VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

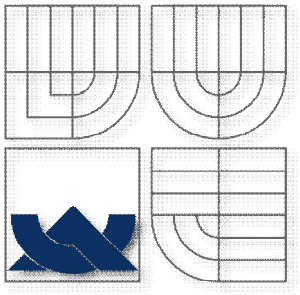
OVLADAČ A HARDWAROVÝ MODUL PROTOKOLU MIWI PRO LINUX

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

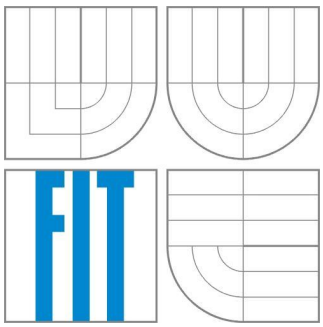
AUTOR PRÁCE
AUTHOR

BC. MARTIN HALA

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

OVLADAČ A HARDWAROVÝ MODUL PROTOKOLU MIWI PRO LINUX

DRIVER AND HARDWARE MODULE OF MIWI PROTOCOL FOR LINUX

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. MARTIN HALA
AUTHOR

VEDOUCÍ PRÁCE — ING. TOMÁŠ NOVOTNÝ
SUPERVISOR

BRNO 2014

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Tomáše Novotného. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

…….
Martin Hala
28. května 2014

Poděkování

Rád bych poděkoval vedoucímu diplomové práce Ing. Tomáši Novotnému za věcné připomínky a rady při zpracování diplomové práce včetně pomoci s návrhem a výrobou desky plošného spoje.

Abstrakt

Diplomová práce se zabývá návrhem a implementací hardwarového modulu, který slouží jako komunikační prvek mezi vestavěným zařízením s Linuxem a senzorovými prvky. Komunikace probíhá prostřednictvím bezdrátového protokolu MiWi. Zadání je součástí projektu IoT, který je vyvíjen na FIT VUT v Brně. Dále práce popisuje návrh ovladače, možná řešení a samotnou implementaci. Závěrem je shrnutí dosažených poznatků z implementace a návrh, jak při tvorbě ovladače pokračovat.

Klíčová slova

MiWi, Linux, ovladač, SPI, Spidev, Microchip, IoT, Raspberry Pi

Abstract

The master's thesis is about a communication element - a hardware module, its design and implementation. The communication is to be maintained between a Linux embedded device and the sensors elements, using the MiWi protocol. The task is part of the IoT project, developed at FIT BUT. Furthermore, the paper describes design of a driver for the module, its likely solution, as well as the very implementation. Finally, the obtained experience is discussed in a summary, along the next step options on how to proceed further with the driver development.

Keywords

MiWi, Linux, driver, SPI, Spidev, Microchip, IoT, Raspberry Pi

Obsah

Kapitola 1

Úvod

Rozvoj moderních informačních technologií má v současné době dopad na téměř jakékoliv odvětví v běžném životě. Mezi nově vzniklé pojmy patří mimo jiné i spojení inteligentní dům.

Integrací inteligence do rodinného domu či kanceláře je možné efektivně spravovat energetické náklady domu. Tento systém se zároveň dokáže zcela automaticky adaptovat na změny svého okolí. Příkladem mohou být prvky jako automatická regulace teploty a vytápění, spínání světel, ovládání spotřebičů, žaluzií, vzduchotechniky, zabezpečení a tak dále.

Moderní, nově stavěné kanceláře jsou již většinou vybaveny množstvím senzorů a čidel. U běžných rodinných domů se s pojmem inteligentní dům setkáváme spíše u novostaveb, a to velmi zřídka. Výhodou tohoto odvětví je neustále vzrůstající poptávka bez ohledu na ekonomickou krizi.

Během výstavby domu je sice možné integrovat kabeláž pro všechny senzory v budově, toto řešení je ale nákladnější jak na realizaci, tak z finančního hlediska. Případná úprava či vylepšení systému již není zcela triviální záležitostí. Nejvhodnější možností je proto připojit čidla do systému pomocí úsporné technologie bezdrátových sítí. Aby něco takového mohlo vzniknout, musely bezdrátové sítě projít dlouhou evolucí.

Běžným příkladem je soukromá domácí WiFi síť, kterou lze nalézt v mnoha domácnostech. Jenomže ta není díky své charakteristice zcela vhodná na komunikaci se senzorovými prvky. Problematice bezdrátových sítí a výběru vhodného řešení se věnuje třetí kapitola.

Následující body shrnují jednotlivé kapitoly této práce:

- Inteligentní domácnost, informace o historii a současnosti. Přehled komerčních řešení v porovnání s projektem IOT vyvíjený na FIT VUT v Brně.
- Popis a kategorizace jednotlivých bezdrátových technologií, jednoduchý popis a porovnání jejich vlastností vůči požadavkům jako důležité a nepodstatné vlastnosti. Porovnání nejvhodnějších technologií, která splňují důležitá kritéria.

- Popis vybraného protokolu MiWi, obecné informace. Informace o protokolu, topologii, směrování a použití protokolu v praxi.
- Obecné seznámení s operačním systémem Linux a jádrem operačního systému Linux. Rozdělení ovladačů dle typu a souhrn nejdůležitějších vlastností.
- Návrh řešení jak z pohledu SW (program pro HW modul, ovladač pro Raspberry Pi), tak z pohledu HW (tvorba a realizace HW modulu).
- Implementace navrženého řešení, opět jak z pohledu softwaru, tak hardwaru.
- Doporučení změn v návrhu a nástin možné cesty, která vede k funkčnímu řešení.

Kapitola 2

Inteligentní domácnost

Definice pojmu inteligentní domácnost lze formulovat jako [1]: Dům, který nazýváme inteligentním, umí maximalizovat uživatelský komfort optimalizováním vnitřního prostředí a zároveň minimalizovat náklady na provoz. Měl by být zároveň adaptivní na změny okolních vlivů a potřeb, což vyplývá obecně z definice pojmu inteligence. Kromě uživatelského komfortu má přinést i zvýšení bezpečnosti objektu, která by měla být složena z více prvků. Přestože je celý objekt vybaven více systémy různých funkcí, tyto jednotlivé prvky by měly být vzájemně kompatibilní a schopné vzájemné komunikace a centrálního řízení.

Výhodou napojení celého systému do sítě internet na základě myšlenky IoT (viz. Kapitola 2.2), je možnost ovládat a kontrolovat aktuální stav objektu odkudkoliv na světě, kde je dostupné internetové připojení. Tento druh inovace zcela mění jak pohled na moderní dům, tak běžně zažité zvyky v bydlení.

Pojem Inteligentní dům lze tak rozdělit do tří základních kategorií: řízení, bezpečnost, zábava.

- **Řízení** – Systém vyhodnocuje svůj stav pomocí senzorů, reaguje na aktuální podmínky pomocí aktorů. Senzory mohou měřit např. teplotu, vlhkost, intenzitu světla, pohyb po domě a aktory reagovat na tyto vstupy spínáním světel, ovládáním topení, ovládáním žaluzií, otevíráním / zavíráním vrat a tak podobně.

- **Bezpečnost** – Zabezpečovací systém, který je sestaven ze senzorů specificky určených pro bezpečnost. Mezi takové patří například detektory pohybu, kouře, vibrační detektory, magnetické kontakty, detektory rozbití skla, bezpečnostní kamery. Propojením této kategorie do zbytku inteligentního domu vzniká velké bezpečnostní riziko, které by mohlo být zneužito potencionálním narušitelem.

- **Zábava** – Kategorie, která se může zdát jako nejméně důležitá, avšak skrývající obrovský potenciál k možnostem rozšíření. Komfort lze zvýšit jednoduchým ovládáním za použití intuitivních gest, které lze zadávat do chytrého mobilního zařízení, jako je tablet. Toto je ale pouze obecný popis celé kategorie. Možnosti jsou nesčetné, záleží jen na lidské představivosti. Lze například ovládat svou chytrou televizi, přenášet obrazová, hudební data mezi jednotlivými přehrávači po celém domě a při spuštění hudby určitého žánru automaticky nastavit osvětlení, barvu a jas světel atp.

2.1 Aktuální stav

Jedna z mála jistot, na které se můžeme v současnosti spolehnout, je fakt, že náklady za energie budou nadále stoupat. A proto snižování nákladů na provoz budovy je jeden z klíčových faktorů, které vedou k rozhodnutí vybudovat inteligentní řízení domácnosti. Paradoxně ekonomická krize zapříčinila rozmach tohoto oboru, za cílem snížení a zefektivnění provozu domácnosti. Stejně tak rozmach chytrých mobilních zařízení, typu telefon a tablet, otevřel nové možnosti v komfortu ovládání celého systému. A i když je aktuálně v České republice stále nízké procento staveb vybavených inteligentní domácností, jiné evropské státy dokazují obrovský potenciál k růstu.

Poptávka po chytrém bydlení neustále stoupá a úměrně s ní i počet firem, které svá řešení nabízejí. Aktuálně je na trhu již tak obrovské množství existujících řešení, že vybrat si to nejvhodnější je velmi obtížné. Uvedu zde alespoň základní popis několika nejvýznamnějších řešení, s kterými se lze na českému trhu v současnosti setkat. Podrobněji popíšu jeden z nich, který se podobá systému vyvíjeným na FIT VUT v Brně.

2.1.1 Existující komerční řešení

Jak jsem se již zmínil, na trhu lze nalézt nepřeberné množství komerčních řešení. Hlavní rozdíl mezi jednotlivými typy je v použitých technologiích, možnostech konfigurace, správy a hlavně výsledné ceny. Ta mnohdy hraje hlavní roli při výběru vhodného řešení.

V této podkapitole se nachází několik vybraných komerčních produktů s krátkým popisem a jeden z nich popsaný podrobněji.

- **Haidy** [http://www.haidy.cz/] – Hlavními přednostmi tohoto řešení je jednoduché ovládání a úprava funkcí inteligentních senzorů přes ovládací prvky. Správa systému HAIDY je možná pomocí chytrých telefonů, tabletů, PC i chytré TV. Typické řešení inteligentní domácnosti o rozloze 200 metrů čtverečních vyjde v průměru na 150 000 Kč + instalace.
- **Belkin WeMo** [www.belkin.com/WeMo] - WeMo je celý balík produktů, speciálně určených pro IoT, vytvořený firmou Belkin. V nabídce jsou zajímavá řešení, která lákají své zákazníky hlavně nízkou cenou. Sortiment produktů není zatím nijak rozsáhlý, firma Belkin ovšem slibuje rozšíření svého portfolia produktů. Dle nejaktuálnějších informací z veletrhu spotřební elektroniky CES 2014, který se každoročně koná v Las Vegas, firma Belkin rozšiřuje svoje produkty například o inteligentní kávovar či vařič. Dálkově ovládaná zásuvka s pohybovým senzorem, který vyvolá nastavenou reakci na pohyb, vychází na 80 dolarů.
- **Jablotron** [www.jablotron.cz]– česká firma s dlouholetou tradicí se na trhu se zabezpečovací a komunikační technikou pohybuje již delší dobu. Díky široké nabídce zařízení lze vytvořit inteligentní domácnost (zde nazvanou jako komfort systém) přesně na míru zákazníka. Jejich řešení existuje jak v drátové, tak v bezdrátové podobě. Vzdálené ovládání je ovšem řešeno pomocí GSM sítí (topení lze např. regulovat SMS zprávou), což není zrovna nejvhodnější řešení z pohledu IoT. Za nevýhodu je možné považovat také vyšší cenu kompletního řešení.
- **Obzor RF Home** [www.obzor.cz] – Mezi největší výhody systému RF Home od firmy Obzor je využití bezdrátové technologie pro ovládání aktorů. Ovládat lze všechny běžné elektrické části domu. Protože je toto řešení zajímavé a zčásti se podobá projektu vyvíjenému na FIT VUT v Brně, rozhodl jsem se ho popsat podrobněji.

2.1.2 Obzor RF Home

Firma Obzor [2] vidí hlavní výhodu svého řešení právě v bezdrátové komunikaci, čímž odpadá nutnost stavebních úprav při budování systému. Systém je tak vhodný pro integraci do již dokončeného objektu. Ovládání probíhá pomocí bezdrátových ovladačů, které stačí pouze napájet z baterie. Bezdrátový přenos zajistí ovládání příslušného aktoru. Další možností je ovládat jednotlivé prvky pomocí bezdrátové klíčenky.

Dle výrobce lze ovládat různé typy osvětlení, ventilace, domovních i garážových bran

a žaluzií. Komunikace mezi ovladačem a aktorem probíhá na frekvenci 868 MHz, čímž se elegantně vyhýbá zarušené bezlicenční frekvenci 2,4 GHz (používající mnohá zařízení typu WiFi a podobně). Dosah signálu je až 160 metrů a lze si vybrat z 8 kanálů. Tím ovšem funkcionalita celého systému končí.

I když je systém na první pohled podobný srovnávanému řešení, ve skutečnosti jde o mnohem jednodušší funkcionalitu a neprobíhá zde žádný sběr dat, která by byla odesílána na server. Komunikace mezi vysílačem a přijímačem je jednosměrná, nedochází k žádnému sběru dat ze senzorů, čímž celý systém nijak nereaguje na jakýkoliv podmět. Přijímače disponují pouze dvěma režimy. Jedná se tedy o klasický spínač nebo stmívač.

V podstatě se ani nedá mluvit o systému jako o inteligentní domácnosti. Řešení pouze přidává možnost bezdrátového ovládání klasických vypínačů. Dle ceníku vyjde kombinace 1 kanálového bezdrátového vypínače se spínacím aktorem na 1000 Kč, což může být pro zákazníky, kteří nevyhledávají složité řešení, jako dostačující. Důležité je také zmínit, že aktuální ceny jsou dlouhodobě zlevněny o 40 % z původní ceny.

Pokud bych měl celé řešení od firmy Obzor shrnout, tak výsledný systém nelze srovnávat s ostatními řešeními, které pracují na principu IoT. Hlavní nevýhodou je jednoduchá funkcionalita, a proto nelze popsané řešení porovnávat s propracovanějšími systémy, které disponují možnostmi správy a ovládáním přes internet.

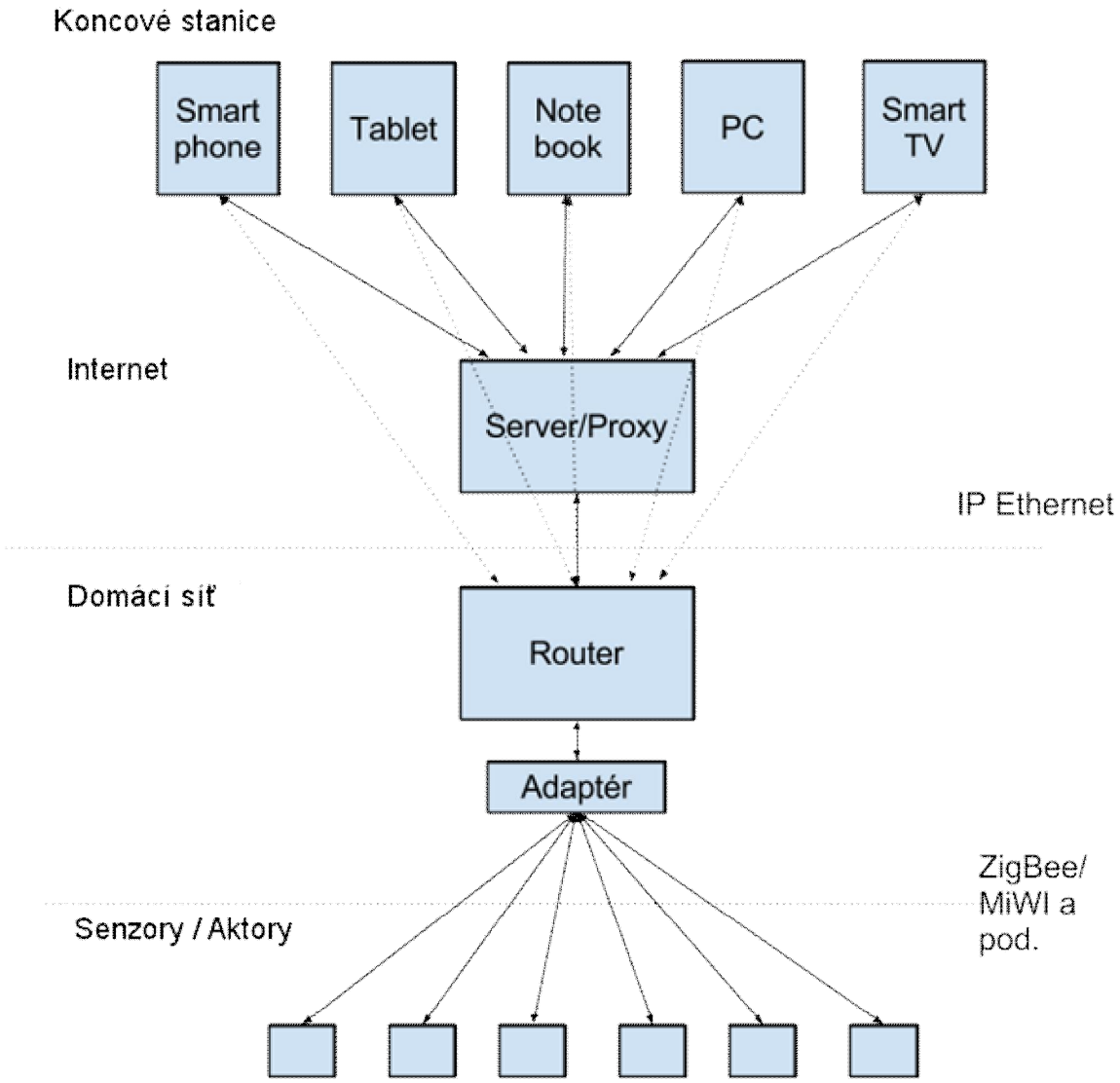
2.2 IoT na FIT VUT v Brně

Definice Internet of Things [3], (IoT) popisuje jednotlivé objekty jako jednoznačně identifikovatelné a jejich virtuální reprezentace je součástí struktury sítě typu internet.

Alespoň takto byl pojem IoT vnímán v roce 1999, v době kdy tento pojem vznikl. Dnes se spíše jedná o sadu autonomních zařízení v rámci internetu, které jsou jednoznačně identifikovatelné. Tyto samostatné uzly sbírají data ze senzorů a nahrávají je na internetové úložiště, ať už to je cloud, pronajatý server, či jakékoliv z internetu dostupné úložiště.

Mezi hlavní výhody IoT na FIT VUT v Brně patří nízké pořizovací náklady a využití opensource řešení pro některé části projektu, z důvodu zjednodušení vývoje celého systému a tím snížení celkových nákladů.

2.2.1 Architektura



Ilustrace 1: Architektura IoT na FIT VUT v Brně

Na obrázku je návrh architektury pro IoT systém vyvíjený na FIT VUT v Brně [4]. Rozdělení je provedeno do tří částí: senzorová část (na ilustraci dole), zpracování dat v domácí síti a část, kde jsou data nahrávána na internetové úložiště. Odsud dochází ke komunikaci ovládacích prvků se serverem.

- **Senzorová část** – Na nejnižší úrovni této architektury dochází ke komunikaci mezi senzory a adaptérem. Ten má na starosti udržování topologie a spojení mezi senzory. Senzorová část je cílem této diplomové práce – komunikace mezi senzory a hardwarovým modulem. Senzory nejsou vlastní výroby, ale využívají se řešení třetích stran. Co se týče nákladů,

lepší je si vybrat již existující a ověřené řešení. Ke komunikaci byl vybrán protokol Zigbee/MiWi – bližší popis výběru vhodného protokolu lze nalézt v následujících kapitolách. Jeden z hlavních důvodů je ale odolnost vůči rušení od běžných domácích spotřebičů jako jsou mikrovlnná trouba, domácí WiFi router a jiná, využívající pro svou komunikaci některé z bezlicenčních pásem (nejčastěji na 2,4 GHz). Senzory musí být jednoduše spárovatelné a komunikace s adaptérem by měla probíhat přes zabezpečený kanál. To vše s co nejmenší energetickou náročností.

- **Adaptérová část** – jeho hlavní funkcí je správa bezdrátových čidel, zajištění zabezpečené komunikace, a to v plně-duplexním režimu. Samotný adaptér je připojený na hraniční router sítě, který se stará o komunikaci mezi adaptérem a serverem. Možností, jak takový adaptér připojit do sítě, je několik. Prvním z nich je integrace přímo do HW a SW routeru. Zmíněné řešení je dosti náročné na realizaci, a to jak z důvodů HW změn, tak zásahem do OS routeru, což není ani jednoduché a ne vždy z licenčních důvodů možné. Druhou možností je propojit adaptér pomocí IEEE 802.3/11 tzn. buď využít propojení přes LAN nebo pomocí WLAN. Zde je nutné adaptér napájet zvlášť nebo pomocí externího PoE (u běžných routerů není PoE přímo integrováno pro napájení připojených zařízení). Nejvhodnějším řešením se jeví připojení adaptéru přes USB rozhraní (běžné routery jsou dnes téměř standardně vybaveny tímto rozhraním). Tím by komunikace a napájení byly elegantně vyřešeny. Adaptér by měl být ovšem schopný fungovat i když není router k dispozici (například regulace teploty topení by měla fungovat neustále). Router by měl fungovat pouze jako prostředník mezi serverem a adaptérem. Pokud by se pro přístup k administraci inteligentního domu využívalo přímo připojení na router, znamenalo by to řešení několika problémů. Router by musel být přístupný z vnější sítě přes staticky přidělenou veřejnou IP adresu. Dalším problémem je nemožnost přístupu na vnější veřejnou IP adresu ve chvíli, kdy by chytrý ovládací prvek (tablet, chytrý telefon atp.) byl připojen do vnitřní sítě. Tyto problémy se dají vyřešit buď proxy serverem, který by směřoval provoz na router i bez veřejné IP adresy (router by tak musel fungovat jako server s databází), a nebo přeposíláním dat z routeru na server. Ten by byl dostupný odkudkoliv s kompletním zpracováním dat. V tom případě by router sloužil tak, jak bylo zamýšleno a to pouze jako prostředník.
- **Internetová část** – Část, kde je server přístupný, a to jak na straně routeru tak pro ovládací příkazy, přicházející z chytrých zařízení. Router je zaregistrován v serveru a komunikuje s ním na přímo, tzn. odesílá a přijímá požadavky dle potřeby. Uvnitř routeru dochází ke zpracování dat, ukládání do databáze, výpočtu různých předpovědí, administrace a jiné.

Ovládací prvky se připojí na server přes vlastní aplikaci a ta se podle nakonfigurovaného profilu spojí s daty a vytvoří náhled pro konkrétní inteligentní domácnost. Nevýhodou se může jevit možné bezpečnostní riziko. Důvod je ten, že jsou veškerá data odesílána do internetu. V této části musí být velmi dobře zpracovaná autentizace uživatele, šifrování přenosu, využití certifikátů pro připojení při zachování co nejvyšší míry zabezpečení. Určité riziko systému může zapříčinit nedostupnost serveru. Proto by takový systém měl být schopen ochránit sám sebe a automaticky ovládat kritické inteligentní prvky i bez přímé komunikace se serverem.

Kapitola 3

Bezdrátové technologie

Tato kapitola popisuje jednotlivé bezdrátové technologie z pohledu senzorových sítí. U každé kategorie je souhrn výhod a nevýhod, které se odrážejí od podstatných a nepodstatných vlastností, jež jsou popsány v kapitole 3.1. Ty jsou posuzovány právě z pohledu ideálního protokolu pro komunikaci mezi adaptérem a senzory.

3.1 Vlastnosti ideální bezdrátové sítě

Ještě před výběrem vhodné bezdrátové technologie, která bude použita pro komunikaci mezi adaptérem a senzory, by bylo dobré vědět, které vlastnosti jsou pro výběr podstatné a které naopak nejsou až tak důležité.

3.1.1 Podstatné vlastnosti

- Nízký odběr energie, senzory jsou napájeny bateriemi (co možná nejdelší interval mezi výměnou).
- Vysoká úroveň zabezpečení komunikace.
- Jednoduché příkazy v rámci komunikace.
- Spolehlivost přenosu, odolnost vůči rušení od domácích spotřebičů.
- Možnost uspání senzoru během klidového stavu.

3.1.2 Nepodstatné vlastnosti

- Přenosová rychlost, přenášený objem dat mezi rádiem a senzorem je malý.

- Vzdálenost senzoru od rádia. Kategorie WPAN je zcela pro tyto účely dostačující.
- Přenosové pásmo, mělo by být bezlicenční.

3.2 Porovnání bezdrátových technologií

Bezdrátové technologie jsou rozděleny do skupin dle IEEE 802.x. Vybrané skupiny jsou tři a to: WWAN, WLAN, WPAN. Každá kategorie je přímo srovnávána se zvolenými požadavky, které by měly být splněny.

3.2.1 WWAN (Wireless Wide Area Network)

Obecné označení pro bezdrátové sítě s širokým geografickým pokrytím. Mezi nejznámější zástupce této kategorie patří standardy jako GSM, GPRS, LTE, WiMAX. V kategorii WWAN je zbytečné rozebírat jednotlivé technologie a vysvětlovat, proč je takové řešení nevhodné pro účely komunikace mezi jednotlivými prvky. Vesměs se jedná o sítě vlastněné soukromými firmami zabývajícími se komunikacemi. Vybudovat si vlastní WWAN síť je v tomto případě nevhodné. Výhodou by sice mohla být široká dostupnost a rozšířenost technologií (GSM protokol podporují všechny mobilní telefony). Nevýhodou je, že jakýkoliv přenos dat by byl zpoplatněn.

Ale i z pohledu inteligentní domácnosti se najde opodstatnění pro využití WWAN kategorie (konkrétně GSM sítí). Jednoduchým příkladem je informování o mimořádné události v rámci systému pomocí SMS zprávy.

3.2.2 WLAN (Wireless Local Area Networks)

802.11 [http://www.ieee802.org/11/] Nejrozšířenější kategorie bezdrátových sítí, využívající pro komunikaci vysokofrekvenční rádiové vlny. Obecný název pro rodinu protokolů IEEE 802.11 je Wi-Fi. Výhodou jsou nižší pořizovací náklady a využití bezlicenčních frekvencí, které obvykle pracují na frekvencích 2,4 GHz a 5 GHz.

- **Wi-Fi** - Technologie bezdrátových sítí využívající ke komunikaci zmíněné frekvence 2,4 GHz a 5 GHz. Typy Wi-Fi sítí jsou odlišeny písmeny sítí: 802.11 a/b/g/n. Rozdíly jsou v přenosové rychlosti, použitém kódování, kmitočtovém pásmu atp. Silnou stránkou je nízká cena a velká rozšířenost této technologie a tím i vcelku nízká pořizovací cena. Hlavní

nevýhodou je, že Wi-Fi sítě v jakékoliv své specifikaci nesplňují základní požadavky pro účely komunikace se senzory. Hlavním nedostatkem je vysoká energetická náročnost celé bezdrátové technologie.

3.2.3 WPAN (Wireless Personal Area Networks)

802.15 [http://www.ieee802.org/15/] Jedná se o soukromé bezdrátové sítě, které mají rádius pokrytí v řádu několika metrů. Běžně jsou používané pro připojení různých zařízení k internetu, či ke komunikaci mezi sebou. Seskupená zařízení spolu většinou komunikují v režimu ad-hoc. Existující typy WPAN sítí jsou popsány v následujících podkapitolách.

- **IrDA** [http://www.irda.org/] - Komunikace pomocí infračerveného světla. Na jedné straně se nachází přijímač a na druhé vysílač. Běžně se používá u dálkových ovladačů a fotobuněk. Mezi výhody určitě patří nízké pořizovací náklady a odolnost vůči rušení od rádiových vln. Hlavními nevýhodami jsou velmi nízká přenosová vzdálenost, náchylnost na intenzitu světla okolí a nutnost přímé viditelnosti přijímače a vysílače. Vyjmenované nevýhody jasně naznačují, že tato technologie je zcela nevhodná pro využití v senzorových sítích.

- **Bluetooth** [http://www.bluetooth.com] - IEEE 802.15.1 Proprietární a otevřený standard pro bezdrátovou komunikaci. Komunikace probíhá ve volném pásmu 2,4 GHz. Přes široké rozšíření není tento standard zcela vhodný pro komunikaci mezi senzory [5]. Bluetooth není navržen pro dlouho trvající operace vzhledem k vyšší energetické náročnosti během provozu. Další omezení protokolu je zapříčiněno podporou maximálně 8 zařízení v rámci topologie.

- **Zigbee** [http://www.zigbee.org] - IEEE 802.15.4 – Standard, speciálně vytvořen jako vysoko úrovňový komunikační protokol pro malá zařízení s nízkým příkonem. Tento standard je běžně využíván senzorovými sítěmi i v rámci inteligentních domácností. Protože je protokol od počátku budován pro senzorové sítě, splňuje tak všechny požadavky, které od ideálního protokolu lze očekávat. Hlavní výhody jsou nízká energetická náročnost, nízký datový tok, bezpečné směrování a nižší pořizovací náklady oproti konkurenčním protokolům.

- **Z-Wave [**http://www.z-wave.com**]** - Bezdrátový komunikační protokol speciálně navržen pro automatizaci domácnosti. Obdobně jako Zigbee využívá RF rádio s nízkým příkonem. Komunikační frekvence je 900 MHz. O tento protokol se stará konzorcium Z-Wave Alliance čítající více jak 250 výrobců. Vyznačuje se jednoduchostí, velkou škálou existujících zařízení od mnoha výrobců. Mezi nevýhody lze zařadit použití pouze pro domácí automatizaci (není možné například použít tuto specifikaci pouze jako dálkové ovládání). Podporovaná topologie síťové hierarchie je omezena pouze na typ strom. Další nevýhodou je menší počet zařízení v rámci stromové struktury a velmi nízká přenosová rychlost oproti ostatním protokolům.

- **MiWi** [https://www.microchip.com/miwi/] - IEEE 802.15.4 – Proprietární bezdrátový protokol vyvíjený společností Microchip technology. Obdobně jako Zigbee a Z-Wave se jedná o malé zařízení s nízkým příkonem, vhodný pro domovní automatizaci. Z pohledu požadavků splňuje MiWi všechny podstatné vlastnosti.

3.3 Nejvhodnější bezdrátová technologie

Po zhodnocení všech výhod a nevýhod jednotlivých protokolů bylo jasné, že jedinou možností je vybírat z protokolů ve standardu IEEE 802.15, což jsou WPAN sítě. A to konkrétně mezi Z-Wave, ZigBee a MiWi. Z-Wave je i přes všechny své výhody a rozšířenost příliš svazující v možnostech použití. Co se týče Zigbee a MiWi, zde jsou rozdíly opravdu minimální. MiWi je novější protokol, který vznikl mimo jiné jako jednodušší alternativa složitějšího Zigbee protokolu. Vývoj s MiWi je jednodušší a tím i rychleji dokončený, což znamená, že výsledný produkt může rychleji na trh. Proto byl jako nejvhodnějším řešením zvolen protokol MiWi od firmy Microchip.

Kapitola 4

Protokol MiWi

MiWi protokol [5] je proprietární bezdrátové řešení od firmy Microchip. Spadá do kategorie WPAN, tedy soukromých bezdrátových sítí. Protokol je navržen tak, aby byl pro vývojáře co nejjednodušší a tím i zkrátil dobu potřebnou od návrhu po vydání produktu na trh.
Vybraný protokol je vyvíjen přímo pro potřeby senzorových sítí. Tedy jako energeticky a datově nenáročný. Důraz je kladen na nízké pořizovací náklady. Stejně tak průměrná velikost plošného spoje je menší oproti řešení od známějšího senzorového protokolu Zigbee.

4.1 Historie

Protokol byl vytvořen firmou Microchip v roce 2008. Odvíjí se od standardu IEEE 802.15.4, což je specifikace pro WPAN sítě s nízkou spotřebou, který byl později rozšířen pro podporu RF vysílačů od výše jmenované firmy. MiWi je jednoduše použitelná alternativa v kategorii bezdrátové komunikace určené pro senzorové sítě. Hlavní kategorií, kde lze MiWi využít, jsou menší zařízení a sítě s nízkým příkonem.

4.2 Síťový model

- **Fyzická –** Vysílače MiWi protokolu jsou vyráběny ve dvou verzích dle frekvence: 2,4 GHz a Sub-GHz. Obě pásma jsou bezlicenční, tudíž jejich použití není nijak zpoplatněno. Vybraný vysílač MRF89XAM8A pracuje ve frekvenčním pásmu 863 – 870 Mhz. Pro modulaci využívá FSK/OOK kódování a přenosová rychlost modulu je 40 kb/s.

- **Linková** – MiWi protokol používá standard IEEE 802.15.4 jako výchozí bod při vytváření linkové vrstvy. Například využívá mechanismu potvrzování datových přenosů. Tato metoda využívá speciální ACK příznak, který je součástí hlavičky paketu. Pokud je tento

příznak nastaven na hodnotu TRUE a po odeslání rámce nedorazí žádná potvrzující zpráva, tak po vypršení časovače dojde ke znovu odeslání rámce. Tento mechanismus sice dokáže odhalit nedoručené rámce, ale nedokáže detekovat poškozený rámec. Toto je již úkol pro aplikace z vyšších vrstev, které by měly být schopny zkontrolovat obsah rámce a odhalit tak případné poškození.

- **Vyšší vrstvy** – V rámci standardu 802.15.4 nejsou vyšší vrstvy síťového modelu definovány.

4.3 Protokol

MiWi protokol je založen na linkové a fyzické vrstvě, v rámci specifikace IEEE 802.15.4. Navržen je pro práci ve frekvencích jako jsou 2,4 GHz a Sub-GHz (frekvence nižší než 1 GHz). Vlastnosti protokolu poskytují podporu při vyhledávání uzlů, vytváření a realizace spojení, skenování ostatních uzlů a směrování mezi uzly.

Balíček **MiWi DE** (MiWi Development Environment) zahrnuje podporu dvou proprietárních protokolů. Oba dva fungují na principu bezdrátové sítě na krátkou vzdálenost.

- **MiWi P2P** – Jednoduchá peer-to-peer síť

- **MiWi PRO** – Podporuje síťovou topologii mesh. Možnost připojení více jak 8000 uzlů s 64 hopy.

4.3.1 Síť s vícenásobným přístupem

Specifikace IEEE 802.15.4 je definována jako síť s vícenásobným přístupem k přenosovému médiu. V praxi to znamená, že každé zařízení může komunikovat bez omezení. Existují zde dva přístupy ke komunikaci: *beacon* a *non-beacon*. V síti podporující režim *beacon* (maják) mají jednotlivá zařízení přidělené časové pásmo, ve kterém mohou vysílat. *PAN coordinator* (viz. další podkapitola) vyšle super-rámec typu *beacon* a ostatní zařízení se díky tomu sesynchronizují. Každé zařízení má pak přidělený určitý slot v super-rámci, během kterého může vysílat data, aniž by došlo na síti ke kolizi. Druhý způsob je *non-beacon* – zde není žádný super-rámec, ale zařízení vysílají pouze v případě, že žádné jiné nevysílá. MiWi protokol podporuje pouze *non-beacon* řešení.

4.3.2 Typy zařízení

Ještě než popíši jednotlivé typy zařízení, které pracují s protokolem MiWi, bylo by dobré se seznámit s rozdělením zařízení dle své funkcionality v síti. Tu lze rozdělit na dvě části následujícím způsobem:

- **FFD** (Full Function Device) – do této kategorie spadají zařízení s plnou podporu nabízených služeb a jsou běžně napájeny ze zdroje.

- **RFD** (Reduced Function Device) – zařízení mají limitovanou nabídku služeb oproti první FFD. Běžně jsou napájeny bateriemi, protože jejich provoz není tak energeticky náročný.

Nyní se zmíním o kategoriích, do kterých lze MiWi zařízení rozdělit. Jsou jimi:

- ***PAN Coordinator*** - Zařízení, které koordinuje celou síť. Toto zařízení je pouze jedno v konkrétní síti a pracuje v módu *FFD*, takže podporuje veškeré služby, které protokol MiWi nabízí. Běžnou funkcí tohoto zařízení je vytvoření a ustanovení sítě, včetně adresování. Při vytváření je zvolen kanál, na kterém bude komunikace probíhat a definuje se *PAN ID* sítě, což je jednoznačný identifikátor. Koordinátor si také udržuje tabulku zařízení, díky které má přehled o topologii sítě.

- ***Coordinator*** – Tento typ zařízení je volitelný a používá se spíše pro zvýšení dosahu dané sítě. Koordinátorů může být v síti více a mohou se starat, mimo jiné, o monitoring sítě a kontrolu funkcí. Pracují v módu *FFD*, stejně jako *PAN Coordinator*.

- ***End Device*** – Koncové zařízení, které je typicky napájeno z baterie. Slouží ke sběru dat, ovládání aktorů atp. Může pracovat v obou režimech, jak *FFD* tak *RFD*.

4.4 Topologie sítě

Dle aktuální definice standardu existují tyto topologie sítě.

- **Star Network** – Běžná topologie tvaru hvězdy. Uprostřed je *PAN coordinator*, který řídí

a komunikuje s ostatními zařízeními typu *End Device*. Pokud chtějí mezi sebou koncová zařízení komunikovat, musí tento proces probíhat přes uzel *PAN Coordinator*.

- **Cluster Tree Network** – Topologie ve stylu stromové struktury. Hlavním uzlem je stále *PAN Coordinator*. Ten ale může na sebe mít navázány jak koncové uzly, tak další uzly typu *Coordinator*. V tomto případě je *PAN Coordinator* kořenem stromu, *Coordinator* jako uzel a koncová zařízení jako listy stromu. Během komunikace se postupuje stromem, takže jakákoliv komunikace jde opět přes kořen stromu.

- **Mesh Network** – Velmi podobná topologie typu *Cluster Tree* s výjimkou, že není nutné dodržovat topologii stromu. Zařízení typu FFD tak mohou komunikovat s libovolným FFD uzlem. Avšak komunikace uzlů typu RFD musí být stále směřována přes nadřazený FFD uzel, v tomto případě se nemusí nutně jednat o *PAN Coordinator*. Výhodou je menší latence a vyšší spolehlivost, protože celá topologie není závislá na jediné cestě.

4.5 Adresování

MiWi protokol používá k adresaci tzv. 16 bitovou *Short adresu* neboli adresu zařízení. Ta je unikátní v rámci PAN sítě a slouží k identifikaci při komunikaci se zařízením. V IEEE 802.15.4 je specifikováno, že *PAN Coordinator* má vždy adresu 0000h.

- **Bity 0-6** jsou pojmenovány jako *Child's number* a slouží k rozlišení uzlu, zda se jedná o koordinátora nebo koncové zařízení. Nejnižší bit označuje typ zařízení (FFD nebo RFD).

- **Bit 7** je označen jako *RxOffWhenIdle* – pokud je tento bit nastaven na log. 1, znamená to, že zařízení má deaktivovaný přijímač a není tak možné doručit zprávy (typicky pro koncové uzly). Nadřazený koordinátor ukládá zprávy do bufferu a odešle je koncovému uživateli, jakmile si zažádá po probuzení o update.
- **Bity 8-10** nesou unikátní číslo koordinátora- protože je tento identifikátor 3 bitový, tak maximální počet koordinátorů včetně PAN je $2^3 = 8$.

- **Bity 11-15** jsou rezervovány pro budoucí využití.

4.5.1 Hlavička paketu

Obsahuje informace nutné pro směrování PAN sítí a je důležitá pro zpracování. Rozděluje se na tyto části:

- **Hops** (1 bajt) – Počet hopů (skoků), které může paket provést před tím, než je znovu odeslán. Lze nastavit na hodnotu 00h, čímž je znovu odeslání zakázáno.

- **Frame Control** (1 bajt) – Definuje chování paketu (vyžádání potvrzení paketu, informace o šifrovaném obsahu a podobně).

- **Cílová PANID** (2 bajty) – PANID cílového uzlu.

- **Cílová Short adresa** (2 bajty) – Short adresa cílového uzlu.

- **Zdrojová PANID** (2 bajty) – PANID zdrojového uzlu.

- **Zdrojová short adresa** (2 bajty) – Short adresa zdrojového uzlu.

- **Sekvenční číslo** (1 bajt) – sekvenční číslo pro snadnější identifikaci doručených paketů.

4.5.2 Směrování

Směrování v bezdrátových sítích patří k obtížným technikám, které jsou velmi složité na vývoj. MiWi protokol využívá pro směrování adresovou alokaci k rozeznání nadřazeného uzlu, kterému zpráva má být směrována.

4.6 MiWi v praxi

V následujících bodech je souhrn výhod a nevýhod protokolu MiWi a příklady, kde lze tento protokol nalézt v praxi.

4.6.1 Výhody a nevýhody MiWi

Výhody:

- Jednodušší návrh a vývoj bezdrátových sítí oproti jiným protokolům.
- Dobrá přenositelnost vytvořeného řešení napříč různými RF vysílači od firmy Microchip.
- Jednoduché rozšíření topologie v rámci MiWi protokol frameworku.
- Podporuje více jak 8000 uzlů s více jak 64 hopy v rámci sítě.
- Stack má podporu pro 8, 16 a 32-bitové mikrokontroléry.
- MiWi protokol pracuje na frekvencích 2,4 GHz a menších jak 1 GHz.
- Možnost využití MiWi P2P protokolu pro jednoduché sítě.

Nevýhody:

- MiWi protokol stack není tak odladěný jako Zigbee stack, což může občas způsobovat problémy.
- Pro využití MiWi protokolu je nutné z licenčních důvodů používat hardware firmy Microchip.
- Dle informací z diskuzních fór ohledně MiWi neměl tento protokol dlouho dobu fail-safe mechanismy a ani teď není stále tak vyzrálý jako jiné příbuzné protokoly.

4.6.2 Typické použití MiWi

- Vzdálený monitoring.
- Bezpečnost.
- Bezdrátový audio systém.
- Průmyslové senzory a aktory.
- Domácí automatizace.
- Lékařské účely.
- Osvětlení.

Kapitola 5

Ovladač pro Linux

Jednou z hlavních částí této práce je vytvoření ovladače pro operační systém Linux. Ten by měl komunikovat přes SPI rozhraní s HW modulem, který má na starosti řízení celé MiWi sítě. Aby bylo možné komunikovat s HW modulem přes uživatelský program, tak je potřeba vytvořit ovladač, který v sobě bude mít integrovaný komunikační protokol.

5.1 Linux

Linux je volně šiřitelný operační systém, který je pojmenovám po svém autorovi Linusu Torvaldsovi. Samotné jádro je volně šiřitelné pod licenční smlouvou GNU GPL. Linux vznikl postupným přepisem z UNIXu, ovšem při zachování volného šíření. Díky tomu má kdokoliv právo dát dohromady linuxové jádro a programy a vytvořit si tak vlastní distribuci. Těch existuje obrovské množství a liší se hlavně účelem, pro který je distribuce sestavena. Z pohledu této práce nás bude nejvíce zajímat distribuce Debian, respektive její upravená verze pro Raspberry Pi, nazvaná Raspbian [7]. Distribuce Debian byla uzpůsobena a optimalizována pro specifický HW a ARMový procesor, kterým Raspberry Pi disponuje.

5.2 Linuxové jádro

Přesněji řečeno jádro operačního systému Linux je, jak jsem již naznačil výše, klasickým příkladem otevřeného systému pod licencí GNU GPL. Řadí se mezi unixové systémy. Autor první verze jádra je identický s tvůrcem Linuxu Linusem Torvaldsem, který se dodnes podílí na jeho vývoji a na revizích změn v jádře [8].

5.2.1 Architektura jádra

Monolitické jádro operačního systému Linux je navrženo pro podporu reálného multitaskingu. Z pohledu designu je jádro implementováno jako jednolitý úsek kódu, podporující načítání externích modulů. Tato vlastnost snižuje velikost jádra, snižuje paměťovou náročnost a zvyšuje rychlost běhu a stabilitu.

5.3 Ovladač

Jádro operačního systému je složeno z velkého množství zdrojových kódů, které jsou značně složité a náchylné na jakoukoli nedokonalost. Chyba v jádře může zároveň znamenat výrazné bezpečnostní riziko pro celý systém.

Ovladač v jádře plní velmi specifickou roli rozhraní mezi hardwarovým zařízením a jádrem, které navenek působí jako black-box. Zapouzdřuje v sobě specifické detaily, které jsou s daným hardwarem spjaty.

Uživatelská aplikace kontaktuje driver příslušného zařízení. Ovladač vyvolá programovou rutinu, která kontaktuje hardwarové zařízení přes systémové rozhraní a postará se o veškerou práci. Ve chvíli, kdy zařízení odešle data zpět k ovladači, ovladač pomocí zpětného volání rutiny kontaktuje program, který driver zavolal.

Každý ovladač je závislý na specifickém typu hardwaru a operačního systému. Díky ovladači je možné psát uživatelské programy bez ohledu na daný typ operačního systému, protože specifické ovladače pro konkrétní operační systém zapouzdřují celou funkcionalitu vůči okolí. Navenek tak různé typy hardwaru působí jakoby bez rozdílu.

5.3.1 Komunikace s uživatelským prostorem

Aby bylo možné ovládat hw zařízení na uživatelské úrovni pomocí klasických programů, je nutné zajistit komunikaci mezi uživatelskou úrovní a úrovní jádra. Za účelem výměny dat mezi jednotlivými režimy linuxového jádra existuje několik typů souborových systémů [9]. Tato rozhraní jsou většinou reprezentována na základě souborů. Soubor většinou obsahuje jednu hodnotu, můžou ovšem obsahovat i více hodnot. Programy z uživatelského režimu mohou přistupovat k těmto souborům pomocí standardních funkcí *read / write*. Výhodou použití *read /write* operací na rozdíl například od *socketů* je fakt, že uživatelský režim podporuje velké

množství nástrojů, které odesílají data do režimu jádra jako jsou *echo*, *cat*, atp.

Komunikační rozhraní mezi uživatelským prostorem a prostorem jádra:

- **Procfs** – Rozhraní nacházející se v souborovém systému v umístění */proc*. Původně bylo navrženo pro export nejrůznějších informací o procesech (aktuální stav procesu, otevřené file-descriptory, a podobně). *Procfs* slouží pro pronásledující účely: Poskytuje informace o běžícím systému, přerušeních, volné paměti a verzi jádra. *Procfs* také obsahuje speciální podadresář */proc/sys*, který slouží ke konfiguraci velkého množství parametrů. Důležitou informací je také to, že ne všechny soubory uvnitř */proc/sys* používají *procfs* rozhraní. Některé z nich využívají *sysctl*, které je popsáno níže.
- **Sysfs** – Navrženo pro vyobrazení celého zařízení z pohledu jádra. Obsahuje informace o zařízeních, ovladačích, sběrnicích a jejich vzájemném propojení. Hierarchie propojení je členěna do značného počtu skupin, které jsou reprezentovány samostatnými adresáři. Mezi nejznámější patří např.
  - ***sys/block/*** – obsahuje všechna známá bloková zařízení jako jsou /hda, /sda.
  - ***sys/bus/*** – všechny registrované sběrnice. /bus obsahuje dva podadresáře.
    - ***device/*** - všechna zařízení připojená ke konkrétní sběrnici.
    - ***driver/*** - všechny ovladače, které jsou sdružené s touto sběrnicí.
  - ***sys/class/*** - každý typ zařízení zde má svůj podadresář např. /printer atd.
  - ***sys/device/*** - všechna zařízení, o kterých jádro ví, že jsou v systému přítomny.
  - ***sys/kernel/*** - drží si cesty pro ostatní souborové systémy jako je třeba debugfs.
  - ***sys/module/*** - seznam všech jaderných modulů, které jsou zavedeny do jádra.
  - ***sys/power/*** - obsahuje soubory s informací o stavu napájení pro určitý typ HW.
- **Configfs** – Jedná se o protiklad k *sysfs*, který vyobrazuje objekty jádra na úrovni souborového systému. Hlavním rozdílem mezi *configsys* a *sysfs* je, že v *configfs* jsou všechny objekty vytvořeny na úrovni uživatelského prostoru pomocí systémového volání *mkdir*. Jádro reaguje na vytvoření adresáře tím, že vytvoří nový atribut – soubor, který lze číst a editovat na uživatelské úrovni. Životní cyklus *configfs* objektu má plně pod kontrolou uživatelský režim, který v případě nečinnosti objekt odstraní včetně jeho obsahu, pomocí systémového volání *rmdir*. Společnou vlastností *configfs* a *sysfs* je fakt, že každý atribut může nabývat pouze jednu hodnotu.

- **Debugfs** – Jak již název napovídá, jedná se o jednoduchý souborový systém určený pro účely debuggingu. Je určen pro vývojáře, kteří jej mohou používat pro odladění svých programů. Pomocí Debugfs si lze nastavit zasílání některých doplňujících informací, které mohou pomoci při doladění vyvíjeného programu.
- **Sysctl** – Infrastruktura **sysctl** slouží k editaci parametrů jádra za běhu systému. Ty lze nalézt v souborech, které jsou umístěny v */proc/sys/**. K obsahu lze přistupovat přes standardní linuxové příkazy typu *cat*, *sysctl* a *echo*. Změny, které jsou provedeny za běhu pomocí echo, jsou platné pouze do restartu jádra. Pro trvalou změnu parametrů je nutné tyto hodnoty zanést do */etc/sysctl.conf*.
- **Ioctl** – Mimo běžné operace *read* / *write*, které jsou popsány u výše zmíněných typů, lze všechny souborově založené mechanismy ovládat pomocí dodatečných řídících příkazů, které jsou podporovány pomocí *ioctl*. To je implementováno jako jednoduché systémové volání, které přistupuje k určitým funkcím v režimu jádra.

  Volání ioctl je složeno ze tří částí:

  - 1) file/socket deskriptor.
  - 2) ID příkazu.
  - 3) argument.

  ID příkazu je unikátní a slouží k rozlišení specifického volání.

  Ioctl lze rozlišit dle argumentu na:

  - Příkaz neobsahující data.
  - Příkaz odesílající data jádru.
  - Příkaz odesílající data z jádra.
  - Čtení a editace dat jaderným modulem.
- **Systémová volání jádra** – ta se využívají, pokud chce uživatelský program použít nějaká data nebo službu, kterou zprostředkovává linuxové jádro. Systémových volání je velké množství (více jak 300) a podporují operace typu: práce se soubory, síťové operace, práce s procesy, atp. Každé systémové volání je identifikováno pomocí unikátního čísla, podle kterého je lze rozlišit. Při každém volání procesor přepne kontext do režimu jádra a provede odpovídající jadernou operaci.
- **Posílání signálu z jádra do uživatelského režimu** – Tento přístup je značně odlišný od

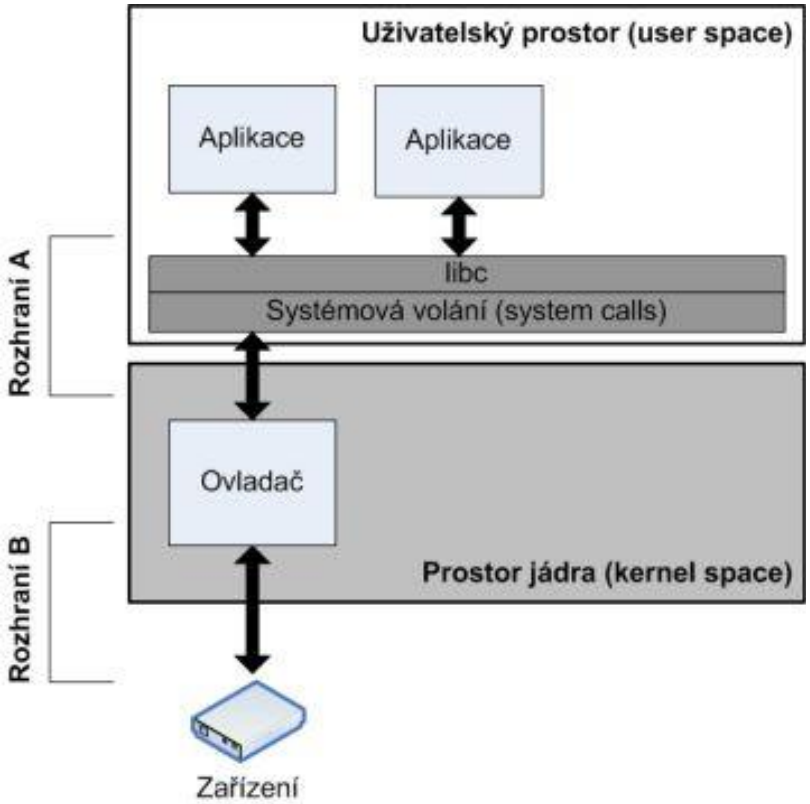
ostatních typů, protože signál lze vyslat pouze ve směru prostor jádra => uživatelský prostor a ne naopak. Stejně tak je množství dat, které lze v rámci signálu odeslat, omezeno.

Existují dva typy signálů:

- **Normální –** signál, který nenese žádná data. Tento signál je odeslán pouze jednou a není řazen ve frontě. V případě, že je víc takovýchto signálů posláno k jednomu procesu ve stejný čas, nemusí dojít k jejímu zpracování.
- **Real-time** - signál může nést až 32 bitů dat a přijímající proces si jej ukládá do fronty – je tedy zpracován vždy.

5.4 Základní rozdělení ovladačů

Ovladače [10] se vyskytují v prostoru jádra (tzv. kernel space). Z uživatelského prostoru (tzv. user space) jsou přístupné přes sadu systémových volání (Ilustrace 5 vyobrazuje Rozhraní A). Ovladače znakových a blokových zařízení jsou v Linuxu prezentovány jako soubory. Proto se při přístupu k ovladačům používají stejné sady systémových volání jako při práci se soubory. Ovladač nepřistupuje přímo k HW, ale používá sadu funkcí pro čtení/zápis do I/O pamětí a portů, které poskytuje jádro (Ilustrace 5 označeno jako Rozhraní B).



Ilustrace 2: Ovladač a jeho rozhraní (zdroj www.ucsimply.cz)

5.4.1 Znakové ovladače

Znaková zařízení jsou nejběžnějšími typy, s kterými se lze setkat. Komunikace probíhá pomocí toku bytů, což znamená, že přístup je velmi podobný jako při práci s běžnými soubory. Čtení a zápis probíhá po znacích a to v obou směrech komunikace. Ovladač znakového zařízení většinou obsahuje systémová volání: *open*, *close*, *read* a *write*. Přistupovat ke znakovým zařízením je možné přes běžný souborový systém. Příkladem může být přístup k */dev/tty1*.

5.4.2 Blokové ovladače

Na rozdíl od znakových ovladačů, které komunikují pomocí proudu bytů, u blokových probíhá komunikace po blocích dat pevné délky. K datům se typicky přistupuje náhodně a to pomocí adresace k určitému bloku. Mezi typické představitele blokových zařízení lze řadit zařízení úložná – pevné disky, optické jednotky a s nimi spojené paměti typu disketa, CD, DVD a jiná. Jedna ze základních vlastností blokových zařízení je možnost vytvoření samostatného souborového

systému.

Z pohledu jádra se k blokovému zařízení přistupuje zcela odlišným způsobem. Operace typu čtení a zápis jsou zde zcela asynchronní. To znamená, že požadavky na tyto operace jsou vytvořeny jádrem jako zcela samostatné entity. Každá z těchto entit může zároveň obsahovat více dílčích požadavků.

Požadavky jsou řazeny ve frontách, nad kterými pracuje plánovač I/O. Ten na základě algoritmu řídí, v jakém pořadí mají být zpracovávány. Plánovač předává ovladači informaci o tom, jaké požadavky má zpracovat a odeslat dále do zařízení a zároveň zpětně nahlásit jejich vyřízení.

Je tedy na ovladači samotném, aby vhodným způsobem zpracovával požadavky od plánovače. Z toho popisu jasně vyplývá, že se jedná o asynchronní operace – bloková zařízení tedy pracují asynchronně.

5.4.3 Síťové ovladače

Mezi síťové ovladače lze zařadit vše, co jakýmkoliv způsobem umožňuje komunikaci síťového charakteru. Na rozdíl od znakových a blokových ovladačů je zde dlouhá cesta mezi uživatelským procesem a samotným zařízením. Uživatelské programy používají pro systémová volání objekt, který je nazýván *socket*. Ten podporuje jak klasická systémová volání (například *read* a write), tak systémová volání vlastní.

Hlavní výhodou je jednoduché uzpůsobení existujících uživatelských programů pro přidání podpory síťové komunikace. K datům se přidávají hlavičky, které se při příjmu musí kontrolovat. Navíc dochází k ověřování, že je komunikace zabezpečena a data jsou nepoškozena.

Síťová zařízení jsou silně asynchronní. Data mohou kdykoliv přijít a ovladač by měl být schopen si je od zařízení převzít a zpracovat. Data se zde přenášejí po blocích (v tomto případě po paketech). Tudíž je tato vlastnost je s blokovými ovladači společná.

5.5 Znakový SPI ovladač

Pro účely této práce jsem vybral jako nejvhodnější možnost implementovat znakový ovladač s podporou komunikace přes rozhraní SPI. Před návrhem a implementací takového ovladače je důležité důkladné prostudování teorie.

5.5.1 IOCTL (Input / Output Control)

Systémové volání určené pro operace typu *vstup / výstup* v rámci komunikace s hardwarovým zařízením. Každé volání v sobě obsahuje určitý kód, který je pro dané zařízení specifický. Systémové volání *ioctl* je dostupné pro operační systémy typu Unix, Linux, MAC OS a podobné. Analogická systémová volání lze také nalézt pro operační systémy od Microsoft Windows a to v rámci jejich Win32 API. Podrobnější popis Ioctl lze nalézt v podkapitole 5.3.1.

5.5.2 Souborový systém /proc

Read/write mechanismus určený pro odesílání informací procesům z jádra. Původně byl navržen pro zjištění aktuálního stavu procesu. Nyní je velmi hojně využíván pokud je potřeba hlásit konkrétní informace. Z pohledu SPI ovladače jsou zajímavé části např. */proc/modules* nebo */proc/meminfo*. První z nich slouží k vylistování seznamu modulů, druhý k zobrazení statistiky ohledně využité paměti.

5.5.3 Rozhraní SPI

Serial Peripheral Interface bus je synchronní sériové periferní rozhraní. Slouží ke full-duplex komunikaci na krátkou vzdálenost mezi mikroprocesory a ostatními IO jako jsou displeje, paměti, převodníky atp. Komunikace probíhá po společné sběrnici a adresace je vedena po zvláštních vodičích.

Zařízení komunikují v režimu master - slave. Master se stará o celou komunikaci, ustanovuje ji, rozhoduje, s kým bude komunikovat a sdílí svůj hodinový signál. Synchronizace slave zařízení probíhá pomocí hodinového signálu od mastera a komunikuje s masterem v případě, že je aktivní dle SS.

Rozhraní SPI sběrnice:

- **SCLK** – Serial Clock (výstup od mastera).
- **MOSI** – Master Output, Slave Input (výstup od mastera, vstup pro slave).
- **MISO** – Master Input, Slave Output (vstup pro mastera, výstup od slave).
- **SS** – Slave Select (výstup od mastera, výběr, který slave bude aktivní).

5.5.4 SPIdev

SPI má velmi limitované API, které podporuje pouze základní half-duplex operace typu read, write a full-duplex *ioctl* operace. *Spidev* zpřístupňuje SPI rozhraní pro uživatelské programy, což přináší celou řadu výhod a zjednodušení. Odpadá tak nutnost psaní jaderných programů pro jednoduché využití SPI rozhraní. Pro běžné programy, které využívají rozhraní SPI, je uživatelský program mnohem lepší volbou, která je zároveň bezpečnější.

Nejčastěji používaný SPI ovladač pro Raspberry Pi je *spidev* [11] – což vyplývá i z nejrůznějších diskuzních fór na téma SPI a Raspberry Pi. Zavedením ovladače *spidev* dojde k vytvoření adresáře uvnitř souborového systému v umístění */dev/*. Díky tomu je možné číst a zapisovat na toto rozhraní pomocí **ioctl** příkazů.

Kapitola 6

Návrh SW a HW řešení

Šestá kapitola popisuje návrh softwarového a hardwarového řešení. V první části se věnuji popisu demonstrační desky od společnosti Microchip, která slouží k seznámení se s protokolem MiWi. Tato deska je zároveň inspirací pro vlastní HW modul. Před samotným popisem tvorby hardwarového modulu se krátce zmiňuji o zařízení postaveném na Linuxu, které se stará o řízení adaptéru. Jako cílovou platformu jsem vybral minipočítač Raspberry Pi. Od demonstrační desky se popis pozvolně přesouvá k návrhu vlastního adaptéru.

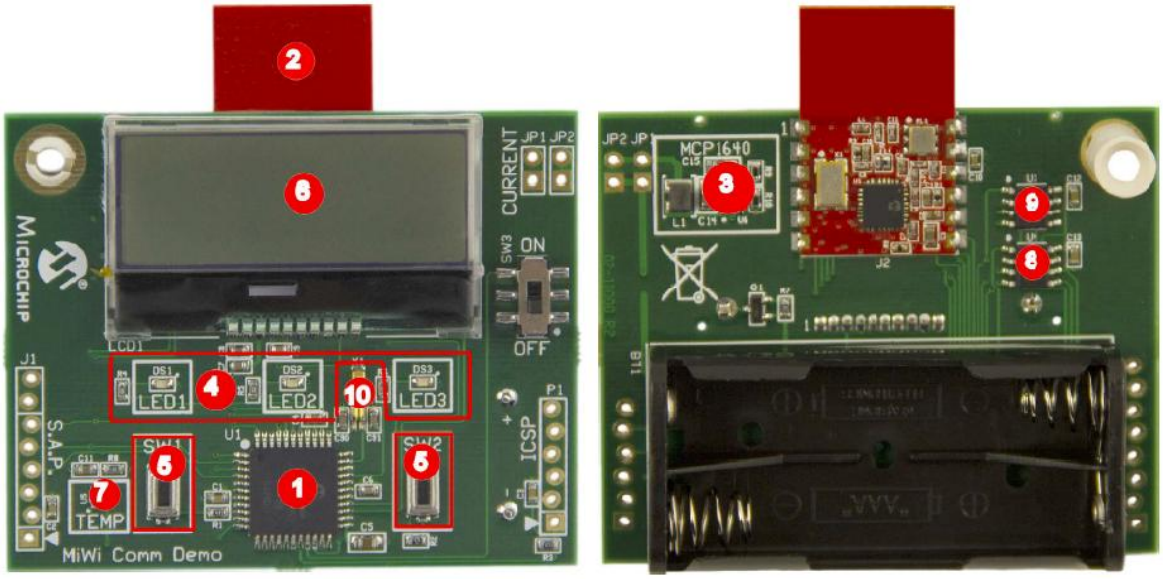
Návrh včetně implementace HW části byl hlavní náplní zimního semestru, na softwarovém řešení jsem pracovat až v průběhu letního semestru.

6.1 Demo board MiWi

Tato vývojová platforma je primárně určena pro vývojáře, kteří si tak mohou osahat a vyzkoušet práci s protokolem MiWi v praxi [12]. Výrobcem demonstrační desky je firma Microchip a oficiální název celého balíčku je MiWi demo kit. Obsahem balení jsou dvě demonstrační desky. Každá tato demonstrační deska je z výroby naprogramována ukázkovým programem. Další ukázkové programy lze stáhnout z oficiálních stránek výrobce. Ty obsahují příklady základních funkcí, například jak se ustanovuje PAN síť, či ukázka komunikace mezi jednotlivými demonstračními deskami, které v tomto případě jsou uzly MiWi sítě.

6.1.1 Hardware desky

Ilustrace 3 vyobrazuje obě strany demonstrační desky s očíslovanými součástkami. Jejich seznam je vypsán pod obrázkem.

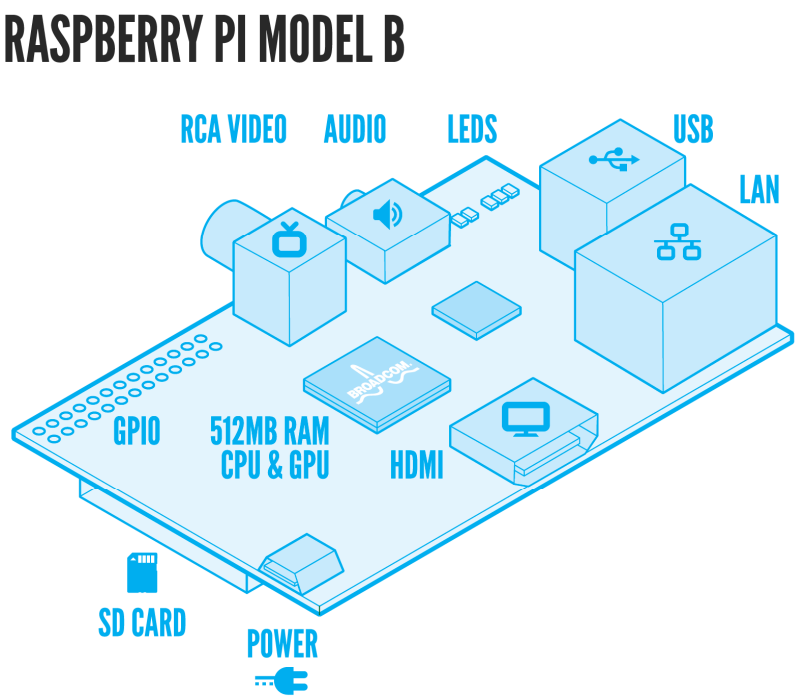


Ilustrace 3: Osazení MiWi demo boardu

1. PIC18F46J50 8-bitový XLP mikrokontrolér
2. MRF89XAM8A RF vysílací modul
3. +3,3 V napěťový regulátor (MCP1640)
4. Tři indikační LED (červená, žlutá, zelená)
5. Dvě mikrospínací tlačítka (SW1 a SW2)
6. 2 X 16 znakový LCD displej
7. MCP9700 teplotní senzor
8. 2K SPI EEPROM s unikátní MAC adresou
9. 1 Mbit SPI Serial Flash paměť
10. 32 kHz krystal podporující sleep mode

Hlavní řídící MCU je zde 8-bitový PIC18F46J50 (1). Hodinový signál mu udává krystal o frekvenci 32 kHz (10) podporující sleep mode. Vysílací modul (2) je RF od Microchipu a pracuje na frekvenci 868 MHz (verze pro EU). Protože je deska napájena z baterií, tak o regulaci hladiny napětí na 3,3 V se stará napěťový regulátor (3). K signalizaci činnosti slouží tří LED diody (4) a LCD displej (6). Uživatel vstupuje do systému pomocí dvojice tlačítek (5). V ukázkových programech slouží k ovládání menu. Jedním z ukázkových programů je vyčtení teploty z integrovaného teplotního senzoru (7). Deska je vybavena EEPROM pamětí, která obsahuje unikátní MAC adresu (8) a uživatelskou flash pamětí o velikosti 1Mb (9).

6.2 Raspberry Pi



Ilustrace 4: Model Raspberry Pi (zdroj www.raspberrypi.org)

HW modul pouze komunikuje se senzory, a proto je potřeba nadřazené zařízení, které zpracuje požadavky, řídí adaptér a tím činnost celé MiWi sítě. K tomuto účelu jsem vybral minipočítač Raspberry Pi [13]. Jeho velikost odpovídá přibližně velikosti kreditní karty a je postaven na procesorové architektuře ARM. Raspberry Pi je vytvořen a vyráběn nadací Raspberry Pi Foundation. K dispozici jsou dva modely: Model A stojí 25 $ a model B 35 $. Levnější model nemá ethernetovou přípojku a pouze jeden USB port. Standardní vybavení prodávanějšího modelu B vyobrazuje Ilustrace 3. Systém běží z SD karty, do které si můžete stáhnout jednu z upravených linuxových distribucí, které lze nalézt na oficiálních stránkách.

Nejznámější linuxovou distribucí je Raspbian, což je upravená linuxová distribuce vycházející z mnohem známější distribuce Debian. Hlavní modifikací této distribuce je podpora HW založeného na procesoru typu ARM.

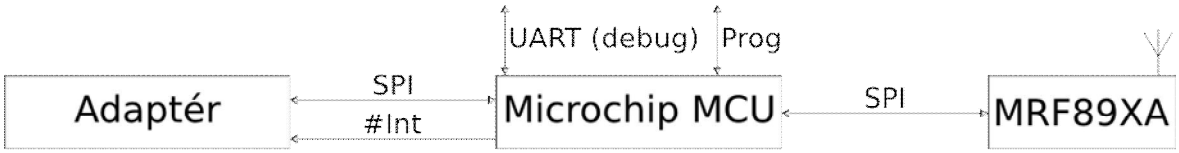
Spousta prodaných kusů tohoto miniPC míří k televizím jako multimediální centrum a to z důvodu, že dokáže přehrávat video ve FullHD rozlišení díky své integrované grafické hardwarové akceleraci.

Adaptér lze připojit k Raspberry Pi pomoci GPIO pinů. Komunikace mezi deskami probíhá pomocí sériového rozhraní SPI, které je u obou z nich integrováno.

Dalšími připojenými piny jsou:

1. Napájení adaptéru.
2. Zem pro adaptér.
3. Signál přerušení od adaptéru k PC. Tento signál tu je z důvodu zamezení pollingu a tím zbytečného vytěžování procesoru.

6.3 Návrh HW modulu



Ilustrace 5: Blokové schéma HW modulu

Demonstrační deska od firmy Microchip je pro funkci HW modulu příliš složitá. Ta má sloužit jako adaptér, který koordinuje ostatní MiWi síťové prvky a obsahuje některé komponenty, které jsou zbytečné a pouze navyšují celkové náklady na výrobu. Vzhledem k tomu, že existující řešení této desky je již odzkoušené a funkční, dohodl jsem se s vedoucím diplomové práce o možnosti inspirovat se při tvorbě nového HW modulu. Jednotlivé části nově navrženého adaptéru jsou popsány v následujících podkapitolách. Obrázky schématu zapojení a DPS (desky plošných spojů) jsou k nalezení v příloze.

6.3.1 Mikrokontrolér PIC18F46J50

Licenční podmínky společnosti Microchip stanovují povinnost použít mikroprocesor jejich výroby v kombinaci s protokolem MiWi. Vzhledem k dostatečnému množství konfigurovatelných portů, které mikroprocesor PIC18F46J50 nabízí, a přijatelné pořizovací ceně, byl tento konkrétní typ MCU zachován i pro HW modul.

6.3.2 RF vysílací modul MRF89XAM8A

Vysílací modul pracující na frekvenci 868MHz zůstal nezměněný. Splňuje vše, co je potřeba, a navíc obsahuje i vlastní integrovanou anténu. Jakýkoliv zásah by mohl způsobit pouze komplikace, a to jak z pohledu funkcionality, tak opět z pohledu porušení licence.

6.3.3 Indikační LED

Tři signalizační LED zůstaly součástí desky, ~~pouze jsou připojeny na jiných pinech MCU~~. V poslední revizi desky jsou opět na původním místě. Tím se zjednodušuje i úprava již existujících demonstračních programů. Jelikož LCD displej není součástí adaptéru, jsou tyto diody jedinými výstupy, které mohou signalizovat funkci adaptéru, v němž se právě nachází.

6.3.4 Spínací tlačítka

Tlačítka jsem odstranil a místo nich jsou na desce pouze piny, které při spojení vyvolají vstupní impuls. Konkrétní funkce těchto pinů není prozatím určena. Jsou vhodná hlavně k testovacím účelům, případně jako hard-reset systému po nečekaném zamrznutí.

6.3.5 Příprava pro EEPROM paměť s MAC adresou

Na adaptéru se nachází příprava pro osazení EEPROM pamětí, která není v rámci návrhu potřeba. Pro komunikaci s MCU se využívá sdílená SPI sběrnice, tudíž nedochází ke zbytečnému obsazení dalších pinů mikroprocesoru. V případě potřeby ji lze bez větších komplikací osadit. MAC adresa je napevno nastavena uvnitř MCU, ovšem odstranění EEPROM paměti přináší nutnost provést určité změny v systému. V rámci demonstračních aplikací se k této paměti přistupuje velmi často za účelem ukládání dat. To ovšem lze vyřešit i jiným způsobem, čímž lze celé řešení elegantně obejít.

6.3.6 Rozhraní

Adaptér obsahuje následující rozhraní:

- **Rozhraní k Raspberry Pi** (SPI + Vcc + GND + #INT) – rozhraní se skládá ze sériového rozhraní SPI, napájení a zem pro adaptér a signál přerušení.
- **ICSP** – rozhraní k obvodovému debuggeru a programátoru MPLAB ICD 3 od firmy Microchip – Rozhraní slouží pro naprogramování MCU přes vývojové prostředí MPLAB.
- **UART** – rozhraní pro sériovou komunikaci. Po připojení UART/USB převodníku lze využít rozhraní jako debuggovací výstup.

6.3.7 Nepoužité součástky

Na demonstrační desce lze nalézt několik součástek, které nemají v adaptéru využití. Napěťový regulátor není potřeba, protože adaptér je napájen přímo z Raspberry Pi. Toho nahradily filtrační kondenzátory, které vyhlazují stejnosměrnou složku. LCD displej byl také nadbytečný, o signalizaci se starají tři SMD LED diody. Teplotní senzor byl také odstraněn, protože nemá u adaptéru žádné využití. Paměť 1Mb SPI flash, je také zbytečná a důležitá data jsou uložena přímo v řídícím zařízení, v tomto případě v Raspberry Pi.

Díky těmto úsporám se podařilo zmenšit velikost desky a snížit tak i výdaje na výrobu hardwarového modulu. Hlavní výhodou toho řešení je snížení proudového odběru. Raspberry Pi má výstupní proud u 3.3V větve limitován na 50mA. Tato hranice je u navrženého hardwarového modulu dostatečná, a tak lze modul napájet interně bez nutnosti externího napájení.

6.4 Návrh DPS HW modulu

Návrh schématu zapojení a DPS byl proveden v editoru plošných spojů Eagle[1]. DPS je dvouvrstvá a veškeré součástky jsou pouze na horní straně, což byl jeden z hlavních požadavků během návrhu. Protože vše nešlo zcela podle plánu, následující podkapitoly popisují jednotlivé revize DPS, které jsem v průběhu práce vytvořil či navrhl.

6.4.1 Revize 1

U první revize je RF vysílač umístěn na desce dle doporučení datasheetu [14]. Běžné součástky jako rezistory a kondenzátory jsou typu SMD se zvolenou velikostí 0805. Adaptér je ovšem kompletně přepinován z důvodu snížení počtu křížení cest za účelem vytvoření co nejmenší a nejjednodušší DPS. Po osazení desky jsem začal upravovat demonstrační aplikaci MiWi demo kitu. Přemapování se ovšem ukázalo jako nedostačující a některé piny nebylo bohužel možné přemapovat na potřebné funkce. Z tohoto důvodu jsem se rozhodl kompletně přepracovat schéma i DPS.

[1] http://www.cadsoftusa.com/

6.4.2 Revize 2

U druhé revize zapojení jsem se rozhodl lépe využít původní funkci pinů MCU a eliminovat nutnost přemapování na minimum. Rozložení pinů u konektoru mezi HW modulem a RPi bylo upraveno pro jednodušší propojení. Anténní MiWi modul je propojen s MCU tak, že není nutné přemapovávat žádné další piny. Nevýhodou tohoto řešení je složitější zapojení, a proto jsem si při tvorbě DPS dal značnou práci, abych zachoval veškeré součástky na jedné straně a druhá strana sloužila pouze na nejnutnější přemostění křížících se cest. Hotové zapojení jsem nechal zkontrolovat od vedoucího mé DP panem Ing. Novotným a panem Ing. Hájkem a bylo mi doporučeno zakomponovat několik dalších změn, ze kterých vznikla revize 2.1. Tato revize se ovšem nedostala do produkce a zůstala pouze v elektronické podobě.

6.4.3 Revize 2.1

Poslední a finální revize schématu zapojení a DPS byla vyrobena firmou PragoBoard. DPS jsem osadil potřebnými součástkami a oživil. Před samotným připojením k Raspberry Pi jsem ověřil proudový odběr na laboratorním zdroji a tím i eliminoval možnost zkratu na desce. V této revizi DPS jsou zakomponovány veškeré poznatky a odstraněny problémy, které měly revize předešlé. I když není změn mnoho oproti revizi 2, tak je z důvodu pootočení MCU o 90°, wiring DPS zcela přepracován. Zapojení jednotlivých komponent k procesoru je nyní více podobné MiWi demo kitu, čímž se zjednodušuje adaptace demonstračních programů. I přes neoptimální rozložení komponent a pinů procesoru se mi povedlo snížit velikost výsledné DPS a je tak rozměrově nejmenší ze všech revizí viz. příloha.

6.5 Návrh ovladače

Návrh ovladače byl jednou z nejdůležitějších částí této práce. Prozkoumání řešení SPI na Raspberry Pi nebyl vůbec snadný úkol. I přes rozsáhlou komunitu fanoušků okolo RPi, včetně oficiálního fóra, jsou informace okolo SPI rozhraní velmi strohé. Většina problémů, které se u SPI řeší bylo hlavně zprovoznění samotného SPI a aktivace *spidev* zařízení uvnitř souborového systému v */dev/*, které dělá u některých RPi desek problém. Po prozkoumání možných typů řešení jsem se rozhodl pro editaci stávajícího ovladače SPI, který je součástí linuxového jádra.

6.5.1 WiringPi

Malá knihovna [15] vyvíjená nadšeným programátorem jménem Gordon Henderson, nabízí možnost ovládat SPI přímo (odesílání / přijímání). Vývojář vybízí případné uživatele jeho knihovny k možnosti rozšíření o potřebnou funkcionalitu, pokud chtějí. Svůj projekt sice dokládá různými statistickými údaji o rychlosti atp., ovšem dle repozitáře Git zde byla poslední aktivita v srpnu 2013. Z toho důvodu není jisté, zda bude v případně nutnosti dostupná podpora z jeho strany, v případě nutnosti. S tímto vývojářem jsem vedl emailovou konverzaci na téma *Spidevu* a dle jeho slov se cca na konci roku 2013 přesunul *Spidev* čistě do jádra a editace, což není zrovna jednoduchou záležitostí. O tom jsem se přesvědčil na vlastní kůži. Své řešení označil jako aktuálně neudržované. To je hlavní důvod, proč jsem se toto řešení rozhodl raději dále nerozvíjet.

6.5.2 Spike

Z pohledu vývoje ovladače pro SPI lze na internetu najít užitečný tutoriál pro tvorbu takového ovladače a to včetně ukázkového zdrojového kódu [16]. Na první pohled se zdá být vývoj jednodušší ve spojitosti s možností dynamického registrace SPI zařízení. Největším problémem jsou ale téměř neexistující doplňující informace, čímž se celý případný vývoj stává mnohem náročnějším. I přesto jsem se rozhodl část *Spidev* ovladače přepsat do formy dynamické registrace. Detailnější popis tohoto řešení lze nalézt v části implementace.

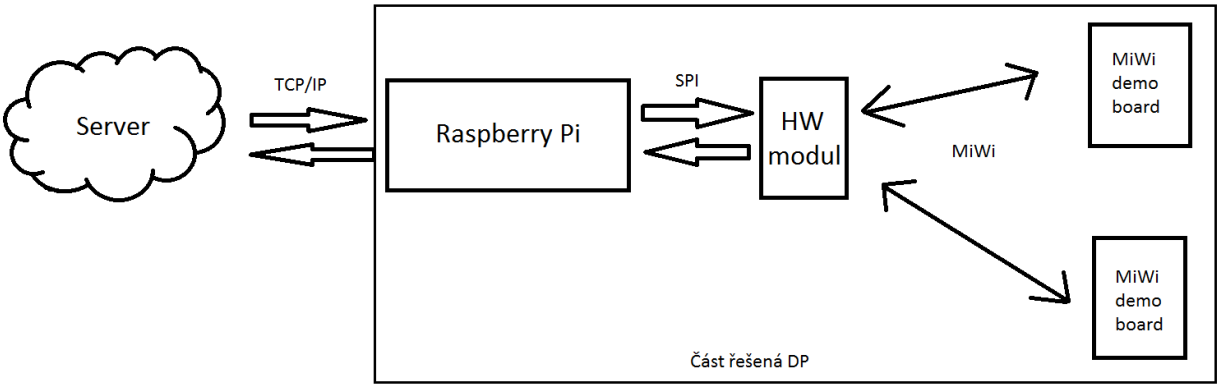
6.5.3 Spidev

Spidev[2] přistupuje k SPI, které RPi podporuje bez nutnosti překompilovat jádro. Jedná se o hlavní SPI jaderný modul, který je po kompilaci jádra ovšem obalen dalšími moduly, a výsledný ovladač se tak nachází pod označením *spi_bcm2708*. Editace modulu *Spidev* se zdá jako nejideálnější řešení, ovšem za předpokladu, že vše půjde dle plánu. Nevýhodou je, že na rozdíl od standardního *Spidevu* se u Raspberry Pi používá upravená verze, což komplikuje případnou částečnou přenositelnost budoucího ovladače na jiná zařízení, postavená také na ARMové architektuře. Celý zdrojový kód jádra lze nalézt na oficiálním hostingu GIT repozitářů, Github.

[2] https://www.kernel.org/doc/Documentation/spi/spidev

6.6 Návrh blokového schématu

Na ilustraci níže je vyobrazen návrh zapojení výsledného řešení. Část označená velkým obdélníkem ilustruje část, která je specifická pro tuto práci.



Ilustrace 6: Návrh blokového schématu finálního zapojení

6.7 Návrh komunikačního protokolu

Návrh komunikačního protokolu mezi HW modulem a senzory je specifikován v rámci projektu IoT FIT VUT v Brně. Variantou bylo vytvořit vlastní komunikační protokol nebo použít již protokol navržený. Druhá možnost přináší větší pravděpodobnost, že výsledné řešení by mohlo být použito v praxi.

Následující tabulky 1 a 2 popisují daný komunikační protokol definovaný v projektu IoT na FIT VUT v Brně. Bližší informace ohledně protokolu lze nalézt na oficiálních wiki stránkách projektu[3] (stránky vyžadují autorizaci).

[3] https://merlin.fit.vutbr.cz/wiki-iot/index.php/Senzory_adapter_server_komunikace

6.7.1 Návrh komunikačního protokolu mezi senzory a adaptérem

Bity	Význam	Použitý typ
0 – 15	Verze protokolu	Unsigned short (2B)
16 – 31	Stav baterie	Unsigned short (2B)
32 – 39	Počet přenášených veličin (1-5)	Unsigned char (1B)
40 – 47	Rezervované	Pad byte (1B)
48 – XX	Seznam dvojic typ:hodnota	

Tabulka 1: Komunikační protokol mezi senzory a rádiem adaptéru

Typ	Označení	Význam	Jendotky	Použitý typ
0x00	t0	Teplota	(°C)	Long (4B)
0x01	t1	Vlhkost	(%r)	Long (4B)
0x02	t2	Tlak	(hPa)	Long (4B)
0x03	t3	Sepnutí-senzor	(ON/OFF)	Long (4B)
0x04	t4	Sepnutí - přepínač	(ON/OFF)	Bool (1B)
0x05	t5	Intenzita světla	(lx)	Bool (1B)
0x06	t6	Intenzita hluku	(dB)	Long (4B)
0x07	t7	Emise (CO2)	(ppm)	Long (4B)
0x08	t8	Rezervované		Long (4B)

Tabulka 2: Detailní popis seznamu dvojic typ:hodnota

6.7.2 Návrh komunikačního protokolu mezi adaptérem a Raspberry Pi

Ve chvíli, kdy dochází ke spárování senzoru s adaptérem, odesílá senzor inicializační hodnotu nastavenou na 0xFF. Tímto dává najevo, že prozatím nedošlo ke spárování s adaptérem. Pokud je již senzor s adaptérem spárován a pouze se identifikuje po ukončení sleep-módu, identifikuje se s inicializační hodnotou nastavenou na 0x00. Tyto hodnoty si Raspberry Pi uchovává pro každý senzor.

Bity	**Význam**	**Použitý typ**
0 – 7	Inicializační hodnota	Unsigned char (1B)
8 - 23	Verze protokolu	Unsigned short (2B)
24 - 39	Stav baterie	Unsigned short (2B)
40 - 55	Kvalita signálu	Unsigned short (2B)
56 - 63	Počet přenášených veličin	Unsigned char (1B)
64 - 71	Rezervované	Pad byte (1B)
72 - XX	Seznam dvojic typ:hodnota	

Tabulka 3: Komunikační protokol mezi adaptérem a Raspberry Pi

Dvojice typ:hodnota zůstává stejná jako u předešlého typu komunikačního protokolu, který je popsán v tabulce 2.

Kapitola 7

Implementace (SW a HW)

7.1 Tvorba HW modulu

Po vytvoření několika kusů HW modulu (první revize) jsem začal pracovat na osazení součástkami a oživení desky. Vytvořil jsem si 3 kopie pro případ, že by některá z nich nebyla funkční (HW chyba, zkrat mezi piny, nefunkční součástky). Všechny tyto 3 kopie jsem úspěšně ověřil. Pro kontrolu jsem si vytvořil jednoduchou demonstrační aplikaci, která testovala korektní zapojení a osazení rozsvícením diod.

Po odzkoušení funkčnosti bylo nutné upravit demonstrační program a tím otestovat zapojení včetně MiWi modulu. Protože demonstrační program obsahuje velké množství zbytečných částí, rozhodl jsem se začít projekt od začátku a inspirovat se demonstračním programem.

Tato metoda se brzy ukázala jako velmi neefektivní, a to jak z důvodu časové náročnosti, tak ze strany složitosti celého programu. Výhodou bylo, že jsem do detailu pochopil funkci samotného programu. Ve chvíli, kdy jsem narazil na problémy, které jsou dle uživatelských fór na stránkách Microchipu jsou velmi sporadické a těžko odhalitelné, rozhodl jsem se začít zcela od začátku ale opačně. Vzal jsem již existující demonstrační příklad a upravil jsem jej na mnou vytvořený HW modul. I přes mnoho změn hlavně z důvodu absence displeje, jsem dospěl k funkčnímu řešení velmi rychle. Důvodem bylo detailní pochopení aplikace z předešlého postupu.

Zde jsem již ale narazil na problém, který jsem nebyl schopen změnou kódu odstranit. Některé piny nepodporovaly funkci přemapování pinů a z toho důvodu nebylo možné zprovoznit MiWi modul.

To byl hlavní podmět ke kompletnímu předělání návrhu a DPS. Bylo nutné změnit zapojení mezi piny MCU a ostatními komponenty. Na základě toho jsem si mohl dovolit celé zapojení předělat. Do nové verze jsem tak zakomponoval i ostatní poznatky, které se u první verze objevily.

Například ne zcela vhodné rozestavění součástek, zbytečné obsazení přemapovatelných pinů a podobně.

U revize 2 jsem se rozhodl pootočit MCU v rámci DPS o 90°, čímž bylo možné vést cesty mezi součástkami s méně kříženími. Zároveň s tím jsem provedl úpravu rozložení součástek, což vedlo k celkovému zmenšení rozměrů desky. K návrhu zapojení jsem dostal ještě několik dalších poznatků, které bylo potřeba zakomponovat do nové verze. Protože to ale znamenalo změnu napojení několika pinů MCU, rozhodl jsem se opět otočit MCU zpět o 90° na původní pozici i přesto, že to znamenalo kompletní předělání DPS. S touto revizí jsem si dal značnou práci při ideálním rozložením součástek, cest a křížením a povedlo se mi výslednou DPS ještě zmenšit. Porovnání revize 1 a revize 2.1 lze nalézt v příloze.

7.2 Implementace SW pro HW modul

V rámci projektu IOT na FIT VUT v Brně probíhá vývoj i dalších částí inteligentní domácnosti. Po domluvě s ostatními členy skupiny jsme došli k názoru, že nejlepší možností je použít navržený komunikační protokol, který je kompatibilní s ostatními částmi inteligentní domácnosti. Díky tomuto rozhodnutí je tak možné výsledný produkt využít v reálném prostředí. Jako adaptér slouží pro senzorovou část již zmiňovaný MiWi demo kit. Existující SW pro tento modul bylo tak potřeba opravit pro mnou navržený HW modul. Díky poslední revizi, která využívá ideální rozložení pinů, nebylo zapotřebí přepisovat celý kód, ale pouze upravit příslušné části a chybějící součásti mého adaptéru odstranit z kódu (displej atp.). Adaptér je tak připraven pro komunikaci přes SPI rozhraní s ovladačem ze strany Raspberry Pi, tak s uzly MiWi sítě na straně druhé.

7.3 Implementace ovladače

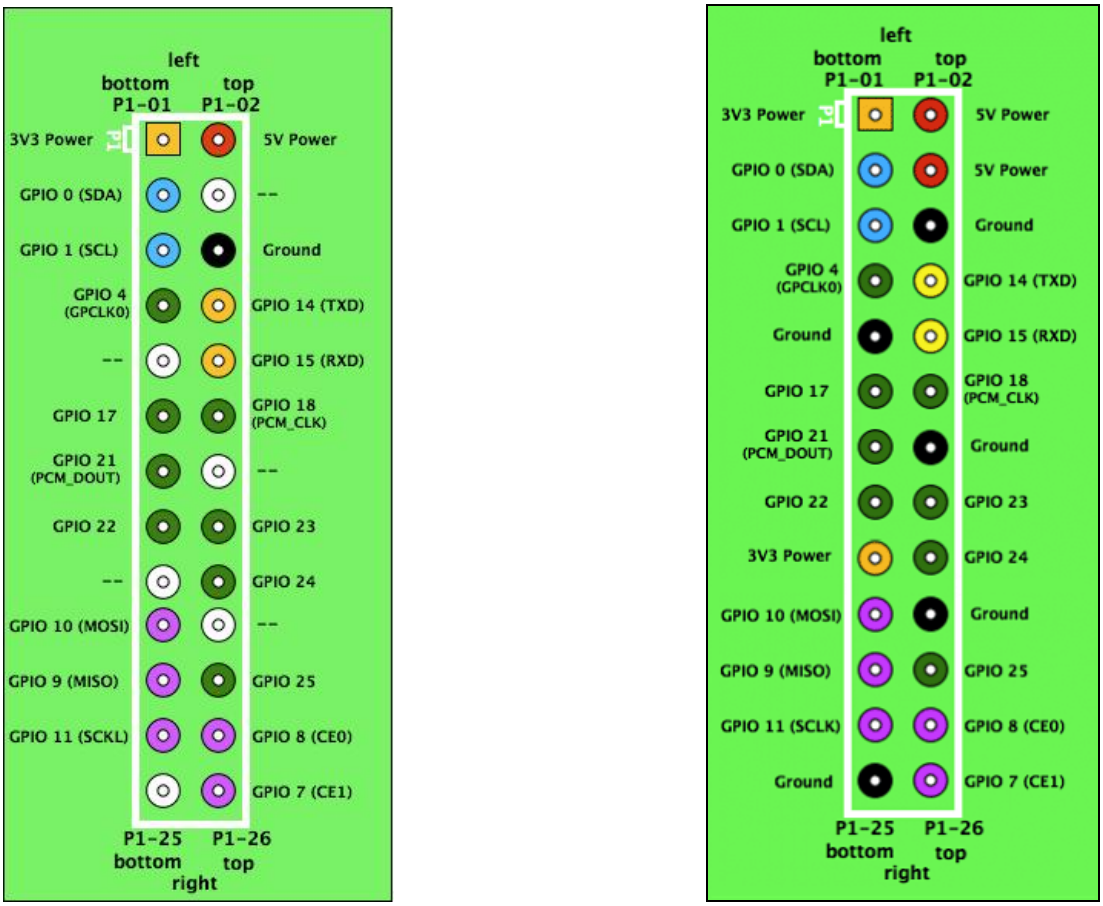
SPI má na Raspberry Pi nativní podporu. Nejdříve je ovšem zapotřebí odstranit záznam z blacklistu RPi, který je umístěn v */etc/modprobe.d/raspi-blacklist.conf*. Ovladač SPI je zde označen pod názvem spi-bcm2708. Po restartu systému je možné nalézt zařízení *spidev0.** uvnitř adresáře */dev/*, což signalizuje aktivní SPI rozhraní. Blacklist slouží právě k tomu, aby SPI nebylo zbytečně aktivováno, když jej spousta uživatelů Raspberry Pi nikdy nepoužije.

Pro ověření správné funkčnosti SPI je vytvořen testovací zdrojový kód, který odesílá testovací data přes MOSI port a zároveň je přijímá na portu MISO. Stačí tedy pouze propojit GPIO piny 9 a 10 (MISO a MOSI) a spustit testovací program. Ten ověří funkčnost SPI, takže je již

možné pokročit dále při tvorbě ovladače.

Další důležitou částí implementace je seznámení se s GPIO pinout u RPi, který se liší dle revize zařízení. Pro SPI jsou vyhrazeny piny GPIO 9,10 a 11 (MISO, MOSI, SCKL). Levý obrázek vyobrazuje pinout u RPi revize 1.0. Pin 17 ani pin 25 není osazen a na celém konektoru se nachází pouze jeden 3,3V napájení a zem. Obrázek vpravo zobrazuje pinout RPi u revize 2.0. Změn je zde více, ovšem z pohledu SPI jsou nejdůležitější změny GPIO u pinů 17 a 25, které nyní slouží jako napájení 3,3 V a zem.

Vzhledem k tomu, že vlastním obě revize RPi, rozhodl jsem pracovat s novější verzí, která má přívětivější pinout z pohledu SPI.



Ilustrace 7: GPIO u Raspberry Pi (vlevo revize 1.0, vpravo revize 2.0)

7.3.1 Kompilace modulu pro Raspberry Pi

Před samotnou implementací jaderného modulu je potřeba otestovat kompilaci modulů.

K tomu může posloužit jednoduchý tzv. hello-world, který neplní žádnou specifickou roli, pouze se zavede do jádra, vypíše zprávu o tom, že je inicializován. Jeho přítomnost lze ověřit příkazem *lsmod* a při odebrání modulu se korektně uvolní. Ukázkový kód ovladače vypadá následovně:

```
#include <linux/module.h>       /* Needed by all modules */
#include <linux/kernel.h>       /* Needed for KERN_INFO */
#include <linux/init.h>         /* Needed for the macros */

static int __init hello_start(void)
{
       printk(KERN_INFO "Loading hello module...\n");
       printk(KERN_INFO "Hello world\n");
return 0;
}

static void __exit hello_end(void)
{
       printk(KERN_INFO "Goodbye Mr.\n");
}
module_init(hello_start);
module_exit(hello_end);
```

Ilustrace 8: hello.c (ukázkový kód jednoduchého ovladače)

Před samotnou kompilací je nutné zjistit aktuální verzi jádra, která je na zařízení přítomna. Protože i verze jádra se neustále aktualizuje a každá nová verze upravuje velkou řadu chyb a přináší nová vylepšení, doporučuje se před samotnou kompilací aktualizovat všechny repozitáře a provést upgrade jádra na poslední verzi.

To lze v linuxové distribuci Raspbian provést pomocí následujících dvou příkazů:

- *Apt-get update*
- *Apt-get upgrade*

Tyto dva příkazy zajistí, že se na zařízení nachází nejaktuálnější verze jádra. Označení verze lze nalézt pod příkazem *uname –r*. V době psaní této práce (květen 2014) je nejaktuálnější jádro ve verzi 3.12.20+.

Kompilaci jaderného modulu lze provést dvěma způsoby. První možností je kompilace přímo na zařízení, tou druhou je tzv. cross-compiling.

Kompilace na Raspberry Pi

1. Stáhnout zdrojové soubory jádra dle jeho aktuální verze, GCC kompilátor a *Makefile*, který celou kompilaci provede.
2. Vytvořit kopii konfigurace jádra, která poslouží při kompilaci. Ta se nachází zde: */proc/config.gz*.
3. Příkazy *make oldconfig* a *make modules_prepare* slouží k vytažení konfiguračních dat z původní konfigurace jádra, které se připraví k zavedení do nově zkompilované verze.
4. Stáhnout soubor z oficiálního Raspbian repozitáře na serveru github s názvem *Modules.symvers*. Ten obsahuje informace o tom, co vše je potřeba pro kompilaci modulů u nově zkompilovaného jádra.
5. Vytvoření symlinků, které vedou z umístění */lib/modules/[kernel_version]/* k místu, kde jsou stažené zdrojové soubory jádra. Ty jsou obvykle umisťovány do */usr/src/**.
6. Pokud jsou kompilační parametry souboru *Makefile* dobře nastaveny, tak se kompilace jádra nijak neliší od kompilace běžného uživatelského programu.

Tato metoda je jednodušší než cross-compiling, ovšem je časově náročnější. Na Raspberry Pi trvá přibližně 11hodin. Ještě než popíši své zkušenosti s kompilací, tak popíši ještě druhou zmiňovanou metodu.

Kompilace přes cross-compiling

Postup je složitější, celý návod je podrobně popsán serveru elinux[4]. Nyní popíši pouze nejzákladnější body.

1. Kompilace jádra probíhá na jiném linuxovém stroji, je tedy nutné trochu více příprav: Stažení všech potřebných programů, kompilátoru, zdrojových kódů, atp. .
2. Po stažení zdrojových souborů jádra se opět extrahuje konfigurace jádra a za pomoci příkazů níže se konfigurace jádra připraví pro kompilaci:

   make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- oldconfig -j 3

   make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- menuconfig
3. Kompilace je časově mnohem méně náročná (na nejnovější generaci 4 jádrového

[4] http://elinux.org/RPi_CANBus

procesoru i5 cca 45minut). Spuštění kompilace:

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- -j 3

4. Po kompilaci je ještě potřeba stáhnou nástroje, které vytvoří z nově zkompilovaného jádra image, který se nahraje na Raspberry Pi. Tím je cross-compiling úspěšně dokončen.

7.3.2 Kompilace modulu v praxi

Po nastudování možných řešení jsem se rozhodl pro první možnost, tzn. kompilace přímo na vlastním zařízení. Vzal jsem svůj upravený zdrojový kód *Spidev*, který obsahoval dynamickou registraci z driveru *Spike*, a pokusil jsem se jej zkompilovat. V té době jsem měl na Raspberry Pi starší verzi jádra 3.2.27. Kompilace se nezdařila z důvodu nekompatibilních zdrojových souborů. Usoudil jsem tedy, že je načase aktualizovat verzi jádra, a tak jsem provedl upgrade jádra na verzi 3.10.37+ (v tu chvíli nejaktuálnější stabilní verze). A pokus jsem opakoval.

Kompilace proběhla úspěšně, ovšem při pokusu o zavedení ovladače do systému došlo ke kompletnímu zatuhnutí systému. Bylo tedy nutné Raspberry Pi natvrdo restartovat odpojením napájení. Pro zjištění příčin pádu jsem analyzoval výpisy uvnitř souboru */var/log/dmesg* a */var/log/messages*. V těchto souborech ale nebyla žádná informace o zavedení ovladače k nalezení. Po hlubším prozkoumání problému jsem narazil na několik diskuzí na serveru stackoverflow, kde se doporučovalo metodě kompilace přímo na Raspberry Pi vyhnout a raději zvolit spolehlivější cross-compiling. Z tohoto důvodu jsem upustil z této možnosti a raději se pokusil zprovoznit kompilaci způsobem cross-compiling.

Po nainstalování nejaktuálnější verze OS Ubuntu (14.04) do Virtualboxu jsem začal pracovat na přípravě všech potřebných programů a nastavení pro cross-compiling.

I přes složitější přípravu na kompilaci mi zde pomohl návod, kde je krok za krokem přesně popsáno, co je potřeba udělat a za jakým účelem (jak jsem se již zmiňoval při popisu cross-compilingu o stranu výše).

V této fázi jsem měl přístup přímo ke zdrojovým souborům jádra, proto jsem se rozhodl před spuštěním kompilace lehce editovat jaderný modul *Spidev*. Do části inicializace a uvolnění jsem přidal pouze informační výpis do bufferu pomocí jaderné funkce *printk*. Tímto jsem si mohl být jistý, že tato změna nebude mít vliv na žádnou z kritických částí jaderného modulu a navíc si ověřím, že zavedený modul je ten mnou editovaný.

```
/* Module main init */
static int __init spi_miwi_init(void)
{
	. . .
	printk(KERN_INFO "SPI_MIWI_MODULE: Module has been initialized\n");
	return 0;
	. . .
	return -1;
}
module_init(spi_miwi_init);

static void __exit spi_miwi_exit(void)
{
	spi_unregister_device(spi_miwi_dev.spi_device);
	spi_unregister_driver(&spi_miwi_driver);
	. . .
	printk(KERN_INFO "SPI_MIWI_MODULE: Module has been released\n");
}
module_exit(spi_miwi_exit);

MODULE_AUTHOR("Martin Hala <xhalam04@stud.fit.vutbr.cz>");
MODULE_DESCRIPTION("SPI-MiWi Module");
MODULE_LICENSE("GPL");
MODULE_VERSION("0.1");
```

Ilustrace 9: spidev.c (úryvek zdrojového kódu s přidanými výpisy do bufferu)

Po úpravě tohoto ovladače jsem spustil kompilaci jádra a provedl veškeré potřebné kroky. Ze zkompilovaného jádra jsem pouze vyjmul zkompilovaný *Spidev* ovladač (v tomto případě *spi-bcm2708.ko*), zazálohoval jsem původní a na jeho místo vložil upravený. Po odebrání modulu a znovu zavedení ovladač nenaběhl. Navíc se v souboru *dmesg* objevilo následující hlášení:

bcm2708_spi bcm2708_spi.0: master is unqueued, this is deprecated

Zároveň s tímto hlášením zmizelo zařízení *spidev0.** z umístění */dev/*.

Další pokus jsem tedy zkusil s tím, že jsem výsledný image zkompilovaného jádra zkopíroval na SD kartu. Předtím jsem ještě provedl zálohu aktuální verze na vlastní PC. Po zkopírování běžela na Raspberry Pi aktuální verze Raspbianu s nově zkompilovaným jádrem. Vylistováním zavedených modulů pomocí příkazu *lsmod* se ukázalo, že modul je zavedený, ale zařízení *spidev0.** nebylo registrováno mezi zařízeními uvnitř */dev/*. Navíc se neobjevilo ani žádné hlášení v souboru *dmesg* a to ani mnou přidaný informační výpis přes funkci *printk* – viz. Ilustrace 9.

Celý cyklus jsem tedy znovu opakoval ovšem s nejnovější verzí jádra 3.12.20+, ale bohužel s naprosto stejným výsledkem. Ve chvíli, kdy jsem se pokusil upravit jiný jaderný modul než *Spidev*, tak se výpis přidaný funkcí *printk* se objevil uvnitř *dmesg*.

Touto analýzou jsem přišel na problém, proč kompilace nefunguje. Problém nebyl ve verzi jádra ale v samotném návrhu *Spidevu* pro Raspberry Pi. *Spidev.c* není jediný modul, který se kompilací spouští, ale obalují jej další části ovladače jako je *spi.c* a *spi-bcm2708.c*. Z tohoto důvodu se ovladač chová velmi citlivě na jakoukoliv změnu. Žádné bližší informace ohledně kompilace SPI jsem nenašel, stejně tak ani přímou spojitost mezi těmito ovladači. K nalezení bylo pouze několik diskuzí na toto téma, žádné ale neobsahovalo faktické vysvětlení.

7.3.3 Analýza poznatků

Ve výsledku to tedy znamená, že navržené řešení není možné reálně zprovoznit na Raspberry Pi. Teoretická šance by mohla být při vytvoření paralelního ovladače, ten by v sobě ovšem musel zapouzdřovat spojitost mezi ovladačem *Spidev* a již zmiňovaným *spi-bcm2708*, který se navenek tváří jako výsledný driver – obsah zdrojového kódu ale nenaznačuje žádnou spojitost s ovladačem *Spidev*. Další z možností by bylo vytvořit jakýsi wrapper, který by byl mezičlánkem mezi *Spidevem* a uživatelskou aplikací. Toto by muselo být realizováno na úrovni jádra a to znamená, že by jednotlivé funkce v rámci *Spidev* musely být implementovány jako EXPORT_SYMBOL[5] a to by znamenalo opět zásah do původního kódu *Spidevu*, což není možné.

Například podpora SPI pro python je realizována pomocí wrapperu na úrovni userspace. Zda je tomu tak z důvodu problémů, spojených s editací ovladače *Spidev,* se mohu pouze domnívat.

[5] http://stackoverflow.com/questions/413955/how-to-write-linux-driver-module-call-use-another-driver-module

Kapitola 8

Návrh možného řešení

Nelze tvrdit, že tvorba SPI ovladače pro Raspberry Pi je nereálná. V praxi to znamená nejen důkladné prostudování kompletní hierarchie kompilace ovladače, ale i vytvoření zcela samostatné větve ovladače, která ke své praci nevyžaduje ovladač *Spidev*. Ale ani tento postup ještě neznamená zaručený úspěch a pravděpodobnost, že čas strávený nad tímto řešením bude mnohem vyšší, než jít zcela jinou cestou, je dost vysoká.

Pokud bych měl shrnout dosavadní poznatky, tak nejlepším řešením by bylo se pokusit o vytvoření SPI ovladače pro jiné embedded linux zařízení, ovšem pod podmínkou analýzy, zda se nenarazí na obdobný problém při tvorbě či editaci ovladače SPI rozhraní. Zařízení Raspberry Pi jsem si totiž pro tuto práci vybral sám. Od mého vedoucího jsem na prvních konzultacích dostal nabídku vytvářet řešení pro Olinuxino[6], za kterým stojí větší komunita vývojářů včetně mého vedoucího a jeho kolegů, kteří by mi v případě nesnází mohli poradit. Tvorba SPI ovladačů by tak mohla být snažší cestou.

Ovšem jako nejjednodušší řešení se jeví implementovat komunikační protokol na úrovni userspace, tak jak to například nabízí py-spidev[7] wrapper, který zpřístupňuje ovladač Spidev pro programy napsané v pythonu. Protokol je tak mimo jaderný modul, čímž nijak nelimituje samotný driver a použije se jen v případě potřeby.

[6] https://www.olimex.com/Products/OLinuXino/open-source-hardware
[7] https://github.com/doceme/py-spidev

Kapitola 9

Závěr

V rámci zimního semestru jsem nastudoval protokol MiWi a seznámil se s vývojovým prostředím MPLAB. Vyzkoušel jsem si práci s obvodovým debuggerem a programátorem MPLAB ICD 3. Dále jsem se seznámil s tvorbou ovladačů pro vestavěný systém Linux a vybral vhodného kandidáta. Navrhl jsem hardwarový modul, který jsem otestoval a na základě poznatků vytvořil novou verzi. Po úspěšném oživení a otestování HW modulu jsem upravil software pro podporu komunikace mezi ním a Raspberry Pi.

Po této části jsem začal pracovat na návrhu a tvorbě ovladače pro Linux. V poslední kapitole jsem shrnul dosažené znalosti, které jsem nabyl během analýzy problému a nastínil možná řešení, která by vedla k vytvoření funkčního ovladače.

V rámci této práce jsem byl detailně seznámem s protokolem MiWi, se psaním jaderných ovladačů a s problematikou během kompilace.

Vytvořený HW modul má velký potenciál uchytit se v praxi a to hlavně kvůli nižším pořizovacím nákladům a vcelku jednoduché editaci programů z MiWi demo kitu. Samotná komunikace mezi embedded Linuxem a HW modulem se dá obejít vytvořením wrapperu na úrovni userspace, který bude mít v sobě integrovaný navržený komunikační protokol. Toto řešení by navíc disponovalo větším potenciálem přenositelnosti, protože ovladač je specifickou záležitostí pro každý konkrétní hardware. Ovladač *Spidev* lze běžně najít na mnoha single-board PC s podporou SPI rozhraní.

Literatura

[1] BRŮHA, Petr. Jak snadno sestavit projekt a realizovat „Inteligentní dům“. *Jak snadno sestavit projekt a realizovat „Inteligentní dům“* [online]. 2008 [cit. 2014-01-13]. Dostupné z: http://www.techpark.sk/technika-782009/jak-snadno-sestavit-projekt-a-realizovat-inteligentni-dum.html

[2] *OBZOR - Bezdrátové ovládání RF Home* [online]. [cit. 2014-01-14]. Dostupné z: http://www.obzor.cz/cz/2-vyrobky/2-bezdratove-ovladani-rf-home.html

[3] BECHYNSKÝ, Štěpán. Internet of Things. [online]. 9.7.2012 [cit. 2014-01-14]. Dostupné z: http://www.zdrojak.cz/clanky/internet-of-things/

[4] Smarthome Architecture. *IoT* [online]. 2013 [cit. 2014-01-14]. Neveřejně dostupné z: https://merlin.fit.vutbr.cz/wiki-iot/index.php/Smarthome_architecture

[5] KETTIMUTHU, Rajkumar a Sankara MUTHUKRISHNAN. Is Bluetooth suitable for large-scale sensor networks?. In: [online]. Bangalore [cit. 2014-01-14]. Dostupné z: http://www.mcs.anl.gov/~kettimut/publications/icwn05.pdf

[6] FLOWERS, David a Yifeng YANG. MiWi Wireless Networking Protocol Stack: AN1066. [online]. 2011 [cit. 2014-01-14]. Dostupné z: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1824&appnote=en520606

[7] Raspbian [online]. [cit. 2014-05-27]. Dostupné z: http://www.raspbian.org/

[8] Linus Torvalds - GitHub. [online]. [cit. 2014-05-27]. Dostupné z: https://github.com/torvalds

[9] *Kernel Space - User Space Interfaces*. KELLER, Ariane. [online]. 2008 [cit. 2014-05-27]. Dostupné z: http://people.ee.ethz.ch/~arkeller/linux/kernel_user_space_howto.html

[10] CORBET, Jonathan. *Linux device drivers*. 3rd ed. Sebastopol: O´Reilly, 2005, xviii, 615 s. ISBN 05-960-0590-3.

[11] Spidev documentation. [online]. [cit. 2014-01-14]. Dostupné z: https://www.kernel.org/doc/Documentation/spi/spidev

[12] MiWi Demo Kit – 2.4 GHz MRF24J40. [online]. 2012 [cit. 2014-01-14]. Dostupné z: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en559628

[13] Raspberry Pi FAQs. [online]. [cit. 2014-05-27]. Dostupné z: http://www.raspberrypi.org/faqs

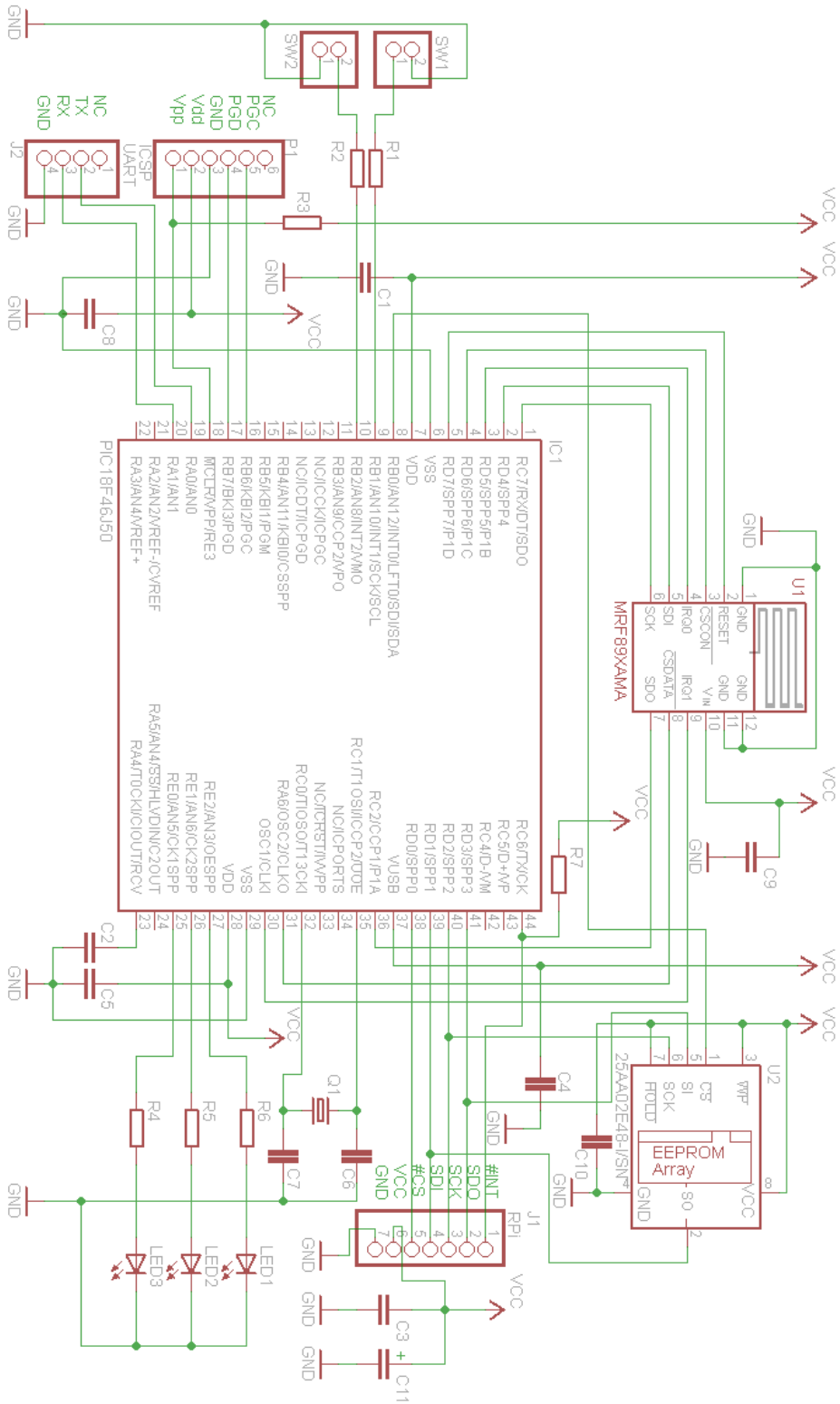
[14] MRF89XAM8A Data Sheet: 868 MHz Ultra-Low Power Sub-GHz Transceiver Module. [online]. 2010 [cit. 2014-01-14]. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/70651A.pdf

[15] HENDERSON, Gordon. Gordons Projects: Understanding SPI on the Raspberry Pi. [online]. 2012 [cit. 2014-01-14]. Dostupné z: https://projects.drogon.net/understanding-spi-on-the-raspberry-pi/

[16] *Writing a Linux SPI driver* [online]. 14.4.2011 [cit. 2014-05-27]. Dostupné z: [1] http://108.163.188.242/~jumpnowt/index.php?option=com_content&view=article&id=57&Itemid=62

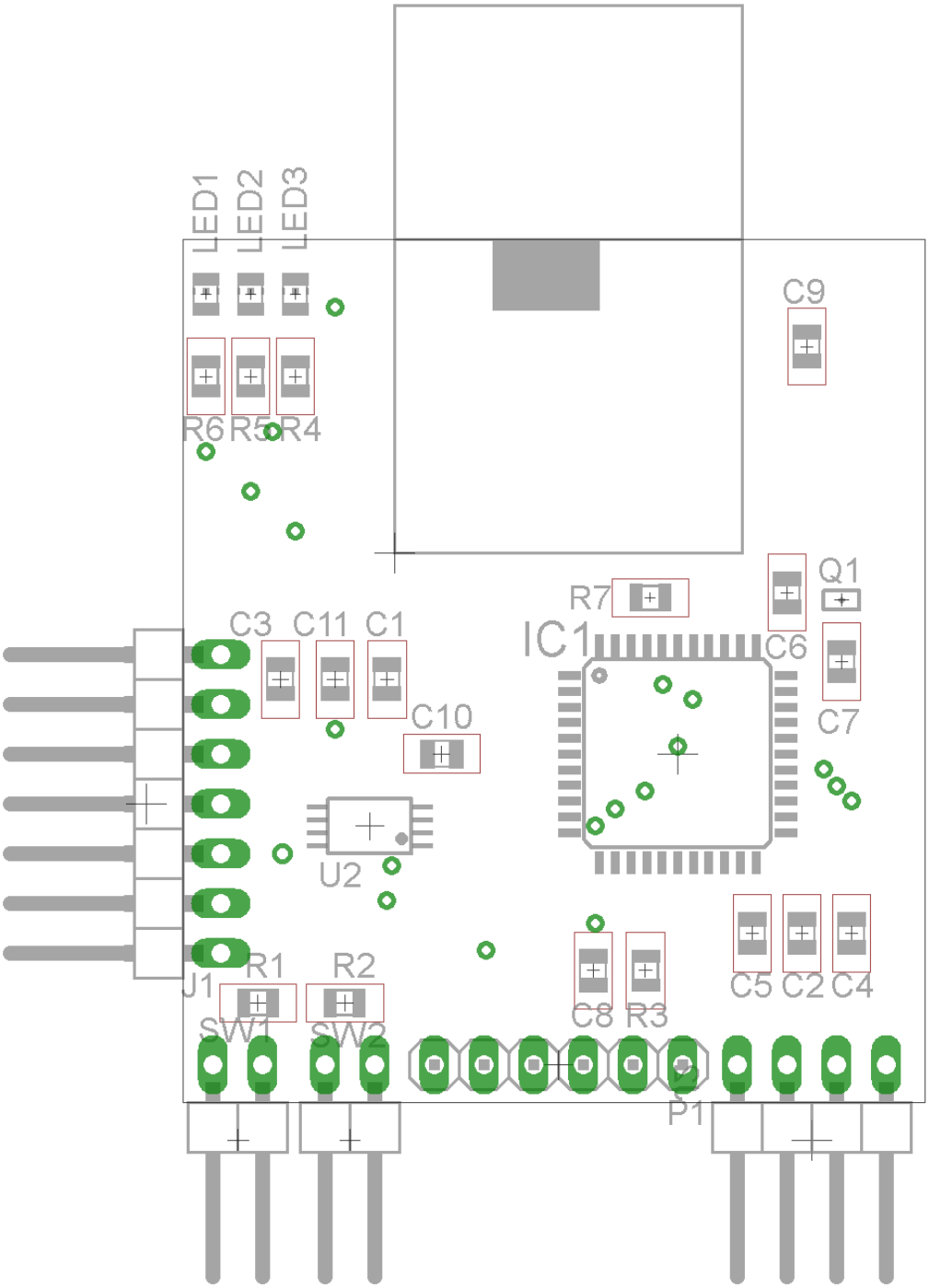
Příloha A

Schéma HW modulu – revize 1



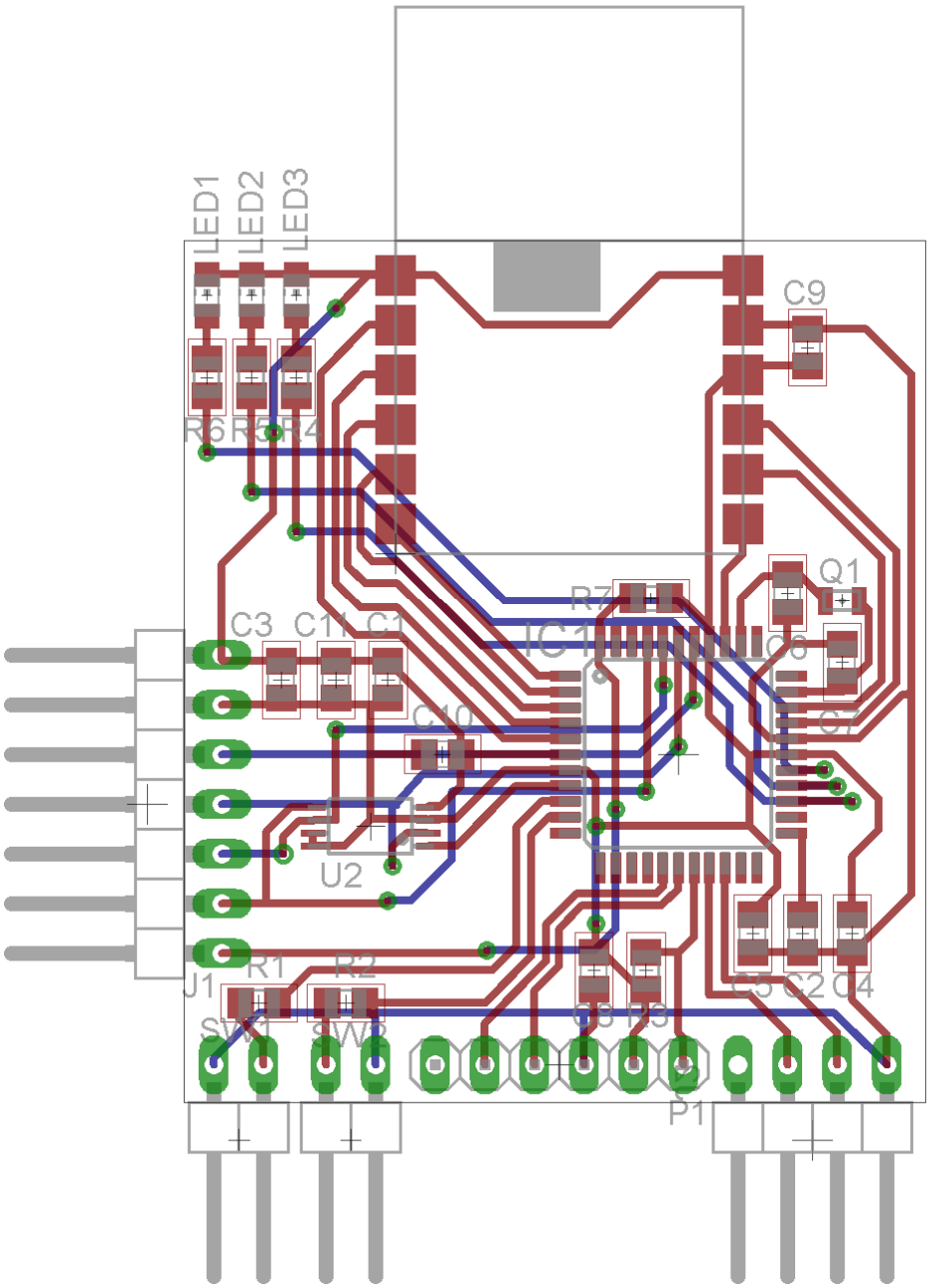
Příloha A

Osazovací plán HW modulu – revize 1



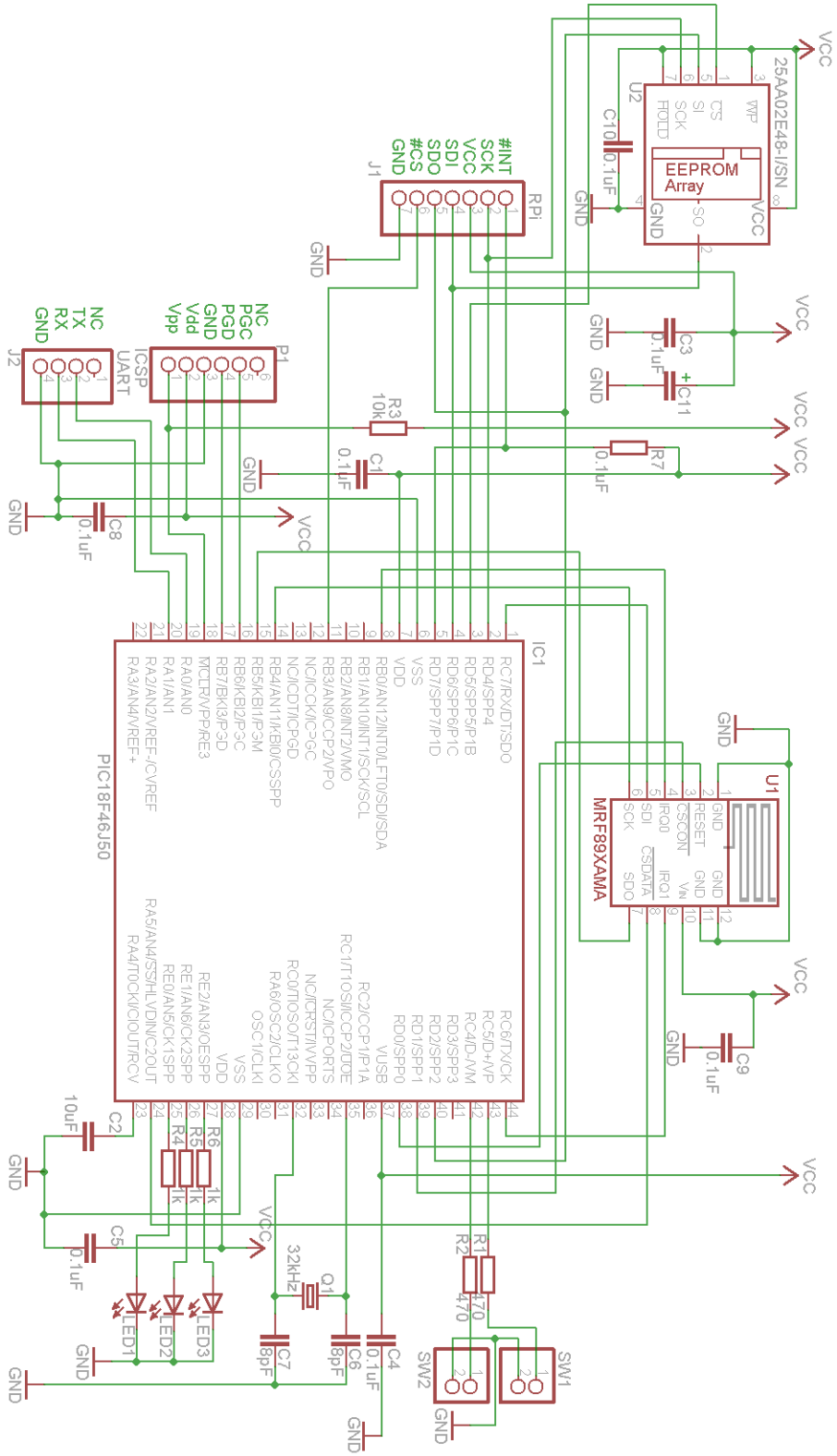
Příloha A

DPS HW modulu – revize 1



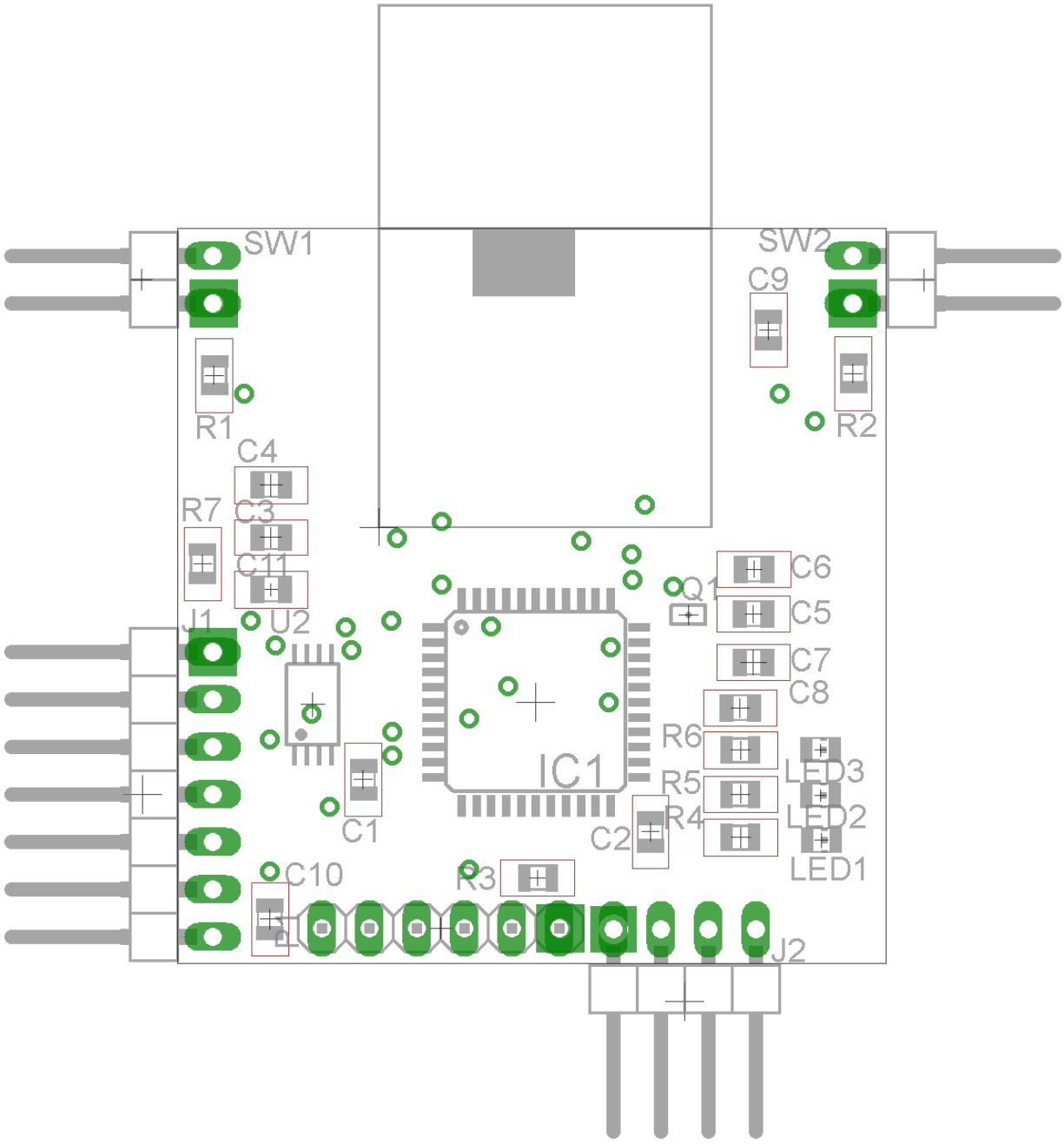
Příloha B

Schéma HW modulu – revize 2.1



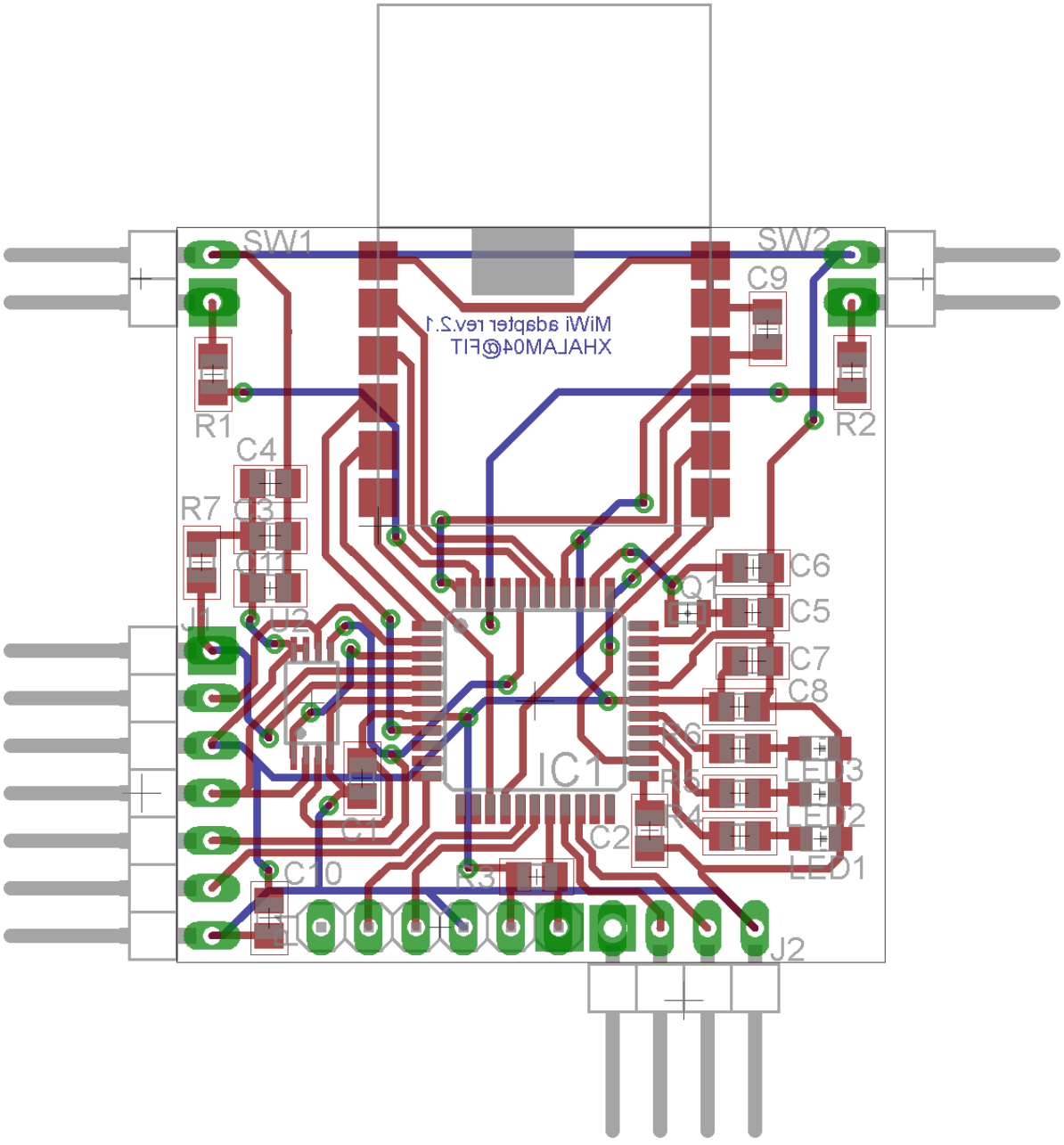
Příloha B

Osazovací plán HW modulu – revize 2.1



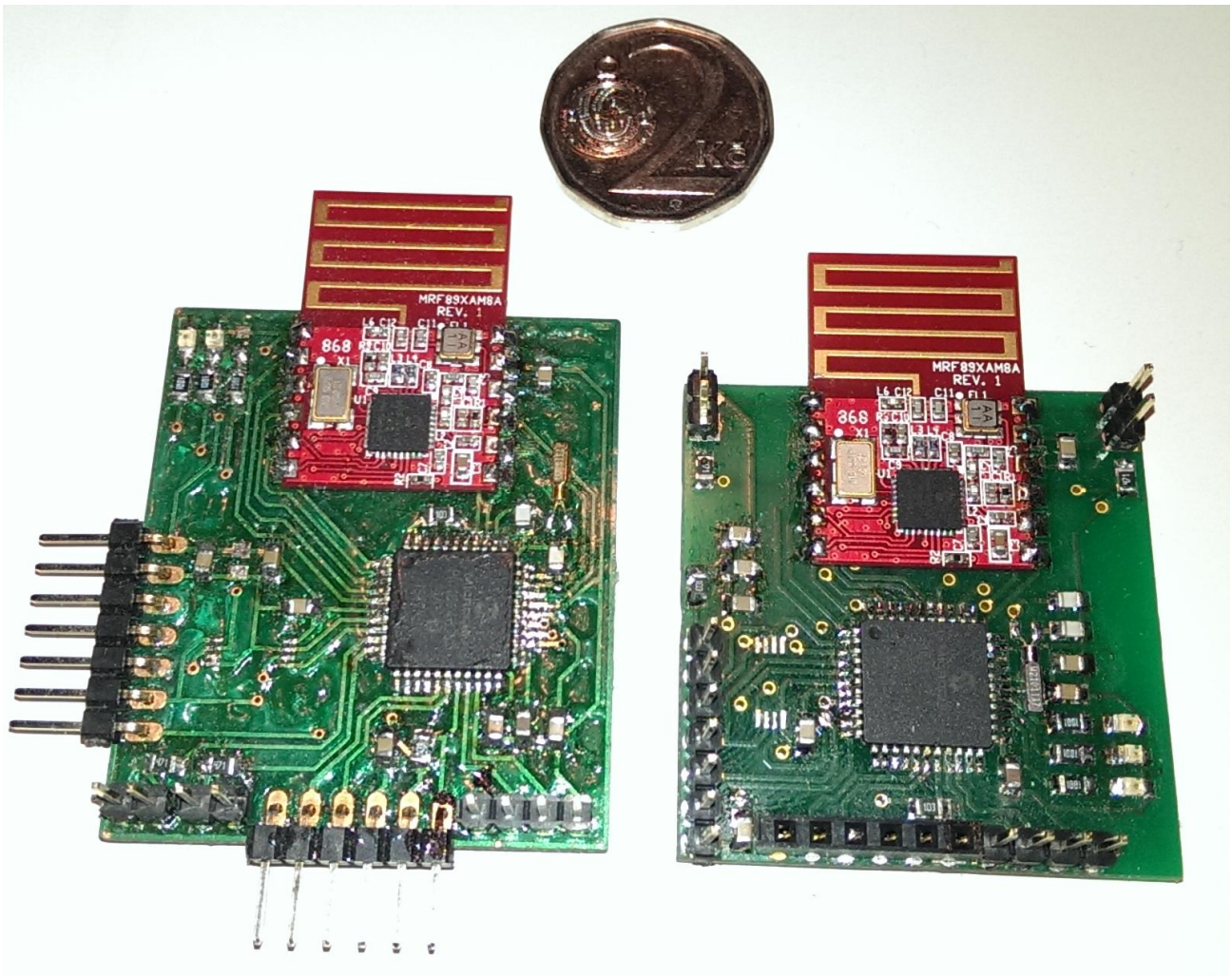
Příloha B

DPS HW modulu – revize 2.1



Příloha C

Porovnání HW modulů rev. 1 a rev. 2.1



Příloha D

Zapojení Raspberry Pi a HW modulu