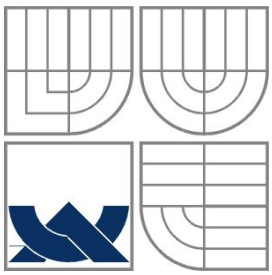
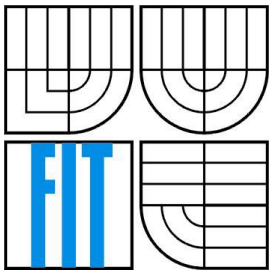


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

FIRMWARE PRO VESTAVNÝ VYSÍLAČ A PŘIJÍMAČ PRO BEZDRÁTOVOU KOMUNIKACI V SRD PÁSMU

EMBEDDED TRANSMITTER AND RECEIVER FIRMWARE FOR WIRELESS COMMUNICATION IN SRD BAND

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN VOMÁČKA

VEDOUCÍ PRÁCE
SUPERVISOR

ING. JOSEF STRNADEL, PH.D.

BRNO 2013

Abstrakt

Tato práce se věnuje problematice bezdrátového přenosu v bezlicenčním ISM pásmu. Cílem bylo navrhnout a implementovat firmware vysílače a přijímače pro použití v drobnějších uživatelských projektech. Pro bezdrátovou komunikaci jsou použity moduly značky Aurel, které pracují v SRD pásmu 868 MHz. Vysílač a přijímač je ovládán mikrokontrolérem MSP430 osazeným na laboratorním přípravku FITkit.

Abstract

This bachelor's thesis analyzes wireless communication in range of unlicensed ISM band. The aim was to design and implement a transmitter and reciever firmware for utilization in minor custom projects. Wireless communication was secured by Aurel modules operating in the 868 MHz SRD band. The transmitter and the receiver are controlled by MSP430 microcontroller mounted on the FITkit laboratory equipment.

Klíčová slova

SRD pásmo, MSP430, FITkit, Manchester, Hammingův kód, firmware, bezdrátová komunikace

Keywords

SRD band, MSP430, FITkit, Manchester, Hamming code, firmware, wireless communication

Citace

Martin Vomáčka: Firmware pro vestavný vysílač a přijímač pro bezdrátovou komunikaci v SRD pásmu, bakalářská práce, Brno, FIT VUT v Brně, 2013

Firmware pro vestavný vysílač a přijímač pro bezdrátovou komunikaci v SRD pásmu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Josefa Strnadela, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Vomáčka
15.05.2013

Poděkování

Velice děkuji vedoucímu bakalářské práce Ing. Josefu Strnadelovi, Ph.D. za cenné rady a nápady při návrhu implementace, stejně tak děkuji mé rodině za psychickou podporu při studiu.

Obsah

Obsah....1
1 Úvod....3
2 ISM pásmo....4
2.1 Bluetooth....4
2.1.1 Specifikace....5
2.1.2 Spojení zařízení....6
2.2 Wi-Fi....6
2.2.1 Přehled nejpoužívanějších standardů....7
2.2.2 Topologie a struktura sítě....7
2.2.3 Rozdělení kanálů pásem....8
2.2.4 Zabezpečení sítí....8
2.3 SRD860....8
2.4 Modulace signálu....10
2.4.1 Modulace digitálního signálu....10
2.4.2 Modulace analogového signálu....11
3 Implementace firmware....12
3.1 Kódování Manchester....12
3.2 Přenosová rychlost....12
3.3 Protokol pro komunikaci....13
3.4 Instrukční kódy....13
3.5 Hammingův kód....14
3.5.1 Kódování paketu firmwaru....15
3.6 Popis funkcí firmwaru vysílače....15
3.6.1 Příprava uživatelských dat....16
3.6.2 Příprava paketu....16
3.6.3 Odeslání paketu....17
3.7 Popis funkcí firmwaru přijímače....18
3.7.1 Naslouchání a příjem dat....18
3.7.2 Dekódování paketu....18
3.7.3 Zpracování uživatelských dat....19
3.8 Možnosti rozšiřitelnosti implementace....20
3.8.1 Rozšíření hlavičky paketu....20
3.8.2 Rozšíření Hammingova kódování – SECDED....20
3.8.3 Obousměrná komunikace....21

4 Hardware a testování....22
4.1 Vysílací modul....22
4.2 Přijímací modul....22
4.3 Testování implementace....23
4.3.1 Rádiový dosah přenosu....24
4.3.2 Úspěšnost přenosu....24
4.3.3 Samostatné testy přenosu....25
5 Závěr....26
Literatura....27
Seznam příloh....28
A. Fotodokumentace přijímače....29
B. Fotodokumentace vysílače....30
C. Algoritmus přijímače načítání z modulu....31
D. Tabulka hodnot z testování přenosu....33
E. Uživatelská příručka....34

1 Úvod

Bezdrátové technologie jsou v dnešní době již nedílnou součástí našich životů. Od nejjednodušších zařízení jako například dálkový ovladač nebo bezdrátové čidlo teploměru až po například mobilní telefony, inteligentní domácnosti nebo bezdrátové počítačové sítě. Co bezdrátové technologie občas postrádají třeba na kvalitě nebo rychlosti přenosu, to v mnoha případech snadno kompenzuje onen fakt bezdrátovosti, tedy absence nutnosti propojování pomocí kabelů a s nimi spojených problémů – cena materiálu při velkých vzdálenostech nebo enviromentální překážky.

Tento projekt je zaměřen na implementaci dvou firmwarů, pro vysílač a přijímač, v takové podobě, aby mohly být univerzálně použity pro jednoduché aplikace vyžadující bezdrátový přenos, jako například snímaní hodnot z čidel nebo dálkové ovládání jiných zařízení.

Bezdrátová komunikační pásma lze rozdělit na licenční a bezlicenční. Protože se v tomto projektu budeme věnovat bezlicenčnímu pásmu ISM, konkrétně pásmu SRD, pojednává druhá kapitola o nejrozšířenějších komunikačních standardech provozovaných v ISM pásmu a shrnuje jejich vlastnosti a použití. Dále také stručně informuje o modulačních technikách digitálních signálů a jejich výhodách a nevýhodách.

Třetí kapitola podrobně popisuje metody a algoritmy použité při návrhu firmwaru, komunikační protokol a zvolený způsob zabezpečení komunikace. Také je zde popsána funkčnost firmwaru vysílače a přijímače a konec kapitoly pojednává o možnostech rozšiřitelnosti firmwaru, možných omezeních, na které je možné v současné implementaci narazit, a případných variantách řešení těchto omezení.

Zapojení komunikačních modulů a nutnosti pro jejich správný provoz jsou popsány v kapitole páté. V této kapitole jsem se také zaměřil na otestování výsledné implementace, abych zjistil, v jakých podmínkách je možné komunikaci provozovat a jak se projevují vlivy prostředí na kvalitě a úspěšnosti přenosu.

V závěrečné kapitole jsou shrnuté veškeré poznatky z implementace a testování navrženého firmware.

2 ISM pásmo

Zkratka ISM vznikla z anglického Industry-Science-Medical a označovala tak rozsah rádiové frekvence, pro využití v průmyslu, vědě a medicíně, vyjma komunikace. Zařízení operující v tomto pásmu mohou generovat vysoké výkony a bylo tedy zapotřebí zavést určitá omezení pro jejich provoz, aby nedocházelo k vzájemnému rušení nebo rušení jiné rádiové komunikace. Pro různé druhy zařízení byly přiděleny určité části pásma, kde každé podléhá svým pravidlům. Například v počtu dostupných kanálů v daném pásmu, v maximálním vyzářeném výkonu nebo ve vysílacím klíčovacím poměru (podíl času, kdy vysílač vysílá na nosném kmitočtu, v rámci jedné hodiny). Z pohledu výkonových zařízení lze najít zástupce, jako mikrovlnná trouba nebo lékařské zařízení pro diathermii, kde se rádiové záření používá pro tepelnou terapii pacientů.

V posledních letech však velmi narostla poptávka po bezdrátové komunikaci a proto byla některá ISM pásma uvolněna pro tyto účely bez nutnosti vlastnit licenci pro jejich využití. Mezi asi nejznámější lze bezesporu zařadit technologie Bluetooth nebo Wi-Fi, které jsou velmi snadno dostupné široké veřejnosti.

Vzhledem k regulačním podmínkám CEPT [6] je celosvětově dostupných hned několik bezlicenčních ISM pásem. SRD pásmo, které je využito při tvorbě tohoto projektu, je oproti ostatním ve výhodě hlavně z důvodu nízké obsazenosti. V tabulce 2.1 je vidět porovnání a využití některých bezlicenčních ISM pásem.

ISM pásmo	využití	výhody	nevýhody
150-160 MHz	dálkově ovládaná zařízení, zabezpečení	velký dosah v zastavěné oblasti	velké vytížení pásma
433-450 MHz	dálkově ovládané zámky automobilů, garáží, čidla v domácnostech	střední dosah v zastavěné oblasti	střední vytížení pásma
868 MHz	senzory, dálková ovládání, RFID	střední dosah, malé vytížení pásma	dostupnost hardware
2,4 GHz	Bluetooth, Wi-Fi 2,4 GHz	nízká cena, vysoká dostupnost, infrastruktura, výborná HW i SW podpora	menší dosah bez použití směrovacích antén, vysoká hustota provozu
5,8 GHz	Wi-Fi 5 GHz	zatím méně vytížené pásmo, vyšší přenosové rychlosti než u 2,4 GHz	menší dostupnost zařízení a SW podpory

Tabulka 2.1: Bezlicenční pásma ISM

2.1 Bluetooth

Technologie Bluetooth je bezpochyby nejrozšířenější a nejdostupnější technologií pro bezdrátovou komunikaci na krátké vzdálenosti pro dvě a více zařízení. Jeho první veřejně dostupná verze (1.0a) byla poprvé prezentována v roce 1999 skupinou BSIG (Bluetooth Special Interest Group), založenou roku 1998 pěti firmami – Ericsson, IBM, Intel, Nokia a Toshiba. Hlavním zadavatelem tvorby této technologie byla již v roce 1994 firma Ericsson. Ihned po uveřejnění si tato technologie našla oblibu nejen u běžných uživatelů, ale také v průmyslových aplikacích. Skupina BSIG byla následně

rozšířena o mnoho dalších firem a v dnešní době je v této skupině přes desetitisíce členů. Zařízení Bluetooth lze v dnešní době najít nejen v mobilních telefonech, ale také v dalších zařízeních vyžadujících komunikaci na krátkou vzdálenost, jako například počítačové příslušenství nebo audio systémy [4].

Díky jeho nízké ceně, rozměrům a všeobecné rozšířenosti je možné Bluetooth najít v široké škále odvětví, včetně průmyslového nasazení.

2.1.1 Specifikace

Technologie Bluetooth byla přesně specifikována standardem *IEEE 802.15.1* v roce 2002. Tato specifikace popisuje Bluetooth v1.1, která opravovala mnoho chyb nalezených v předchozích původních verzích 1.0a a 1.0b. V současnosti je několik verzí Bluetooth, vždy zpětně kompatibilních s verzí předešlou. Nejvíce rozšířenou verzí je Bluetooth 2.0.

Z tabulky 2.1 je zřejmé, že zařízení Bluetooth pracují v ISM pásmu 2,4 GHz. Jelikož toto pásmo má velmi vysoký provoz, je pro komunikaci mezi zařízeními používána metoda FHSS[1], kdy při komunikaci dochází k přepínání mezi 79 kanály, kde každý zabírá šířku pásma 1 MHz a přepínání je prováděno minimálně 2,5krát za sekundu. Zařízení Bluetooth pracují s přepínáním 1600krát za sekundu. Takový mechanismus by měl být dobře odolný proti rušení na stejných frekvencích. K dispozici jsou 3 výkonové úrovně, které se liší maximálním možným vyzářeným výkonem a s tím souvisejícím přibližným dosahem - viz. tabulka 2.2. Je však nutné dodat, že uvedený dosah je uveden pro volný prostor. V zastavěné oblasti se dosah rychle zmenšuje, především z důvodu chybně doručených paketů.

	Maximální vyzářený výkon [mW]	Přibližný dosah [m]
Třída 1	100	~100
Třída 2	2,5	~20
Třída 3	1	~1

Tabulka 2.2: Výkonnostní třídy Bluetooth

Rychlosti přenosu dat se pohybují od 1 Mbit/s u BT v1.2 až po 24 Mbit/s u BT v3.0. Tyto rychlosti jsou však teoretické a reálná rychlost se pohybuje průměrně v 0,7 násobku těchto teoretických rychlostí. V následujícím seznamu jsou uvedeny 3 nejvíce známe a používané verze Bluetooth.

- **Bluetooth v1.2** - značné vylepšení základu, který položil standart IEEE u verze 1.1. Mezi hlavní vylepšení patřilo rychlejší vyhledání a navázání spojení, vyšší přenosové rychlosti, vylepšení hlasových přenosů pomocí synchronních přenosů a také přidání technologie AFHSS – adaptivní výběr vhodných frekvencí pro skákání.
- **Bluetooth v2.0 + EDR** - novinkou je rozšíření EDR (Enhanced Data Rate), umožňující přenosové rychlosti až 2,2 Mbit/s. Vzápětí na to byla vydána ještě vylepšená verze 2.1, která přinášela rychlejší a efektivnější vyhledávání zařízení a jejich filtrování a bezpečnější párování zařízení. Bohužel tato inovovaná verze nebyla implementována ve velkém množství zařízení a verze 2.0 je oproti ní značně rozšířenější.
- **Bluetooth v3.0 + HS** – opět oproti předchozím verzím dochází k nárůstu přenosové rychlosti až na hodnotu 24 MBit/s, avšak pro dosažení takových rychlostí už nedochází čistě přes Bluetooth zařízení, ale komunikaci už probíhá pomocí Wi-Fi – zařízení Bluetooth obstará pouze navázání spojení. Zařízení podporující tuto schopnost nesou ono označení „+HS“, bez tohoto označení není možné dosáhnout vysokých přenosových rychlostí.

1 *Frequency Hopping Spread Spectrum* - jedna z metod přenosu v rozprostřeném spektru. Její princip spočívá v přepínání mezi několika frekvencemi při přenosu bitu nebo bitů.

Protože Bluetooth je architekturová vrstva, tvoří ji několik komunikačních protokolů. Mezi ty povinné patří protokoly LMP, L2CAP, SDP a dále také dva všeobecné používané protokoly HCI a RFCOMM.

LMP	– stará se o samotný rádiový přenos
L2CAP	– využívá protokolů vyšší úrovně k navázání logického spojení mezi dvěma zařízeními
SDP	– slouží ke zjištění jednotlivých služeb, které nalezená zařízení podporují
HCI	– zajišťuje standard komunikace mezi samotným řadičem Bluetooth a hostitelem (počítač, mobilní telefon)
RFCOMM	– zajišťuje virtuální sériové datové spojení, podobné RS-232 a obstarává jednoduchý a spolehlivý tok dat, jako například TCP protokol.

2.1.2 Spojení zařízení

Zařízení v síti Bluetooth pracují na principu Master-Slave. Jedno zařízení je vždy nadřazené – Master a ostatní jsou podřazené – Slave. Vznikají tak takzvané pikosítě, kde Master řídí tok dat. V jedné pikosíti může být celkem až 255 zařízení, přičemž těch aktivních může být zároveň až 8 (včetně Master). Ostatní neaktivní zařízení jsou v pasivním módu. Při propojení pouze dvou zařízení mluvíme o Point-to-Point spojení (nebo také Mono Slave), při třech a více zařízeních se jedná o Multi-to-Point spojení (nebo také Multi Slave).

Tyto vytvořené sítě je možné kombinovat do tzv. Scatternet, což je možné si představit jako síť pikosítí, tvořící ve své podstatě stromovou strukturu. Víme, že v každé pikosíti musí být jeden Master, avšak tento Master může být zase v rámci jiné pikosítě jako Slave. Sítě scatternet se však příliš nepoužívají.

Každé zařízení Bluetooth má svou unikátní 48 bitovou adresu, podle které je možné zařízení identifikovat. Tato adresa nebývá standardně pro uživatele viditelná, proto dochází k pojmenovávání zařízení uživatelem pro snadnější identifikaci jednotlivých zařízení. Všechna zařízení, která jsou v režimu viditelnosti, tedy jsou pomocí Bluetooth vyhledatelná, na vyžádání poskytují čtyři základní informace. K této výměně informací dochází hlavně při pokusu o navázání spojení se zařízením, se kterým komunikujeme poprvé – je neznámé. Jedná se o tyto informace:

- **Název zařízení**
- **Typ zařízení** (třída)
- **Seznam dostupných služeb**
- **Technické informace o zařízení** (výrobce, podporovaná BT specifikace, funkce apod.)

Pokud se pokouší zařízení navázat komunikaci se zařízením známým, dojde přímo k pokusu o navázání spojení. Některé služby mohou vyžadovat párování, nebo potvrzení spojení uživatelem. Jedná se o formu zabezpečení komunikace a zabránění případného ovládání Bluetooth zařízení jiným uživatelem. Při párování dochází mezi dvěma zařízeními k výměně párovacích klíčů, které je možné uložit pro příští spojení. Pokud je pak znova požadováno propojení těchto dvou zařízení, není již potřeba je znovu párovat a automaticky dojde k propojení. Párování je velmi hojně používáno hlavně u audio zařízení, jako jsou sluchátka nebo headsety, kdy se nastaví párování společně s automatickým připojením, které zajišťuje okamžité propojení se spárovaným zařízením v momentě, kdy je zařízení v dosahu Bluetooth.

2.2 Wi-Fi

Zkratka Wi-Fi stojí za standardem pro bezdrátové lokální sítě (WLAN). Zajišťuje vzájemné bezdrátově propojení přenosných zařízení a jejich propojení s drátovými lokálními sítěmi (LAN) a

také připojení těchto zařízení k Internetu. Veškeré specifikace Wi-Fi jsou popsány ve standardu IEEE 802.11 a jeho více než dvaceti doplňcích. Wi-Fi pracuje ve dvou ISM pásmech – původní specifikace používala nejprve pásmo 2,4 GHz (stejně jako Bluetooth). Postupem času a také snahou vylepšování Wi-Fi se začalo používat také pásmo 5 GHz, převážně také z důvodu velké hustoty provozu v pásmu 2,4 GHz. První oficiální specifikace IEEE 802.11 byla vydána roku 1997, pracovala s již zmíněným pásmem 2,4 GHz a přenosové rychlosti se pohybovaly do 2 MBit/s [3].

V dnešní době lze Wi-Fi najít v obrovské škále zařízení, počínaje přenosnými počítači a mobilními telefony, přes zabezpečovací systémy (CCTV kamery, snímače, senzory) až po dálkově ovládané domácí spotřebiče a jejich integrace do inteligentních domácností. Velká obliba Wi-Fi však sebou přinesla také bezpečnostní rizika v podobě nevyžádaných nabourávání do Wi-Fi sítí. Z toho důvodu se několik dodatků standardu IEEE 802.11 věnuje pro zvýšení bezpečnosti také zabezpečení a šifrování síti.

2.2.1 Přehled nejpoužívanějších standardů

Od prvního standardu uvedeného roku 1997 bylo vydáno mnoho jeho dalších doplňků. Úpravy propustností pásma a maximálních přenosových rychlostí, využití pásma 5 GHz, dodatky týkající se zabezpečení a šifrování pásem, správou sítí a nebo řízení QoS. Tabulka 2.3 ukazuje přehled těch nejvíce známých dodatků, které Wi-Fi sítě ovlivnili především použitým ISM pásmem a maximálními přenosovými rychlostmi.

Standard IEEE	Pásmo ISM [GHz]	Max. přenosová rychlost [Mbit/s]	Druh modulace signálu
IEEE 802.11	2,4	2	DSSS[2]
IEEE 802.11a	5	54	OFDM[3]
IEEE 802.11b	2,4	11	DSSS
IEEE 802.11g	2,4	54	OFDM
IEEE 802.11n	2,4 / 5	540	OFDM / MIMO[4]

Tabulka 2.3: Přehled standardů IEEE 802.11

2.2.2 Topologie a struktura sítě

Hlavním prvkem pro vytvoření bezdrátové sítě mezi dvěma a více zařízeními je identifikátor sítě, tzv. SSID (Service Set Identifier). Jedná se řetězec až 32 znaků v ASCII podobě, kterým je možné jednotlivé sítě identifikovat. Rozlišujeme dva druhy sítí - *Ad-hoc*, kdy dochází ke spojení pouze dvou zařízení mezi sebou, a dále *Infrastruktura*, kde dochází k propojení více zařízení pomocí přístupových bodů, tzv. Access Points (AP). SSID každé sítě je v pravidelných intervalech vysíláno pomocí broadcast, tudíž zájemci o připojení mohou tímto způsobem sítě vyhledat a připojit se k nim.

Pokud propojujeme pouze dvě zařízení (Ad-hoc režim), jedná se o pouhé peer-to-peer spojení. Pro navázání spojení je zapotřebí, aby obě zařízení byly ve vzájemném rádiovém dosahu a obě zařízení používaly stejné SSID.

2 *Direct Sequence Spread Spectrum* - technika přímého rozprostřeného spektra. Je jednou z metod pro rozšíření spektra při bezdrátovém přenosu dat.

3 *Orthogonal Frequency Division Multiplexing* - jedná se o přenosovou techniku pracující s tzv. rozprostřeným spektrem, kdy je signál vysílán na více nezávislých frekvencích

4 *Multiple-input multiple-output* - abstraktní matematický model pro multi-anténní komunikační systémy. Významný nárůst datové propustnosti a dosahu při zachování šířky pásma a celkového výdeje vyzařovací energie.

V případě Infrastruktury hrají hlavní roli přístupové body, které obstarávají komunikaci s připojenými zařízeními. Vysílají SSID své sítě a umožňují přístup více zařízením. Více různých AP může vysílat stejné SSID se stejným nastavením, což je vhodně využito ve rozsáhlejších řešeních, kdy je potřeba Wi-Fi signálem pokrýt větší prostor a jeden AP by na toto nestačil. Zařízení připojující se do této sítě tak mohou, bez nutnosti ručního připojování se k nové síti, přihlásit z různých míst a k různým AP, ale stále tak budou přistupovat do stejné sítě. Umožňuje to také zařízení pohybovat se ve větším prostoru a podle síly signálu jednotlivých AP se tak připojit vždy k tomu nejvhodnějšímu. Jedná se o tzv. roaming.

2.2.3 Rozdělení kanálů pásem

Aby nedocházelo k rušení jednotlivých sítí, obzvláště pokud se v jedné lokalitě nachází sítí několik, jsou pásma 2,4 GHz a 5 GHz rozdělena na jednotlivé kanály [5].

Pásmo 2,4 GHz je rozděleno celkem na 14 kanálů. Počet použitých kanálů se liší. Pro Ameriku a Kanadu je k dispozici 11 kanálů, pro Evropu 13 a pro Japonsko 14. Podle generální licence ČTÚ R/12/08.2005-34 [5] je pásmo rozděleno na celkem 13 kanálů s frekvenčním rozestupem 5 MHz v rozmezí 2,412–2,472 GHz. Protože je pásmo 2,4 GHz mnohem více využívané oproti 5 GHz, v místech s velkou hustotou zalidnění je někdy velice velký problém najít volné pásmo pro připojení.

Pásmo 5 GHz, opět podle regulací ČTÚ, je rozděleno na 19 použitelných kanálů s frekvenčním rozestupem 20 MHz, avšak některé kanály jsou určeny pouze pro vnitřní použití a některé naopak pouze pro venkovní použití. Podrobné informace lze taktéž nalézt v prohlášení ČTU VO-R/12/08.2005-34, včetně informací o povolených vysílacích výkonech apod. Pásmo 5 GHz není zatím pod náporem velkého provozu, proto je zde zatím malá míra rušení a snadněji se hledají volné kanály.

2.2.4 Zabezpečení sítí

Zabezpečit síť proti připojení nežádaných zařízení je určitě potřebné. Od prosté ochrany osobních dat nebo třeba sdíleného internetu po ochranu před nabouráním do sítě a útoku na zařízení.

Tou naprosto nejjednodušší, avšak ne příliš kvalitní ochranou, je prosté zakázání vysílání SSID dané sítě. Pro připojení do takové sítě je pak zapotřebí dané SSID znát, obyčejné vyhledání sítě danou síť neodhalí. Bohužel taková ochrana je snadno prolomitelná, protože při připojování zařízení, které SSID skryté sítě zná, dochází k nezašifrované komunikaci, tím pádem také přenos SSID sítě. Případný útočník tak nemá potíž si tuto komunikaci odchytit a z ní poté získat ono SSID.

Další možností je filtrování MAC adres připojujících se zařízení, kde přístup do sítě je buď zakázán určitým MAC adresám a všem ostatním je povolen – tzv. blacklisting, nebo povolen pouze určitým MAC adresám a všem ostatním je zakázán – tzv. whitelisting. Zde je ochrana určitě lepší oproti pouhému skrytí sítě, avšak existují způsoby, kdy je možné z komunikace již připojeného zařízení zjistit MAC adresu, tu pak virtuálně *emulovat* na svém zařízení a poté se do sítě připojit pomocí této *emulované* adresy.

Pokročilé metody zabezpečení pracují s šifrováním. Je možné použít statické šifrování pomocí WEP klíčů, kdy dochází k ručnímu nastavení klíčů na obou komunikujících stranách. Na šifrování WEP navazují metody WPA a WPA2, kdy se používá již dynamické šifrování klíčů a pro přístup do sítí probíhá také autentizace pomoc PSK klíčů nebo pomocí služby RADIUS.

2.3 SRD860

Pásmo SRD, z anglického Short Range Device neboli zařízení na krátkou vzdálenost, je pásmo vyhrazené v rozsahu kmitočtů 863–870 MHz, dle podmínek stanovených CEPT (Konference evropských správ pošt a telekomunikací) [6]. Podle těchto podmínek lze v Evropě provozovat SRD zařízení ve stanovených pásmech a za daných parametrů.

Dle prohlášení ČTÚ [2] je možné využít celý rozsah 863–870 MHz, s omezením ERP (maximální vyzářený výkon) na 25 mW a maximálního klíčovacího poměru ≤ 0,1%, přičemž lze provozovat:

a. zařízení s modulací FHSS s kanálovou roztečí ≤ 100 kHz
b. zařízení s modulací DSSS nebo s jinou širokopásmovou modulací kromě FHSS bez omezení kanálové rozteče; u těchto zařízení je spektrální hustota výkonu omezena na:
   - –4,5 dBm/100 kHz v případě využití celého kmitočtového pásma
   - +6,2 dBm/100 kHz v případě využití pouze kmitočtového úseku 865–868 MHz
   - +0,8 dBm/100 kHz v případě využití pouze kmitočtového úseku 865–870 MHz
c. úzkopásmové zařízení s kanálovou roztečí ≤ 100 kHz.

Lze využít rozdělení rozsahu 868-870 MHz s kmitočtovým rozestupem 25 kHz, čímž dostáváme celkem 80 kanálů, na které se vztahují omezení v tabulce 2.1, čerpající jak z regulací ČTÚ tak z regulací CEPT. Zařízení pracující v SRD pásmu nelze používat pro vysílání analogových hovorových a akustických signálů, s výjimkou kanálů 9a, a to pouze za předpokladu dodržení ERP ≤ 5 mW a použití pokročilých technik zmírňujících rušení.

Kategorie	Rozsah frekvence [MHz]	Maximální ERP [mW]	Klíčovací poměr	Využití
1	868,00-868,60	25	≤ 1 %	obecné aplikace
2	868,60-868,70	10	≤ 1 %	bezpečnostní alarmy
3	868,70-869,20	25	≤ 0,1 %	obecné aplikace
4	869,20-869,25	10	≤ 0,1 %	sociální alarmy
5	869,25-869,30	10	≤ 0,1 %	bezpečnostní alarmy
6a	869,30-869,40	25	≤ 100 %	domácí automatizace
6b	869,30-869,40	10	≤ 1 %	bezpečnostní alarmy
7a	869,40-869,65	500	≤ 10 %	obecné aplikace
7b	869,40-869,65	25	≤ 0,1 %	obecné aplikace
8	869,65-869,70	25	≤ 10 %	bezpečnostní alarmy
9a	869,70-870,00	5	≤ 100 %	hlasové hovory, akustické signály
9b	869,70-870,00	25	≤ 1 %	obecné aplikace

Tabulka 2.4: Rozdělení subpásem v pásmu SRD

Kompletní tabulky regulací lze dohledat na stránkách CEPT [6] a ČTÚ [2]. V tomto projektu budeme pracovat v kategorii 1, konkrétně na frekvenci 868,3 MHz. Jelikož pracujeme s kompletním modulem, kterému posíláme data, nemusíme řešit maximální ERP – o to se stará výrobce. Měli bychom však při implementaci komunikace dbát na nepřekračování klíčovacího poměru, tedy nepřesáhnout dobou vysílání 1% hodiny, což je 36 sekund. To nám dává možnost přenosu přibližně 4,6kB dat za hodinu (vyjdeme-li z přenosové rychlosti specifikované v pasáži 3.2).

2.4 Modulace signálu

Modulací rozumíme zakódování vstupního signálu na vysílači pro jeho rádiový (analogový) přenos k přijímači a následně jeho dekódování zpět na přenášený signál. V rádiové komunikaci rozlišujeme dvě hlavní odvětví modulace – modulaci digitálního signálu a modulaci analogového signálu. Jelikož v tomto projektu pracujeme s přenosem binárních dat, větší pozornost budu věnovat digitálním modulačním technikám

2.4.1 Modulace digitálního signálu

Jde o modulaci, kde je nosný signál modulovaný digitálním signálem – buď přímo jeho obdélníkovým průběhem nebo skupinami bitů. Ve své podstatě se jedná o D-A převod při modulaci a zpětný A-D převod při demodulaci. Mezi nejznámější techniky patři:

- PSK (phase-shift keying) – modulace posunutím fáze
- FSK (frequency-shift keying) – modulace posunutím frekvence
- ASK (amplitude-shift keying) – modulace posunutím amplitudy
- QAM (quadrature amplitude modulation) – kombinace PSK a ASK modulace

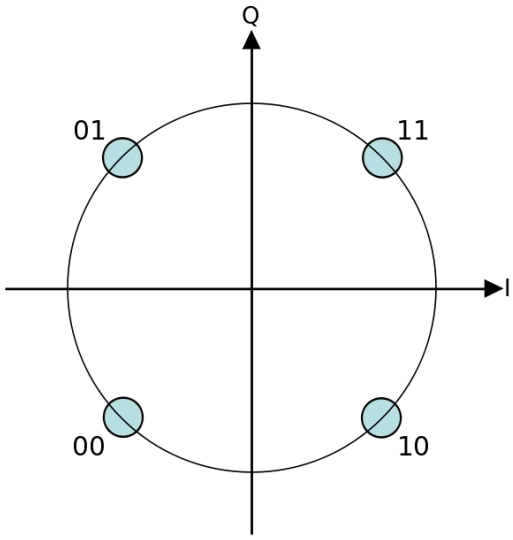
2.4.1.1 Modulace PSK

Modulaci PSK lze třídit podle toho, kolik datových bitů kóduje zároveň. Pokud chceme kódovat po jednotlivých bitech, budeme potřebovat dvě fáze – stavy, v tomto případě o 180° otočené. U kódování 2bitů zároveň už budeme potřebovat čtyři stavy – každý otočený o 90°. Pro zjištění počtu fází, pro kódování po více bitech zároveň, tedy platí vzorec: $k = 2^n$, kde k je počet fází a n je počet zároveň kódovaných bitů. Podle počtu stavů – fází, se pak metoda PSK označuje, tedy například 8-PSK označuje kódování vstupních dat po 3 bitech (8stavů). Výhoda takového kódování je vyšší přenosová rychlost, případná možnost snížení šířky pásma, avšak nevýhodou oproti tomu je menší odolnost vůči rušení (šum, přeslechy). Demodulace tohoto signálu probíhá rozpoznáváním změn, mezi jednotlivými fázemi. Čím více fází používáme, tím přesnější musí být technika rozpoznávání při demodulaci.

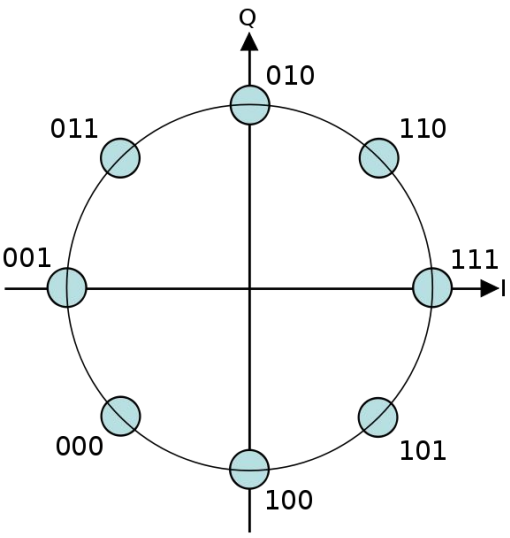
Metodu 8-PSK lze najít například v systémech GSM[5], její fázový diagram je zobrazen na obrázku 2.2. Pro porovnání je na obrázku 2.1 zobrazení fázového diagramu pro 4-PSK modulaci, která se používá například u DVB-S/DVB-T[6] vysílání. Na fázových diagramech je také možné si všimnout, že kódování jednotlivých bloků bitů je uspořádáno tak, aby mezi každými vedle sebe ležícími fázemi byl rozdíl vždy jen v jedné bitové pozici.

5 *Global System for Mobile Communications* - původně francouzsky *„Groupe Spécial Mobile“* - jedná se o nejrozšířenější standard pro komunikaci mobilních telefonů. GSM je buňková síť - mobilní telefony se připojují do sítě prostřednictvím nejbližší buňky (vysílače).

6 *DVB-S/DVB-T* – Standardy digitálního televizního vysílání. DVB-S značí satelitní přenos, DVB-T značí pozemní přenos. Datové toky jsou komprimovány, čímž je značně ušetřeno frekvenční spektrum a na jedné frekvenci je možné naladit více programů, tzv. multiplex.



Obr. 2.1: Fázový diagram pro 4-PSK



Obr. 2.2: Fázový diagram pro 8-PSK

2.4.1.2 Modulace FSK

Při základní (také označovanou jako binární – BFSK) modulaci je nosná modulována přímo obdélníkovým průběhem vstupního signálu. Logické jedničky jsou modulovány jako „značky“, tedy frekvence s vyšším kmitočtem a logické nuly jsou modulovány jako „mezery“ - frekvence s nižším kmitočtem. Takováto základní modulace je velmi snadná na implementaci a je tak vhodná pro jednoduché aplikace, avšak není moc vhodná pro přenos na větší vzdálenosti a na vstup modulátoru nemohou přicházet delší sledy nul nebo jedniček.

Často používanou variantou FSK je audio modulace, neboli AFSK (Audio frequency-shift keying), kde se vstupní signál převání na zvukové tóny. Logická jednička bývá nejčastěji zakódována na 1200 Hz a logická nula na 2400 Hz. Vzniká tak tón, který byl velmi typický a známý u starých telefonních modemů.

2.4.1.3 Modulace ASK

Tento druh modulace se velmi podobá klasické analogové amplitudové modulaci, tedy každá různá amplituda představuje jinou informaci. Každé amplitudě je přiřazen určitý vzorek binárních dat. Těmto přiřazením se říká symboly a vždy tvoří symbolovou sadu. Tuto symbolovou sadu musí mít jak modulátor, tak demodulátor stejnou, aby bylo možné data zpětně dekódovat. Nejjednodušší variantou ASK je metoda OOK (ON-OFF Keying), která při vstupní logické jedničce vysílá na stejné amplitudě a při logické nule nevysílá nic. Stejný způsobem byl také vysílán například Morseův kód.

2.4.2 Modulace analogového signálu

Stejně jako u modulace digitální, tak také u modulace analogové rozlišujeme několik základních druhů. Mezi ně patří amplitudová modulace, frekvenční modulace a fázová modulace. Jedná se vždy o kombinaci nosného signálu s jednou ze tří složek modulovaného signálu, tedy amplitudou, frekvencí nebo fázovým posunem.

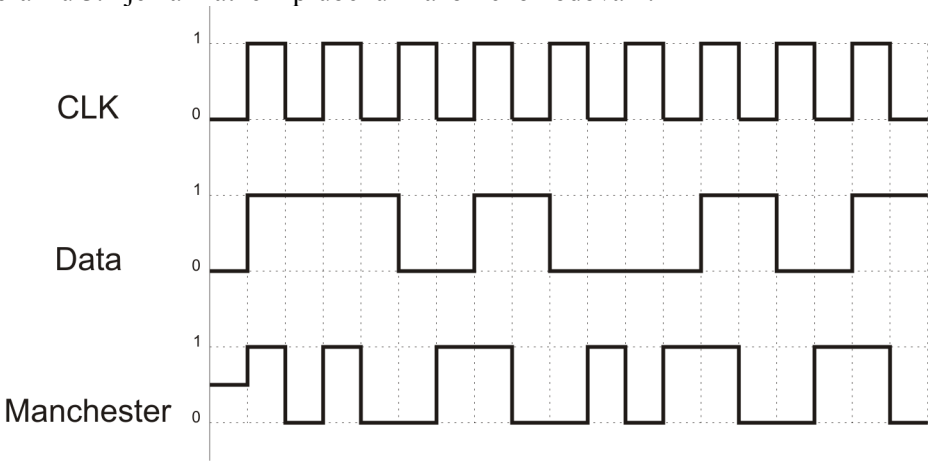
3 Implementace firmware

Tato sekce popisuje podrobnou implementaci navrženého komunikačního firmware, včetně metod použitých při řešení samotné komunikace a jejího ladění. Implementace zdrojových kódů probíhala v programovacím jazyce C a celá implementace je svou funkčností zaměřena na použití s mikrokontrolérem MSP430 laboratorních kitů FITkit. Celý projekt byl vyzkoušen a odladěn na FITkitech verze 1.2, avšak stejná funkčnost by měla být zajištěna také na verzích 1.0 a 2.0.

3.1 Kódování Manchester

Jelikož do modulátoru vysílače nesmí přicházet dlouhé sledy logických jedniček ani nul, je zapotřebí zvolit takový signál, který má pokud možno vyváženou střední hodnotu. Jako nejlepší varianta se v tomto případě jeví dvoufázový signál v kódování Manchester [1].

Kódování dat probíhá tím způsobem, že log. 0 je zakódována jako posloupnost sestupné a nástupné hrany, naopak log. 1 je zakódována jako posloupnost nástupné a sestupné hrany. Toto zakódování nám zaručí podmínku, že na vstupu modulátoru nebudou dlouhé sledy logických jedniček a nul. Na obrázku 3.1 je na krátkém průběhu znázorněno kódování.



Obr. 3.1: Kódování Manchester

Drobnou nevýhodou tohoto kódování je fakt, že je při přenosu dat zapotřebí dvakrát vyšší vzorkovací frekvence, než je původní frekvence vstupních dat. Pokud například chceme zakódovat data o frekvenci 1 kHz, výsledný signál v kódování Manchester bude potřeba odesílat frekvencí 2 kHz, abychom dodrželi původní rychlost. Je tedy zapotřebí brát ohled na možnosti vysílacích a přijímacích modulů a dbát na tato omezení při volbě přenosové rychlosti.

3.2 Přenosová rychlost

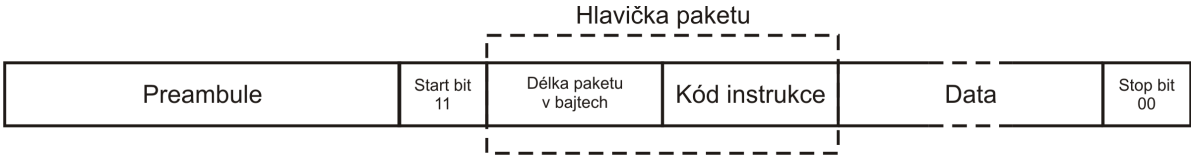
Volba přenosové rychlosti se v tomto projektu odvíjela od dvou hlavních faktorů. Prvním z nich bylo omezení maximální frekvence vstupních dat do vysílacího modulu. Dle parametrů modulu [11] to je maximálně 3 kHz. Dále je zde faktor druhý, a to vhodné nastavení časovače na MSP430 osazeném na FITkitu v1.2. Hlavní hodiny FITkitu běží na 32,768 kHz. Vhodným zvolením předděličky časovače tedy potřebujeme dosáhnout frekvence nižší, než 3 kHz. V tomto projektu jsem zvolil předděličku 16, která nám dá celé číslo 2048 Hz (nedojde tak k oříznutí případných zbytků za desetinou čárkou) a

nechává zároveň případnou rezervu, pro budoucí úpravy přenosové rychlosti – například použití jiného krystalu.

2048 Hz je tedy frekvence, kterou budeme přivádět na vstup vysílače. Pokud ale chceme znát přímo přenosovou rychlost dat, je potřeba tuto frekvenci ještě upravit. Nejdříve je zapotřebí ji podělit dvěma, protože vysíláme v kódování Manchester. Dostáváme tak přenosovou frekvenci dat 1024 Hz, jinak řečeno 1024 bit/s.

3.3 Protokol pro komunikaci

Pro zajištění správného přenosu a bezproblémovou komunikaci bylo potřeba navrhnout vlastní komunikační protokol, který by byl dostatečně variabilní pro více druhů použití. Pro takové účely jsem zvolil variantu odesílání po paketech s předem určenou délkou. Při novém vysílání je nejprve vyslána preambule neboli předem definovaný sled logických jedniček zakódovaný v Manchester kódu – tedy posloupnost po sobě jdoucích log. 1 a 0. Tato preambule trvá přibližně 50 ms, což při vysílací frekvenci 2048 Hz odpovídá 102 logickým jedničkám v Manchester kódu. Po preambuli následuje start bit – dvě po sobě jdoucí logické jedničky – označující začátek paketu. Hned za start bitem se nachází 8bitová hlavička nesoucí dvě 4 bitové hodnoty. První 4 bity udávající celkovou délku paketu v bajtech, počítanou bez start a stop bitu a druhé 4 bity udávající kód instrukce. Za hlavičkou následuje již zbytek zprávy, rozdělený po jednotlivých bajtech až po poslední, za kterým následuje stop bit – dvě po sobě jdoucí logické nuly. Vizuální znázornění paketu je znázorněno na obrázku 3.2.



Obr. 3.2: Komunikační protokol

3.4 Instrukční kódy

Sada instrukčních kódů popisuje jednotlivé instrukce a jejich syntaxi v paketu. Tato sada byla zavedena z důvodu variability využití firmware a také pro případy budoucího rozšiřování. Aktuálně je možné namapovat až 16 různých instrukcí. Pokud by byla nutnost přidat ještě více instrukcí, musel by se upravit komunikační protokol rozšířením 4 bitového pole kódu instrukce v hlavičce paketu. Tabulka 3.1 obsahuje aktuální mapování instrukcí.

Jednotlivé délky dat vycházejí z knihovny `limits.h` dle specifikací MSP430. Některé datové typy nejsou do instrukční sady zahrnuty z důvodu stejných délek dat jako již implementované instrukce. Pro příklad je možné uvést, že datový typ **short** má stejnou délku (16bitů) jako datový typ **int**, proto je zbytečné ho implementovat a pro jeho odeslání stačí pouhé přetypování ve zdrojovém kódu programu. Kompletní informace o délkách datových typů je možné najít v již zmíněném hlavičkovém souboru limits.h.

Kód instrukce	Zkratka	Délka paketu [B]	Délka dat [B]	Popis
0000	START	1	0	Zapnutí činnosti na přijímači
0001	INTEGER	3	2	Číslo typu **int**
0010	UINTEGER	3	2	Číslo typu **unsigned int**
0011	LONG	5	4	Číslo typu **long**
0100	ULONG	5	4	Číslo typu **unsigned long**
0101	DOUBLE	5	4	Číslo typu **double**
0110	STRING	2 - 15	1-14	Řetězec délky 1 – 14 znaků
0111	USERCMD1	1	0	Uživatelský příkaz bez dat
1000	USERCMD2	1	0	Uživatelský příkaz bez dat
1001	USERCMD3	1	0	Uživatelský příkaz bez dat
1010	USERCMD4	1	0	Uživatelský příkaz bez dat
1011	USERCMDDATA1	1-15	0-14	Uživatelský příkaz s daty
1100	USERCMDDATA2	1-15	0-14	Uživatelský příkaz s daty
1101	USERCMDDATA3	1-15	0-14	Uživatelský příkaz s daty
1110	USERCMDDATA4	1-15	0-14	Uživatelský příkaz s daty
1111	STOP	1	0	Ukončení činnosti přijímače

Tabulka 3.1: Sada instrukčních kódů

3.5 Hammingův kód

Pro snížení šance neúspěšného přenosu paketu mezi vysílačem a přijímačem byl implementován Hammingův kód [9], který je schopen detekovat a opravit **jednobitovou chybu** v každé takto zakódované části. Kódování probíhá tak, že ke zprávě kterou chceme zakódovat přidáváme kontrolní paritní bity. Zvolený poměr kódování se pak zapisuje ve tvaru *(celkový počet bitů, počet datových bitů)*. Pokud tedy použijeme Hammingův kód (7,4), znamená to, že čtyřem datový bitům přísluší tři paritní bity. Tyto paritní bity jsou v kódované zprávě rozmístěny tak, že každá pozice, jejíž hodnota je mocninou 2, je paritní bit a všechny ostatní bity jsou bity datové (tedy nesoucí samotná data). Každý paritní bit má pak na starost kontrolu jen určitých datových bitů a obecně se indexují podle jejich pozice v kódované zprávě. Takový způsob indexování nám pak ulehčuje určení, které datové bity daný paritní bit kontroluje.

Index		1	2	3	4	5	6	7	8	9	10	11	12
Bity zprávy		**p1**	**p2**	d1	**p4**	d2	d3	d4	**p8**	d5	d6	d7	d8
Pokrytí par. bitů	p1	x		x		x		x		x		x	
	p2		x	x			x	x			x	x	
	p4				x	x	x	x					x
	p8								x	x	x	x	x

Tabulka 3.2: Rozmístění datových a paritních bitů v Hammingově kódu (12,8)

Obecně platí: p_n = *prvních n-1 bitů přeskoč, dále n bitů zkontroluj, n bitů přeskoč, n bitů zkontroluj, atd.*

Toto rozmístění jednotlivých bitů a pokrytí můžeme vidět v tabulce 3.2, která znázorňuje Hammingův kód (12,8). Tento kód jsem rovněž použil při implementaci mého firmware. Výpočet hodnoty jednotlivých paritních bitů se provádí již jednoduše pomocí operace XOR (operátor ⊕). Jednotlivé hodnoty paritních bitů pak vypočteme následovně:

p1 = d1 ⊕ d2 ⊕ d4 ⊕ d5 ⊕ d7
p2 = d1 ⊕ d3 ⊕ d4 ⊕ d6 ⊕ d7
p4 = d2 ⊕ d3 ⊕ d4 ⊕ d5 ⊕ d8
p8 = d5 ⊕ d6 ⊕ d7 ⊕ d8

Po výpočtu hodnot těchto paritních bitů je zpráva připravená k odeslání. Po přijetí na přijímači následuje zpětná kontrola přijaté zprávy. Znovu dojde k výpočtu hodnot paritních bitů dle výše uvedených vzorců a jejich porovnání s hodnotami přijatých paritních bitů. Jedná se o tzv. výpočet syndromu chyby. Zde se nám opět hodí indexování paritních bitů dle jejich pozice v zakódované zprávě, protože hodnoty indexu nám ve výsledku udávají, na které pozici ve zprávě došlo k chybě.

Pokud nedošlo při přenosu k žádné chybě, musí hodnoty nově vypočtených a přijatých paritních bitů odpovídat. Syndrom chyby je v tomto případě nulový.

Pokud nebude odpovídat hodnota pouze jednoho paritního bitu, znamená to, že data přišla v pořádku a chyba nastala přímo v paritním bitu. Syndrom chyby je v takovém případě některou mocninou čísla 2. Podle indexování je možné si to ověřit – když nebude odpovídat například pouze paritní bit p4, znamená to, že chyba nastala na 4. pozici, což je samotný paritní bit p4. Takovou chybu můžeme, ale také nemusíme opravovat, protože se nejedná o chybu v datech.

Pokud nebude odpovídat dva a více paritních bitů, došlo k chybě na některé z pozic datových bitů. Tuto pozici opět zjišťujeme pomocí indexů jednotlivých paritních bitů, které se neshodují. Pro příklad, pokud nebudou odpovídat paritní bity p1, p2 a p8, stačí sečíst indexy paritních bitů a dostaneme syndrom chyby (její pozici), tedy 1 + 2 + 8 = 11. Chyba nastala na pozici 11 což odpovídá datovému bitu d7.

Následná oprava nalezené chyby je již jednoduchá. Pokud je na dané pozici nalezena chyba, dojde k invertování této hodnoty. Toho lze docílit snadno operací XOR původní hodnoty s log. 1.

3.5.1 Kódování paketu firmwaru

Při implementaci firmware jsem použil Hammingův kód (12,8), kdy každý jeden bajt dat je zakódován pomocí Hammingova kódu včetně samotné hlavičky paketu. Tímto způsobem zvyšuji odolnost proti případným jednobitovým chybám v každém přeneseném bajtu dat, než kdybych volil kódování celého paketu zároveň. Tímto ale také vzniká redundance datového toku, protože na každý bajt dat připadají 4 kontrolní bity, což nám zavádí 33% redundanci dat při přenosu. Tím pádem dochází také ke snížení efektivní přenosové rychlosti z 1024bit/s na 682bit/s.

Kódování a dekódování do a z Hammingova kódu obstarávají funkce:
vysílač: `void codeHamming8to12(char, unsigned int *)`
přijímač: `void decodeHamming12to8(unsigned int *, unsigned int *)`

3.6 Popis funkcí firmwaru vysílače

Firmware vysílače tvoří tři stěžejní části. První připravujue uživatelská data k odeslání, druhá provádí sestavení paketu, včetně zakódování jeho obsahu pomocí Hammingova kódu a třetí realizuje samotné odeslání dat. Samotnému zpracování dat a odesílání předchází krátká inicializace firmware, která spočívá v nastavení výstupních pinů FITkitu pro správnou komunikaci s vysílacím modulem. Vysílací modul je připojen přes PORT4 a pro jeho správnou funkčnost je potřeba určitě směr komunikace tohoto portu. To je provedeno zápisem hodnoty 0x20 (hexadecimálně) do registru

P4DIR. Tím určíme, že pin 5 portu 4 bude jako výstupní, všechny ostatní pak jako vstupní. Tento pin odpovídá pinu číslo 4 na patici JP9 (viz. 4.1 Vysílací modul).

3.6.1 Příprava uživatelských dat

Pro přípravu uživatelských dat pro odeslání je ve firmwaru připraveno celkem 8 funkcí zajišťujících jejich zpracování. Tyto obslužné funkce vycházejí z navržené tabulky instrukčních kódů. Jsou zde dvě funkce univerzálnějšího použití a dalších šest funkcí pro jednotlivé datové typy. První z univerzálních funkcí, `send_NODATA(instructions)`, obsluhující všechny instrukce pracující bez přídavných dat (START, STOP a instrukce USERCMD1-4), bere jako jediný parametr samotnou zkratku instrukce. Druhá univerzální funkce, `send_USERDATA(instructions, char*)`, obsluhuje čtveřici instrukcí USERCMDDATA1-4, bere dva parametry, kde prvním z nich je zkratka instrukce a druhým je ukazatel na pole obsahující data pro odeslání. Následující šestice funkcí, pojmenovaných `send_{DATA_TYPE}({DATA_TYPE} number)`, kde *{DATA_TYPE}* zastupuje datový typ, je ve své podstatě velmi podobná, jen se liší v přijímaném parametru dle datového typu, který chceme odesílat. Každá z těchto funkcí už si zkratku instrukce nastavuje sama, není tedy potřeba ji jako u předešlých dvou univerzálních funkcí explicitně specifikovat v parametru.

3.6.2 Příprava paketu

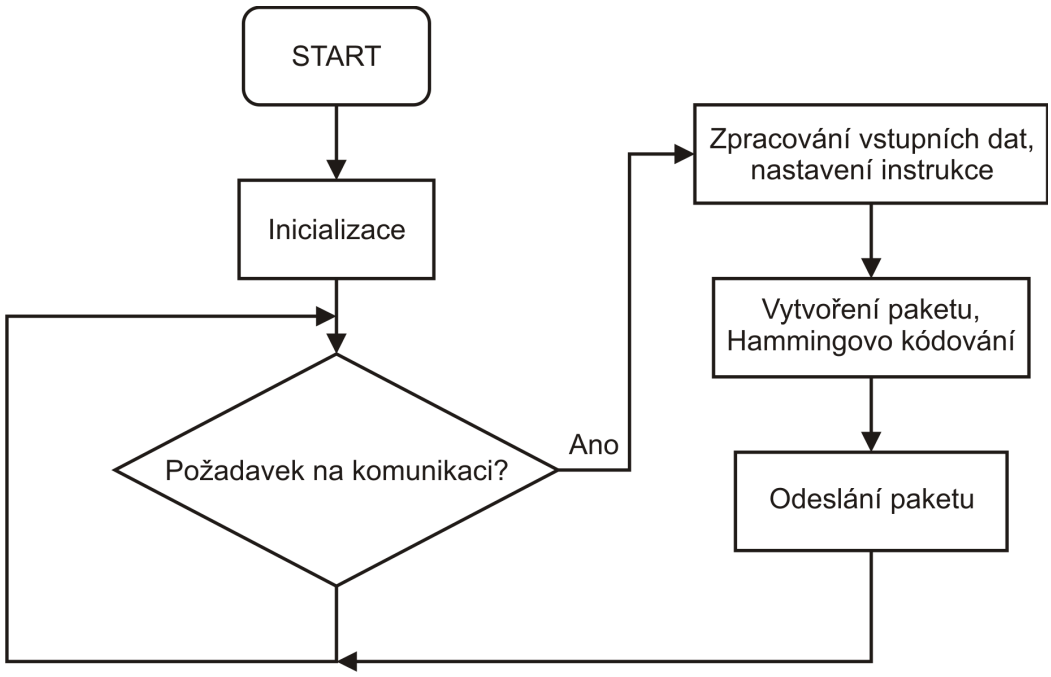
Všech 8 výše zmíněných funkcí, po zpracování vstupních dat, volají funkci `transmit(char*, instructions, unsigned int)`, která je stěžejní funkcí pro celou komunikaci. Jako první dojde ke zpracování všech odesílaných informací do struktury `tPacket`, zavoláním funkce `createPacket(unsigned int, struct tPacket*, char*)`. Před každým voláním je celá struktura `tPacket` nulována pomocí `memset`, aby se nemohlo stát, že se do nového paketu dostanou data z nějakého paketu předchozího. Nejdříve je vytvořena hlavička paketu nesoucí celkovou délku paketu a instrukční kód, která je zakódována pomocí Hammingova kódu. Pak proběhne také zakódování dat po 8-bitových blocích (pokud jsou nějaká data pro odeslání nachystána). Po vytvoření paketu je již vše připraveno k samotnému odeslání dat. Následující výtažek z kódu ukazuje způsob zakódování 8bitové zprávy na 12bitovou. Prvním parametrem funkce je 8bitová hodnota, kterou chceme zakódovat, druhým je ukazatel na pole, do kterého je výsledná kódovaná zpráva uložena.

```
void codeHamming8to12(char bajt, unsigned int *pole) {
    unsigned int i, j;
    for(i=0, j=0; i<8; i++)
    { //napleni 12bit pole datovimy bity
        if(i==4 || i+1==8) //osetreni preskoceni p4 a p8 bitu
        {
            pole[10-i-j]=bajt & 1; //zapis datoveho bitu
            j++;
        }
        else
            pole[11-i-j]=bajt & 1; //zapis datoveho bitu
        bajt >>=1; //log. posun zapisovane bajtove hodnoty
    }
    //vypocet vsech paritnich bytu
    pole[0]=pole[2]^pole[4]^pole[6]^pole[8]^pole[10];
    pole[1]=pole[2]^pole[5]^pole[6]^pole[9]^pole[10];
    pole[3]=pole[4]^pole[5]^pole[6]^pole[11];
    pole[7]=pole[8]^pole[9]^pole[10]^pole[11];
}
```

3.6.3 Odeslání paketu

Při odesílání dat je nejdříve zapotřebí nastavit správnou odesílací frekvenci. Pro odesílání je využit čítač **TimerA** [8] a správná obsluha časování je zajištěna pomocí generování přerušení tohoto časovače. Jako první dojde k vyresetování nastavení časovače před každým novým vysíláním – je tak zabráněno tomu, že při novém vysílání dat nebude v registrech časovače již nějaká předchozí hodnota. Dále je zápisem do registru **CCTL0** povoleno přerušení. Jak je uvedeno v sekci 3.2, je potřeba nastavit předděličku časovače hlavních hodin na hodnotu 16, aby bylo dosaženo přenosové rychlosti 2048Hz. To je zajištěno zapsáním hodnoty 16 do registru **CCR0**. Posledním krokem je již volba hlavních hodin a čítacího módu – zde je použit hodinový zdroj **ACLK** a mód čítače **UP**. Mód **UP** počítá od nuly rychlostí hodin **ACLK** a po dosažení hodnoty zapsané v registru **CCR0** dojde k vygenerování přerušení a k vynulování čítače, takže v dalším cyklu zase počítá od nuly. Samotná obsluha přerušení pak už provádí pouze nulování globální proměnné `wait_break`, která slouží jako povolovací signál pro vysílání – tedy při každé změně `wait_break` dojde k přerušení čekací smyčky funkce `com_wait()` a tím k vystavení právě odesílané bitové hodnoty na výstup, tedy na samotný vysílací modul.

Na začátku dojde k odeslání preambule v Manchester kódu, dále start bitu a následují všechna data z připraveného paketu. Veškeré odesílání dat obstarává funkce `sendData(unsigned int, unsigned int*)`, která odesílá data připraveného paketu po blocích, jejichž délka je uvedena v prvním parametru. K odesílání dat dochází také v kódování Manchester, tedy odesílaný bit je reprezentován dvěma bity v Manchester kódu. Pro usnadnění tohoto vysílání jsem si pro každou bitovou hodnot dat připravil dvě drobné pomocné funkce `manch_zero()` a `manch_one()`, které jednoduše odešlou (vystaví na výstup na vysílací modul) kombinaci bitů 0,1 nebo 1,0. Po odeslání všech dat z paketu je ještě odeslán stop bit a tím končí práce funkce `transmit()`. Obrázek 3.3 znázorňuje vývojový diagram firmwaru vysílače.



Obr. 3.3: Vývojový diagram firmwaru vysílače

3.7 Popis funkcí firmwaru přijímače

Stěžejními částmi firmwaru přijímače jsou, stejně jak u vysílače, funkce pro samotný příjem dat z modulu přijímače, dále dekódování přijatého paketu v Manchester kódu a taktéž dekódováním Hammingova kódu. V případě nalezení jednobitových chyb v jednotlivých blocích dat dochází automaticky i k jejich opravě. Poslední částí příjmu je pak volání jednotlivých obslužných funkcí dle přijatého instrukčního kódu paketu.

Aby po zapnutí přijímače nedocházelo k aktivnímu čekání na příjem dat (tedy blokující čekání na příjem dat, které by zamezovalo mikroprocesoru ve výkonu jiných funkcí), je pro samotné naslouchání dat také zaveden časovač vypršení naslouchání, tzv. **timeout**. Nastavení timeoutu je možné upravovat dle vlastních potřeb, v této implementaci pracuji s timeoutem zhruba 500 ms. Tedy pokud nebude v časovém rozmezí 500 ms přijat žádný paket, je automaticky naslouchání paketu přerušeno a tím je dána možnost mikroprocesoru zpracovávat jiná data nebo funkce dále implementované uživatelem.

3.7.1 Naslouchání a příjem dat

Hlavní funkcí, zajišťující naslouchání na výstupu z modulu přijímače a zpracování příchozího paketu, je funkce `recieve()`. Tato funkce, stejně jak bylo zmíněno v části 3.6.3 Odeslání paketu, pracuje s časovačem **TimerA** a taktéž je před samotným příjmem dat zapotřebí nastavit správnou frekvenci. Jelikož se jedná o naprosto stejnou obsluhu jako u vysílače, nebudu tento princip již rozebírat.
Po nastavení naslouchací frekvence je taktéž resetován čítač pro timeout a naslouchání přechází do své první části – hledání preambule. Ze sekce 3.3 je zřejmé, že preambule se skládá ze 102 po sobě jdoucích jedniček, zakódovaných v Manchester kódu (tedy sled 101010...). Po každém zachycení jedné jedničky v Manchester kódu (posloupnosti 10) je zvýšen čítač preambule a taktéž čítač pro timeout je navyšován v každém přerušení vyvolaném při naslouchání pro příchozí data, tedy v obslužné rutině pro přerušení časovače TimerA. Tento cyklus hledání preambule skončí buď nalezením preambule, tedy přečtením minimálně 60ti po sobě jdoucích jedniček v Manchester kódu nebo dojde k dosažení hranice timeoutu (500ms). Při vypršení timeoutu dojde k ukončení celé funkce `recieve`, naopak při nalezení preambule dojde pouze k vyskočení z čekací smyčky a postoupení k další části – nalezení start bitu.

Nalezení **startbitu** probíhá stejně jak načítání preambule. Start bit je zakódován jako po sobě jdoucí dvě jedničky, tedy posloupnost 11. Jakmile je start bit nalezen, dojde opět k ukončení smyčky a následuje již poslední část – načtení dat paketu.

Data paketu se ukládají do **bufferu**, implementovaného jako globální pole hodnot **unsigned int**. Ze vstupu je čteno vždy po dvou bitech, opět kvůli kódování Manchester. Pokud je na vstupu posloupnost 10, je do bufferu zapsána hodnota 1, naopak pokud je na vstupu posloupnost 01, je do bufferu zapsána hodnota 0. Takto zápis do bufferu probíhá do té doby, než je na vstupu načtena dvojice bitů 00, která značí **stopbit**, tedy konec paketu. Po nalezení **stopbitu** je ukončen cyklus načítání dat a tím také končí celá funkce `recieve`.

3.7.2 Dekódování paketu

Po načtení surových dat z modulu přijímače a uložení do globálního bufferu, je zapotřebí celý paket dekódovat z Hammingova kódu. Protože je použito kódování (12,8), je dekódování prováděno po 12ti bitových blocích. Celé dekódování obstarává funkce `parsePacket()`, která je volána pokaždé, když jsou úspěšně přijata data (tedy nedojde k timeoutu). Pro celý dekódovaný paket je připravená struktura `tPacket`, kterou před každým dekódováním pomocí funkce `memset` nuluji. Jako první je z Hammingova kódu dekódována hlavička pro zjištění délky paketu a druhu instrukce a tato data jsou

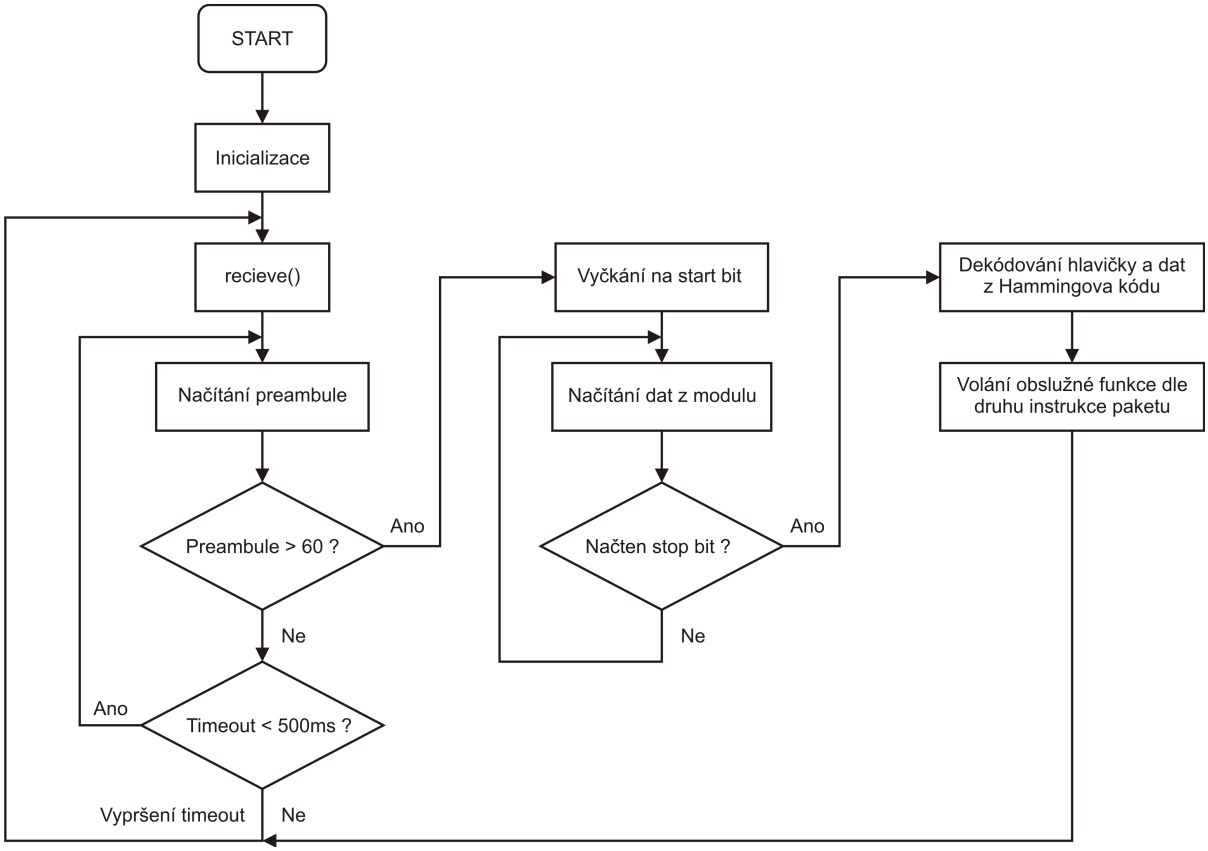
uložena do struktury. Poté následuje ve for cyklu postupně dekódování datové části paketu (pokud nějaká data obsahovala) do nového bufferu ve struktuře `tPacket`.

Dekódování z Hammingova kódu zajišťuje funkce `decodeHamming12to8(unsigned int*, unsigned int*)`, kde prvním parametrem je ukazatel na **buffer** se surovými daty z modulu přijímače a druhým parametrem je ukazatel do výstupního pole (dekódovaná data) ve struktuře `tPacket`. Tato funkce z každého 12bitového bloku dat vypočítá nové hodnoty paritních bitů (p1, p2, p4 a p8 – viz. Sekce 3.5) a tyto hodnoty porovná s paritními bity v přijatém bloku dat. Pokud je zjištěna chyba (rozdíl) pouze v jednom paritním bitu, došlo k chybě právě v tomto paritním bitu, pokud je chyba ve více paritních bitech, je opravena chyba na pozici udávané součtem těchto paritních bitů (více v sekci 3.5). Opravit je však možné pouze jednobitovou chybu v každém 12bitovém bloku.

Po dekódování celého paketu už následuje pouze větvení `case`, kde jsou podle druhu instrukce přijatého paketu volány jednotlivé obslužné funkce pro zpracování přijatých dat.

3.7.3 Zpracování uživatelských dat

Opět, stejně jako v části 3.6.1 je pro toto zpracování přichystáno 8 funkcí, kde dvě jsou univerzální a dalších 6 odpovídá jednotlivým datovým typům. Parametrem všech těchto funkcí je vždy pole dekódovaných dat ze struktury `tPacket` a jednotlivé návratové hodnoty těchto funkcí odpovídají datovým typům, které zpracovávají – tedy `parse_INT` vrací datový typ **int**, `parse_ULONG` vrací **unsigned long**, atd. Celá sada těchto obslužných funkcí je přehledně okomentována ve zdrojovém kódu, pro bližší informace o jednotlivých funkcích je možné nahlédnout přímo tam. Vývojový diagram firmwaru vysílače znázorňuje obrázek 3.4.



Obr. 3.4: Vývojový diagram firmwaru přijímače

3.8 Možnosti rozšiřitelnosti implementace

Výše popsaná implementace je spíše vhodná pro drobné použití na kratší a střední vzdálenosti a také instrukční sada a délka paketu nemusí být vyhovující pro rozsáhlejší projekty. Komunikace je sice chráněna Hammingovým kódem, který však umí detekovat a opravovat pouze jednobitové chyby. To v kombinaci s pouze jednosměrnou komunikací od vysílače k přijímači může při komunikaci na delší vzdálenosti nebo ve více rušeném prostředí (zastavěná oblast, přítomnost jiných zařízení komunikujících ve stejném pásmu, atd.) mít za následek ztrátu dat bez možnosti jejich opravy nebo opětovného odeslání. Řešením všech těchto nevýhod se budu zabývat v následujících sekcích.

3.8.1 Rozšíření hlavičky paketu

V případě, že uživatel firmwaru bude potřebovat zasílat delší pakety nebo mu nebude dostačovat stávající rozsah instrukční sady, je možné rozšířit celou hlavičku paketu z aktuálních 8 bitů na 16 bitů nebo i více bitů, avšak bylo by vhodné držet se násobků 8mi kvůli snadnějšímu kódování paketu Hammingovým kódem (12,8). Taková úprava by neznamenala žádný razantní zásah do zdrojového kódu a značně by rozšířila použitelnost.

3.8.1.1 Rozšíření instrukcí

Jak je patrné ze sekce 3.3, první část hlavičky, nesoucí kód instrukce, je 4 bitová, což nám dává celkem 16 různých instrukcí, z toho 6 instrukcí je čistě pro přenos datových typů a zbytek je použitelný dle potřeb uživatele. Není to mnoho, ale základní potřeby menšího projektu tato instrukční sada pokryje. Rozšířením na 8 bitů dostáváme 256 použitelných instrukcí, které by bez problému stačily třeba i na řízení domácích spotřebičů ve větším rodinném domě nebo řízení více jiných složitějších zařízení zároveň.

Ve spojitosti s řízením jiných zařízení se také nabízí varianta délky instrukčního kódu 6 bitů (64 instrukcí) a omezení délky paketu pouze na 2 bity (délka paketu maximálně 4 bajty, tj. maximálně 3 bajty dat), protože v řízení jiných zařízení většinou není příliš zapotřebí přenášet velké bloky dat. Dohromady by tak tvořily 8bitovou hlavičku snadno kódovatelnou pomocí Hammingova kódu (12,8) a krátké pakety, což by znamenalo menší režii komunikujících zařízení při jejich zpracování.

3.8.1.2 Rozšíření délky paketu

Stejně jako kód instrukce je také hodnota délky paketu v hlavičce 4 bitová. Tím dostáváme maximální délku paketu 16 bajtů, v čistých datech však jen 15 bajtů, protože jeden bajt zabírá pouze hlavička. Pokud by to situace vyžadovala a bylo by zapotřebí přenášet velké bloky dat (tj. > 16 bajtů), rozšířením části délky paketu na 8 bitů dostaneme opět délku až 256 bajtů na paket, což je už poměrně velký objem dat. Nevýhodou takto dlouhých paketů však může být větší náchylnost na chyby, způsobené při bezdrátovém přenosu. Pokud by došlo k rozšíření jak délky instrukce, tak délky paketu na 8bitů, dohromady by hlavička tvořila 2 bajty. Maximální délka dat přenesená v jednom paketu by pak činila 254 bajtů.

3.8.2 Rozšíření Hammingova kódování – SECDED

Hammingův kód, popsaný v sekci 3.5, nese určité omezení, a to schopnost detekovat a opravit pouze jednobitovou chybu. Pokud se naskytne dvou a vícebitová chyba, není možné takovou chybu opravit. Je tu však možnost rozšíření Hammingova kódu o jeden paritní bit navíc, který ještě paritně kontroluje jednotlivé původní paritní bity. Takto upravený kód pak dokáže detekovat i dvoubitovou

chybu, i když ji nedokáže opravit. Takto upravený Hammingův kód nese označení SECDED[7] a nepatrně zvyšuje bezpečnost při přenosu dat. Tento jeden paritní bit navíc se přidává před celé kódové slovo a pokrývá jednotlivé paritní bity kódované zprávy.

Stejně jako ostatní paritní bity je i tento bit, označme jej **p0**, počítán pomocí bitové operace XOR mezi všemi paritními bity. Výpočet pro Hammingův kód (13,8) je tedy následovný:

p0 = p1 ⊕ p2 ⊕ p4 ⊕ p8

Co se týče implementace, je to pouze přidání jednoho výpočtu navíc ve firmwaru vysílače a jedna kontrola navíc ve firmwaru přijímače. Na krátké vzdálenosti a v nerušených oblastech je toto rozšíření pravděpodobně zbytečné, ale v zastavěných oblastech s vyšším rušením nebo při komunikaci na větší vzdálenost se může hodit.

3.8.3 Obousměrná komunikace

Při implementaci firmwaru vysílače a přijímače jsem měl k dispozici pouze po jednom modulu vysílače a přijímače. Tím je celá komunikace implementována pouze jednosměrně a to má za důsledek ztrátu paketu při neúspěšném přenosu. Protože přijímač nemůže požádat vysílač o znovu-odeslání paketu, dochází tak k nepříjemné ztrátě informací Neúspěšný přenos může nastat v několika případech:

- **chyby v preambuli způsobené rušením** – paket nebude vůbec přijat
- **vícebitové chyby v paketu** – data v paketu nebude možné opravit Hammingovým kódem
- **nekompletní paket** – rušením bude načten stop bit dříve, než budou obdržena všechna data

Pokud by byly k dispozici dva vysílací a dva přijímací moduly, nebylo by příliš složité upravit celou implementaci tak, aby v případech neúspěšného přenosu byl paket odeslán znova. Když bych se zaměřil na tři výše zmíněné případy špatného přenosu, volil bych úpravy komunikačního protokolu následovně.

Nejdříve je nutné rozšířit hlavičku paketu o nové pole, nesoucí identifikátor paketu. Pro menší projekty by stačila 8bitová hodnota, která by nám tak dala 256 hodnot pro číslování paketu, které by se mohly ve smyčce recyklovat. Pro rozsáhlejší projekty by bylo vhodné volit násobky 8, pro snadnější zakódování do Hammingova kódu.

Dále by bylo zapotřebí přidat dvě nové instrukce, které by určovaly buď potvrzení správně přijatého paketu nebo naopak požádání vysílače o znovu-odeslání paketu. Obě instrukce by v hlavičce nesly vždy číslo potvrzovaného paketu a také svůj instrukční kód. Ve své podstatě by se jednalo o obdobu synchronní komunikace, kdy každý úspěšně přenesený paket je odesílací straně buď potvrzen odesláním *ACK* nebo je naopak indikován neúspěšný přenos odesláním *NACK*.

V rámci firmwaru vysílací strany by také musel být doplněn buffer pro veškeré odeslané pakety. Nejvhodnějším řešením by bylo buď pole struktur o velikost maximálního počtu paketů, (pokud bychom měli k dispozici dostatek paměti), anebo jednosměrný seznam, do kterého by se odeslané pakety vkládaly. Po každém odeslání by byl celý paket uložen do tohoto bufferu a až v případě, že vysílací strana obdrží od přijímací strany potvrzení o úspěšném přenosu, mohl by tento paket z bufferu smazat pro šetření pamětí.

Tímto by byly vyřešeny problémy s neúplnými pakety – přijímač by kontroloval přijatou délku paketu s hodnotou délky paketu uvedenou v hlavičce obdrženého paketu, stejně jako problém s vícebitovými chybami v paketu. Pro případ narušené preambule by bylo zapotřebí ještě přidat v bufferu odeslaných paketů vysílače ke každému paketu časové razítko odeslání, které by se po určitých časových krocích kontrolovalo. Pokud by v uživatelem zadaném časovém rozmezí nedošlo k přijetí *ACK* nebo *NACK* potvrzení daného paketu, byl by opětovně odeslán a taktéž by bylo aktualizováno jeho časové razítko.

---

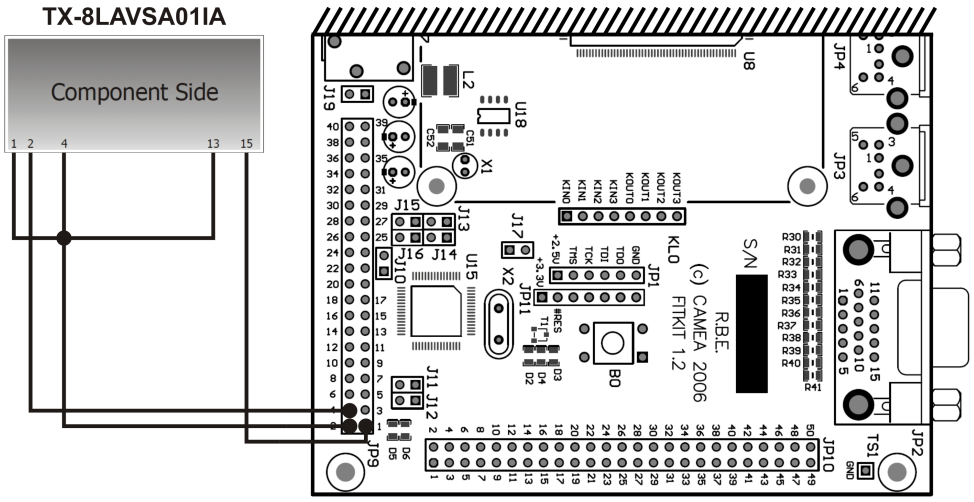
7 *Single Error Correction, Double Error Detection* – volným překladem tedy oprava jednobitových chyb, detekování dvoubitových chyb

4 Hardware a testování

Tato pasáž se věnuje popisu samotného zapojení modulů k FITkitům [7] tak, aby byla zajištěna korektní funkčnost. Dále také obsahuje i stručné informace o testech implementace firmwaru, kterými jsem ověřoval hlavně samotný bezdrátový přenos. Cílem testování bylo zjistit, na jaké vzdálenosti je možné komunikaci provozovat, jak se zvětšující vzdálenost projeví na kvalitě přenosu a také jestli vzdálenost razantně ovlivní počet chyb při přenosu.

4.1 Vysílací modul

Vysílání signálu obstarává modul firmy Aurel s označením TX-8LAVSA01IA, který vysílá na frekvenci 868,3 MHz. Tento modul má, oproti přijímacímu, výhodu většího rozsahu napájecího napětí, přičemž optimální napětí je +3 V [11]. Je tedy možné modul napájet přímo z FITkitu, pomocí pinů 1 (+3,3 V) a 2 (GND) patice JP9. Tento modul má anténu zabudovanou, není tedy potřeba její externí připojení. Vysílací modul pracuje se signálovou modulací OOK (ON-OFF keying), což je základní varianta ASK modulace. Vstupní data pro modul jsou přivedena z pinu 4 patice JP9. Modul je osazen v nepájivém kontaktním poli a s FITkitem propojen pomocí propojovacího USB kabelu pro interní připojení k základní desce osobního počítače. Schéma zapojení je vidět na obrázku 4.1.

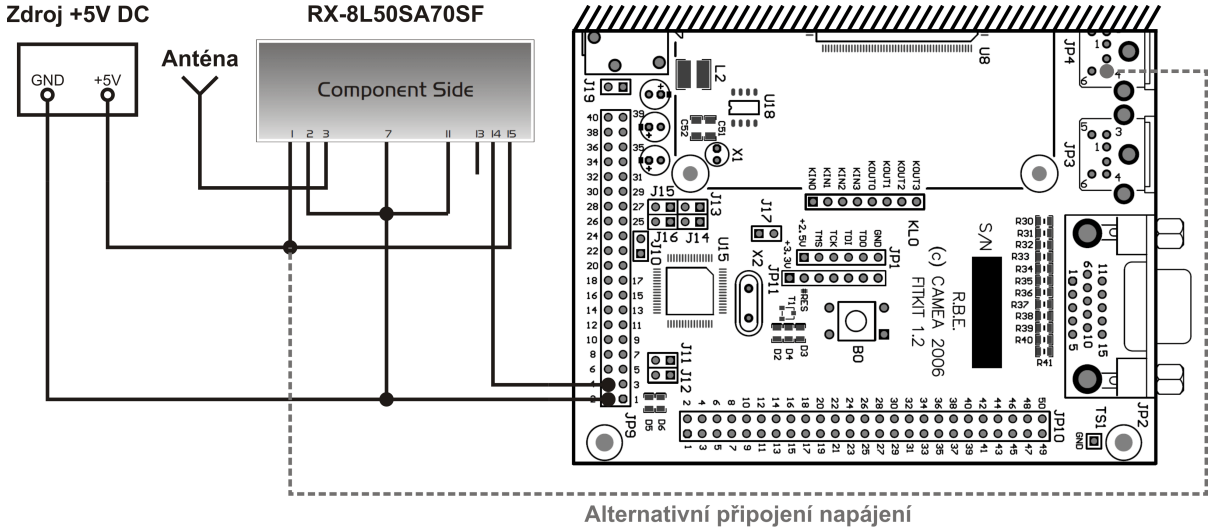


Obr. 4.1: Schema zapojení vysílacího modulu

4.2 Přijímací modul

Pro příjem signálu byl zvolen modul firmy Aurel s označením RX-8L50SA70SF, který pracuje na frekvenci 868,3 MHz. Tento modul, na rozdíl od vysílacího, vyžaduje vyšší provozní napájení, které však na patici JP9 FITkitu není k dispozici. K napájení je optimálně zapotřebí stejnosměrné napětí +5 V s možnou odchylkou ±0,25 V, viz. tabulka parametru v datasheetu modulu [10]. Z tohoto důvodu jsem použil vlastní externí zdroj napájení, generující stabilních +5,04 V. Možnou alternativou, v případě nedostupnosti vlastního zdroje napájení modulu, je získání +5V z pinu číslo 4 na PS/2 konektoru dostupného na FITitu, avšak toto řešení se mi nezdá příliš vhodné. Jako záložní řešení však funguje bez potíží.

Při následné implementaci a zprovozňování komunikace se také objevil problém plovoucí log. 1 na datovém výstupu modulu. Bohužel varianta MSP430 osazená na FITkitu v1.2, nemá k dispozici pullup/pulldown rezistory, proto byl tento problém vyřešen propojením uzemnění externího zdroje s uzemněním FITkitu přes pin 2 patice JP9. Výstupní data z modul jsou připojena na pin 4 patice JP9. Modul je, stejně jako modul vysílače, osazen v nepájivém kontaktním poli a s FITkitem propojen pomocí propojovacího USB kabelu pro interní připojení k základní desce osobního počítače. Schéma zapojení je vidět na obrázku 4.2, včetně vyznačení alternativního připojení napájení přijímacího modulu.



Obr. 4.2: Schema zapojení přijímacího modulu

4.3 Testování implementace

Při testování jsem se chtěl zaměřit hlavně na dosah komunikace a také na chybovost komunikace při rozdílných vzdálenostech. Testování jsem prováděl na třech různých vzdálenostech komunikujících modulů a dvou různých druzích paketů. Testované vzdálenosti byly 1 metr, 10 a 50 metrů. Pro jednu sadu testů jsem použil instrukci USERCMD3, tedy paket bez přídavných dat – zasílala se jen hlavička paketu. Druhá sada testů probíhala s paketem ULONG, tedy paket nesoucí číslo ve tvaru **unsigned long.**

Test na vzdálenost jednoho metru byl prováděn na přímou viditelnost, na psacím stole, vzdálenost 10 metrů v rámci rodinného domu, tedy přes stěny a s jistou možností rušení jinými elektrospotřebiči a vzdálenost 50 metrů byla testována na zahradě, tedy přímá viditelnost, zanedbám-li stromy. Při všech testech jsem k modulu přijímače připojil provizorní externí anténu délky 1,5 m, vyrobenou z izolovaného zvonkového drátu průměru [x].

Každý test v rámci jedné vzdálenosti a druhu paketu byl prováděn celkem 5 krát a zjištěné výsledky byly zprůměrovány. Cyklus jednoho testu obsahoval postupné odeslání 100 paketů dané instrukce z vysílacího modulu v rozestupech 500ms. Modul přijímací pak počítal přijaté pakety a taktéž rozlišoval, zda paket došel v pořádku, či nikoliv. Kontrolovány byla jak korektnost hlavičky – tedy, že došel správný paket o správné délce, tak samotná data (u instrukce ULONG) – zda odesílaná hodnota souhlasí s přijatou. Celkem jsem tedy provedl 3000 odeslání paketů (5*3 (vzdálenost)*2 (instrukce)*100 (paketů v jednom testu)), plus navíc 3 testy zvlášť – o těch je zmíněno v sekci 4.3.3. Tabulka všech naměřených hodnot je uvedena v příloze D.

4.3.1 Rádiový dosah přenosu

Rádiový dosah přenosu je poměrně uspokojující, vezmu-li v potaz fakt, že vysílací modu disponuje pouze integrovanou anténou a modul přijímací improvizovanou drátěnou anténou. V procentech jsem z naměřených hodnot vyjádřil, nehledě na korektnost obsahu paketu, že byl paket přijat přijímačem. Souhrn těchto procentuálních vyjádření je vidět v tabulce 4.1. každý zbytek procent, který chybí do 100%, představuje pakety, které nebyly doručeny vůbec – nebyla správně zachycena preambule a paket tak nebyl vůbec zpracován a započítán. Tento fakt může být zaviněn právě vysokým požadavkem na zachycení preambule – vysílač odesílá 102 jedniček v Manchester kódu a přijímač pro úspěšné zachycení paketu vyžaduje zpracování sledu minimálně 60 po sobě jdoucích jedniček v tomtéž kódu.

Druh paketu	Vzdálenost		
	1 m	10 m	50 m
USERCMD3	98,6%	93%	93,4%
ULONG	99,4%	93,4%	85,2%

Tabulka 4.1: Procentuální vyjádření úspěšného přenosu

Jak je z tabulky vidět, na krátké vzdálenosti je komunikace velice dobrá, avšak s narůstající vzdáleností i při čisté viditelnosti dochází k poklesu, hlavně pak u paketů nesoucí data. Tato procentuální úspěšnost by určitě šla vylepšit zvýšením tolerance načítané preambule a hlavně použitím kvalitní antény na přijímači. Pokud by vysílací modul měl možnost připojení externí antény, domnívám se, že by to také dosahu pomohlo.

4.3.2 Úspěšnost přenosu

Úspěšnost přenosu jsem také vyjádřil procentuálně. Opět, pro každý druh instrukce a každou vzdálenost, jsem spočítal, kolik procent ze všech přijatých paketů došlo poškozených. Počítání poškozených paketů jsem prováděl i v těch případech, že byly opravitelné pomocí Hammingova kódu. Znovu se ukázalo, že s rostoucí vzdáleností a velikostí paketů se zvětšuje náchylnost k chybám. Souhrn chybovosti vyjádřený v procentech je ukázán v tabulce 4.2.

Druh paketu	Vzdálenost		
	1 m	10 m	50 m
USERCMD3	1,18%	0,564%	0,188%
ULONG	2,38%	5,024%	7,488%

Tabulka 4.2: Procentuální vyjádření výskytu chyby v přenosu

Data použitá pro výpočty brala v potaz pouze doručené pakety. Ty, které nebyly vůbec doručeny jsem do těchto výpočtů nezahrnul. Zaznamenal jsem neúměrnost nárůstu chybovosti u paketů USERCMD3, ale předpokládám, že ve větším měřítku testů by se tyto hodnoty srovnaly na zhruba stejné hranicí. Stejně jako v předchozím případě by použití externích antén a zvýšení tolerance preambule zajisté statistiku těchto testů vylepšily.

4.3.3 Samostatné testy přenosu

Při testování jsem vyzkoušel i 3 druhy testů, kdy jsem ověřoval tyto parametry – maximální dosah a úspěšnost s a bez použití improvizované antény. Všechny tři testy proběhly pouze s paketem typu USERCMD3, tedy bez přídavných dat. První testem byl přenos na 10m uvnitř budovy bez přídavné antény. Jak se ukázalo, kvalita přenosu velmi klesla, obzvláště pak velmi přibylo vůbec nedoručených paketů. Druhým testem, testem vzdálenosti jsem chtěl vyzkoušet hlavně přenos dat na nepřímou viditelnost. Konkrétně se jednalo o přenos na 100 m a přes terénní nerovnost – horizont. Bohužel, u takto malých modulů a s absencí kvalitní vysílací antény, byl přenos v takových podmínkách v podstatě neuskutečnitelný. Poslední pokusem jsem si chtěl ověřit pokus druhý a opakoval jsem test na vzdálenost 75 m s přímou viditelností. Zde byla úspěšnost opět v uspokojivých hodnotách, v podstatě srovnatelné s přenosem na 50m. Všechny hodnoty z těchto tří testů je možné vidět v tabulce 4.3.

10 m bez antény			100 m přes horizont			75 m na přímo		
ALL	GOOD	BAD	ALL	GOOD	BAD	ALL	GOOD	BAD
34	12	22	12	1	1	93	93	0

Tabulka 4.3: Výsledky samostatných testů

5 Závěr

Cílem této práce bylo navrhnout a odladit firmware vysílače a přijímače pro komunikaci v SRD pásmu. Po seznámení se s teorií a standardy ISM pásma a prostudování dokumentace samotných komunikačních modulů a laboratorního zařízení FITkit, jsem postupným obohacováním kódu o nové funkce nakonec dospěl k funkční implementaci.

Navržený firmware v programovacím jazyce C, určený pro mikrokontrolér MSP430, zvládá základní jednosměrnou komunikaci mezi jednotlivými moduly. Pro komunikaci jsou případnému uživateli přístupné jednoduché obslužné funkce pro správné odeslání a příjem dat a odpadá tak potřeba bližšího studování samotného přenosu.

Při testování komunikace, popsaném v sekci 4.3, jsem zjistil, že se navržený firmware v případě komunikace v zastavěné oblasti hodí více pro komunikaci na kratší vzdálenosti do 10 metrů, avšak při venkovním použití je dosah a spolehlivost možné zajistit až na několik desítek metrů. Hlavním problémem při komunikaci je fakt, že vysílací i přijímací modul nedisponují kvalitními anténami a přenášený signál je více náchylný na rušení a těžko se dostává přes fyzické překážky. Opravu paketů pomocí Hammingova kódu nebylo možné prokazatelně ověřit, protože ve většině případů docházelo k blokovým chybám v paketech a ty nebylo možné opravit.

V případě budoucí práce na projektu by bylo vhodné se zaměřit na možnosti rozšiřitelnosti a vylepšení implementace, které jsou popsané v sekci 3.8, jako například obousměrná komunikace a potvrzování paketů. Použití jiných modulů s kvalitnějším výkonem, případně s lepšími anténami by mohlo značně vylepšit kvalitu a vzdálenost přenosu.

Literatura

[1] STALLINGS, William. *Data and computer communications*. 5th ed. Upper Saddle River, N.J.: Prentice Hall, 1997, p. 97-107. ISBN 00-241-5425-3.

[2] *Všeobecné oprávnění č. VO-R/10/04.2012-7 k využívání rádiových kmitočtů a k provozování zařízení krátkého dosahu* [Online]. Český Telekomunikační Úřad, 24. dubna 2012. Dostupné z: <http://www.ctu.cz/cs/download/oop/rok_2012/vo-r_10-04_2012-07.pdf>.

[3] KUCHAŘ, Martin. *Bezdrátová technologie Wi-Fi zbavená roušky tajemství* [Online]. PCTuning.cz, 24. března 2005. Dostupné z: <http://pctuning.tyden.cz/hardware/site-a-internet/4444-bezdratova_technologie_wi-fi_zbavena_rousky_tajemstvi>.

[4] ČEPIČKA, David. *Základy technologie Bluetooth: původ a rozsah funkcí* [Online]. PCWorld.cz, 10. února 2009. Dostupné z: <http://pcworld.cz/hardware/Zaklady-technologie-Bluetooth-puvod-a-rozsah-funkci-6635>.

[5] *Všeobecné oprávnění č. VO-R/12/08.2005-34 k využívání rádiových kmitočtů a k provozování zařízení pro širokopásmový přenos dat na principu rozprostřeného spektra nebo ODFM v pásmech 2,4 GHz a 5 GHz* [Online]. Český Telekomunikační Úřad, 9. srpna 2005. Dostupné z: <http://www.ctu.cz/1/download/Opatreni obecne povahy/VO_R_12_08_2005_34.pdf>.

[6] *Decision 2011/829/EU* [Online]. CEPT ECC, 8. December 2011. Dostupné z: <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2011:329:0010:0018:EN:PDF>.

[7] *FITkit v1.2 schematic* [Online]. Fakulta Informačních Technologií, VUT Brno. Dostupné z: <http://merlin.fit.vutbr.cz/FITkit/download/schematic_v12.pdf>.

[8] WANG, Yin. *MSP430 Clock System and Timer* [Online]. CCIS, Northeastern University, 2007. Dostupné z: <http://www.ccs.neu.edu/home/noubir/Courses/CSU610/S07/MSP430-Clock-Timers.pdf>.

[9] WAGNER, Neal R. *The Laws of Cryptography* [Online]. Department of Computer Science, The University of Texas at San Antonio, 2003, p. 44-47. Dostupné z: <http://www.cs.utsa.edu/~wagner/lawsbookcolor/laws.pdf>.

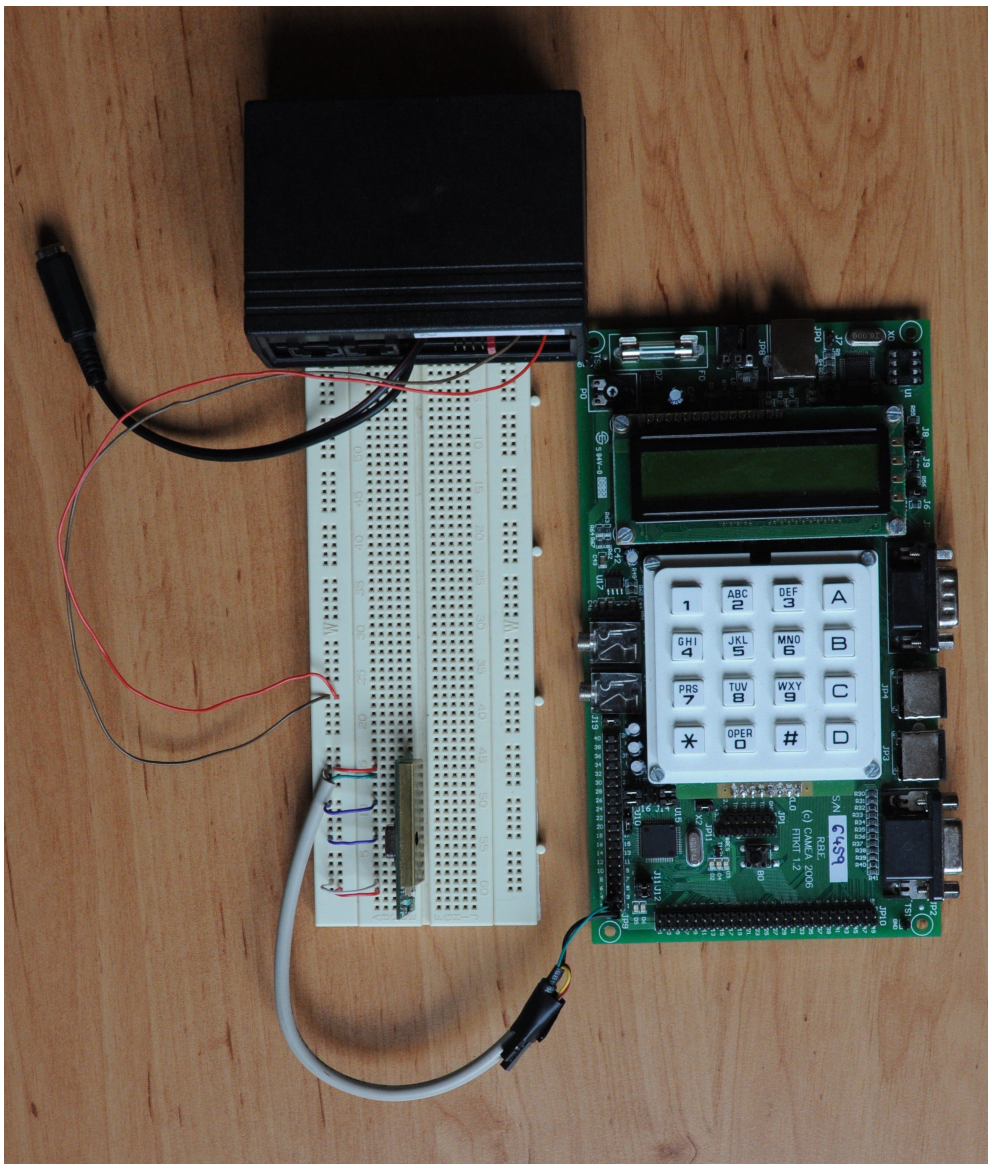
[10] *868 MHz Super-Het AM (OOK) Receiver RX–8L50SA70SF* [Online]. Aurel S.p.A., 2002. Dostupné z: <http://www.ges.cz/sheets/r/rx8l50.pdf>.

[11] *868 MHz SAW RF Transmitter Module TX-8LAVSA01IA* [Online]. Aurel S.p.A., 2002. Dostupné z: <http://www.ges.cz/sheets/t/tx805ia.pdf>.

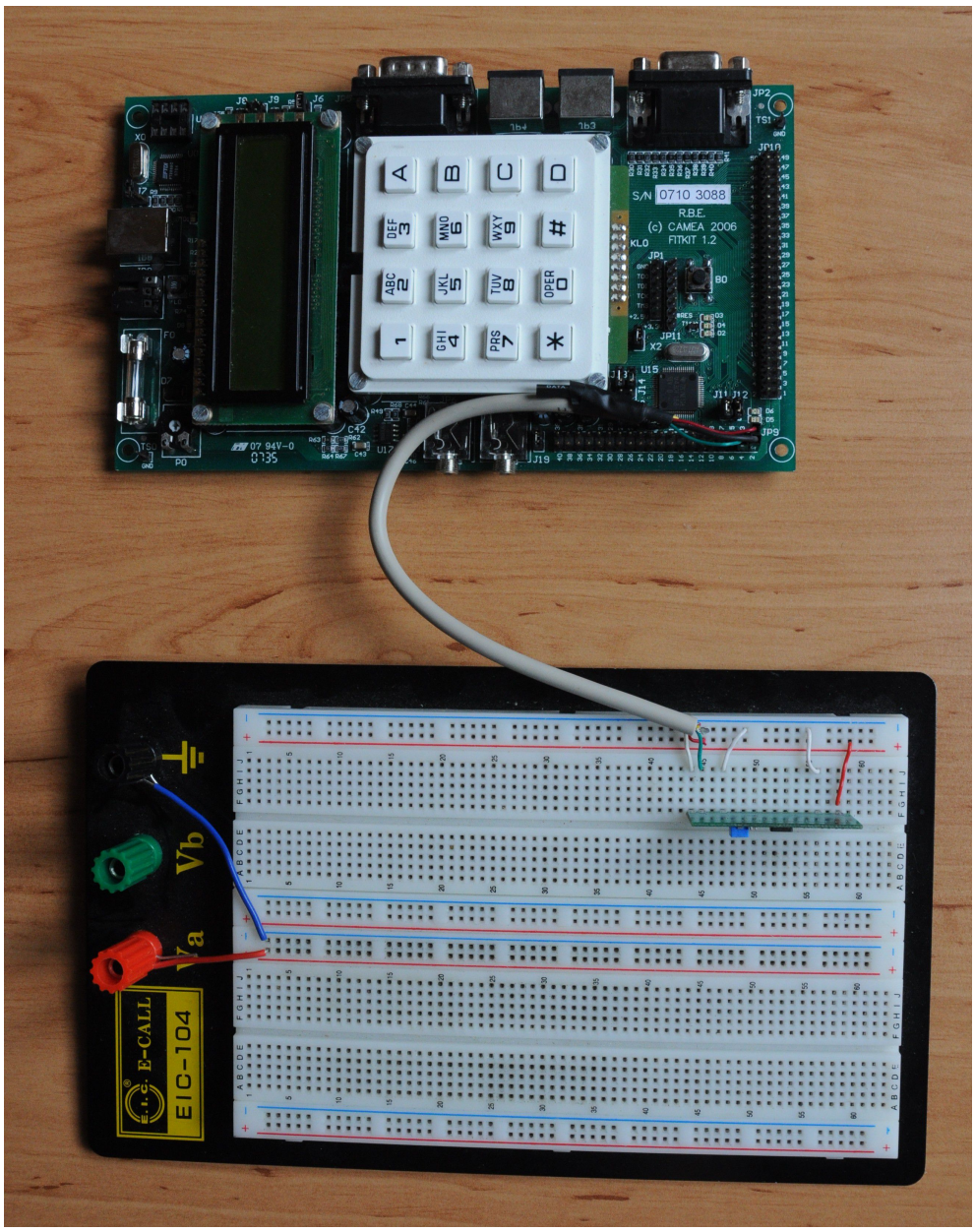
Seznam příloh

Příloha A. Fotodokumentace přijímače
Příloha B. Fotodokumentace vysílače
Příloha C. Algoritmus přijímače načítání z modulu
Příloha D. Tabulka hodnot z testování přenosu
Příloha E. Uživatelská příručka

A. Fotodokumentace přijímače



B. Fotodokumentace vysílače



C. Algoritmus přijímače načítání z modulu

```
int recieve(void) {
      unsigned int sync_reg, i, startbit, preamb, stopbit;
      startbit=preamb=stopbit=timeout=0;
      TACTL = TACLR; //reset casovace pred kazdym novym vysilanim
      CCTL0=CCIE; //povoleni preruseni pri naplneni registru CCR0
      CCR0=16; //nastaveni preddelicky - 32768Hz/16 = 2048Hz
      TACTL = TASSEL_1 + MC_1; //ACLK zdroj, UP mod
      //cekani na preambuli - minimalne 60 jednicek v Manch. kodu
      while(preamb<60)
      {
        com_wait();
        if(rx_input())
        {
          com_wait();
          if(!rx_input()) //nactena jednicka v Manchester kodu
          {
            preamb++; //zvyseni citace preambule
          }
          else
            preamb=0; //sled prerusen - nulovani citace preambule
        }
        if(timeout>1024)
          return 1; //vyprseni tiemout - ukonceni nacitani
      }
      //preambule zachycena, nulovani bufferu
      memset(data_raw,0,sizeof(unsigned int)*168);
      //hledani start bitu
      while(!startbit)
      {
        com_wait();
        if(rx_input())
        {
          com_wait();
          if(rx_input()) //start bit nalezen
          {
            startbit=1;
          }
        }
      }
      i=timeout=0; //nulovani indexu bufferu a timeoutu
      //nacitani obsahu paketu do bufferu
      while(!stopbit && i<168)
      { //nacita dokud není stop bit nebo po max. velikost bufferu
        sync_reg=0;
        com_wait();
        sync_reg |= rx_input(); //prvni polovina cisla manch. kodu
        sync_reg <<= 1;
        com_wait();
        sync_reg |= rx_input(); //druha polovina cisla manch. kodu
```

```
    if(0x01 == sync_reg)
    {
      data_raw[i]=0; //prijata hodnota log.0
    }
    else if(0x02 == sync_reg)
    {
      data_raw[i]=1; //prijata hodnota log.1
    }
    else if(!sync_reg)
    {
      stopbit=1; //prijat stop bit
    }
    i++; //zvyseni indexu bufferu
    if(timeout>1024)
      return 1; //vyprseni tiemout - nezachycen stopbit
  }
  return 0;
}
```

D. Tabulka hodnot z testování přenosu

Test č.	1 m					
	ULONG			USERCMD3		
	ALL	GOOD	BAD	ALL	GOOD	BAD
1	100	98	2	98	98	0
2	99	97	2	99	99	0
3	100	98	2	98	95	3
4	100	98	2	100	99	1
5	98	94	4	98	96	2
Dosah	99,4			98,6		
Úspěšnost	2,38			1,18		

Test č.	10 m					
	ULONG			USERCMD3		
	ALL	GOOD	BAD	ALL	GOOD	BAD
1	92	87	5	94	93	1
2	93	86	7	91	91	0
3	90	84	6	93	93	0
4	95	89	6	94	92	2
5	97	94	3	93	93	0
Dosah	93,4			93		
Úspěšnost	5,024			0,564		

Test č.	50 m					
	ULONG			USERCMD3		
	ALL	GOOD	BAD	ALL	GOOD	BAD
1	85	72	13	94	93	1
2	88	79	9	93	93	0
3	89	82	7	94	94	0
4	84	78	6	93	93	0
5	80	71	9	93	93	0
Dosah	85,2			93,4		
Úspěšnost	7,488			0,188		

Samostatné měření								
10 m bez antény			75 m na přímo			100 m přes horizont		
ALL	GOOD	BAD	ALL	GOOD	BAD	ALL	GOOD	BAD
34	12	22	93	93	0	12	1	11

E. Uživatelská příručka

Firmware sám o sobě nemá svou předem danou funkčnost. Obsahuje pouze sadu funkcí pro bezdrátovou komunikaci, kterou může uživatel použít k implementaci vlastních projektů. V případě vysílače to jsou funkce:

```
void send_NODATA(instructions code);
void send_USERDATA(instructions code, char *data);
void send_INT(int number);
void send_UINT(unsigned int number);
void send_LONG(long number);
void send_ULONG(unsigned long number);
void send_DOUBLE(double number);
void send_STRING(char *retezec);
```

Více informací k těmto funkcím je možné nalézt v hlavičkovém souboru `srd_tx.h` ve zdrojových souborech vysílače. Stejně tak je tomu na straně přijímače, kde je opět připravena sada funkcí na dekódování přenesených dat, spolu se switch diagramem ve funkci `parsePacket()`. Opět podle druhu přijatého paketu jsou volány obslužné funkce zajišťující správné zpracování přijatého paketu a uživateli pomocí návratové hodnoty vrací konkrétní data:

```
int parse_INT(unsigned int *pole);
unsigned int parse_UINT(unsigned int *pole);
long parse_LONG(unsigned int *pole);
unsigned long parse_ULONG(unsigned int *pole);
double parse_DOUBLE(unsigned int *pole);
void parse_STRING(unsigned int *pole, char *zprava, unsigned int delka);
```

Bližší informace jsou k dispozici po prostudování algoritmu funkce `parsePacket()`. Lze také nahlédnout na popis obslužných funkcí v hlavičkovém souboru `srd_rx.h` ve zdrojových souborech přijímače.

Správné zapojení modulů k FITkitům je popsáno v sekci 4.1 a 4.2.