MRF24J40, MRF89XA a MRF49XA zajišťující mj. přístup k fyzické vrstvě. Podle zvoleného modulu lze technologii provozovat ve frekvenčních pásmech 433/868/915 MHz a 2,4 GHz. Softwarové prostředky jsou distribuovány ve formě knihoven (tzv. stacku), které jsou nahrány do nadřazeného mikrokontroléru a realizují samotný protokol MiWi.

Výhodou této technologie je zmíněný stack, který je vyvíjen společností Microchip pro mikrokonteroléry PIC, tudíž bude možné jej implementovat na mikrokontroléru uvedeném v zadání. Příznivá je i cena vybraného modulu MRF24J40MA, jehož cena je u distributora Farnell Element14 161 Kč bez DPH (k 14. 12. 2016). Problematická ale může být jeho implementace, která vyžaduje poměrně rozsáhlé zkušenosti v oblasti programování mikrokontrolérů PIC32MX.

2.6 IQRF

Technologie IQRF představuje kompletní platformu zahrnující bezdrátové moduly, vlastní vývojové prostředí, programovací nástroje včetně SDK (Software Development Kit) a připojení do cloudové služby IQRF Cloud. Samotný modul obsahuje mikrokontrolér PIC16LF1938 s integrovaným operačním systémem, transceiver SPIRIT1, konfigurovatelné LED diody, teplotní senzor, paměť EEPROM a napěťový regulátor. Moduly lze konfigurovat pro vysílání ve frekvenčních pásmech 433/868/915 MHz, k hostitelskému mikrokontroléru jej lze připojit skrze rozhraní SPI nebo UART. Zajímavostí je i možnost bezdrátového programování modulů připojených do sítě.

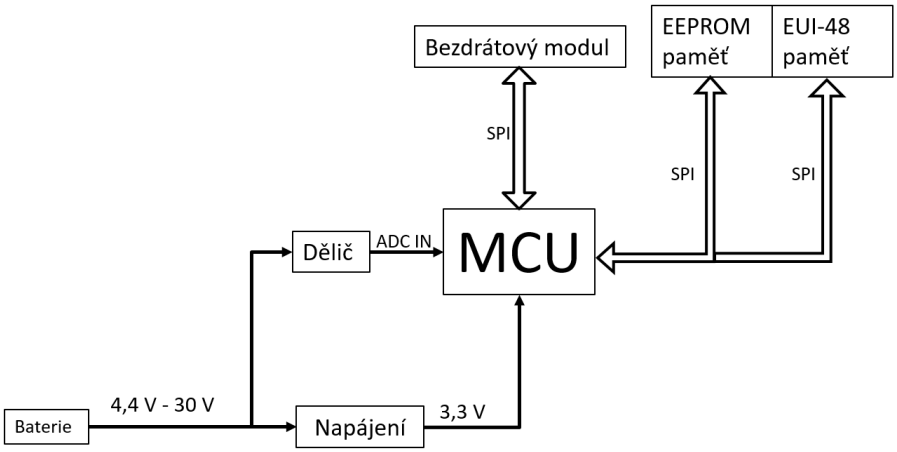
Tato platforma vznikla v roce 2004 jako reakce na tehdejší komplikované a v mnoha případech i zpoplatněné protokoly vyžadující drahý hardware, tudíž se tato technologie vyvíjela tak, aby byla její implementace snadná, rychlá a cenově dostupná. Cena vybraného modulu DCTR-72DA u výrobce a současně distributora, Microrisc s. r. o., činí 579 Kč bez DPH (k 14. 12. 2016).

2.7 Závěrečné zhodnocení

Po společné diskusi s vedoucím a konzultantem práce a zhodnocení požadavků byly vybrány dvě technologie: MiWi, jejíž hardwarové parametry odpovídají kategorii RF modulů dle Obr. 2.1, a IQRF, která dle Obr. 2.1 spadá do kategorie RF modulů s integrovaným operačním systémem, protokolem pro přenos dat a vzdálené řízení periferií ostatních modulů.

3 Elektronika a hardware

Schéma zapojení s mikrokontrolérem PIC32MX795F512H částečně vychází z vývojové desky *Microchip MiWi Demo Kit - 2.4 GHz MRF24J40*[4]. Jelikož obě vybrané platformy používají pro komunikaci s mikrokontrolérem sběrnici SPI (Serial Peripheral Interface), výsledná deska plošných spojů je navržena tak, aby byly moduly obou technologií vyměnitelné. Pro vzájemnou výměnu je použita patice *mikroBUS*^{TM} navržená srbskou společností Mikroelektronika d. o. o. Princip vyměnitelnosti jednotlivých komponent byl aplikován i na napájecí část, mohou tedy být testovány různé typy napěťových stabilizátorů bez nutnosti tvořit kompletně nový plošný spoj. Symbolické znázornění důležitých komponent popisuje Obr. 3.1.



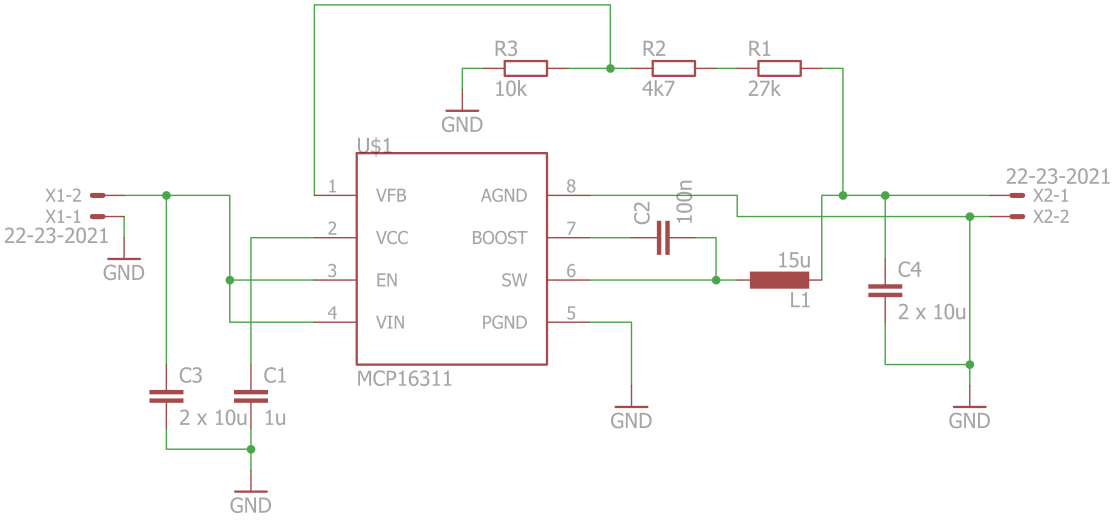
Obrázek 3.1: Symbolické znázornění schématu

3.1 Napájení

Napájecí část musí zajišťovat stabilní napětí 3,3 V, přičemž vstupní napětí se bude v čase měnit. Dalším požadavkem je také vysoká míra energetické účinnosti, která zajistí dlouhou výdrž baterie. Z tohoto důvodu nebude možné použít lineární stabilizátor napětí, proto bude nutné se zaměřit na zapojení s tzv. DC/DC měničem. Prvotní koncept napájení byl směrován k použití co nejmenšího počtu bateriových článků tak, aby vstupní napětí bylo blízké napájecímu napětí systému. Tomuto požadavku nejlépe vyhovoval obvod TPS63031, který je schopen generovat napětí 3,3 V při vstupním napětí v rozsahu od 1,8 V do 5,5 V (jedná se o tzv. *buck-boost*

měnič). Obecná architektura *buck-boost* měniče ale vykazuje poměrně vysoký vlastní odběr proudu, který se v případě obvodu TPS63031 pohybuje kolem 3 mA.

Proto byl zvolen odlišný přístup, který počítá pouze s napětím vyšším než je napájecí napětí systému, použitý typ měniče tedy bude step-down (*buck*), konkrétně obvod MCP16311. Tento měnič je schopen dodávat napětí 3,3V při rozsahu od 4,4 V do 30 V, přičemž disponuje velmi nízkým vlastním odběrem proudu, který se pohybuje v rozmezí od 55 μA do 140 μA.



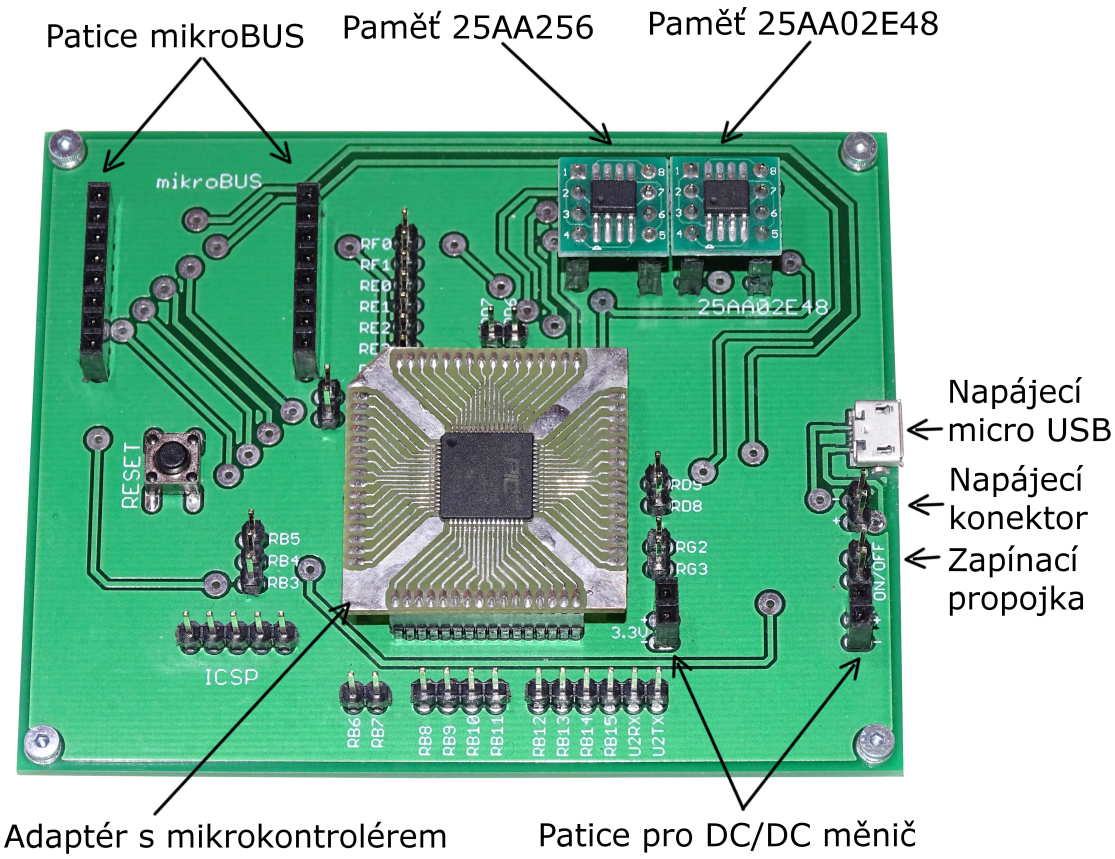
Obrázek 3.2: Schéma zapojení obvodu MCP16311

Zapojení na Obr. 3.2 vychází ze schématu uvedeném v [6]. Výsledný plošný spoj je realizován ve formě vyměnitelného modulu, který je připojen k hlavní desce.

3.2 Hlavní deska

Hlavní deska obsahuje následující komponenty: konektor pro připojení baterie, spínací propojka, patice pro napájecí modul, odporový dělič, mikrokontrolér, paměť 25AA02E48T, paměť 25AA256, patice *mikroBUS*^{TM}, 8 MHz krystal a 32,768 kHz krystal.

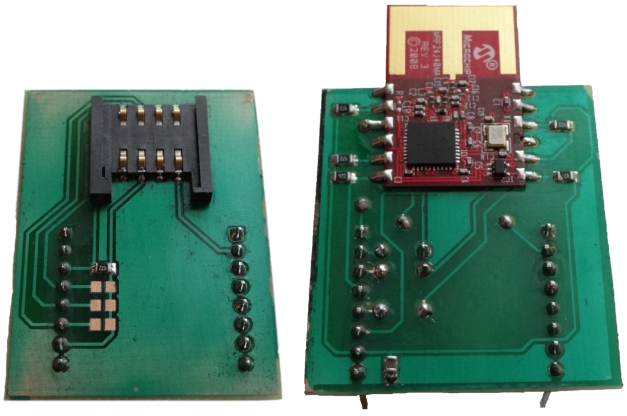
Odporový dělič je připojen na kladný pól baterie a jeho výstup je přiveden na vstup A/D převodníku mikrokontroléru. Jelikož je výše zmíněný integrovaný obvod MCP16311 schopen pracovat do napětí 30 V, je nutné snížit měřené napětí na A/D převodníku tak, aby nedošlo k jeho poškození. Paměť 25AA02E48T obsahuje výrobcem definované unikátní číslo EUI-48 (Extended Unique Identifier), které používá technologie MiWi pro adresaci uzlů v síti. Paměť 25AA256 rovněž slouží technologii MiWi pro ukládání informací o síti, které jsou obnoveny po výpadku napájení. Primární hodinový signál je tvořen krystalem o pracovní frekvenci 8 MHz, sekundární hodinový signál tvoří krystal o pracovní frekvenci 32,768 kHz a je použit pro taktování hodin reálného času v režimu spánku. Schéma zapojení hlavní desky je uvedeno v příloze.



Obrázek 3.3: Hlavní deska

3.3 Adaptéry pro připojení bezdrátových modulů

Oba adaptéry, resp. použité moduly, používají pro komunikaci s mikrokontrolérem sběrnici SPI. Modul IQRF je dle návrhu výrobce připojen pomocí konektoru SIM. Schéma zapojení obou modulů je uvedeno v příloze.

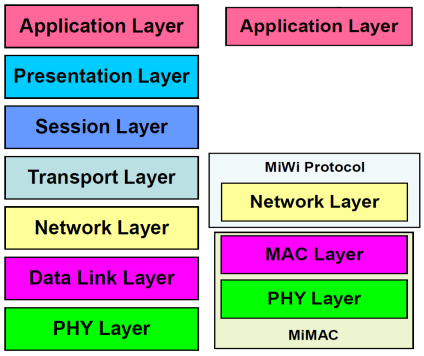


Obrázek 3.4: Adaptéry: vlevo IQRF, vpravo MiWi

4 Technologie MiWi

Technologie MiWi je soubor hardwarových a softwarových prostředků, protokolů a vývojových nástrojů, které se souhrnně nazývají *MiWi*^{TM} *DE* (Development Environment). Mezi hardware lze zařadit příslušné moduly (pro tuto práci byl vybrán typ MRF24J40MA), softwarové prostředky tvoří knihovny, které se souhrně nazývají *stack* a do vývojových nástrojů lze zařadit analyzátor bezdrátově přenášených zpráv *Microchip ZENA Wireless Adapter* a počítačovou aplikaci *Microchip Wireless Development Studio* určenou pro zobrazování zachycených zpráv. Pro přenos zpráv byly vyvinuty celkem tři nezávislé protokoly s následujícími vlastnostmi:

- MiWi P2P
  - Podporuje topologie typu P2P (peer-to-peer) a hvězda
  - Pouze jeden přeskok zprávy, žádné směrování zpráv
- MiWi Mesh
  - Podporuje topologii mesh sítě a směrování zpráv
  - Čtyři přeskoky (hops) zpráv, max. osm koordinátorů v síti, ke každému koordinátorovi je možno připojit až 127 uzlů
- MiWi PRO
  - Rozšíření protokolu MiWi Mesh
  - Podpora až 64 koordinátorů a až 65 přeskoků zprávy



Obrázek 4.1: Porovnání síťových modelů: vlevo OSI, vpravo model MiWi

Jak je psáno v [7], protokol MiWi PRO je funkční pouze v balíku *Microchip Libraries for Applications v2013-06-15*, jehož součástí je i MiWi Stack. Současná verze balíku MLA použitá v této práci je v2016-08-08, tudíž používaným protokolem bude MiWi Mesh.

> *MiWi PRO is only functional in the Legacy MLA (2013-06-15). The Legacy MLA is not supported and is not recommend for development. The current version of the MLA and MCC do not support MiWi PRO.*
>
> *Earlier versions of the MLA contained the "miwi_pro.c"code but this was mistakenly included. None of the MLA demos depended on this file and if the file was included in a project it would fail to compile as it was just copied directly from the Legacy MLA and was not updated for the current versions of the MLA.* [7]

Protokol MiWi je založen na standardu IEEE 802.15.4, konkrétně na fyzické vrstvě (PHY) a vrstvě přístupu k médiu (MAC). Protokol poskytuje funkcionality k vyhledávání, sestavení a připojení k síti, objevování nově připojených uzlů a směrování zpráv. Naopak nepokrývá aplikačně specifické problémy jako je rozhodování, ke které síti se připojit nebo jak často by měly být odesílány zprávy.

Podobně jako standard IEEE 802.15.4, protokol MiWi používá mechanismus potvrzovaných zpráv. Pokud je v odesílané zprávě aktivován příznak pro odeslání potvrzovací zprávy, protistrana musí toto potvrzení odeslat v předem stanoveném intervalu. Pokud potvrzení není včas doručeno, odesílatel znovu odešle původní zprávu.

4.1 Typy zařízení v síti

Standard IEEE 802.15.4 definuje dva typy zařízení:

- Full Function Device (FFD) - zařízení poskytující všechny funkcionality v rámci sítě, které je trvale v aktivním stavu (nepřechází do režimu spánku)
- Reduced Function Device (RFD) - zařízení s omezenými funkcionalitami, typicky koncový uzel napájený z baterie

Protokol MiWi definuje celkem tři typy zařízení. Jejich popis a vztah k typům zařízení dle IEEE 802.15.4 vystihuje Tabulka 4.1.

Typ zařízení	IEEE Typ Zařízení	Funkce
PAN Koordinátor	FFD	Pouze jeden v síti. Sestavuje síť, přiděluje síťové adresy uzlům.
Koordinátor	FFD	Volitelný typ v rámci sítě. Rozšiřuje fyzický dosah sítě. Umožňuje připojování dalších koncových uzlů. Může provádět provádět funkcionality uzlu (např. měřící nebo řídící funkce).
Koncové zařízení	FFD nebo RFD	Provádí pouze měřící nebo řídící funkce.

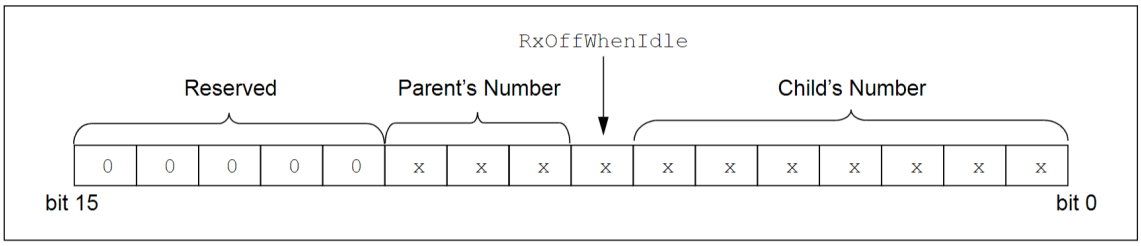
Tabulka 4.1: Typy zařízení v sítích MiWi

4.2 Přidělování adres

Miwi používá tři typy adres:

- Fyzická adresa - tzv. EUI (Extended Organizationally Unique Identifier) adresa je 8 bajtů dlouhá a globálně unikátní
- PANID - identifikátor PAN (Personal area network) sítě, všechny uzly v síti mají stejné PANID, délka 2 bajty
- Zkrácená (logická) adresa - dva bajty dlouhá adresa přidělovaná rodičovským uzlem, unikátní v rámci sítě, používaná k adresaci zpráv

Logická adresa dlouhá 16 bitů je rozdělena na část obsahující 3 bity z adresy rodičovského uzlu, jeden bit adresy (*RxOffWhenIdle*) indikuje, že zařízení vypíná transceiver v režimu spánku. Část vyhrazená pro adresaci uzlu je dlouhá 7 bitů. Uspořádání a význam jednotlivých částí popisuje Obr. 4.2.



Obrázek 4.2: Zkrácená (logická) adresa

Z výše uvedených informací vyplývá, že v síti musí být právě jeden PAN koordinátor, až sedm koordinátorů, přičemž ke každému koordinátorovi může být připojeno až 127 koncových uzlů. Příkladem mohou být následující adresy:

- 0x0000h - tuto adresu má vždy PAN koordinátor sítě

- 0x0300h - adresa koordinátora má vždy část adresy pro koncový uzel a bit *RxOffWhenIdle* nastaveny na hodnotu 0
- 0x0531h - adresa koncového uzlu, jehož transceiver je vždy zapnutý, rodičovský koordinátor má adresu 0x0500h
- 0x07A2h - adresa koncového uzlu, jehož transceiver může být vypnut, rodičovský koordinátor má adresu 0x0700h
- 0xFFFFh - adresa pro všesměrové vysílání (broadcast)

4.3 Datový rámec

Datový rámec protokolu MiWi se skládá ze čtyř částí: hlavičky fyzické vrstvy (PHY), hlavičky vrstvy síťového přístupu (MAC), síťových informací a samotných dat. Hlavička fyzické vrstvy je použita k synchronizaci komunikace v síti (mechanismem unikátního číslování rámců) a přenášení informací o fyzické vrstvě (použitý komunikační kanál, síla signálu, kvalita linky).

Část rámce obsahující MAC hlavičku v sobě nese řídící bajt, který udává informace o přenášeném rámci, např. typ paketu, použití šifrování nebo požadavek na potvrzení doručení. MAC hlavička následně přenáší data (payload), která jsou interpretována vyššími vrstvami. Poslední část, která je přenášena, je kontrolní součet. Strukturu datového rámce popisuje Obr. 4.3.

LAYER	PHY HEADER			MAC HEADER		
NAME	Preamble	SFD	Packet Len	MAC Header	MAC Payload	CRC
BYTE	Various	Various	0 - 1	2 - 21	Various	0 - 2

NAME	Frame Ctrl	Extra Ctrl	Sequence Number	Destination Address	Source Address
BYTE	1	0 - 3	1	0 - 8	0 - 8

NAME	Packet Type	Broadcast	Security	Repeat	Ack	DstPrsnt	SrcPrsnt
BIT	2	1	1	1	1	1	1

Obrázek 4.3: Struktura datového rámce

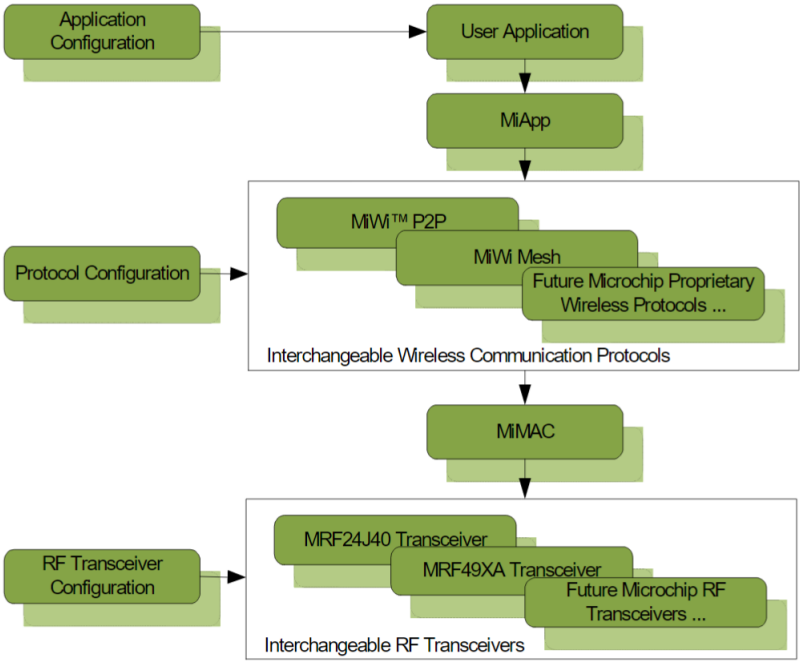
Část rámce obsahující data pro vyšší vrstvy (MAC payload na Obr. 4.3) v sobě nese informace o zdrojové a cílové adrese, PANID a samotná uživatelem přenášená data. Struktura těchto informací je vyobrazena na Obr. 4.4.

Obrázek 4.4: Struktura datového paketu vyobrazeném v prostředí Wireless Development Studio

4.4 Programové rozhraní protokolu MiWi

Programové rozhraní protokolu MiWi (nazývané *MiApp*) se skládá ze dvou částí: konfiguračního souboru a sady funkcí použitých k vývoji. Jak naznačuje Obr. 4.5, programové rozhraní umožňuje vyměnitelnost různých verzí protokolů a transceiverů pouze s minimální nebo žádnou úpravou aplikačního kódu. Celé rozhraní je rozděleno do pěti částí: inicializace, navazování spojení, odesílání zpráv, příjem zpráv a speciální funkce jako je detekce rušení nebo úsporné režimy.

Samotná konfigurace protokolu je realizována pomocí definičních maker uvedených v [9] v Tab. 1.



Obrázek 4.5: Struktura programového rozhraní

Následující souhrn popisuje všechny funkce programového rozhraní *MiApp*.

- `BOOL MiApp_ProtocolInit(BOOL bNetworkFreezer);`
  - Inicializace RF transceiveru a celého stacku. Jediným vstupním parametrem je hodnota typu boolean, která rozhoduje, zda má být načteno předešlé nastavení sítě z paměti EEPROM. Návratová hodnota poté indikuje, zda byl proces inicializace úspěšný.
- `BOOL MiApp_SetChannel(BYTE Channel);`
  - Nastavení komunikačního kanálu, jehož hodnota může být v rozsahu 11 až 26. Návratová hodnota indikuje, zda byla operace úspěšná.
- `BOOL MiApp_StartConnection(BYTE Mode, BYTE ScanDuration, DWORD ChannelMap);`
  - Funkce pro vytvoření PAN sítě. Parametr `Mode` určuje, jestli bude před vytvořením sítě provedena detekce rušení na příslušných kanálech definovaných parametrem `ChannelMap`. Parametr `ScanDuration` určuje dobu skenování kanálů a je dán vztahem $scanTime(\mu s) = 960 * (2^{\text{ScanDuration}} + 1)$.
- `BYTE MiApp_SearchConnection(BYTE ScanDuration,DWORD ChannelMap);`
  - Vyhledává dostupné sítě, resp. koordinátory, výsledky uloží do globálně přístupného pole typu `ACTIVE_SCAN_RESULT` uvedeném ve Zdrojovém kódu 4.1. Parametr `ScanDuration` má stejný význam jako v předcházející funkci,

parametr `ChannelMap` udává, na kterých kanálech bude probíhat vyhledávání. Návratová hodnota indikuje počet nalezených zařízení, ke kterým se lze připojit.

```
typedef struct{
 BYTE Channel;
 BYTE Address[];
 WORD_VAL PANID;
 BYTE RSSI;
 BYTE LQI;
 union
 {
  BYTE Val;
  struct
  {
   BYTE Role: 2;
   BYTE Sleep: 1;
   BYTE SecurityEn: 1;
   BYTE RepeatEn: 1;
   BYTE AllowJoin: 1;
   BYTE Direct: 1;
   BYTE altSrcAddr: 1;
  } bits;
 } Capability
} ACTIVE_SCAN_RESULT;
```

Zdrojový kód 4.1: Struktura ACTIVE_SCAN_RESULT

- `void MiApp_RemoveConnection(BYTE ConnectionIndex);`
  - Provede vymazání spojení z tabulky připojení, jejíž struktura je uvedena ve Zdrojovém kódu 4.2, na pozici udané vstupním parametrem `ConnectionIndex`. Pokud je vstupní hodnota rovna 0xFFh, jsou z tabulky připojení vymazány všechny záznamy.

```
typedef struct __CONNECTION_ENTRY
{
 API_UINT16_UNION PANID;
 API_UINT16_UNION AltAddress;
 uint8_t Address[MY_ADDRESS_LENGTH];
 CONNECTION_STATUS status;
 #if ADDITIONAL_NODE_ID_SIZE > 0
  uint8_t PeerInfo[ADDITIONAL_NODE_ID_SIZE];
 #endif
} CONNECTION_ENTRY;

extern CONNECTION_ENTRY ConnectionTable[CONNECTION_SIZE];
```

Zdrojový kód 4.2: Struktura CONNECTION_ENTRY a pole připojených uzlů

- `BYTE MiApp_EstablishConnection(BYTE ActiveScanIndex, BYTE Mode);`
  - Funkce naváže spojení s jedním nebo více uzly. Návratová hodnota ukazuje na index v tabulce připojení. Pokud je návratová hodnota rovna 0xFFh, spojení se nezdařilo. Parametr `ActiveScanIndex` je index pole typu `ACTIVE_SCAN_RESULT`. Pokud je tato hodnota rovna 0xFFh, pak se funkce pokusí navázat spojení se všemi dostupnými uzly zaznamenané v uvedeném poli. Parametr `Mode` udává způsob připojení. Ten může být buď formou přímého rádiového dosahu nebo v nepřímého.
- `void MiApp_ConnectionMode(BYTE Mode);`
  - Funkce nastaví režim přijímání požadavků o spojení od ostatních uzlů: příjem všech požadavků, příjem požadavků pouze od uzlů, které byly uloženy v paměti EEPROM, příjem pouze skenovacích požadavků a zakázání všech požadavků.
- `void MiApp_FlushTx(void);`
  - Tato funkce resetuje ukazatel na pole, do kterého se nahrávají odesílaná data.
- `void MiApp_WriteData(BYTE OneByteTxData);`
  - Nahraje jeden bajt zprávy daný vstupní hodnotou `OneByteTxData` do pole (bufferu), pomocí kterého se odesílají data. Obvykle je nejprve zavolána funkce `MiApp_FlushTx`, následně `MiApp_WriteData` a poté příslušná funkce pro odeslání zprávy buď ve formě všesměrového vysílání (broadcast) nebo ve formě cíleného vysílání.
- `BOOL MiApp_BroadcastPacket(BOOL SecEn);`
  - Odešle data uložená v transportním poli pomocí všesměrového vysílání. Návratová hodnota indikuje zda byl proces úspěšný. Parametr `SecEn` nastavuje možnost šifrování. odeslaného paketu.
- `BOOL MiApp_UnicastConnection(BYTE ConnectionIndex, BOOL SecEn);`
  - Odešle data uložená v transportním poli, cílový uzel je dán parametrem `ConnectionIndex`, který udává pozici v poli připojených uzlů.
- `BOOL MiApp_UnicastAddress(BYTE *Address,BOOL PermanentAddr, BOOL SecEn);`
  - Odešle data uložená v transportním poli, cílový uzel je dán parametrem `PermanentAddr`, který udává cílovou adresu.
- `BOOL MiApp_MessageAvailable(void);`
  - Funkce vrací hodnotu, která indikuje novou doručenou zprávu připravenou k dalšímu zpracování. Zpráva je uložena v proměnné `rxMessage`, která je typu `RECEIVED_MESSAGE` ve Zdrojovém kódu 4.3.

```
typedef struct
{
 union
 {
  uint8_t Val;
  struct
  {
   uint8_t broadcast:  2;
   uint8_t ackReq:     1;
   uint8_t secEn:      1;
   uint8_t repeat:     1;
   uint8_t command:    1;
   uint8_t srcPrsnt:   1;
   uint8_t altSrcAddr: 1;
  } bits;
 }  flags;

 API_UINT16_UNION SourcePANID;
 uint8_t *SourceAddress;
 uint8_t *Payload;
 uint8_t PayloadSize;
 uint8_t PacketRSSI;
 uint8_t PacketLQI;
} RECEIVED_MESSAGE;
```

Zdrojový kód 4.3: Struktura RECEIVED_MESSAGE

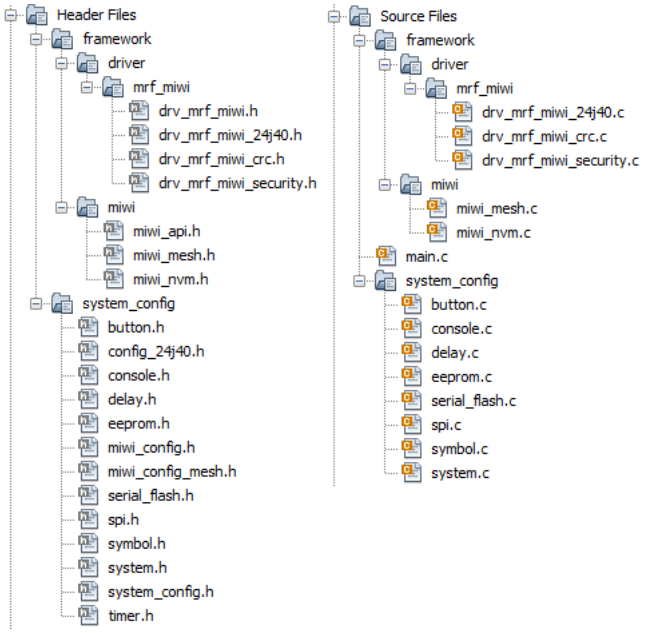
- `void MiApp_DiscardMessage(void);`
  - Zavoláním této funkce dáváme najevo, že příchozí paket byl zpracován a jsme připraveni přijmout další.
- `BYTE MiApp_TransceiverPowerState(BYTE Mode);`
  - Způsobí uspání nebo probuzení transceiveru podle vstupní hodnoty `Mode`. Návratová hodnota indikuje, zda byl proces úspěšný.

5 Vývoj bezdrátové aplikace založené na technologii MiWi

Jak již bylo naznačeno v Tab. 4.1, technologie MiWi rozeznává tři typy zařízení, proto budou vyvinuty tři projekty: PAN koordinátor, koordinátor a koncové zařízení. Vývoj zdrojového kódu probíhal v prostředí MPLAB X IDE v3.45 za pomoci kompilátoru MPLAB XC32 v1.42 a sady knihoven Microchip libraries for Applications (MLA) v2016-08-08 obsahující MiWi stack.

5.1 Obecná struktura projektu v prostředí MPLAB X

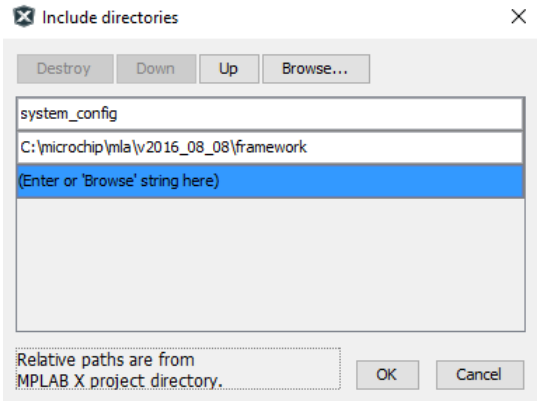
Struktura projektu vychází z příkladové úlohy nacházející se v uvedeném balíku knihoven a je ilustrována na Obr. 5.1.



Obrázek 5.1: Struktura projektu v prostředí MPLAB X[1]

[1]Struktura uvedená na Obr. 5.1 neodpovídá reálné struktuře složek na pevném disku. Tzv. virtuální složky slouží především pro lepší přehlednost projektu.

Zdrojové a hlavičkové soubory MiWi stacku jsou fyzicky uloženy v balíku MLA, který je obvykle instalován do složky `C:\microchip\mla`. Tyto soubory jsou v projektu začleněny do virtuální složky `framework`. Projektově specifické soubory jsou zařazeny do virtuální složky `system_config` a fyzicky uloženy v projektové složce. Dále je nutné ve vlastnostech projektu (Project properties → XC32-gcc → Preprocessing and messages → Include directories) nastavit složky, ze kterých budou načítány hlavičkové soubory.



Obrázek 5.2: Složky, ze kterých budou načítány hlavičkové soubory

Konfigurace konkrétní aplikace a stacku je rozdělena na několik částí. V souboru `system_config.h` jsou především definovány názvy konkrétních pinů, které jsou následně používány celým stackem. Jejich název tedy musí být striktně dodržován. V souboru `system.h` jsou definovány taktovací frekvence příslušných součástí mikrokontroléru. Soubor `miwi_config.h` obsahuje konfiguraci aplikační vrstvy stacku (např. typ použitého protokolu, role uzlu v síti, délka adresy, PAN ID, velikost bufferů, typ použitého komunikačního modulu atd.). Soubor `miwi_config_mesh.h` definuje chování logiky mesh sítě a vrstvy přístupu k médiu (MAC). Jedná se hlavně o časové limity jednotlivých operací. Poslední konfigurační soubor `config_24j40.h` definuje vlastnosti fyzické vrstvy jako je například šifrovací klíč nebo dodatečná nastavení konkrétních modulů. Za zmínku stojí i konfigurace sběrnice SPI, jejíž názvy funkcí musí být také striktně dodržovány neboť jsou používány stackem pro komunikaci s modulem. Důležité je i nastavení časovače. Konkrétně byly použity časovače Timer2 a Timer3, které dohromady tvoří jeden 32bitový časovač. Ten je nastaven dle požadavku standardu IEEE 802.15.4. Tento standard totiž definuje základní časové kvantum (tzv. symbol) o délce 16 μs, což je nastavená perioda inkrementace časovače.

Zdrojové soubory jsou rozděleny stejně jako hlavičkové, tedy na část tvořící stack a na část realizující konkrétní aplikaci včetně podpůrných souborů. Soubor `system.c` obsahuje nastavení konfiguračních bitů mikrokontroléru a funkci, která inicializuje všechny potřebné periferie. Funkce pro inicializaci a čtení hodnoty časovače jsou definovány v souboru `symbol.c`. Hodnota časovače je ukládána do struktury `MIWI_TICK`. Definice funkcí pro používání sběrnice SPI se nacházejí v souboru `spi.c`. V souboru `eeprom.c` jsou definovány funkce pro čtení adresy EUI-48 z paměti

EEPROM. Soubory `delay.c`, `console.c` a `button.c` obsahují pomocné funkce pro generování zpoždění, obsluhu sběrnice UART a tlačítek. Zálohování informací o síti a jejich zpětné načítání zajišťují funkce definované v souboru `miwi_nvm.c`. Logiku sítě, resp. fungování síťové vrstvy, obstarávají funkce definované v souboru `miwi_mesh.c`. Vrstvu přístupu k médiu a fyzickou vrstvu včetně komunikace s modulem zajišťují funkce v souboru `drv_mrf_24j40.c`. Samotná aplikační logika je poté v hlavním souboru `main.c`.

5.2 Úprava zdrojových kódů pro architekturu PIC32MX

Jelikož je MiWi stack distribuován pouze pro osmibitové mikrokontroléry, bylo jedním z úkolů této práce přizpůsobit stack tak, aby mohl fungovat i na architektuře PIC32MX. Hlavní změny byly provedeny v oblasti používání systémových zdrojů mikrokontroléru jako např. úprava časovačů nebo změny v přerušení.

Zatímco osmibitové mikrokontroléry používají systém s jedním vektorem přerušení (v kódu je definována pouze jedna funkce, ve které jsou kontrolovány příslušné přerušovací příznaky), architektura PIC32MX používá více-vektorový systém přerušení. To znamená, že kód pro mikrokontrolér PIC32MX může obsahovat více funkcí obsluhující přerušení, přičemž se zavolá funkce, která obsluhuje daný přerušovací vektor. Navíc používá architektura PIC32MX systém priorit, který obsluhuje přerušení vyvolaná ve stejný okamžik. Výslednou inicializaci přerušení popisuje Zdrojový kód 5.1. Ve zdrojovém kódu `drv_mrf_miwi_24j40.c` je poté nutné nahradit definici přerušovací funkce `void interrupt high_isr (void)` definicí `void __ISR(_EXTERNAL_0_VECTOR, IPL1AUTO) _RxInterrupt(void)`. Velkou pozornost je nutné věnovat nastaveným prioritám. Pokud má přerušení nastavenou jinou prioritu než přerušovací vektor, nedojde k aktivaci přerušení.

```
1  //pouziti vicevektoroveho preruseni
2  INTConfigureSystem(INT_SYSTEM_CONFIG_MULT_VECTOR);
3  //povoleni preruseni
4  INTEnableInterrupts();
5  //vynulovani priznaku preruseni na pinu INT0
6  INTClearFlag(INT_INT0);
7  //povoleni externiho preruseni,
8  //vyvolani preruseni na sestupnou hranu
9  ConfigINT0(FALLING_EDGE_INT | EXT_INT_ENABLE);
10 //nastaveni priority
11 IPC0SET = 0x04000000;
12 //povoleni preruseni
13 INTEnable(INT_INT0, INT_ENABLED);
```

Zdrojový kód 5.1: Inicializace přerušení

Další oblastí, která vyžadovala úpravu, byl časovač. Osmibitové mikrokontroléry disponují pouze 16bitovými časovači a rozšíření na 32 bitů je řešeno softwarovou cestou. U architektury PIC32MX je možné spojit dva 16bitové časovače (v tomto případě Timer 2 a Timer3) a vytvořit tak jeden 32bitový. Dále je nutné nastavit

korektně hodnotu `ONE_SECOND` v souboru `system.h`, jejíž hodnota se řídí vztahem $\frac{1(s)}{perioda\ inkrementace\ casovace(s)}$. V tomto případě je perioda inkrementace časovače 16 μs, výsledné číslo je tedy 62 500. Ve stejném souboru je nutné upravit i převodní makra `SYMBOLS_TO_TICKS(a)` a `TICKS_TO_SYMBOLS(a)`. Pojem *tick* v tomto kontextu znamená jeden takt časovače, zatímco pojem *symbol* znamená již zmíněné časové kvantum o délce 16 μs. Pokud perioda taktu časovače není rovna 16 μs, je nutné uplatnit převodní vztah pro převod těchto jednotek. Pokud je takt časovače roven jednomu symbolu (což je tento případ), pak mohou tato makra mít tvar
`#define SYMBOLS_TO_TICKS(a) a`.

5.3 Aplikační kód - PAN koordinátor

Zdrojový kód koordinátora vychází z doporučené sekvence příkazů uvedené v [9]. Aby uzel zastával roli koordinátora, je nutné v hlavičkovém souboru `miwi_config.h` zavést definici `#define NWK_ROLE_COORDINATOR`.

Nejprve je inicializován veškerý potřebný hardware a poté je načtena EUI-48 adresa. Následně je zahájena inicializace protokolu včetně transceiveru, dále je nastaven komunikační kanál a režim připojení uzlů, zahájen provoz sítě a poté je v nekonečné smyčce kontrolováno, zda byla přijata zpráva. Pokud byla zpráva přijata, je na sběrnici UART vypsána zdrojová adresa zprávy a její obsah. Uvedený Zdrojový kód 5.2 zjednodušeně vystihuje posloupnost příkazů.

```
1  void main(void)
2  {
3   //inicializace desky
4   SYSTEM_Initialize();
5   //nacteni EUI-48 adresy
6   Read_MAC_Address();
7   //inicializace protokolu a transceiveru
8   MiApp_ProtocolInit(false);
9   //nastaveni komunikacniho kanalu
10  MiApp_SetChannel(myChannel);
11  //nastaveni rezimu pripojovani uzlu
12  MiApp_ConnectionMode(ENABLE_ALL_CONN);
13  //vytvoreni site
14  MiApp_StartConnection(START_CONN_DIRECT, 0, 0);
15  while(1)
16  {
17    if(MiApp_MessageAvailable())
18    {
19     int i = 0;
20     CONSOLE_PutString("New message available");
21     sprintf((char *)buf2,
22     (char*)"Source address:<Addr:%02x%02x>",
23     rxMessage.SourceAddress[1],
24     rxMessage.SourceAddress[0]);
```

```
25      CONSOLE_PutString(buf2);
26      for(i = 0; i < rxMessage.PayloadSize; i++)
27      {
28       buf[i] = rxMessage.Payload[i];
29      }
30      CONSOLE_PutString(buf);
31      MiApp_DiscardMessage();
32     }
33  }
34
35 }
```

Zdrojový kód 5.2: Zdrojový kód představující roli PAN koordinátora

5.4 Aplikační kód - koordinátor (router)

Zdrojový kód koordinátora (neplést s PAN koordinátorem) je v určitých ohledech stejný, zvláště v inicializaci a konfiguraci, která probíhá stejně jako u koordinátora. Hlavní rozdíl spočívá v zahájení vyhledávání okolních uzlů a následném připojení ke koordinátorům, které mají hodnotu síly signálu vyšší než 50. Část programu routeru je uvedena ve Zdrojovém kódu 5.3.

```
1  void main(void)
2  {
3   .
4   .
5   inicializace
6   .
7   .
8
9   uint8_t scanResult = MiApp_SearchConnection(10,
10  MiWi_CHANNEL);
11  for(i = 0; i < scanResult; i++)
12  {
13   if(ActiveScanResults[i].RSSIValue > 50)
14   {
15    if (MiApp_EstablishConnection(i,
16    CONN_MODE_INDIRECT) != 0xFF)
17    {
18     ...
19    }
20    else if (MiApp_EstablishConnection(i,
21    CONN_MODE_DIRECT) != 0xFF)
22    {
23     ...
24    }
25    else
```

```
26      {
27       sprintf(buf, "FAILED to connect to 0x%02x%02x",
28       ActiveScanResults[i].Address[1],
29       ActiveScanResults[i].Address[0]);
30       CONSOLE_PutString(buf);
31      }
32     }
33    }
34
35    while (1)
36    {
37     if (MiApp_MessageAvailable())
38     {
39      MiApp_DiscardMessage();
40     }
41    }
42   }
```

Zdrojový kód 5.3: Zdrojový kód představující roli koordinátora (routeru)

5.5 Aplikační kód - koncový uzel

Koncový uzel se od koordinátorů liší především v konfiguraci. V souboru `miwi_config.h` by měla být zavedena definice `#define NWK_ROLE_END_DEVICE` a `#define ENABLE_SLEEP`. V aplikačním kódu je opět standardní inicializace až na nastavení připojování koncových uzlů, která je zakázaná. Zkrácený program je uveden ve Zdrojovém kódu 5.4

```
1   void main(void)
2   {
3    .
4    .
5    inicializace
6    .
7    .
8    MiApp_ConnectionMode(DISABLE_ALL_CONN);
9    uint8_t scanResult = MiApp_SearchConnection(10,
10   MiWi_CHANNEL);
11   for(i = 0; i < scanResult; i++)
12   {
13    if(ActiveScanResults[i].RSSIValue > 50)
14    {
15     if (MiApp_EstablishConnection(i,
16     CONN_MODE_INDIRECT) != 0xFF)
17     {
18      ...
19     }
20     else if (MiApp_EstablishConnection(i,
```

```
21      CONN_MODE_DIRECT) != 0xFF)
22      {
23        ...
24      }
25      else
26      {
27        sprintf(buf, "FAILED to connect to 0x%02x%02x",
28        ActiveScanResults[i].Address[1],
29        ActiveScanResults[i].Address[0]);
30        CONSOLE_PutString(buf);
31      }
32    }
33  }
34
35  while(1)
36  {
37    ADCSample();
38    MiApp_FlushTx();
39    MiApp_WriteData(ADCData[1]);
40    MiApp_WriteData(ADCData[0]);
41    MiApp_UnicastAddress(0x0000, false); //adresa koordinatora
42    ADCOff();
43    MiApp_TransceiverPowerState(POWER_STATE_SLEEP);
44    initRTCC();
45    PowerSaveSleep(); //zde se program zastavi
46    MiApp_TransceiverPowerState(POWER_STATE_WAKEUP);
47    ADCOn();
48  }
49 }
```

Zdrojový kód 5.4: Zdrojový kód představující roli koncového uzlu

6 Technologie IQRF

Technologie IQRF představuje kompletní platformu, která zahrnuje široké portfolio bezdrátových transceiverů (v této práci bude použit modul DCTR-72DA), USB programátor a debugger, vlastní vývojové prostředí a SDK (Software Development Kit) umožňující vytvářet počítačové programy v jazyce Java a C++.

Moduly DCTR lze řídit dvěma způsoby:

- Pomocí protokolu DPA (Direct Peripheral Access) - Protokol DPA lze aktivovat nahráním rozšiřujícího kódu, který v závislosti na verzi vytvoří z modulu buď koordinátora sítě nebo uzel, který má ovšem možnost směrovat zprávy ostatním uzlům. Koordinátor sítě poté může ovládat periferie ostatních modulů pouze pomocí definovaného datového protokolu přenášeného po sběrnici SPI. Případnou úpravu funkcionalit lze řešit pomocí tzv. *Custom DPA Handleru*.

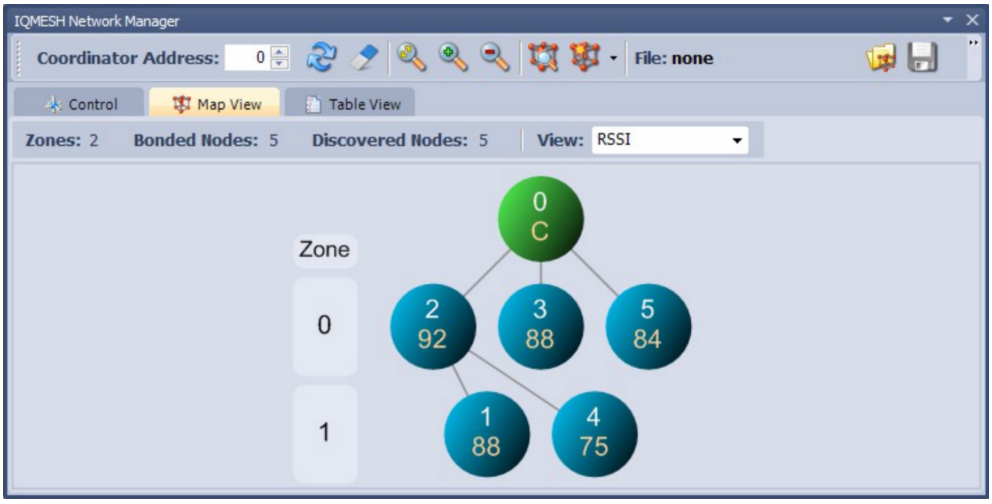
- Programováním v jazyce C - Chování modulu lze upravovat pouhým programem bez implementace protokolu DPA. K tomuto účelu disponuje operační systém IQRF OS sadou funkcí popsanou v [10].

Chceme-li použít protokol DPA, všechny uzly musí být připojeny do sítě a mít nahraný příslušný hardwarový profil. Připojení uzlu do sítě se v terminologii IQRF nazývá *bonding*. Tento proces lze aktivovat buď pomocí vývojového prostředí nebo protokolem DPA. Po úspěšném připojení všech modulů je vhodné tyto moduly rozmístit na příslušné místo a poté provést tzv. *node discovery*, což je v podstatě proces optimalizace sítě. Tento proces, stejně jako ostatní funkce, lze spustit buď pomocí vývojového prostředí nebo zasláním DPA paketu přes rozhraní SPI. Výsledná síťová topologie může být zobrazena v prostředí IQRF IDE tak jak je ilustrováno na Obr. 6.2.

Obrázek 6.1: Struktura DPA paketu

Jak naznačuje Obr. 6.1, DPA paket se skládá z adresy uzlu (2 bajty), identifikátoru periferie (1 bajt), identifikátoru příkazu (1 bajt), identifikátoru hardwarového profilu (2 bajty) a volitelně i dat. Úplný výčet všech periferií, příkazů a jejich použití je uvedeno v [11]. V režimu DPA je maximální počet uzlů omezen na 240,

přičemž adresa 0xFFh je vyhrazena pro broadcast, adresa 0x00h je vyhrazena pro koordinátora sítě a adresa 0xFCh je vyhrazena pro lokální rozhraní (tzv. loopback).



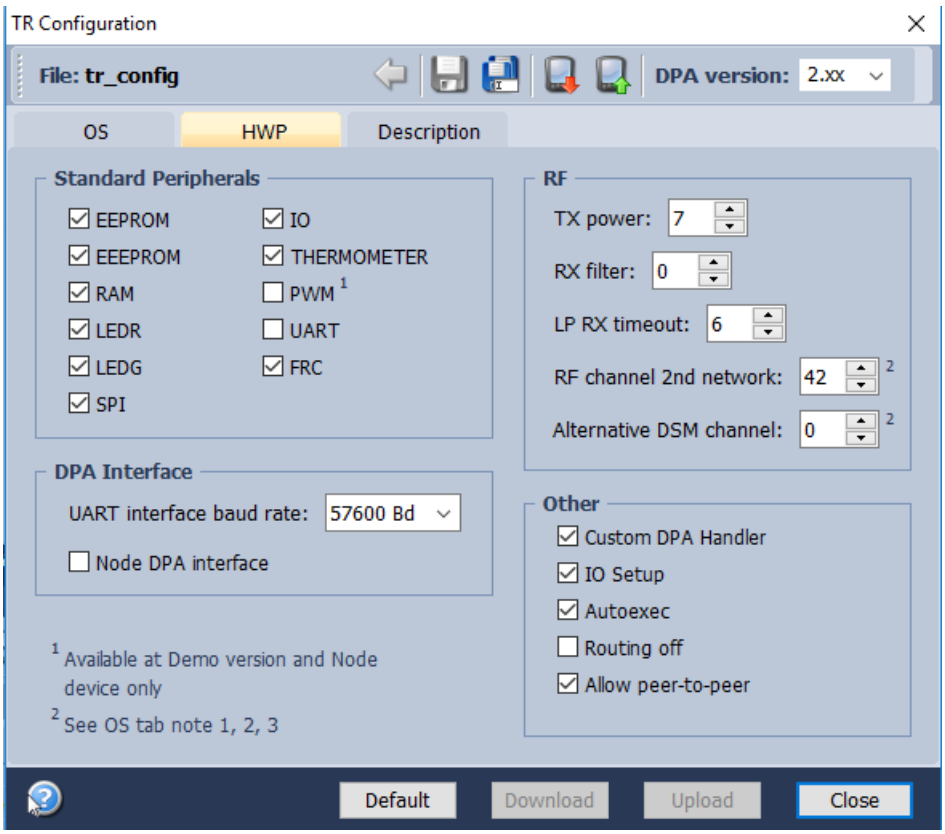
Obrázek 6.2: Síťová topologie vytvořená po procesu *discovery*

7 Vývoj bezdrátové aplikace založené na technologii IQRF

Bezdrátová aplikace založená na technologii IQRF se skládá ze dvou částí: vývoje programu pro bezdrátový modul a vývoje programu pro mikrokontrolér.

7.1 Programování modulu IQRF

Po vytvoření projektu je zapotřebí nahrát do jednoho modulu hardwarový profil koordinátora a do zbylých modulů nahrát profil uzlu. Následně je nutné provést konfiguraci všech modulů, tj. nastavení frekvence, komunikačního kanálu a především nastavení hardwarového profilu podle Obr. 7.1.



Obrázek 7.1: Nastavení hardwarového profilu

Důležitá je v tomto ohledu položka *Custom DPA Handler*, jejíž povolení nám umožní částečně modifikovat chování transceiveru v určitých situacích. Tato konfigurace je následně nahrána do všech transceiverů. Dalším krokem pro úspěšný přenos dat je vytvoření zdrojového souboru, který bude zachytávat události související s protokolem DPA. Jedná se například o události vyvolané před odesláním paketu, před uspáním modulu atd. Zdrojový soubor ve formě předlohy obsahující všechny tyto události je dostupný v balíku IQRF Startup Package. V tomto souboru byly modifikovány dvě události. První je událost `DpaEvent_ReceiveDpaRequest`, která je zavolána po přijetí paketu a pokud paket obsahuje příkaz k uspání transceiveru je část paketu nesoucí informaci době spánku zkopírována na sběrnici SPI spolu se třemi znaky "SLP", které mikrokontrolér vyhodnotí jako příkaz k uspání. Ukázka této modifikace je ve Zdrojovém kódu 7.1.

```
case DpaEvent_ReceiveDpaRequest:
// Called after DPA request was received
if(_PNUM == PNUM_OS && _PCMD == CMD_OS_SLEEP)
{
 memoryLimit = 2;
 copyBufferRF2COM();
 bufferCOM[2] = 'S';
 bufferCOM[3] = 'L';
 bufferCOM[4] = 'P';
 startSPI(5);
 memoryLimit = 0;
}
break;
```

Zdrojový kód 7.1: Událost zavolaná po přijetí DPA požadavku

Druhou modifikovanou událostí je `DpaEvent_BeforeSendingDpaResponse`. Tato událost je zavolána před odesláním DPA odpovědi (response). V této konkrétní modifikaci uvedené ve Zdrojovém kódu 7.2 transceiver čeká na dva bajty (hodnota načtená z A/D převodníku) odeslané mikrokontrolérem, které začlení do DPA odpovědi.

```
case DpaEvent_BeforeSendingDpaResponse:
// Called before sending DPA response back to originator
//of DPA response

if(_PNUM == PNUM_SPI)
{
 enableSPI(); // Master is allowed to transmit from now
 Receive:
 clrwdt();
 if (getStatusSPI()) // Wait until SPI is not busy
 goto Receive;
 if (_SPIRX) // Anything received?
 { // Yes:
  // BufferCOM is automatically protected now
```

```
15    // not to be overwritten by next SPI packet.
16    // Thus, stopSPI is not necessary here.
17    // Packet length is in SPIpacketLength.
18    copyBufferCOM2INFO(); // Store received packet
19    startSPI(0); // and then allow Master to transmit again.
20   }
21   else
22    goto Receive;
23
24   _DpaDataLength = _DpaDataLength + 2;
25   FSR0 = _DpaMessage.Response.PData + _DpaDataLength - 2;
26   setINDF0( bufferINFO[0]);
27   FSR0 = _DpaMessage.Response.PData + _DpaDataLength - 1;
28   setINDF0( bufferINFO[1]);
29  }
30  break;
```

Zdrojový kód 7.2: Událost zavolaná před odesláním DPA odpovědi

Po zkompilování a nahrání zdrojového souboru budou uzly schopny odpovídat na datové požadavky koordinátora a interpretovat příkaz k uspání uzlu a mikrokontroléru současně.

7.2 Programování mikrokontroléru PIC32MX

Pro mikrokontrolér PIC32MX byly vytvořeny dva programy. První program realizuje koordinátora sítě. Mikrokontrolér nejprve provede inicializaci potřebných sběrnic, poté může volitelně provést objevování uzlů a optimalizaci jejich cest (*discovery*). Následně mikrokontrolér odešle jednomu vybranému uzlu požadavek na stav baterie a cca každých 10 ms odesílá dotaz (`SPI_CHECK`), který má hodnotu 0x00h, transceiver současně vysílá jednu z hodnot uvedených v Tab. 7.2. Podrobný popis způsobu přenášení zpráv pomocí sběrnice SPI je uveden v [12] na str. 5. Jakmile je odpověď doručena, stav baterie daného uzlu je odeslán na sběrnici UART. Tento proces se opakuje pro všechny uzly. Pokud uzel neodpoví do pěti sekund, je vynechán. Po ukončení sběru dat mikrokontrolér odešle broadcast obsahující příkaz k uspání na stanovenou dobu. Kompletní zdrojový kód je uložen na DVD příloze ve složce IQRF.

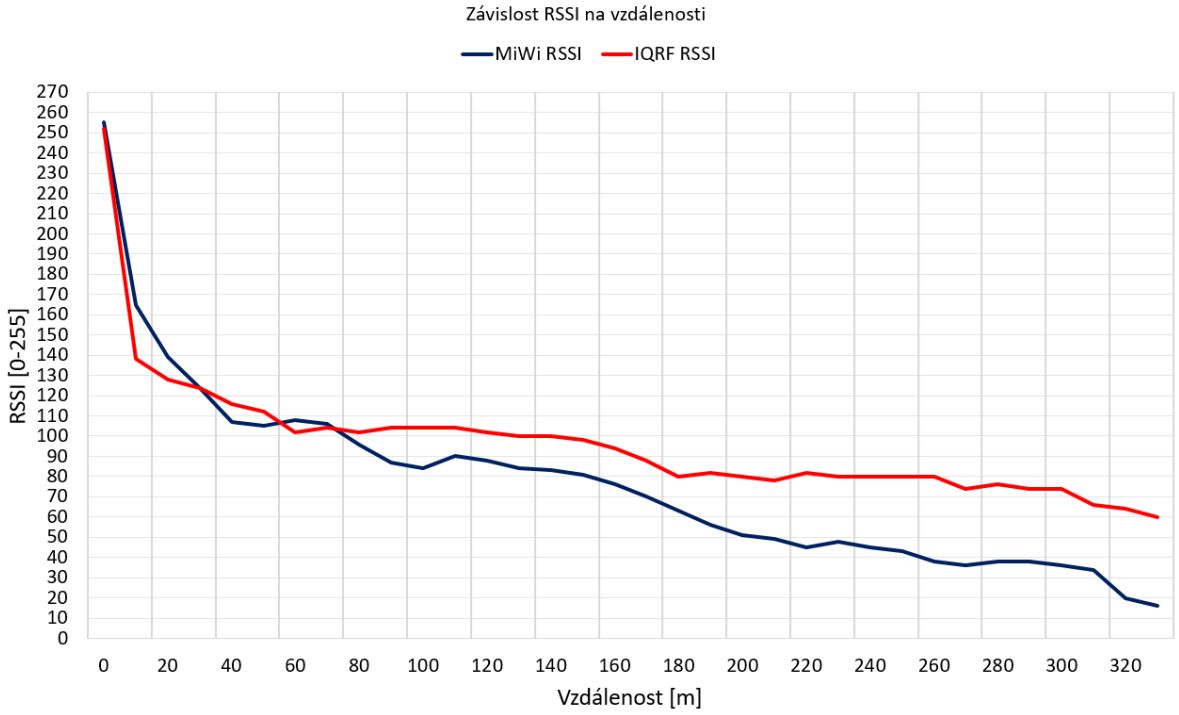
Druhý program realizuje uzel, který provádí měření napětí na baterii. Program vysílá každých cca 10 ms příkaz (`SPI_CHECK`). Pokud jsou načtena nová data, je provedena jejich analýza a obsahují-li posloupnost znaků GD (Get Data), je spuštěna konverze AD převodníku a načtená hodnota je ve formě dvou bajtů odeslána do transceiveru. Pokud přijatá data obsahují posloupnost znaků SLP (sleep), je ze zprávy načtena hodnota doby spánku, která je přepočítána tak, aby bylo možné jí nastavit alarm hodin reálného času, který vyvolá přerušení a probudí tak mikrokontrolér a transceiver z režimu spánku. Kompletní zdrojový kód je uložen na DVD příloze ve složce IQRF.

Hodnota hex	**Stav SPI**
00	SPI není aktivní
07	SPI je pozastavena
3F	SPI není připravena (plný buffer, CRC součet v pořádku)
3E	SPI není připravena (plný buffer, CRC součet není v pořádku
40 až 40 + N_{max}	SPI je připravena, v bufferu jsou data, délka dat = tato hodnota - 0x40
80	SPI je připravena - komunikační režim
81	SPI je připravena - programovací režim
82	SPI je připravena - režim ladění
FF	SPI není aktivní (chyba v hardwaru)

Tabulka 7.1: Možné odpovědi na dotaz `SPI_CHECK`, $N_{max} = 56$

8 Testování v reálných podmínkách

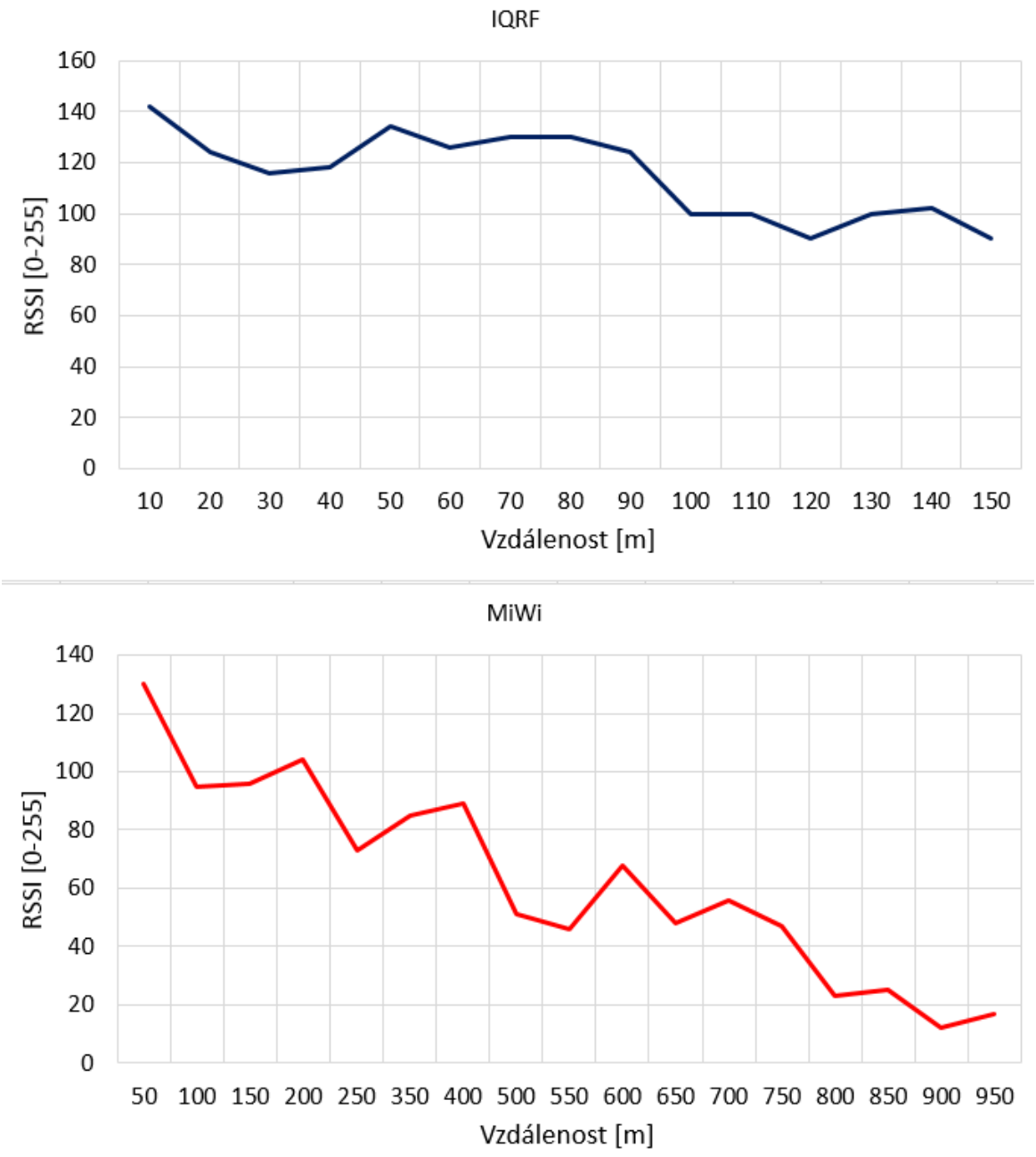
Testování síly signálu probíhalo na dvou lokalitách: na volném rovinatém prostranství a ve vodovodní štole v obci Bedřichov. Transceivery byly po celou dobu měření v přímém dohledu ve výšce cca 150 cm nad zemí.



Obrázek 8.1: Závislost síly signálu na vzdálenosti ve volném prostranství

Jak naznačuje Obr. 8.1, technologie IQRF ve vzdálenosti 330 m disponuje sílou signálu kolem hodnoty 60, což je pro stabilní spojení dostatečné, zatímco technologie MiWi má v této vzdálenosti sílu signálu pod hodnotou 20, což pro stabilní spojení není dostatečné.

Naproti tomu měření síly signálu ve vodovodní štole prokázalo zcela opačné výsledky. Technologie IQRF, ač měla signál dostatečně silný, nedokázala stabilně komunikovat ani na vzdálenost 150 m, technologie MiWi naopak prokázala schopnost stabilně komunikovat až do vzdálenosti 750 m, kde se hodnota síly signálu ustálila na hodnotě 50, což je pro stabilní spojení dostatečné.



Obrázek 8.2: Závislost síly signálu na vzdálenosti ve vodovodní štole v obci Bedřichov

Důvodem této anomálie může být existence tzv. Fresnelových zón. První fresnelova zóna je oblast, kterou se šíří zhruba 90 % energie signálu a má tvar elipsoidu. Její průměr lze spočítat vztahem $r_1[m] = \sqrt{\frac{\lambda d_1 d_2}{D}}$, přičemž λ [m] je vlnová délka signálu, d_1 [m] a d_2 [m] jsou vzdálenosti od překážky (pokud jsou si d_1 a d_2 rovny a současně se rovnají polovině D, pak se jedná o nejširší místo zóny) a D [m] je vzdálenost vysílačů. Pomocí uvedeného vztahu lze spočítat, že průměr r_1 při vzdálenosti 700 m a frekvenci 2,4 GHz je 4,67 m, zatímco při frekvenci 868 MHz je hodnota r_1 7,78 m.

9 Závěr

Byla provedena rešerše, ze které vzešly dvě technologie: MiWi a IQRF. Obě technologie prokázaly možnost vytvoření sítě typu mesh. Liší se především ve způsobu implementace. Jak ukazuje Obr. 2.1, pro kategorii RF modulů je třeba expertních programátorských znalostí a zkušeností, což se u protokolu MiWi potvrdilo. Je nutné podrobně znát a pochopit všechny součásti tohoto protokolu. Tyto vlastnosti v konečném důsledku prodlužují dobu vývoje.

Naopak pomocí technologie IQRF je možné postavit mesh síť během jednoho dne, což dokazuje technologickou vyspělost této platformy. Nicméně fyzikální možnosti nedovolují provoz této technologie v prostředí vodovodní štoly, proto z praktického hlediska vítězí technologie MiWi.

U platformy MiWi je třeba snížit odběr proudu transceiveru v režimu spánku. Současná hodnota odběru činí 1,1 mA, což pro systém napájený z baterie není přijatelné. Dalšího snížení odběru lze docílit připojením tranzistoru na napájecí vodič transceiveru a provádět jeho úplné vypnutí v režimu spánku. Po zapnutí stačí provést reinicializaci modulu a celý systém může pokračovat ve své funkci.

Spotřeba obou technologií v režimu spánku, včetně vlastního odběru DC/DC měniče, nepřesahuje hodnotu 200 μA. V aktivním režimu spotřeba nepřevšuje hodnotu 40 mA.

Literatura

[1] HYNČICA, Ondřej (ed.). Bezdrátové sítě typu mesh. Časopis Automa [online]. 2005 [cit. 2016-12-13]. Dostupné z: http://automa.cz/cz/casopis-clanky/bezdratove-site-typu-mesh-2005_12_30826_1141/

[2] IQRF - bezdrátová technologie, která láme bariéry. In: Pandatron.cz - Elektrotechnický magazín [online]. 2013 [cit. 2016-12-13]. Dostupné z: https://pandatron.cz/?3673&iqrf_-_bezdratova_technologie,_ktera_lame_bariery

[3] Malé, chytré, české... Bezdrátová platforma IQRF. In: Vývoj.HW.cz — Vše o elektronice a programování [online]. 2010 [cit. 2016-12-17]. Dostupné z: http://vyvoj.hw.cz/rf/male-chytre-ceske-bezdratova-platforma-iqrf.html

[4] MICROCHIP TECHNOLOGY INC. MiWi™ Demo KitUser's Guide [online]. 2012 [cit. 2016-12-17]. ISBN 978-1-62076-575-3. DS70687A. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/70687A.pdf

[5] Sub-1GHz Low Cost Mesh Network. Texas Instruments [online]. [cit. 2016-12-14]. Dostupné z: http://www.ti.com/tool/TIDM-SUB1GHZ-MESH-NETWORK

[6] MCP16311/2 30V Input, 1A Output, High-Efficiency,Integrated Synchronous Switch Step-Down Regulator [online]. 2013 [cit. 2016-12-18]. ISBN 978-1-63276-806-3. SLVS696C. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/20005255B.pdf

[7] Support for MiWi Pro. In: Microchip Support Community [online]. 2016 [cit. 2016-12-20]. Dostupné z: https://www.microchip.com/support/

[8] FLOWERS, David a Yifeng YANG. Microchip MiWi™ Wireless Networking Protocol Stack [online]. 2010 [cit. 2016-12-20]. ISBN 978-160932-346-2. AN1066. Dostupné z: http://ww1.microchip.com/downloads/en/AppNotes/AN1066%20-%20MiWi%20App%20Note.pdf

[9] YANG, Yifeng. Microchip Wireless (MiWi™) Application Programming Interface – MiApp [online]. 2009 [cit. 2016-12-21]. AN1284. Dostupné z: http://ww1.microchip.com/downloads/en/AppNotes/01284A.pdf

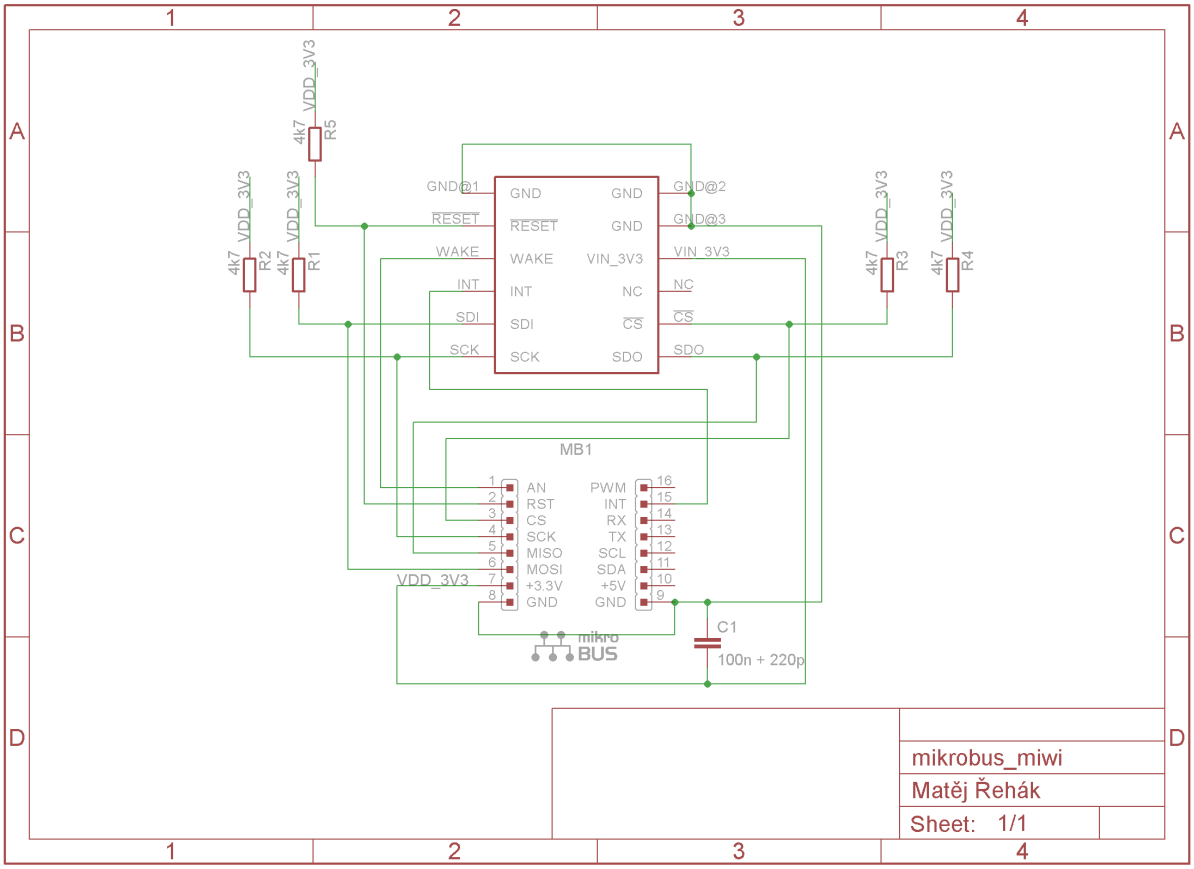
[10] IQRF OS Operating System: Version 3.08D for (DC)TR-7xD Reference Guide [online]. 2016 [cit. 2016-12-23]. Ref_Guide_IQRF-OS-308D_TR-7xD_160921. Dostupné z: http://www.iqrf.org/weben/downloads.php?id=156

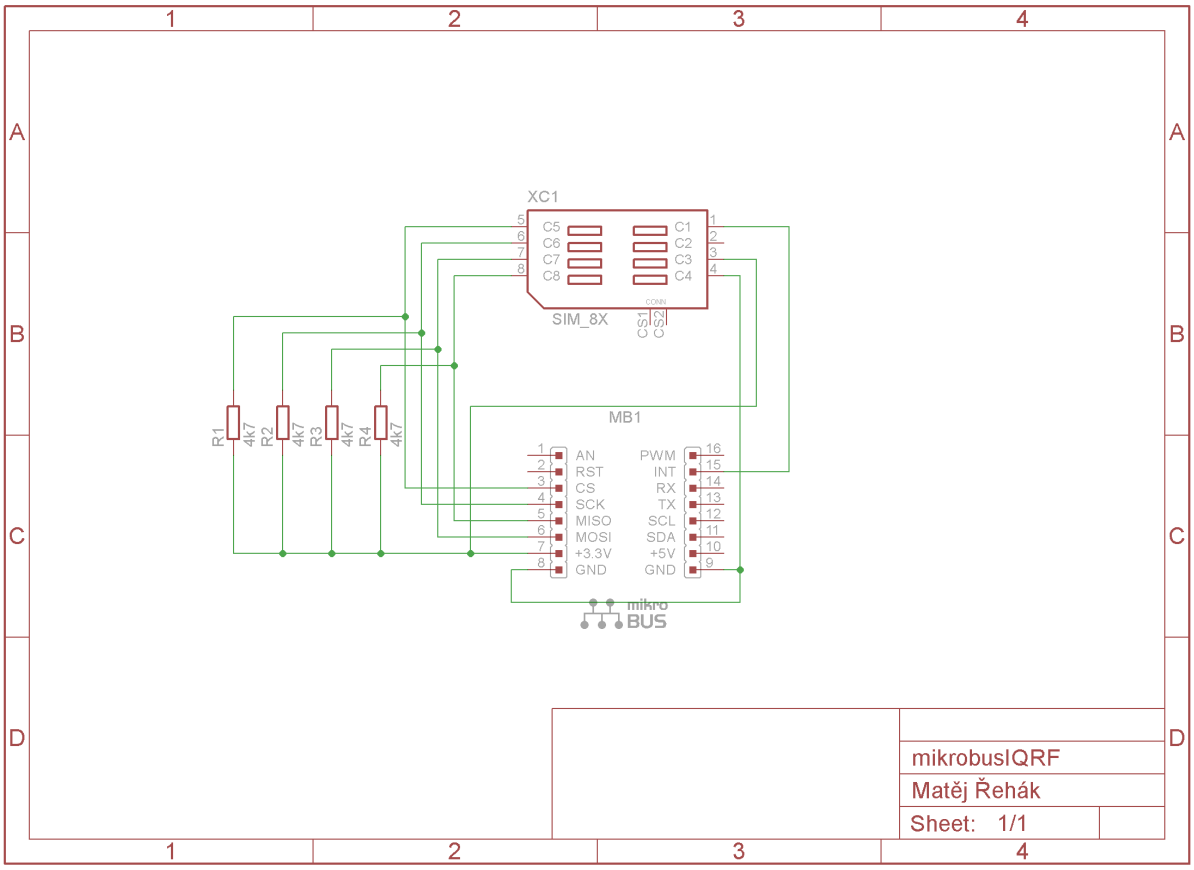
[11] IQRF DPA Framework Technical Guide: Version v2.28 IQRF OS v3.08D [online]. 2016 [cit. 2016-12-23]. Dostupné z: http://www.iqrf.org/weben/downloads.php?id=481

[12] SPI Implementation in IQRF For (DC)TR-7xD Technical guide [online]. 2015 [cit. 2016-12-23]. Dostupné z: http://www.iqrf.org/weben/downloads.php?id=85

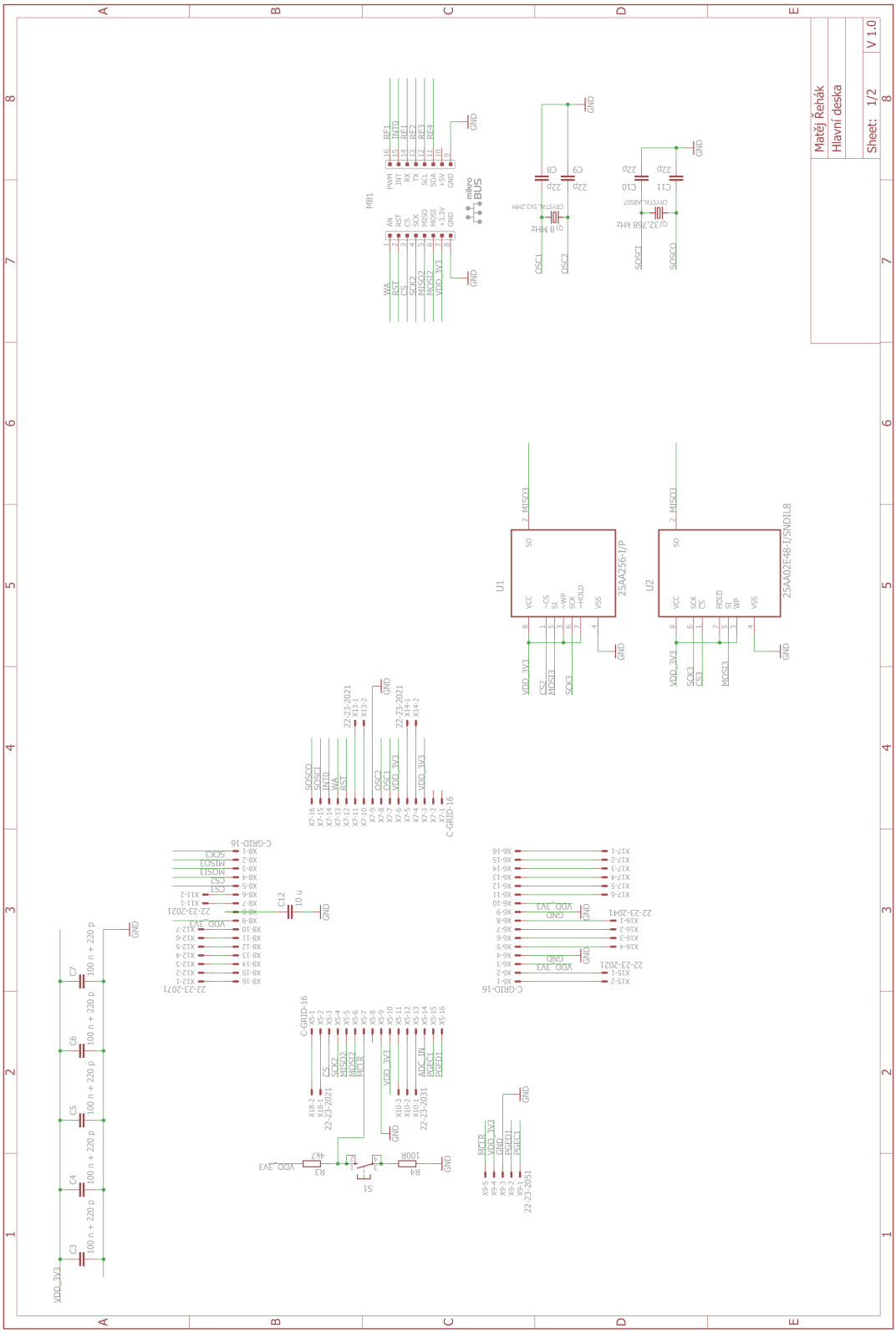
Přílohy

Příloha 1 - Schémata adaptérů bezdrátových modulů




1
2
3
4
A
B
C
D
XC1
C5
C6
C7
C8
C1
C2
C3
C4
SIM_8X
CS1
CS2
R1
4k7
R2
4k7
R3
4k7
R4
4k7
MB1
AN
RST
CS
SCK
MISO
MOSI
+3.3V
GND
PWM
INT
RX
TX
SCL
SDA
+5V
GND
mikro
BUS
mikrobusIQRF
Matěj Řehák
Sheet: 1/1

Příloha 2 - Schéma hlavní desky



Příloha 3 - DVD

DVD obsahuje následující složky:

- MIWI - obsauje všechny projekty pro MPLAB X
- IQRF - obsahuje všechny projekty pro MPLAB X a IQRF IDE
- HARDWARE - obsahuje schémata navržená v prostředí Eagle 7.2.0
- zprava - obsahuje tento dokument ve formátu .PDF a .TEX včetně obrázků