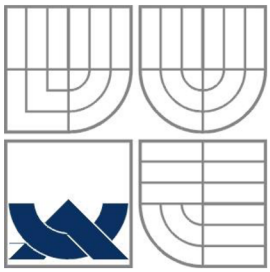
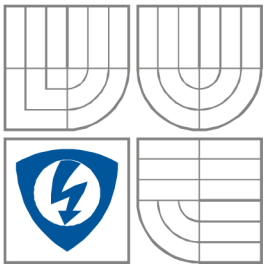


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF RADIO ELECTRONICS

POHYBOVÝ SENZOR S IDENTIFIKACÍ

MOTION SENSOR WITH IDENTIFICATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Václav Halbich
AUTHOR

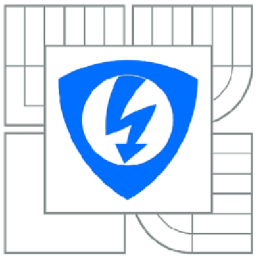
VEDOUCÍ PRÁCE Ing. Zbyněk Fedra, Ph.D.
SUPERVISOR

BRNO, 2013

Výzkum realizovaný v rámci této diplomové práce byl finančně podpořen projektem CZ.1.07/2.3.00/20.0007 **Wireless Communication Teams** operačního programu **Vzdělávání pro konkurenceschopnost**.

Finanční podpora byla poskytnuta Evropským sociálním fondem a státním rozpočtem České republiky.

Tento příspěvek vzniknul za podpory projektu CZ.1.07/2.3.00/20.0007 WICOMT, financovaného z operačního programu Vzdělávání pro konkurenceschopnost



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Diplomová práce

magisterský navazující studijní obor
Elektronika a sdělovací technika

Student: Bc. Václav Halbich ***ID:*** 115173
Ročník: 2 ***Akademický rok:*** 2012/2013

NÁZEV TÉMATU:

Pohybový senzor s identifikací

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s možnostmi RF senzoru společnosti APLS Electric. Navrhněte systém detekce pohybu s možností itentifikace pohybujícího objektu založený na možnostech tohoto senzoru.
Otestujte možnosti vašeho návrhu. Systém by měl využívat možnosti paralelní detekce pohybu a rádiové digitální komunikace, kterou modul umožňuje, k tomu, aby dokázal odlišit obecný pohyb od pohybu objektu (osoba, vozidlo), který má u sebe aktivní klíč. Součástí práce je návrh identifikačního a komunikačního protokolu mezi senzorem a klíčem a implementace tohoto protokolu do firmware pro senzor i klíč.
Zohledněte bezpečnost protokolu, optimalizaci návrhu pro bateriově napájený klíč a vlastní HW klíče. Zvažte možnost úpravy protokolu pro větší počet klíčů a pro správu nastavení senzoru. Výsledkem by měla být jednoduchá demonstrace celého systému.

DOPORUČENÁ LITERATURA:

[1] HAGEN, W. The Definitive Guide to GCC. 2nd ed. New York: Apress, 2006.

[2] MANN, B. C pro mikrokontroléry.Praha: BEN - technická literatura, 2003.

Termín zadání: 11.2.2013 ***Termín odevzdání:*** 24.5.2013

Vedoucí práce: Ing. Zbyněk Fedra, Ph.D.
Konzultanti diplomové práce:

prof. Dr. Ing. Zbyněk Raida
Předseda oborové rady

ABSTRAKT

Návrh a implementace systému datové komunikace s paralelním snímáním pohybu na stejném rádiovém kanálu. Systém dokáže odlišit obecný pohyb od pohybu s aktivním klíčem. Diplomová práce je postavena na RF senzorovém modulu od ALPS electric (produkt stále ve vývojové fázi). Po obecném úvodu popisujícím senzor a interní transciever Nordic nRF24LE1 (fyzická vrstva) je v práci popsán návrh paketového komunikačního protokolu neovlivňujícího detekci pohybu (komunikační vrstva), FW návrh aplikace pro senzor a klíč identifikačního protokolu (aplikační vrstva) a HW návrh prototypu aktivního klíče. Každá sekce je doplněna sérií testů.

KLÍČOVÁ SLOVA

RF senzor, ALPS, nRF24LE1, RF identifikace.

ABSTRACT

The solution of the motion detection system with the parallel radio identification on the same channel/carrier. The work is based on the RF sensor from ALPS Electric (product still in development phase), which works as an RF motion sensor and a digital transceiver as well. System can differenitiate general moving object from an object with active key. First part of this work describes the sensor module and the internal Nordic nRF24LE1 transciever (physical layer). Design of packet based communication protocol which does not influence motion sensing (communication layer) is discussed. Next part consist of FW design of application and identification protocol for sensor and key (application layer) and reference HW design of the active key. There are series of tests in every section.

KEYWORDS

RF motion sensor, ALPS Electric, Nordic nRF24LE1, RF identification.

Halbich, V. *ALPS RF pohybový senzor s identifikací*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky, 2012. 65 s., 8 s. příloh. Diplomová práce. Vedoucí práce: ing. Zbyněk Fedra, PhD.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma Pohybový senzor s identifikaci jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce ing. Josefovi Nevrlému za účinnou metodickou a odbornou pomoc a další cenné rady při zpracování tohoto projektu.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

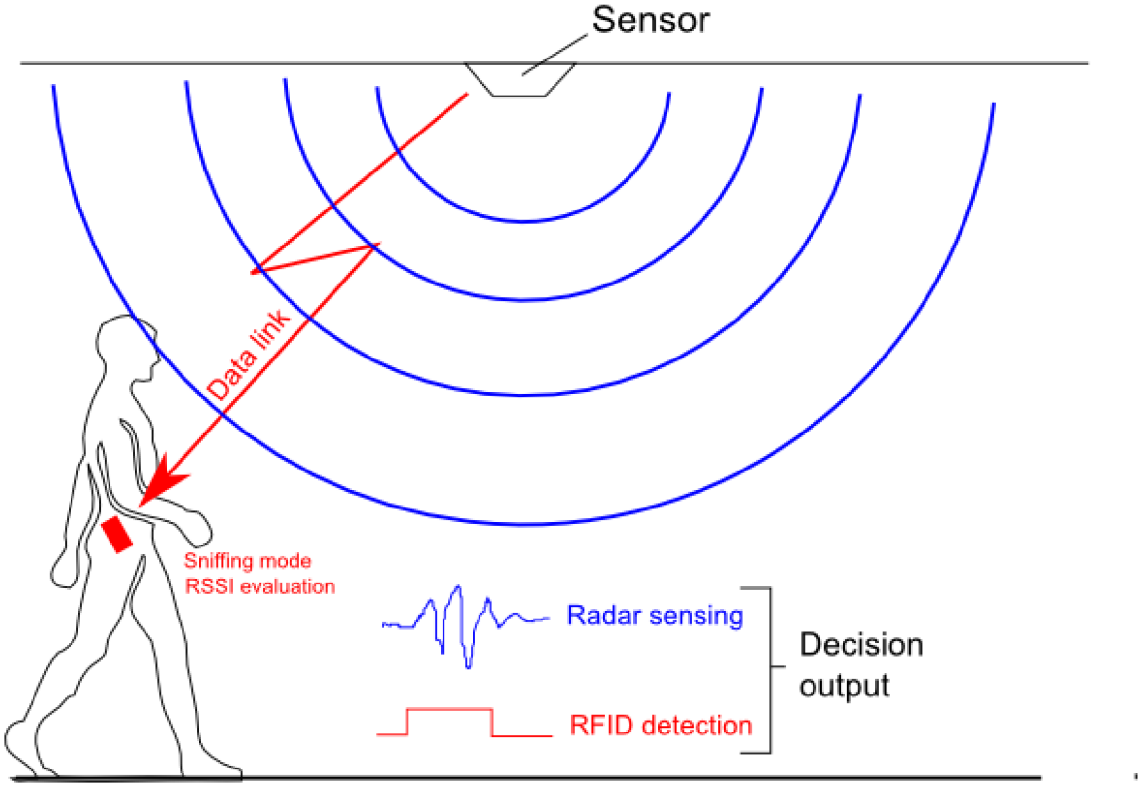
SEZNAM TABULEK

ÚVOD

Diplomová práce je řešením systému detekce pohybu doplněného o rádiovou identifikaci na stejném rádiovém kanálu, založeném na RF senzoru modulu společnosti ALPS Electric. Tento modul funguje paralelně jako RF pohybový senzor a digitální transciever.

Výsledkem diplomové práce je funkční systém (viz **Chyba! Nenalezen zdroj odkazů.**). Systém se sestává z pevně umístěného senzoru a přenosného aktivního klíče. Oba prvky využívají stejný modul. V případě detekce pohybu objektu (osoba, vozidlo), který má u sebe klíč, dojde k ověřovací komunikaci mezi senzorem a klíčem, a v případě pozitivní identifikace senzor provede požadovanou akci (řízení periferie apod.). Může se jednat například o ovládání osvětlení, nebo otvírání garážových vrat atp.

Obsahem práce je především tvorba firmware pro čip Nordic nRF24LE1, který v ALPS modulu zajišťuje funkci transcieveru a zároveň zdroje RF vstupu pro senzor. Důraz je kladen na zajištění řádného souběhu senzorické a komunikační funkce, a na bezpečnost této komunikace. K tomuto účelu by mělo být využito HW kódování a dekódování implementované na nRF24LE1. Kvůli testování byl navržen a implementován prototyp aktivního klíče.



Obr. 1.1 Výsledný systém (převzato z [4])

1 nRF24LE1

Následující kapitola popisuje čip nRF24LE1, bezdrátový systém s důrazem na nízkou spotřebu umístěný na čipu norské společnosti Nordic Semiconductor.

1.1 Základní vlastnosti

Zde jsou přehledně uvedeny základní vlastnosti čipu nRF24LE1 [1].

- nRF24L01+ 2.4 GHz transceiver (přenosové rychlosti: 250 kbps, 1 Mbps a 2 Mbps)
- Rychlý mikrokontrolér (kompatibilní s 8051)
- 16 kB programové paměti (on-chip Flash)
- 1 kB datové paměti (on-chip RAM)
- 1 kB NV datové paměti
- 512 bytů NV datové paměti (s rozšířenou výdrží)
- AES šifrovací HW akcelerátor
- 16-32bit násobící/dělící co-procesor (MDU)
- 6-12 bit ADC
- Vysoce flexibilní IOs
- Nabízí několik výkonových módů, od ultra nízké spotřeby, až k pracovně účinnému aktivnímu módu.
- Několik verzí v různých QFN pouzdrech:
  - 4×4mm QFN24
  - 5×5mm QFN32
  - 7×7mm QFN48
- Podpora pro HW debugger
- HW podpora pro aktualizaci firmwaru

1.2 Komunikace s RF transceiverem

Komunikace mezi samotným transceiverem a MCU probíhá interně pomocí sběrnice SPI, kde MCU je master a transceiver je slave. Nicméně pro nastavení registrů transceiveru a samotné posílání a přijímání dat není potřeba řešit tuto komunikaci na úrovni SPI, protože je k dispozici knihovna od výrobce čipu Nordic Semiconductor, která obsahuje funkce pro jednotlivé operace [1].

1.3 Šifrování

Pro zajištění bezpečnosti rádiové identifikace, jako prevence proti zachycení identifikační sekvence a následně zneužití třetí stranou bude využito šifrování pomocí vestavěného šifrovacího/dešifrovacího akcelerátoru. Akcelerátor je 8 na 8 Galois Field Multiplier s 8 bitovým výstupem. Je použit následující polynom (1.1) používaný standardem AES [1].

$$m(x) = x8 + x4 + x3 + x + 1 \tag{1.1}$$

Samotné provedení šifrování je provedeno pomocí knihoven od Nordic Semiconductor.

1.4 Přehled spotřeby

V následující tabulce (viz Tab. 1.1) je uvedena spotřeba proudu pro některé vybrané situace, ve kterých se může čip nacházet.

Tab. 1.1 Spotřeba proudu pro různé situace (převzato z [1])

Módy	Spotřeba	Poznámka
Deep sleep	0,5uA	
Standby	1mA	
Active	2,5mA	
TX (P_{OUT} = 0dBm)	11,1mA	a
TX (P_{OUT} = -6dBm)	8,8mA	
TX (P_{OUT} = -12dBm)	7,3mA	
TX (P_{OUT} = -18dBm)	6,8mA	
TX (P_{OUT} = -6dBm) průměrně při ShockBurst	0,12mA	
Během nastavování TX	7,8mA	b
RX mode (2Mbps)	13,3mA	
RX mode (1Mbps)	12,9mA	
RX mode (250kbps)	12,4mA	
Během nastavování RX	8,7mA	c

a) Údaje pro impedanci antény Z = 15 + j88 Ω

b) Průměrná spotřeba proudu při nastartování TX módu (130 μs) a při změně z RX módu do TX módu (130 μs).

c) Průměrná spotřeba proudu při nastartování RX módu (130 μs) a při změně z TX módu do RX módu (130 μs).

2 RF TRANSCEIVER

Transceiver je umístěný na jediném čipu nRF24L01+. Je navrhnut pro práci v mezinárodním bezlicenčním frekvenčním spektru ISM od 2,400GHz do 2,835GHz. Je vhodný pro využití v bezdrátových aplikacích dbajících na extra nízkou spotřebu. Modul RF transceiveru je nastavován a spravován pomocí vlastní mapy registrů. MCU přistupuje k těmto registrům pomocí na čipu umístěném SPI rozhraní. Přístup je možný ve všech operačních módech transceiveru.

Má implementovaný protokol Enhanced ShockBurst™, o kterém bude blíže pojednáno níže, a který umožňuje komunikaci pomocí datových paketů a podporuje různé módy použití, od manuálně řízených operací až k pokročilým autonomním operacím protokolu. Přechodné úložiště dat FIFO v modulu RF transceiveru umožňují hladký tok dat mezi modulem transceiveru a nRF24LE1 MCU.

Následující popis je psán z pohledu modulu RF transceiveru jako jádra a zbytku nRF24LE1 jako vnějšího obvodu připojeného k tomuto modulu [1].

2.1 Transceiver obsahuje

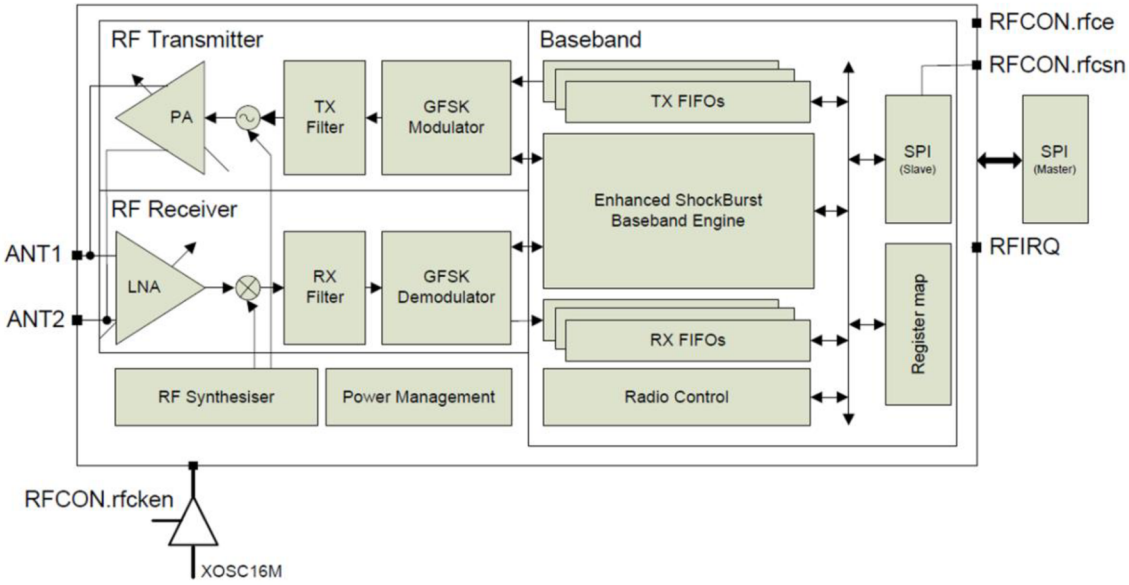
Přehled obsahu transceiveru [1].

- Obecné
  - Pracuje v mezinárodním 2.4 GHz ISM pásmu
  - Společná anténa pro vysílač i přijímač
  - GFSK modulace
  - Přenosové rychlosti 250kbps, 1 a 2Mbps
- Vysílač
  - Programovatelný výstupní výkon: 0, -6, -12 nebo -18dBm
  - 11.1mA pro 0dBm výstupní výkon
- Přijímač
  - Integrované kanálové filtry
  - 13.3mA pro 2Mbps
  - -82dBm citlivost pro 2 Mbps
  - -85dBm citlivost pro 1 Mbps
  - -94dBm citlivost pro 250 kbps
- RF Syntetizátor
  - Plně integrovaný syntetizátor
  - 1 MHz programovatelné frekvenční rozlišení
  - Využívá úsporný ±60 ppm 16 MHz krystal
  - 1 MHz rozestupy nepřekrývajících se kanálů na 1 Mbps

- 2 MHz rozestupy nepřekrývajících se kanálů na 2 Mbps
- Enhanced ShockBurst™
  - 1 až 32 bytů dynamickou délku dat
  - Automatická tvorba paketů
  - Automatické řízení paketového provozu (auto ACK, auto znovuzaslání)
  - 6 datových spojení zaráz MultiCeiver™ pro 6:1 sítě typu hvězda

2.1.1 Blokové schéma

Na následujícím obrázku (viz Obr. 2.1) je blokové schéma transceiveru.



Obr. 2.1: Blokové schéma RF transceiveru (převzato z [1]).

2.2 Operační módy

V této kapitole budou popsány různé operační módy RF transceiveru a parametry užívané k jejich kontrole. RF transceiver se dá nastavit do následujících módů: power down, standby-I, standby-II, RX (přijímacího) a TX (vysílacího) [1].

2.2.1 Power down mód (vypnuto)

V módu vypnuto je RF transceiver zakázán a má minimální proudovou spotřebu. Všechny hodnoty registrů dostupné z SPI jsou zachovány a SPI může být aktivováno. Časy aktivace jsou v tabulce (viz Tab. 2.2). Mód vypnuto je aktivován nastavením nízké úrovně `PWR_UP` bitu v `CONFIG` registru [1].

2.2.2 Standby módy

Standby-I mód

Nastavením vysoké úrovně `PWR_UP` bitu v `CONFIG` registru vstupuje RF transceiver do standby-I módu. Tento mód je využíván k minimalizaci průměrné proudové spotřeby a zároveň udržování krátkých startovacích časů. Změna na aktivní mód nastane pouze při nastavení bitu `rfce` vysokou úroveň [1].

Standby-II mód

V tomto módu jsou aktivovány extra časovače vyrovnávací paměti, a díky tomu je větší spotřeba proudu, než v módu standby-I. Bit `rfce` musí být udržován na vysoké úrovni během operace PTX s prázdným TX FIFO. Pokud je do TX FIFO nahrán nový paket, PLL neprodleně začne a paket je odeslán po normálním zpoždění PLL.

Hodnoty registrů jsou zachovány na stejném nastavení a SPI může být aktivováno během obou módů [1].

2.2.3 RX mód

Pokud chce být RF transceiver používán jako přijímač, je nutné aktivovat tento mód. Bity `PWR_UP, PRIM_RX, rfce` musí být nastaveny na vysokou úroveň.

V RX módu přijímač demoduluje signály z RF kanálu a neustále posílá data k základnímu zpracování protokolem, který neustále hledá validní paket. Pokud je nalezen (sedí adresa a CRC), obsah paketu předá na do prázdného slotu v RX FIFO. Pokud není žádný slot prázdný, je paket zahozen.

RF transceiver zůstává v tomto módu, dokud ho MCU nenastaví do standby-I módu, nebo do módu vypnuto. Pokud základní protokol dovoluje Enhanced ShockBurst™, může RF transceiver přejít do jiných módů, aby mohl provést všechny úkony tohoto protokolu.

V RX módu je také k dispozici detektor síly přijímaného signálu (RPD). RPD je nastaveno na vysokou úroveň, jestliže detekuje na přijímacím frekvenčním kanálu signál vyšší než -64dBm [1].

2.2.4 TX mód

Mód pro vysílání paketů. Pro aktivování tohoto módu je nutné nastavit bity `PWR_UP` na vysokou hodnotu, `PRIM_RX` na nízkou hodnotu, v TX FIFO musí být naplněno a na bitu `rfce` musí být vysoký pulz delší než 10μs.

RF transceiver zůstává v tomto módu, dokud nedokončí odeslání paketu. Poté, pokud je `rfce=0`, vrátí se do standby-I módu. Pokud je `rfce=1`, rozhoduje zaplněnost TX FIFO. Pokud je prázdné RF transceiver přejde do standby-II módu, jinak zůstává v TX módu a vysílá další paket. V TX módu pracuje PLL vysílače v otevřené smyčce. Je důležité se v tomto módu nedržet déle než 4ms. Pokud se používá Enhanced ShockBurst ™, tak k tomu nikdy nedojde [1].

2.2.5 Tabulka nastavení operačních módů

V následující tabulce (viz Tab. 2.1) je přehledně zobrazené nastavení jednotlivých bitů, pro jednotlivé módy a jejich závislost na zaplnění FIFO.

Tab. 2.1: Nastavení bitů pro jednotlivé módy (převzato z [1])

Mode	PWR_UP	PRIM_RX	rfce	stav FIFO
RX mode	1	1	1	
TX mode	1	0	1	Data v TX FIFO. Vyprázdní celé TX FIFO.
TX mode	1	0	Minimálně 10us pulz 1	Data v TX FIFO. Vyprázdní jedno patro TX FIFO.
Standy-II	1	0	1	Prázdné TX FIFO
Standy-I	1	-	0	Žádný paket k přenosu
Power Down	0	-	-	-

2.2.6 Čas přechodů mezi módy

V tabulce (viz Tab. 2.2) jsou uvedeny časy, za které přejde transceiver z jednotlivých módů do jiných módů.

Tab. 2.2: Čas přechodů mezi módy (převzato z [1])

Módy	Max.	Min.
Power Down -> Standby	1us	
Standby -> TX/RX mode	130us	
Minimum `rfce` na vysoké úrovni		10us
Zpoždění z `rfce` sestupné hrany na 0		4us

2.2.7 Přenosové rychlosti

Přenosové rychlosti modulovaného signálu, které umožňuje RF transceiver pro příjem a vysílání dat. Jsou to 250kbps, 1Mbps, 2Mbps. Nižší přenosová rychlost zajistí lepší citlivost přijímače. Vyšší přenosová rychlost oproti tomu přináší nižší spotřebu proudu a menší šanci na přenosové kolize.

Přenosová rychlost se nastavuje pomocí bitu `RF_DR` v registru `RF_SETUP`. Přijímač i vysílač musí být nastaveny na stejnou přenosovou rychlost [1].

2.2.8 Frekvence RF kanálu

Frekvence RF kanálu určuje střední frekvenci kanálu použitého RF transceiverem. Kanál obsazuje šířku pásma méně než 1MHz pro 250kbps a 1Mbps, a šířku pásma méně než 2MHz pro 2Mbps. V tomto případě musí být vzdálenost sousedních kanálů více než 2MHz. RF transceiver pracuje na frekvencích od 2,400GHz do 2,525GHz. Rozlišení programovatelných frekvencí kanálu je 1MHz.

Frekvence rádiového kanálu je nastavena pomocí `RF_CH` registru. Pro výpočet

nastavené frekvence je použit vzorec (2.1) [1].

$$F_0 = 2400 + RF\_CH \tag{2.1}$$

Vysílač i přijímač musí mít nastavený stejný frekvenční kanál, jinak nejsou schopni vzájemné komunikace [1].

2.2.9 Nastavení výkonového zesilovače

Výkonový zesilovač RF transceiveru lze nastavit pomocí `RF_PWR` bitů v `RF_SETUP` registru na čtyři výkonové úrovně s různou spotřebou stejnosměrného proudu uvedené v následující tabulce (viz. Tab. 2.3).

Tab. 2.3: Nastavení výkonového zesilovače (převzato z [1]).

RF_PWE	RF výstupní výkon	DC spotřeba proudu
11	0dBm	11,1mA
10	-6dBm	8,8mA
01	-12dBm	7,3mA
00	-18dBm	6,8mA

2.3 Enhanced ShockBurst™

Enhanced ShockBurst™ je datová linková vrstva založená na přenosu paketů, která obsahuje automatické sestavení a časování paketů, automatické potvrzení a znovuzasílání paketů. Tento protokol je použit jako základní vrstva pro implementování oboustranně rovnocenného komunikačního protokolu.

Během vyslání zprávy ShockBurst™ sestaví paket a nastaví bity pro vyslání paketu. Během příjmu ShockBurst™ neustále vyhledává v demodulovaném signálu platné adresy. Jakmile je nalezena platná adresa, vezme se zbytek paketu a je proveden kontrolní součet CRC. Pokud je vše v pořádku, jsou data z paketu přesunuty do prázdného slotu RX FIFO.

Enhanced ShockBurst™ je protokol určený odesílání paketů z jednoho transceivru na druhý. Z nichž se jeden chová jako primární přijímač (PRX) a druhý se chová jako primární vysílač (PTX). Paketový přenos je vždy inicializován vysláním paketu z PTX, přenos je hotov když PTX přijme potvrzovací paket (ACK) od PRX. PRX může připojit k potvrzovacímu paketu uživatelská data, což umožňuje oboustrannou datovou komunikaci. PRX nemůže samostatně odeslat zprávu.

Samotná komunikace probíhá následovně:

1. Přenos začne vysláním paketu z PTX do PRX. Enhanced ShockBurst™ automaticky přepne PTX do přijímacího módu pro čekání na ACK paket.
2. Pokud je přijat paket v PRX Enhanced ShockBurst™ automaticky sestaví a vyšle ACK paket do PTX a vrátí se do přijímacího módu.
3. Pokud PTX nepřijme ACK paket, je po nastavitelném zpoždění automaticky znovu odeslán původní datový paket a PTX nastaveno do přijímacího módu pro počkání na ACK.

Je možné naprogramovat počet znovuzaslání a čekání na další znovuzaslání. Všechny automatické akce probíhají bez zásahu MCU na baseband periferii. ShockBurst™ také umožňuje vícekanálovou komunikaci pomocí přepínání jednotlivých kanálů - pipes, kdy více PTX komunikuje s jedním PRX [1].

2.4 Srovnání Enhanced ShockBurst™ a manuálního řízení

Enhanced ShockBurst™ má výhodu, že se jedná o již hotový komunikační protokol, který je využit i jako základní komunikační vrstva pro aplikaci pevného senzoru a mobilního klíče. Takové využití však komplikuje fakt, že celý protokol je původně určen pro jednoduché systémy dálkového ovládání (DO), a v systému senzor – klíč jsou role PTX – PRX obráceny.

Ačkoliv se od klíče očekává, že bude plnit i funkci DO, je nutné, aby neustále vysílal senzor a klíč by pouze v určitých intervalech přepnul ze standby módu do přijímacího módu. Pokud by přijal nějaký signál od senzoru, tak by jako odpověď zaslal své identifikační data. Tato funkcionalita vyžaduje nutnost značných změn ve využití protokolu.

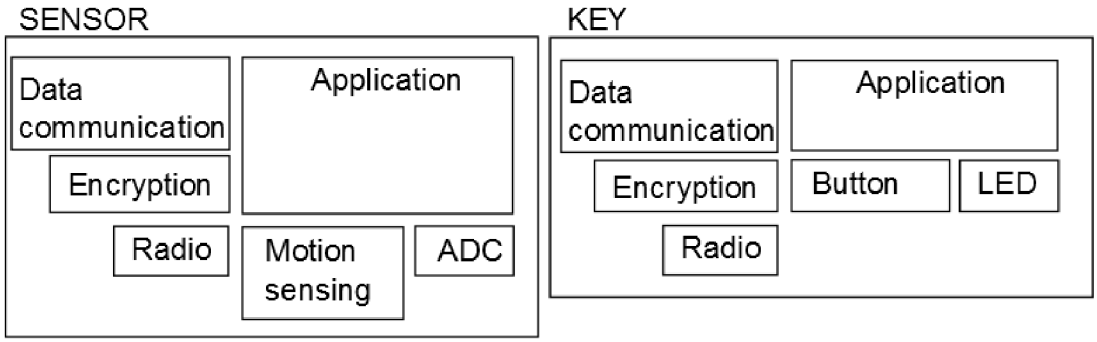
Nicméně se ukázalo, že je výhodnější celou komunikační vrstvu implementovat pomocí Enhannced ShockBurst™ než za použití radio funkcí nižší úrovně.

3 ROZBOR SYSTÉMU

Rozbor jednotlivých součástí systému z různých úhlů pohledu. Součástí kapitoly je softwarový blokový diagram, organizační diagram hierarchie řízení, softwarový diagram z pohledu vrstev. Na závěr je vysvětlen princip měření pohybu.

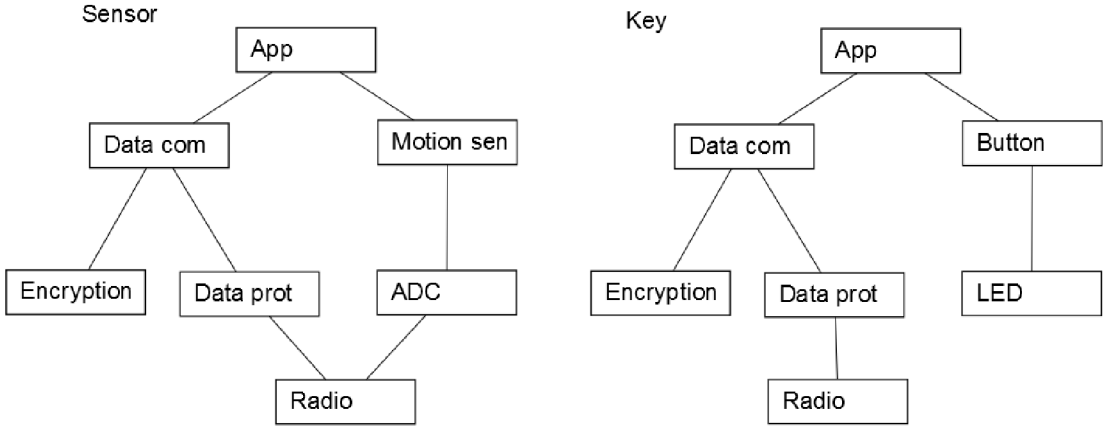
3.1 Softwarový rozbor

Senzor má dva vstupy informací, jednak informaci o pohybu v blízkosti senzoru a jednak informaci z klíče. Tyto informace obstará pohybový senzor přes ADC převodník a datová komunikace. Komunikace je kódovaná. Tyto data se vyhodnocují v aplikační vrstvě programu a následně se vykoná požadovaný úkon. U klíče je situace jednodušší, pouze se předávají data a vyhodnocuje se jenom stisk tlačítka, který zadává úkol klíči, přijetí úkolu je na klíči signalizováno bliknutím LED. Z obrázku (viz Obr. 3.1) jsou patrné základní softwarové bloky systému.



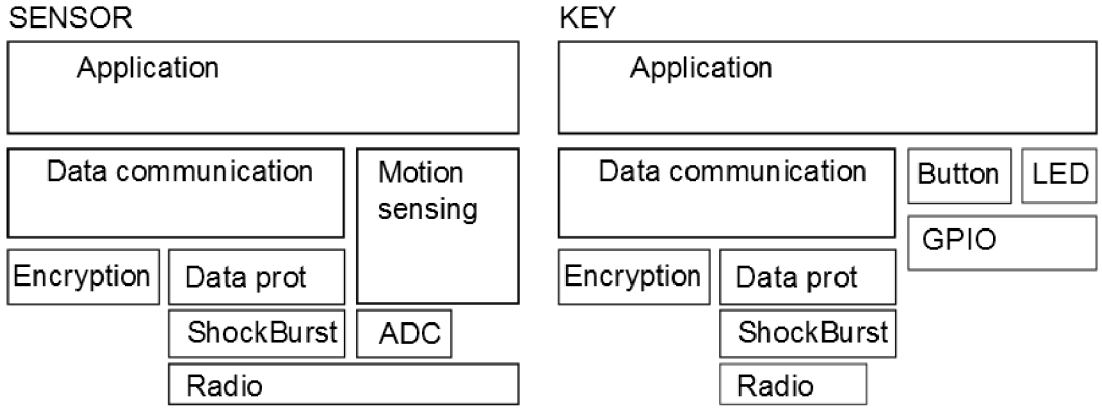
Obr. 3.1 Softwarové blokové schéma

Nyní se na systém podíváme z hlediska hiearchie řízení. U senzoru i klíče je na vrcholu programu aplikační vrstva. U senzoru má pod sebou datovou část a část snímání pohybu. Rádio je využíváno jak pro datovou komunikaci, tak pro detekci pohybu. U klíče je obsluhována pouze datová komunikace a stisk tlačítka. Hiearchie programu je patrná z obrázku (viz Obr. 3.2).



Obr. 3.2 Hiearchie programu

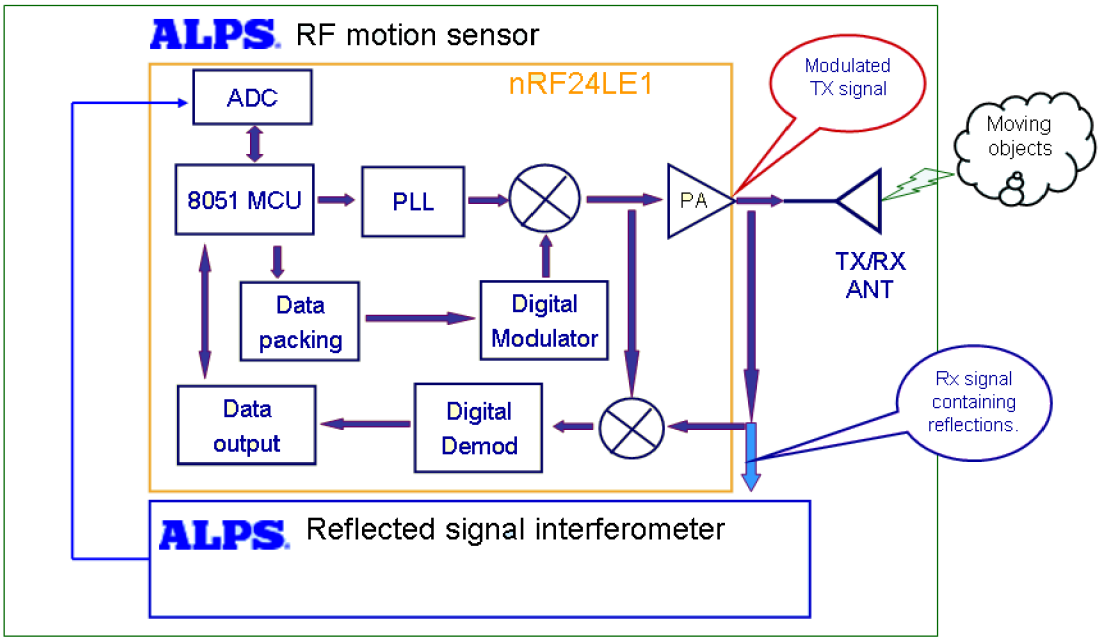
V následujícím odstavci je podrobnější pohled na systém z hlediska vrstev softwaru. Z obrázku (viz Obr. 3.3) je patrné, že datová komunikace bude využívat již na čipu implementovaný Enhanced ShockBurst™. Pro tlačítko a LED je naprogramováno několik GPIO pinů. Jinak platí, co je řečeno výše.



Obr. 3.3 Vrstvy softwaru

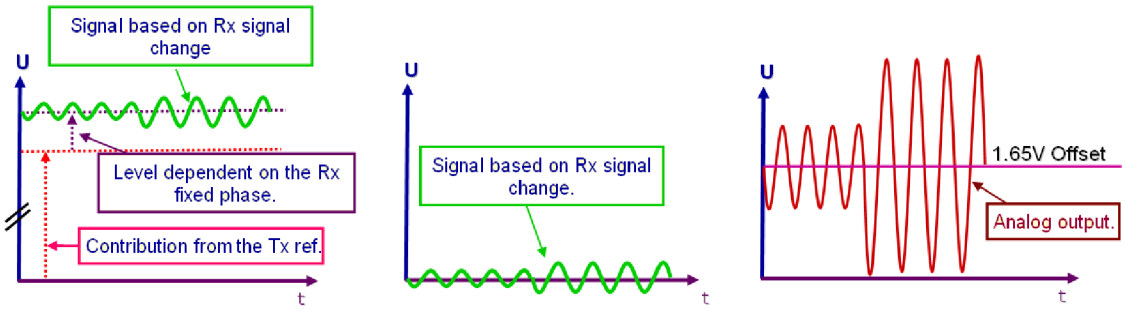
3.2 Senzor pohybu

Měření pohybu je prováděno pouze na senzoru a to pomocí interferometru. Prototyp senzoru je již hotov od firmy ALPS, na obrázku (viz Obr. 3.4**Chyba! Nenalezen zdroj odkazů.**) je blokové schéma senzoru s interferometrem.



Obr. 3.4 Blokové schéma ALPS senzoru (převzato t [4])

Na vstupu interferometru je fázový komparátor a na něj je přiveden vyslaný signál a přijatý signál odražený od okolních objektů. Výstupem z komparátoru je napěťový signál závislý na velikosti kombinovaného signálu. Změna fáze přijatého signálu oproti vyslanému signálu způsobí změnu velikosti napětí kombinovaného signálu a to vede ke změně výstupního napětí. Odraz od nepohybujících se objektů nezpůsobí změnu fáze, na výstupním signálu komparátoru se podílí jako stejnosměrná složka. Odraz od pohybujících se objektů způsobí postupné proměňování fáze odraženého signálu a to způsobí zvlnění na výstupním signálu. Na obrázku (viz Obr. 3.5) je vidět postup zpracování výstupního signálu. Nejprve je odstraněna stejnosměrná složka, a poté je signál zesílen a je přidán 1,65V offset. Toto je výsledný výstup interferometru, který je níže měřen a který jde do AD převodníku.



Obr. 3.5 Zpracování výstupu komparátoru (převzato z [4])

3.2.1 Synchronizace a pseudopohyb

Interferometr pro měření pohybu vyžaduje odesílání zpráv ze senzoru v pravidelných intervalech. Nezáleží na tom, jaká data jsou ve vyslaném burstu přenášena, důležitá je pravidelnost intervalu odesílání zpráv, a pokud možno konstatní energie zprávy (počet jedniček a nul ve dvou po sobě jdoucích zprávách, o tomto je pojednáno v kapitole 4.3.1).

Pokud není dodržen pravidelný interval vysílaní, dochází v interferometru k výkyvům napětí, které se na výstupu jeví obdobně jako výkyvy způsobeny detekovaným pohybem. Tento zásadní problém této metody detekce nazýváme pseudopohybem. V průběhu návrhu FW je jedním ze základních kritérií zamezit výskytu tohoto problému.

4 TESTOVÁNÍ VLASTNOSTÍ MODULU

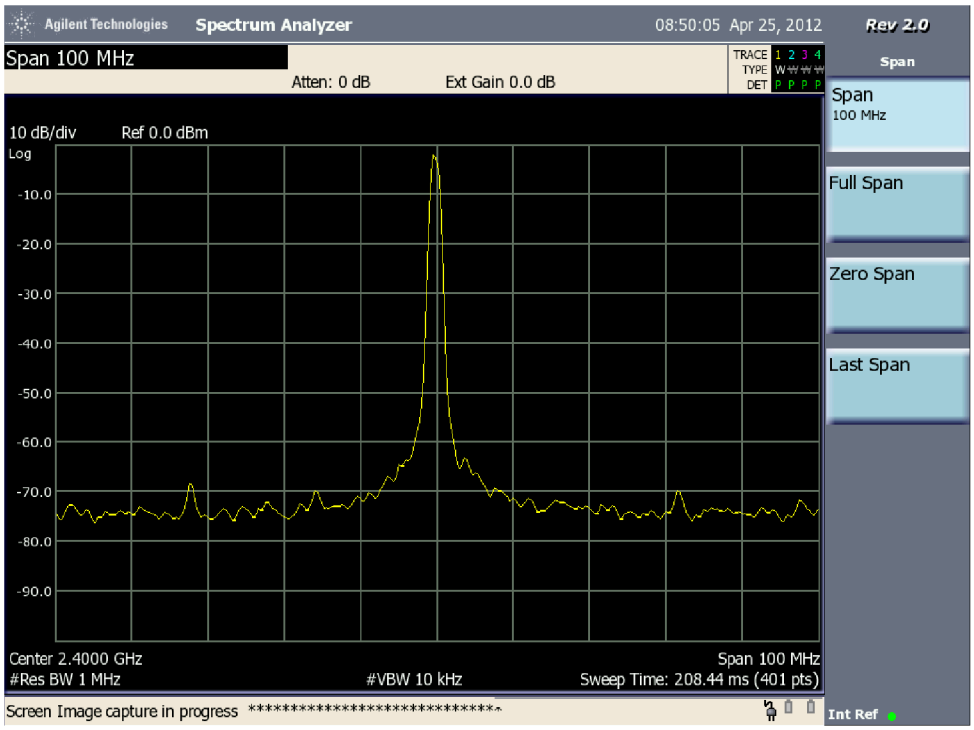
V této kapitole je rozebrán a otestován soubor pravidel, které by měly být dodrženy pro bezchybný průběh komunikace.

4.1 Základní zprovoznění desky

Pro vývoj a testování RF transceiveru byla použita vývojová deska nRFgo Starter Kit (nRF6700) od firmy Nordic Semiconductor, do které je vložen modul s programovaným čipem, v provedení balení 24 pinů 4 x 4 mm QFN. Programování bylo provedeno v programu Keil uVision 4 [2].

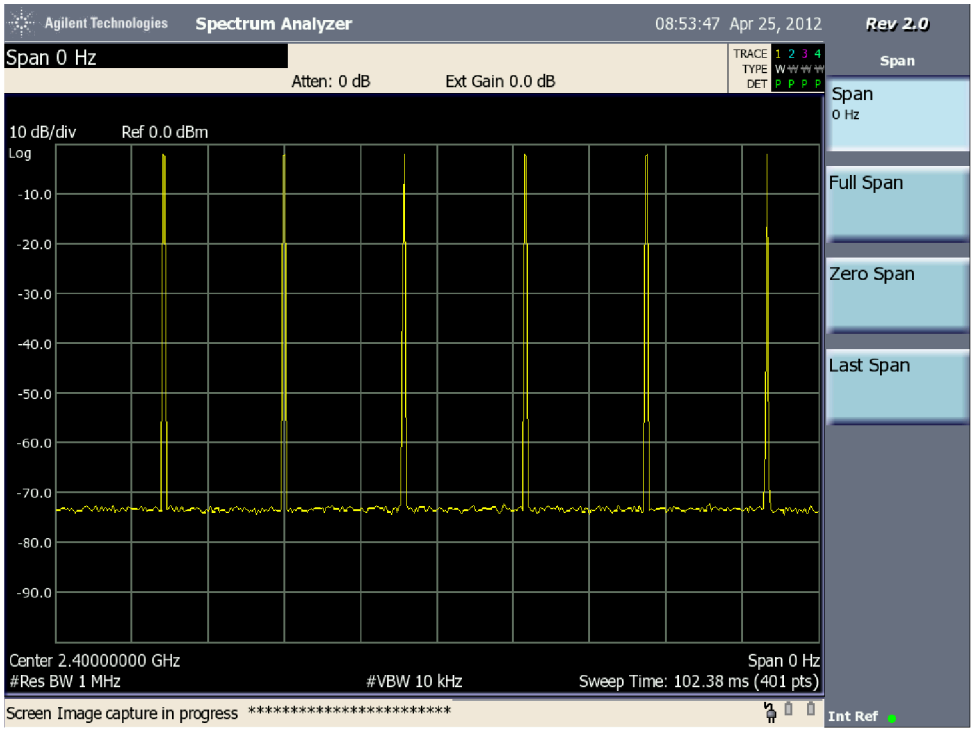
Nejprve bylo zkušebně zprovozněno blikání LED a grafický displej. Následovala UART komunikace s terminálem Realterm na počítači. UART stabilně komunikoval s rychlostí 9600 Baud, při vyšších rychlostech docházelo k selhání aplikace. Na grafický displej i UART byly použity dostupné knihovny od Nordic Semiconductor. Po zprovoznění těchto periferií bylo přistoupeno k testování samotného RF transceiveru.

Pro naprogramování RF transceiveru byly opět využity knihovny od Nordic Semiconductor. Na prvotní vyzkoušení, jestli vůbec transceiver vysílá, a zdali sedí nastavená frekvence kanálu byl místo antény k modulu připojen spektrální analyzátor. RF transceiver byl naprogramován tak, aby vysílal pouze nosnou vlnu na určité frekvenci. Na obrázku (viz Obr. 4.1) je vidět spektrální čára na nastaveném kmitočtu vysílače.



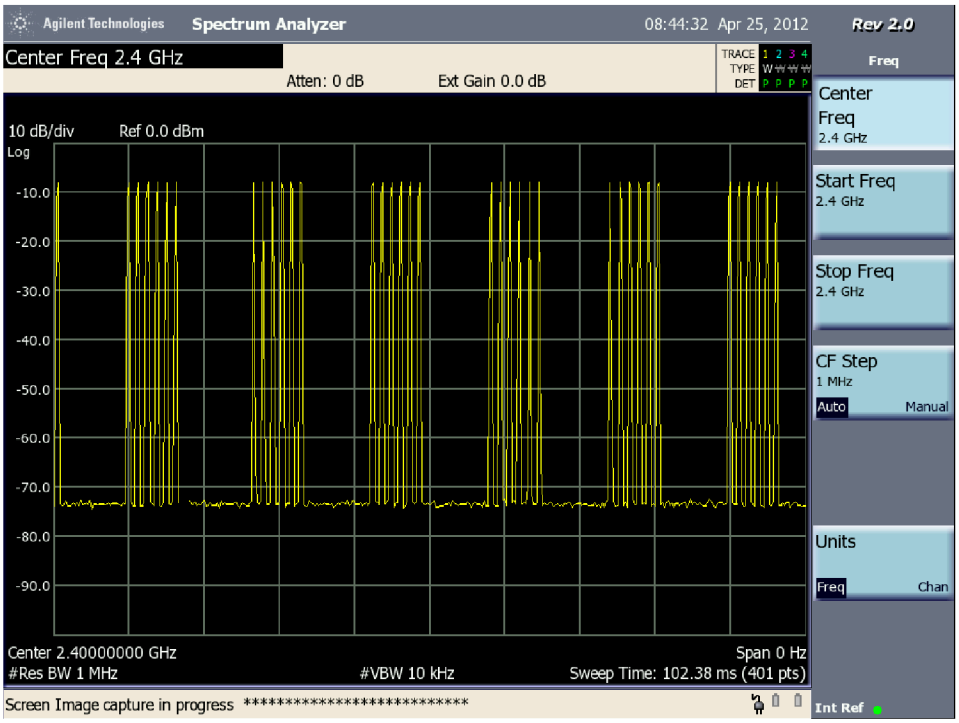
Obr. 4.1 Spektrální čára na nastaveném kmitočtu

Na dalším obrázku (viz Obr. 4.2), který je sejmut v režimu zero span a x je časová osa, je zobrazeno opakované odesílání paketu v nekonečné smyčce protokolem Enhanced ShockBurst™, s vypnutým automatickým odesláním při nepřijmutí potvrzení.



Obr. 4.2 Odesílání paketu v nekonečné smyčce

Na následujícím obrázku (viz Obr. 4.3) je vidět stejný případ, ale s nastavenými pěti automatickými znovuodesláními, při nepřijetí potvrzení o příjmu.



Obr. 4.3 Opakované odesílání pomocí ShockBurst™

Po ověření toho, že vysílač opravdu vysílá, byl jako přijímač použit programovatelný prototyp senzoru s nRF24LE1 firmy ALPS Electric, v této kapitole bude označován jako klíč. Klíč, naprogramovaný jako primární přijímač PRX, byl připojený přes UART k terminálu Realterm na počítači. Na obou senzorech byla nastavena stejná 5 bytová adresa a stejný data pipe viz Enhanced ShockBurst™. Pro automatickou detekci a opravu chyb byl použit již implementovaný CRC16, 16 bitový cyklický redundantní součet. Na klíči bylo nastaveno odesílání ACK s povoleným připojením payloadu.

Nejprve byla využita funkce pro detekování nosné z výše zmíněné knihovny, kterou přijímací senzor, signalizoval odesláním znaku ‘1’ na terminál. Podobně bylo testováno i zaplnění RX FIFO. Následně se podařilo odeslat i jednotlivé znaky ze senzoru na klíč a přijmout znak z ack payloadu z klíče na senzor. Jakmile se podařilo odeslat jednotlivé znaky, tak se přešlo k posílání řetězce s délkou 16 byte. S touto délkou zprávy se již počítá v samotné aplikaci. Nakonec byla komunikace zakódována pomocí vestavěného šifrovacího generátoru.

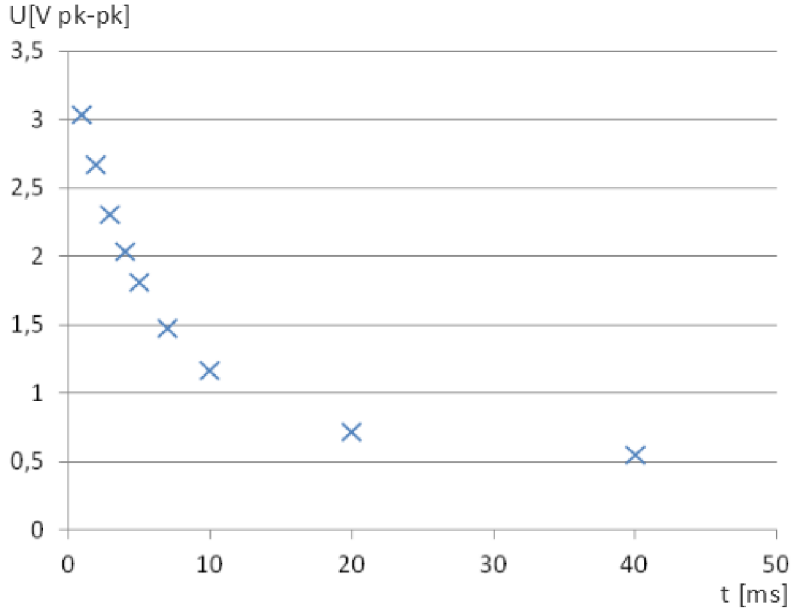
Tato fáze testování potvrdila, že je výhodnější navrhovaný oboustranný komunikační protokol nevystavět na rádio funkcích nižší úrovně. Protokol Enhanced ShockBurst™ poskytuje robustní základ pro tvorbu komunikačního protokolu vyšší úrovně.

4.2 Časová náročnost a synchronizace

Pro hladký chod zamýšlené aplikace je třeba dbát na správnou časovou synchronizaci jak na senzoru při odesílání dat a měření pohybu, tak mezi senzorem a klíčem. Experimentálně bylo ověřeno, že interferometr správně pracuje v intervalu odesílání až do cca 20ms, nicméně jak je vidět z tabulky (viz Tab. 4.1) a grafu (viz Obr. 4.4), čím kratší interval, tím lepší je citlivost zachycení pohybu. Použitá délka intervalu je tedy vždy kompromisem mezi citlivostí, a intervalem dostatečným k tomu, aby se stíhaly vykonávat programové úkony mezi odesíláním.

Tab. 4.1 Závislost citlivosti senzoru na čase mezi odesíláním

t [ms]	1	2	3	4	5	7	10	20	40
U_{pk-pk} [V]	3,03	2,67	2,31	2,03	1,81	1,47	1,17	0,72	0,55



Obr. 4.4 Závislost citlivosti senzoru na délce intervalu mezi odesíláním zpráv

Největší časovou náročnost při testování vykazovalo vypisování 16 bytových zpráv na UART, způsobený jednak nízkou rychlostí UART, druhak metodou obsluhy portu. Ve výsledné aplikaci se toto vypisování používat nebude (pouze pro testovací účely), pro přiblížení k tomuto stavu byl na UART odesílán pouze první znak, který se inkrementoval od 0 do 9 a sloužil zároveň ke kontrole ztrátovosti paketů. Druhým nejdelším úkonem je šifrování. Bez šifrování se aplikace neobejde, časování je tedy nutné podřídit tomu, aby se stíhlo v jednom vysílacím cyklu rozšifrovat přijatou a zašifrovat odeslanou zprávu. Samotné odesílání je z těchto úkonů nejkratší (viz

Tab. 4.2).

Tab. 4.2 Doba trvání jednotlivých úkonů

Úkon	t [ms]
Odeslani 16 byte	0,1
Příjem 16 byte	0,15
Kryptování 16 byte	1,4
UART 16 byte	7,3

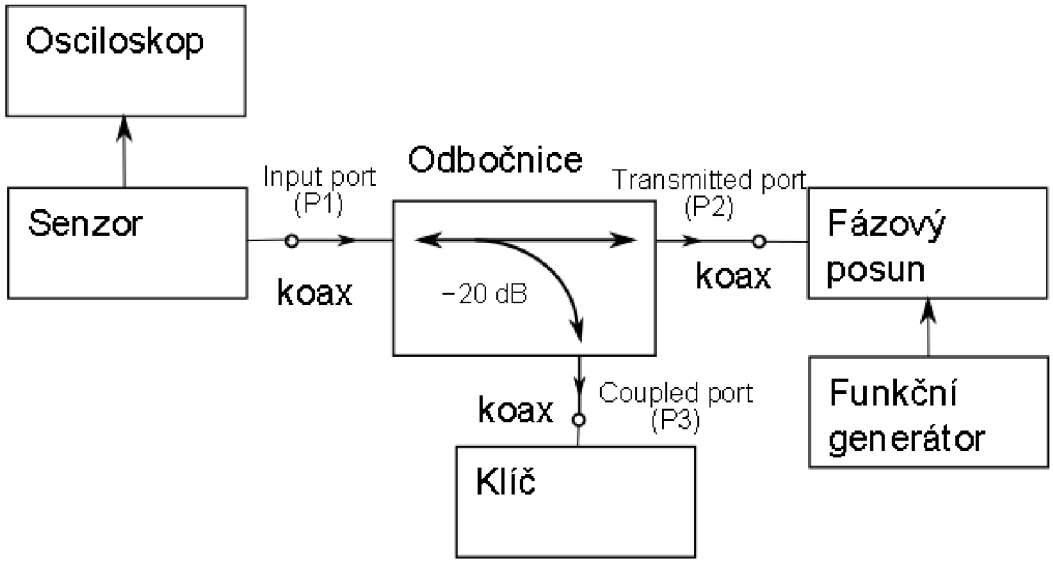
Na straně senzoru je časování pevně dané. K vyslání zprávy dochází každých 5 ms. Časování je řízeno pomocí přerušení od interního časovače TIMER2.

Na straně klíče žádný interní časovač nefiguruje, synchronizovat se musí pouze pomocí zpráv přijímaných od senzoru. Jako nejstabilnější řešení se ukázalo nastavit v přerušení při příjmu zprávy návěští. Na jeho základě pak mít v hlavní smyčce podmínku a v ní všechny potřebné úkony: rozšifrování, odeslání na UART, zašifrování a nastavení zprávy do ack payloadu.

4.3 Měření pseudopohybu

Při různých změnách nastavení jak senzoru, tak klíče dochází na interferometru k detekci pseudopohybu. Proto byla provedena série měření situací, ke kterým by mohlo dojít. Jedná se o zapnutí/vypnutí senzoru i klíče, vliv změny nosné, povolení a opětovné zakázání připojení payloadu k acknowledgy změna délky payloadu, změna vysílacího výkonu a změna obsahu payloadu na obou stranách komunikačního řetězce.

Měření bylo prováděno jednak pro zobrazení samotného, čistého pseudopohybu, a jednak pro chování při zachycení skutečného pohybu (pro účely měření byl tento pohyb simulován). Aby nedocházelo k rušení pohybem z okolí, byl místo vzduchem přenos uskutečněn po koaxiálním kabelu (viz Obr. 4.5).



Obr. 4.5 Blokové schéma pracoviště

Pro měření skutečného pohybu byla použita simulace pomocí fázového posunu (senzor detekuje pohyb na bázi detekce fázového posunu odeslané a přijaté vlny). Pro tuto simulaci byl pomocí směrové vazební odbočnice přidán napětím řízený blok fázového posuvu. Ten při změně napětí na vstupu bloku mění fázi odraženého RF signálu na svém vstupně-výstupním portu. Blok byl řízen harmonickým napětím $0{,}1V_{p\text{-}p}$ s frekvencí 2 Hz z funkčního generátoru. Výsledným efektem simulace je harmonický průběch na výstupu ze senzoru (výstup interferometru), který byl připojen na osciloskop. Vazební útlum odbočnice částečně simuloval ztrátu úrovně signálu šířením volným prostorem, zároveň bránil saturaci přijímače v klíči.. Odbočnice pracuje v pasmu 1 – 4 GHz. Na následujících obrázcích je situace se simulovaným pohybem vždy vpravo. Při všech měřeních probíhala oboustranná zašifrovaná datová komunikace. V tabulce (viz

Tab. 4.3) jsou uvedené typy použitých měřících zařízení.

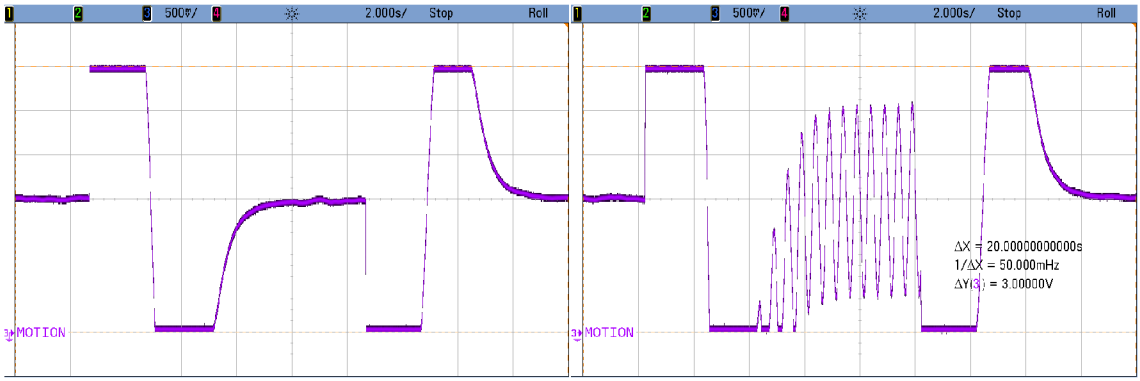
Tab. 4.3 Použité přistroje

Zařízení	Typ
Spektrální analyzátor	Agilent CSA N1996A 100kHz – 3GHz
Osciloskop	Agilent InfiniiVision DSO07014B 100MHz
Funkční generátor	TTi TG2000 20MHz
Multimetr	Metex M-3890D USB

Zapnutí a vypnutí senzoru

Na obrázku níže (viz Obr. 4.6) je nejvýraznější zaznamenaný pseudopohyb, po zapnutí senzoru je potřeba čekat cca 6 sekund na ustálení senzoru. Při samotném chodu aplikace tento pseudopohyb nevadí, senzor musí být neustále zapnutý.

Při přidání simulovaného pohybu je patrné zpoždění, se kterým by byl pohyb díky tomuto přechodnému stavu zachycen, celou tuto dobu je také znemožněna všechna datová komunikace.



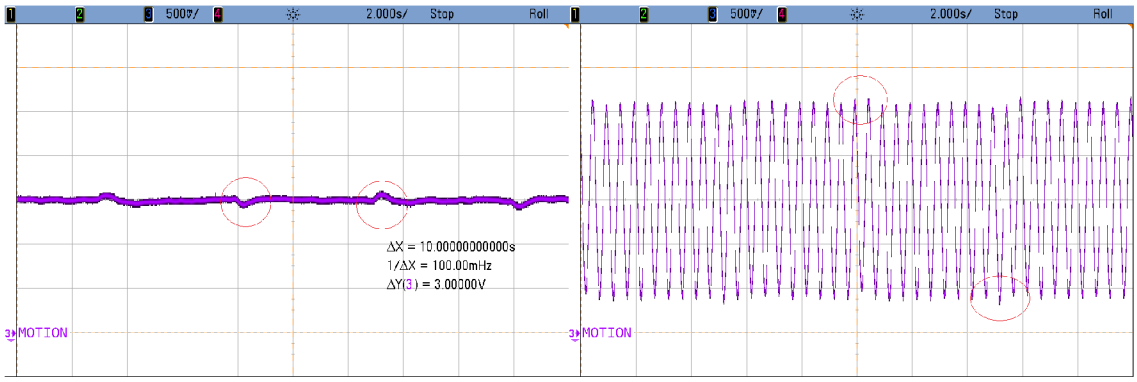
Obr. 4.6 Zapnutí a vypnutí senzoru (vlevo: klid, vpravo: simulovaný pohyb)

Zapnutí a vypnutí klíče

Zapnutí a vypnutí klíče způsobuje pouze nepatrné pseudopohyby.

V případě klíče bylo uvažováno o využití tohoto pseudopohybu pro potvrzení prvotní detekce klíče v rozlišovacím dosahu senzoru. Nicméně jak je vidět na

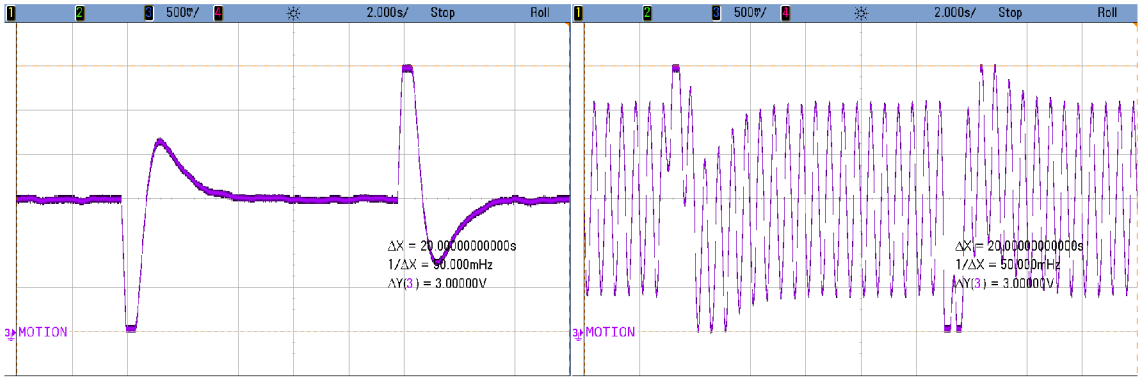
následujícím obrázku (viz Obr. 4.7), tento pseudopohyb, který je označen červenými kolečky, je natolik nepatrný, že je k tomuto účelu nepoužitelný.



Obr. 4.7 Zapnutí a vypnutí klíče (vlevo: klid, vpravo: simulovaný pohyb)

Změna blízké nosné na senzoru

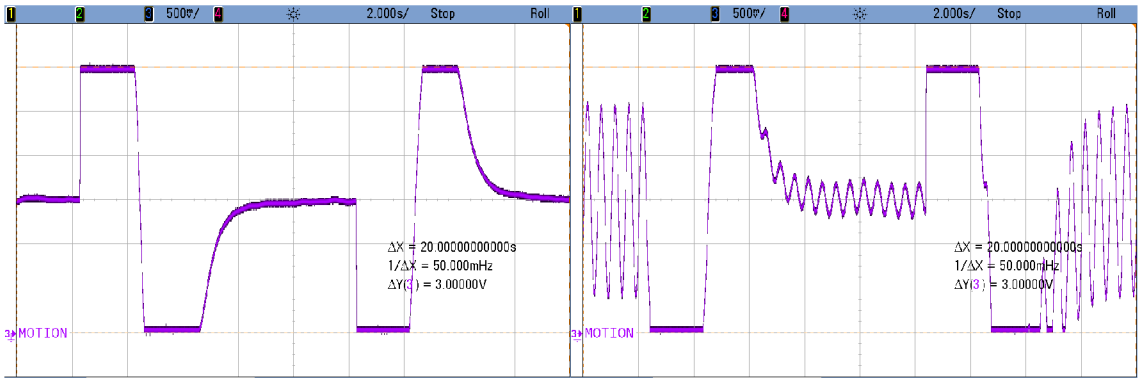
Změna nosné z 2401 MHz na 2402 MHz je na obrázku (viz Obr. 4.8).



Obr. 4.8 Změna blízké nosné na senzoru (vlevo: klid, vpravo: simulovaný pohyb)

Změna vzdálené nosné na senzoru

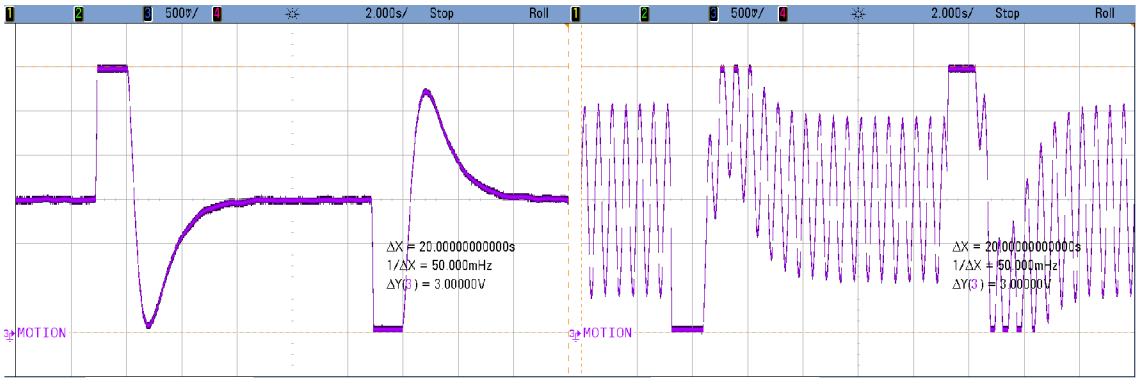
Změna nosné z 2401 MHz na 2450 MHz. Pro případnou kooperaci více klíčů bylo zamýšleno, že bude po prvotní identifikaci nového klíče přidělena unikátní frekvence nosné a senzor bude v krátkých časových intervalech přepínat mezi nosnými. Jak je vidět z obrázku (viz Obr. 4.9) přepnutí nosné vyřadí interferometr na cca 5 sekund z činnosti, takže je tento koncept nepoužitelný.



Obr. 4.9 Změna vzdálené nosné na senzoru (vlevo: klid, vpravo: simulovaný pohyb)

Změna délky datového payloadu na senzoru

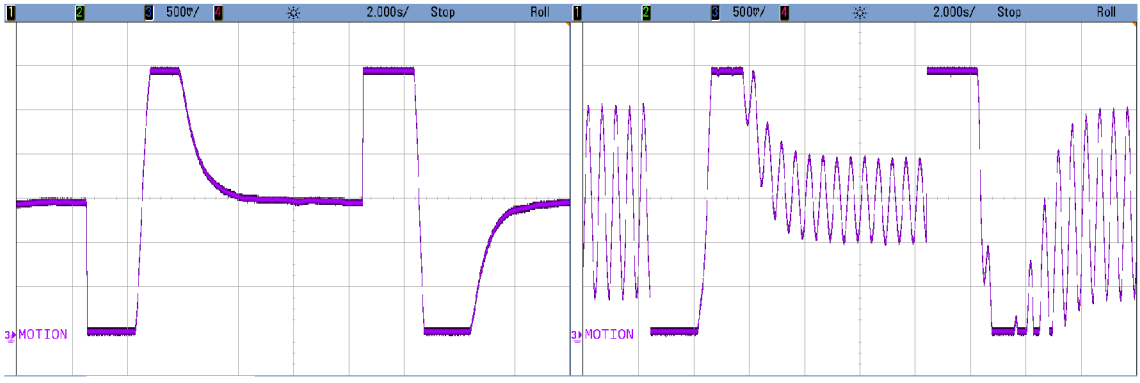
Jak je vidět z následujícího obrázku (viz Obr. 4.10), pro nepřerušovanou práci senzoru je potřeba po celou dobu komunikace ponechat stejnou délku payloadu.



Obr. 4.10 Změna délky payloadu na senzoru (vlevo: klid, vpravo: simulovaný pohyb)

Změna vysílacího výkonu na senzoru

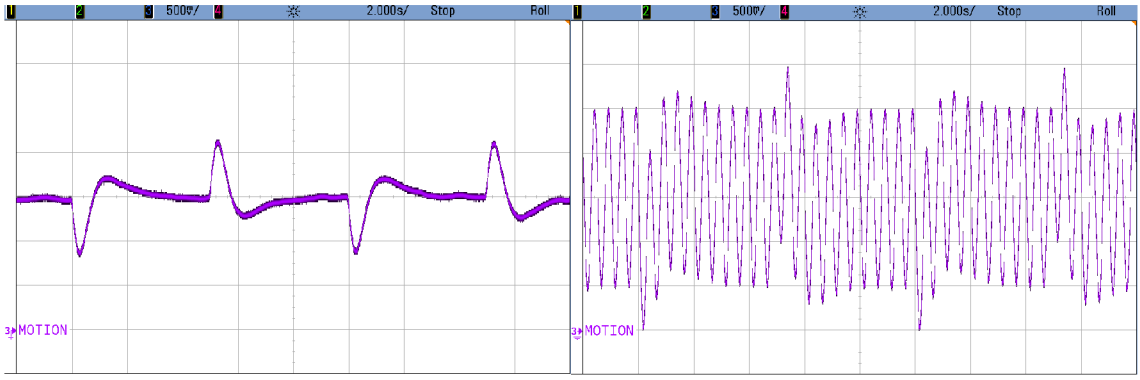
Stejná je situace i se změnou vysílacího výkonu, senzor nebude napájen z baterie, takže pojede stále na nejvyšší možný výkon. Výsledek tohoto měření je vidět na obrázku (viz Obr. 4.11)



Obr. 4.11 Změna vysílacího výkonu na senzoru (vlevo: klid, vpravo: simulovaný pohyb)

Změna obsahu datového payloadu na senzoru

Pseudopohyb nastává pouze v případě, že se změní celková energie zprávy (změní se celkový počet jedniček a nul). Na pořadí nezáleží, pokud se počet jedniček a nul nezměnil, pseudopohyb nenastane. Na obr (viz Obr. 4.12), je vidět extrémní případ, kdy se dvě zprávy liší ve všech bitech. Protože v průběhu aplikace bude nutné posílat různé zprávy je níže provedené ještě další měření tohoto pseudopohybu.



Obr. 4.12 Změna obsahu datového payloadu (vlevo: klid, vpravo: simulovany pohyb)

Ostatní změny na klíči

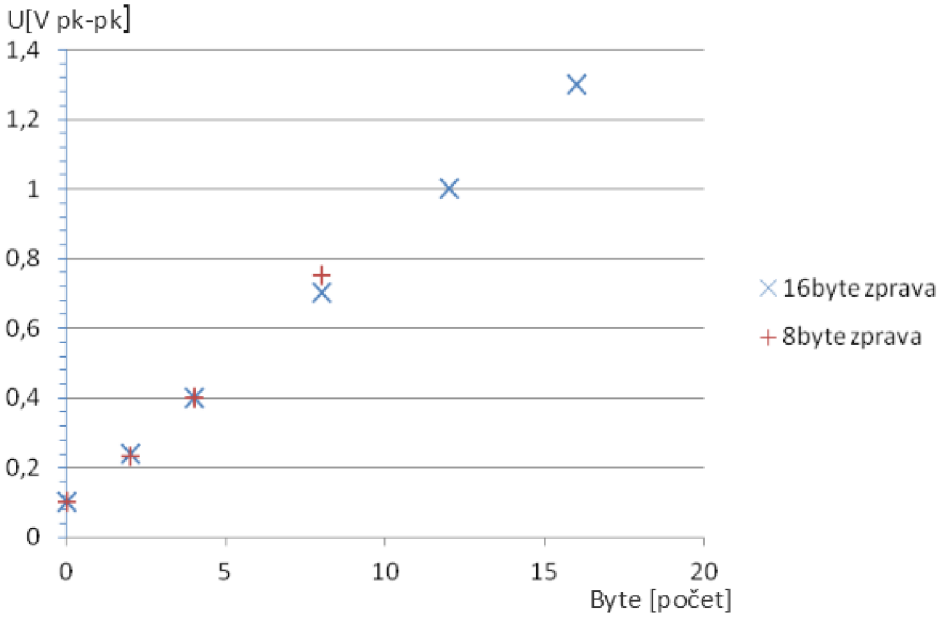
Změna nosné, obsahu payloadu, délky payloadu, zakázání a povolení payloadu v acknowledge a změna výkonu na klíči způsobily natolik nepatrné pseudopohyby (ještě menší než bylo zaznamenáno u vypnutí a zapnutí klíče), že byly z hlediska použité měřící aparatury téměř nezaznamenatelné. Pro účely aplikace se tak dá předpokládat, že mohou být zanedbány.

4.3.1 Vliv počtu odlišných bytů v po-sobě následujících zprávách

Cílem měření je zjistit vliv počtu bytů v po sobě jdoucích zprávách na amplitudu simulovaného pohybu. Měření bylo prováděno pro dvě různé délky zprávy 8 a 16 byte. Vyzkoušeny byly také varianty měnění různých bitů (první, poslední, uprostřed..). Tyto operace měli za následek poze drobné změny amplitudy. Z měření (viz Tab. 4.4 a Obr. 4.13) je patrné, že na délce vysílaného payloadu nezáleží. Pro amplitudu pseudopohybu je rozhodující počet různých byte.

Tab. 4.4 Změna amplitudy v závislosti na počtu odlišných byte

Počet odlišných byte	**0**	**2**	**4**	**8**	**12**	**16**
16byte payload $U_{pk\text{-}pk}$ [V]	0,10	0,24	0,40	0,70	1,00	1,30
8byte payload $U_{pk\text{-}pk}$ [V]	0,10	0,23	0,40	0,75		



Obr. 4.13 Změna amplitudy v závislosti na počtu odlišných byte

5 KOMUNIKAČNÍ PROTOKOL RFSP2

V následující části práce je popsán samotný návrh a implementace komunikačního protokolu RF sensor protocol 2. Komunikační protokol je rozdělen do dvou, respektive tří vrstev, základní, nejnižší vrstvu tvoří již na čipu vestavěný protokol Enhanced ShockBurst™. Na něm je postavený paketový komunikační protokol RFSP2 (RF Sensor Protocol 2.0) pro obousměrný přenos zpráv, který tvoří základ pro aplikační vrstvu identifikačního protokolu a snímání pohybu.

Základní problém při tvorbě komunikačního protokolu byl, že klíč nemůže sám o sobě poslat zprávu. Protože komunikace Senzor-Klíč probíhá formou Master-Slave, zpráva z klíče do senzoru je možná pouze jako připojení k acknowledge, takže pouze jako reakce na přijatou zprávu od senzoru.

Jedná se o paketový komunikační protokol. Velikost jednoho paketu je limitována maximální možnou délkou zprávy, kterou dokáže najednou přenést Enhanced ShockBurst™ – 16 byte, nicméně kvůli zabránění pseudopohybu je třeba zachovat celou dobu stejný poměr jedniček a nul ve zprávě. Zprávy jsou tedy vyvažovány a jedna zpráva obsahuje 8 byte vyvažovacích a 8 byte užitečných, které budou dále nazývány paketem. Z toho je 5 byte vyhrazeno na přenos zprávy, 1 byte na číslo paketu, 1 byte na acknowledge a 1 byte na informaci o konci zprávy a o počtu validních byte z 5 byte nesoucích zprávu. Původně bylo zamýšleno 6 byte nesoucích zprávu a 2 byte na řešení acknowledge (číslo paketu a acknowledge, princip bude vysvětlen níže), a o konci zprávy by informoval ukončovací znak, který by musel být v těle zprávy maskován pomocí escapingu. Z důvodu návaznosti zpráv v přijímacím bufferu by musel ještě jiný znak značit začátek zprávy a escapování dvou znaků přineslo příliš vysokou složitost a výpočetní nároky. Proto se vyhradil ještě jeden byte v paketu na informaci o konci zprávy. Výsledná podoba použitého paketu je vidět na následujícím obrázku, paket

s vyvažovacím polem je vidět v dolní části. (viz Obr. 5.1).

Pakcket num 1 byte	Acknowledge 1 byte	Info 1 byte	Payload 0 - 5 byte

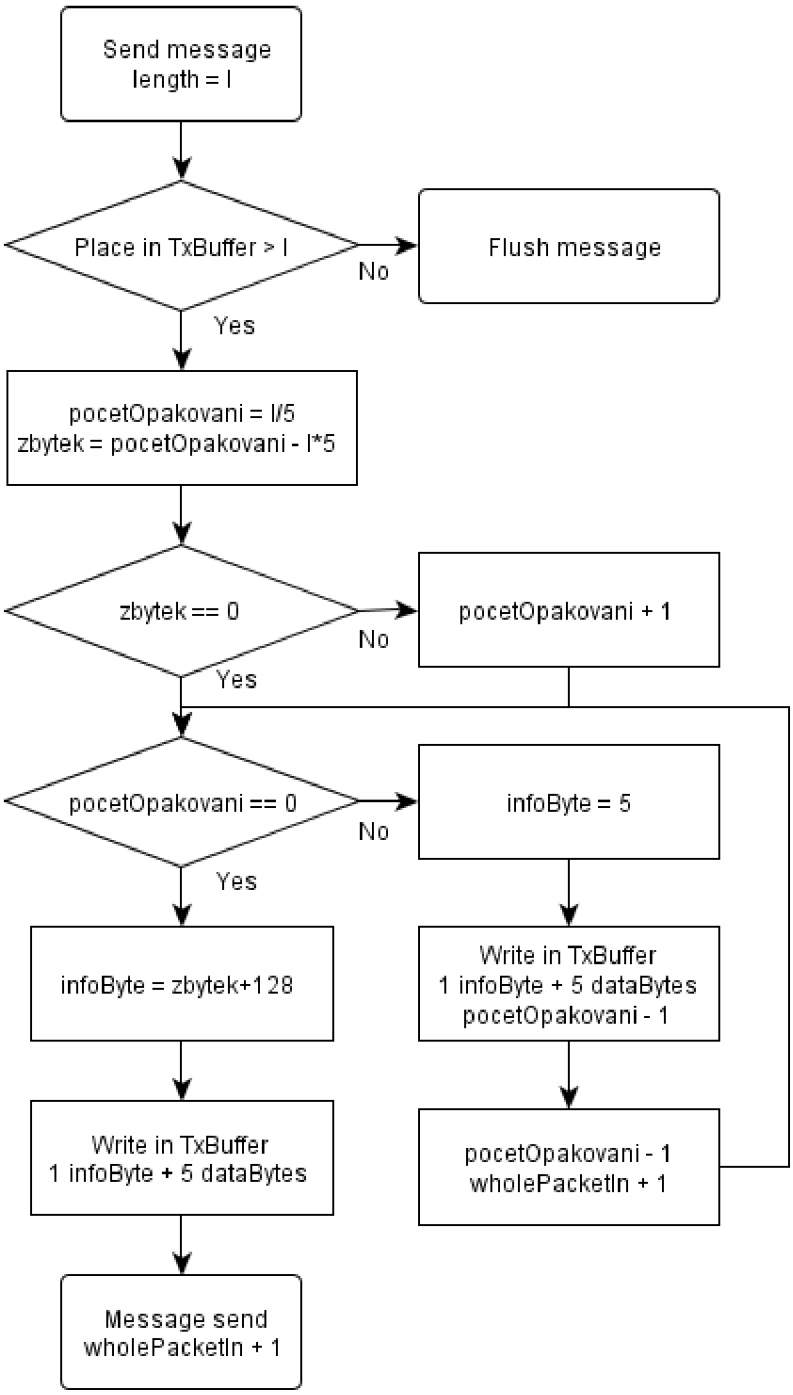
Pakcket num 1 byte	Acknowledge 1 byte	Info 1 byte	Payload 0 - 5 byte	Balance 8 byte

Obr. 5.1 Schema paketu

5.1 Popis programu

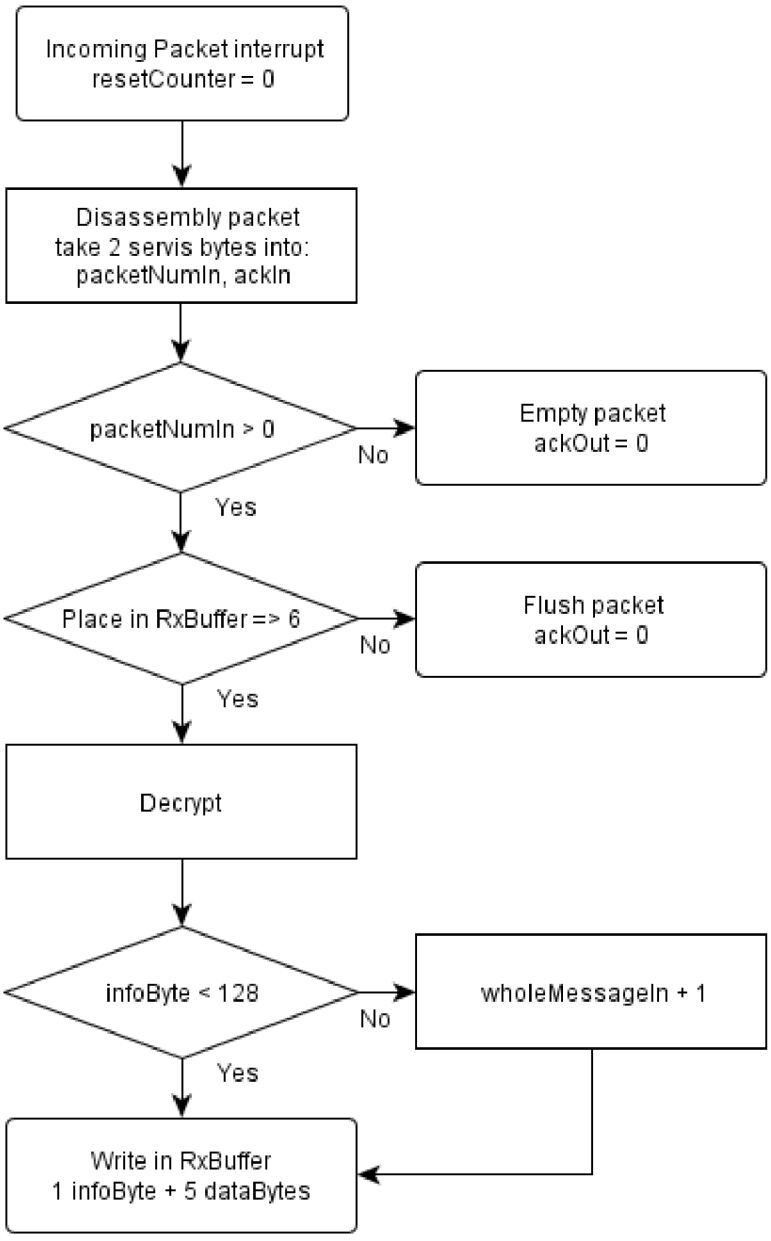
Program je v podstatě rozdělen do čtyř základních rutin, které si mezi sebou vyměňují data, ale jinak jsou na sobě funkčně nezávislé. Nejdůležitější rutina je okolo funkce `assemblyPacket`, jsou to jediné hodiny systému a zajišťuje odeslání libovolného paketu v pevně daném časovém intervalu. Následuje rutina okolo `Send` pro zapsání zprávy do odesílacího paketu, dále přerušení od příchozích paketu, které zajišťuje zapsání dat z paketu do přijímacího bufferu a vypsání přijaté zprávy okolo `assemblyMessage`.

Nejprve se podrobněji podíváme na rutinu okolo funkce `Send`, je to počátek každého odesílání zprávy. Kvůli velikosti odesílacího bufferu může být jedna zpráva maximálně 50 bytu dlouhá. Odesílací i přijímací buffer má velikost 64 byte, oba dva jsou implementovány jako kruhové buffery. Na každých 5 datových bytu v bufferu připadá 1 informační byte. Pokud je tedy na zprávu v bufferu místo, tak se zjisti kolik zabere zpráva paketů a kolik je bytu v poslednim paketu zprávy. Poté se sleduje index počtu opakování a v každém cyklu, dokud není nula se zapíše do bufferu jeden informační byte, pokud není poslední paket, tak vždy s hodnotou 5 a 5 datových bytů. Pokud se jedná o poslední paket, tak se do info bytu zapíše počet bytu v poslednim paketu plus 128, což značí poslední paket, několik validních bytů a paket se doplní na celou délku náhodnými byty. Po každém zapsání jednoho paketu se zvýší flag `wholePakcetInFlag` a tím je tento paket nabídnut k odeslání. (viz Obr. 5.2)



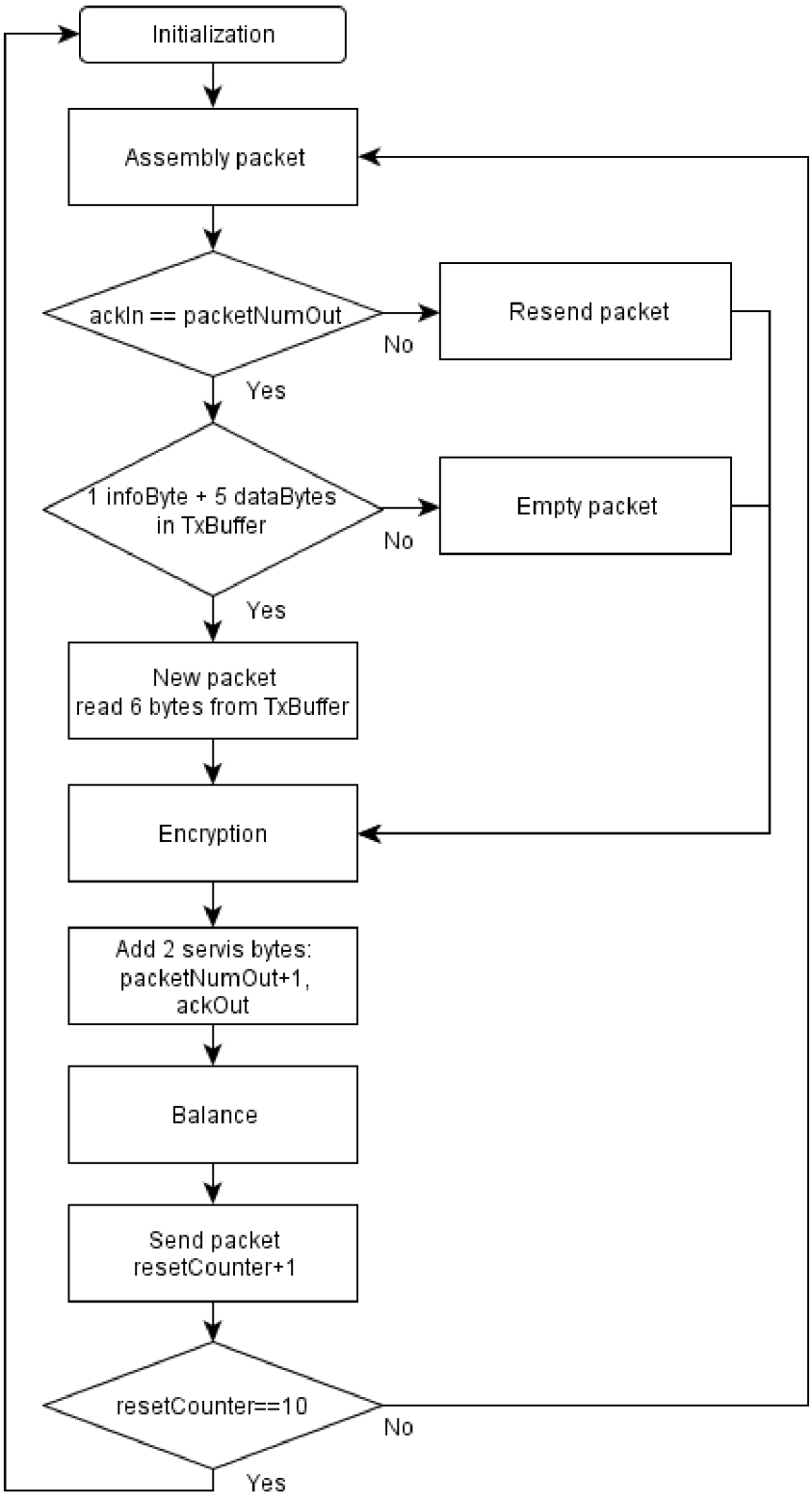
Obr. 5.2 Vývojový diagram funkce odesílání

Další rutina se týká funkce příjmu nového paketu. Při každém rádiovém příjmu se spustí interrupt. Nejprve se odeberou servisní pakety, pokud je číslo příchozího paketu rovno nule, tak tím tato rutina končí. Když není rovno nule, tak se zkontroluje jestli se jedná o nový paket (čísla příchozích paketů se musí lišit) a když je v přijímacím bufferu dostatek místa, tak se rozkóduje 1 info byte a 5 datových bytů. Ty se zapíší do přijímacího bufferu Na velikosti info bytu závisí, z se jedná o poslední paket, pokud je větší než 128, tak to sedí, zvýší se flag `wholeMessageInFlag` informující o přijaté celé zprávě a do bufferu se vypíše spolu s pěti datovými byty. (viz Obr. 5.3)



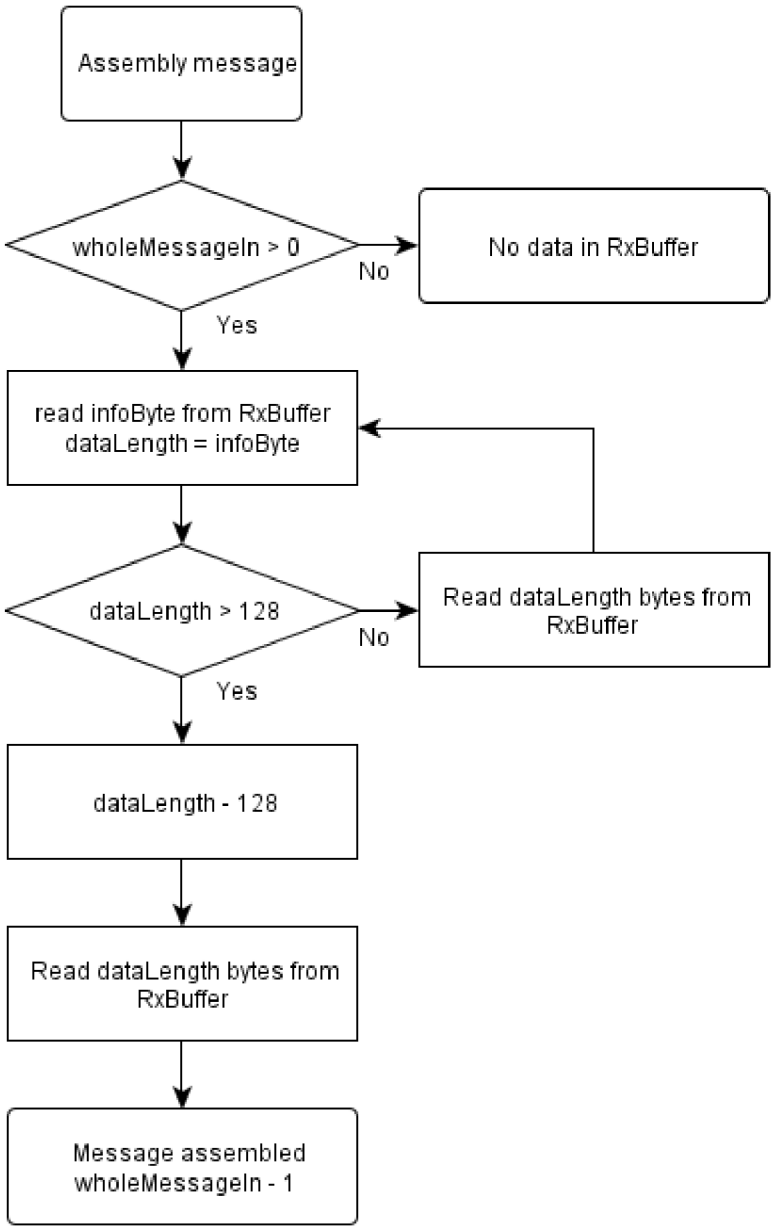
Obr. 5.3 Vývojový diagram příjmu datového paketu

`AssemblyPacket` – nejdůležitější rutina, řídí hodiny celého systému. Volá se v každém přerušení od časovače Timer2 a to v pevně daných 5ms intervalech. Jak je vidět z vývojového diagramu (viz Obr. 5.4), tak při každém volání `assemblyPacket` se nejprve zkontroluje zdali souhlasí acknowledge. Ten pracuje na principu porovnávání čísla posledního odeslaného paketu a čísla poslaného v acknowledge bytu obratem zpět, Pokud čísla nesouhlasí, tak je paket odeslán znovu se stejným číslem, v tomto případě je šestice bytů znovu zapkryptována. Na základě shody čísel se může odeslat další paket. Pokud je v odesílacím bufferu celých 6 bytů k odeslání, tak se z bufferu přečtou a jsou zakryptovány. Sledování místa v bufferu má na starosti `bufferTxPointerTrucker`. Jinak se posílá tzv. prázdný paket, nicméně kvůli zachování časování (kryptování zabírá nejvíce času, viz výše) je kryptován i tento paket s pseudonáhodnými daty, kterými jsou v podstatě data z minulého paketu. Následuje přidání dvou servisních bytů: čísla paketu a čísla acknowledge, těchto osum bytů je vyváženo přidáním dalších osmy bytů vzniklých jednotkovým doplňkem s předchozími byty. Následuje samotné odeslání paketu. Zde je také implementován `resetCounter`, který se po každém odeslání zvyšuje a při každém příjmu snižuje, pokud se Klíč – primárně přijímací strana odmlčí na několik paketů, tak je systém resetován. Při každém resetu je odeslán resetovací paket na klíč. Ten žádné takové počítadlo nemá v časových a resetovacích záležitostech je plně odkázán na senzor.



Obr. 5.4 Vývojový diagram funkce pro složení paketu

Posledni hlavní rutina je složení a vypsání přijaté zprávy. (viz Obr. 5.5) Pokud je flag `wholeMessageInFlag` větší než nula, tak se vždy napřed přečte z přijímacího bufferu informační byte, který rozhoduje o konci zprávy. Dokud se na něj nenarazí, tak je vypsáno vždy 5 bytů, jakmile dojde na informační byte větší než 128, tak se 128 odečte a zbyde puoze počet validních bytů k vypsání. Na závěr se dekrementuje `wholeMessageInFlag`. Funkce `assemblyMessage` také vrací délku přijaté zprávy.



Obr. 5.5 Vývojový diagram funkce slozeni zpravy

Podoba informačního bytu je zřejmá z následující tabulky (viz Tab. 5.1). Nejvyšší bit slouží k rozpoznání posledního paketu ve zprávě, jeden bit slouží k identifikaci resetovacího paketu a jeden bit zajišťuje identifikaci paketu pro re-inicializaci protokolu. Zbylých pět bitů je nevyužito.

Tab. 5.1 Informační byte

Bit	Využití
7 (MSB)	Poslední paket zprávy
6	Resetovaní paket
5	Paket pro re-inicializaci protokolu
4	Nevyužito
3	Nevyužito
2	Nevyužito
1	Nevyužito
0 (LSB)	Nevyužito

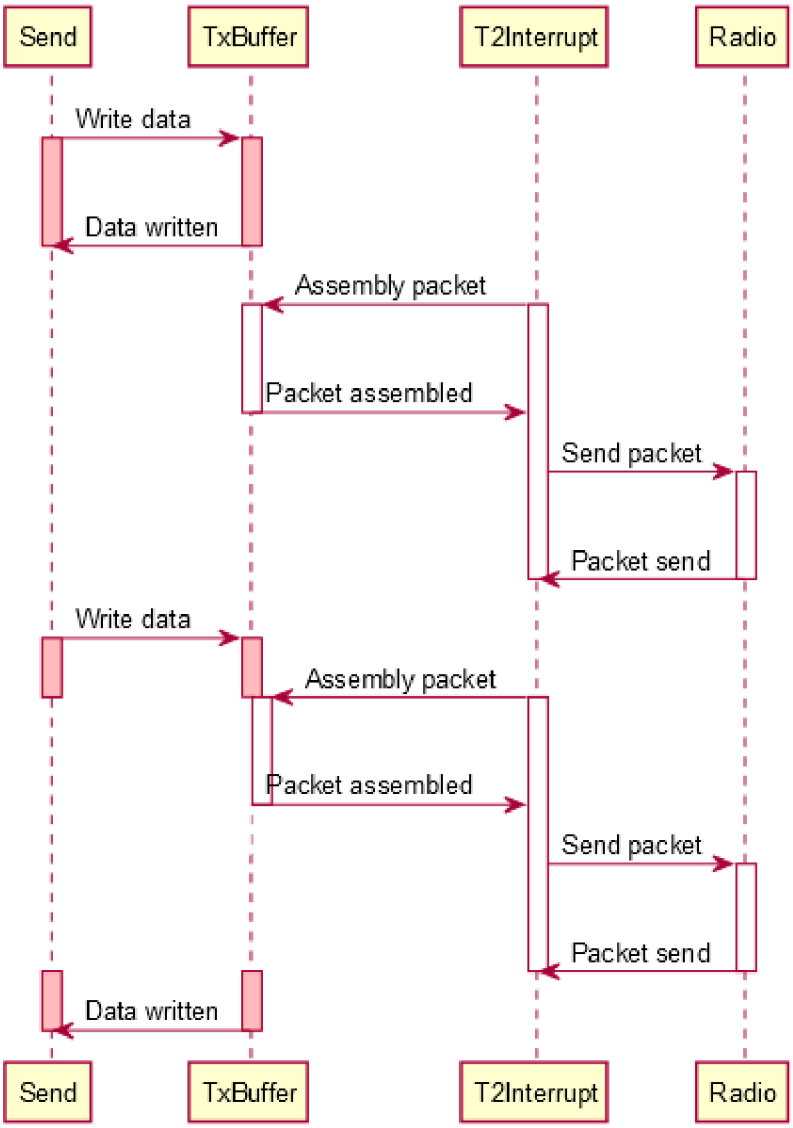
5.2 Realizace

Při samotné realizaci programu došlo k několika zádrhelům, které jsou zde podrobněji popsány. Hlavním problémem byla stabilita komunikace.

Jednou z příčin nestability byla potřeba vypisování výsledků průběžných testů přes UART na počítač. Jak bylo zmíněno výše, tato operace zabírá velkou část systémového času a při vypisování dlouhých zpráv, nebo příliš častém vypisování docházelo k častým kolizím s odesíláním v přerušení od Timer2 a selhání komunikace. Bylo proto třeba tyto zprávy omezit na nejnutnější minimum.

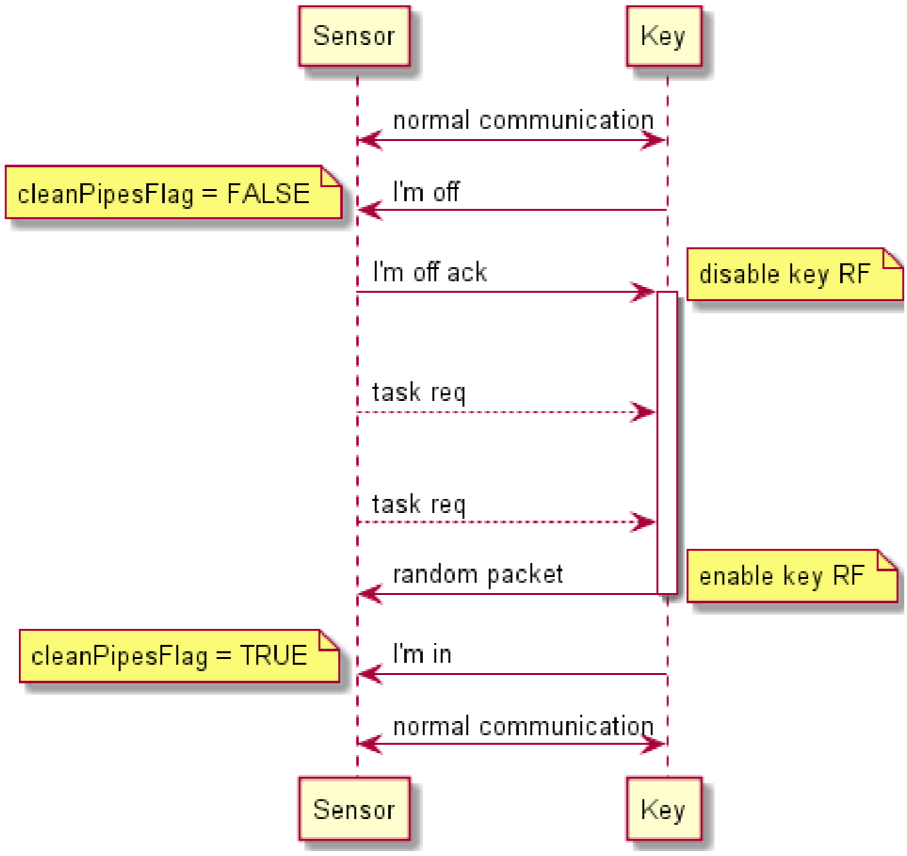
Další problém byl způsoben zvoleným vývojovým nástrojem a spočíval v reentrantním volání jedné funkce. Jedná se o funkci, která byla zavolána a v průběhu jejího konání přišlo přerušení, během kterého byla zavolána znovu. Tento způsob volání použitý překladač Keil C51 v základním nastavení nepodporuje. Pro umožnění těchto operací bylo nutné přidání klíčového slova `reentrant` za definici funkce. Identifikace tohoto problému byla časově náročná.

Jeden z hlavních problému, při kterém se projevovala i výše zmíněná skutečnost způsobovala interakce mezi funkcí `Send`, která může být volána kdykoliv vyšší vrstvou systému a pravidelném přerušení od Timer2. Normální průběh (sledováno osciloskopem), kdy k žádným problémům nedocházelo je zobrazen v horní polovině obrázku (viz Obr. 5.6), přerušení a funkce `Send` se míjejí. V dolní části je vyobrazen případ způsobující chyby, v průběhu vykonávání funkce `Send` přijde přerušení. Stejné problémy nastávaly i na druhé straně při interakci funkce `Send` a přerušení od rádiové aktivity. Projevil se problém s opětovným voláním funkce, dále bylo potřeba velmi pečlivě ošetřit každé možné nežádoucí změny jakéhokoliv flagu, posuv v bufferech, změnu proměnných v přerušení.



Obr. 5.6 Sekvenční diagram interakce mezi odesíláním a přerušením

Další problém spočíval v tom, že při každém restartu klíče byl znovu odeslán paket, který byl odeslán jako poslední před vypnutím. Nepomáhalo vymazání obsahu všech bufferů, ať už před vypnutím nebo hned po zapnutí klíče. Vymazávány byly jak buffery na úrovni RFSP2, tak v nižší úrovni Enhanced ShockBurst™. Tento problém byl nakonec vyřešen pomocí re-inicializace protokolu. Průběh je patrný z obrázku (viz Obr. 5.7) Před odmlčením klíč odešle zprávu že se chce odmlčet. Senzor zprávu přijme, nastaví `cleanPipesFlag = 0` (tento flag neovlivňuje vyřizování ack a čísel paketu, pouze se zahazují přijaté data). Klíč se po přijmutí ack odmlčí. Jakmile se klíč aktivuje, tak o tom pošle zprávu senzoru a ten nastaví `cleanPipesFlag = 1` a komunikace pokračuje. Při restartu senzoru se vždy nejprve resetuje celý systém pomocí resetovací zprávy, takže se tento problém neobjevuje.



Obr. 5.7 Sekvenční diagram odmlčení klíče

5.2.1 Rozdíly mezi senzorem a klíčem

Rozdíly jsou rozebrány především z hlediska rádiové komunikace a z pohledu této vrstvy komunikačního protokolu. Hlavním rozdílem je, že na senzoru běží přerušení od Timer2, jediná smyčka v tomto systému. Na klíči z hlediska protokolu RFSP2 žádná smyčka neběží. Jakákoliv změna na klíči se projeví pouze pokud jej senzor synchronizuje. Například pro resetování, musí klíč přijmout resetovací zprávu od senzoru.

Další rozdíly spočívají i v samotném kódu programu pro klíč a senzor, v knihovně pro RFSP2 je potřeba na začátku nastavit define zdali se jedná primárně o Tx – senzor a Rx – klíč. Nejedná se pouze o jiné nastavení nejnižší vrstvy – Enhanced ShockBurst™, ale i o jiné uspořádání funkcí v programu. Jedinou opakovanou smyčku na klíči představuje pouze přerušení od příjmu. Veškeré operace se tedy musejí odehrát v tomto přerušení. Sensor má krom tohoto přerušení ještě hlavní synchronizační přerušení od Timer2.

Pro komunikaci s vyšší (aplikační) vrstvou byl zvolen na senzoru i klíči odlišný přístup. Informaci o proběhlém pollu na senzoru předávala callback funkce. Pro informaci, zdali probíhá komunikace na klíči, bylo použito flagu. Na nižší vrstvě funkce na základě inkrementujícího se počítadla při příjmu paketu kontrolovala průběh komunikace a podle toho vracela hodnotu. Vyšší vrstva může v jakékoliv chvíli

kontrolovat, jestli komunikace probíhá. Předávání informace o přijaté zprávě probíhá na stejném principu jako kontrolování průběhu komunikace je stejné pro klíč i senzor.

Také funkce pro mazání bufferů jsou mírně odlišné. Funkce umožňují vymazat buffery jak RFSP2, tak v Enhanced ShockBurst™. Odchylka spočívá v tom, že na odesílacím bufferu se musí navíc nulovat `bufferTxPointerTracker`, který sleduje místo v odesílacím bufferu, kvůli nutnosti striktně zapisovat pakety v jednoznačné délce 6 bytů.

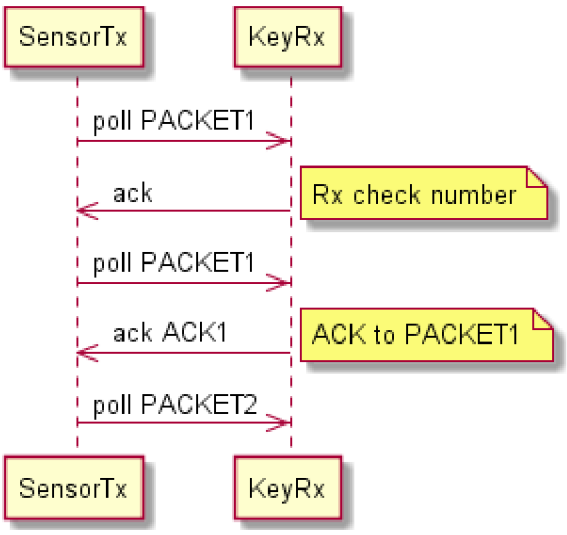
5.3 Testování

S implementovanou vrstvou komunikačního protokolu RFSP2 byla provedena série testů. Konkrétně měření latence mezi odesláním a příjmem, měření propustnosti, přenos paketů v závislosti na útlumu a test spolehlivosti.

5.3.1 Měření latence

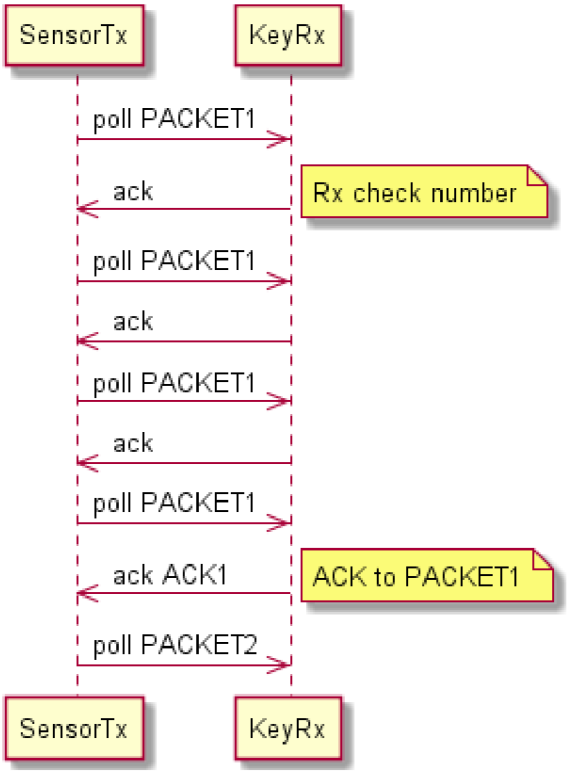
Měření doby od zadání příkazu send po přijetí zprávy. Měření latence probíhalo v podmínkách ve kterých bude senzor pracovat - přenos dat probíhal vzduchem bez žádných vložených útlumů. Časové rozmezí místo jednoho údaje je proto, že čas odesílání není nijak korelován s časem mezi jednotlivými polly. Tomu odpovídá i velikost časového rozpětí cca 5ms.

Při tomto měření se naplno projevil problém, který se průběžně objevoval už od začátků implementace a testování. Teoretický předpoklad průběhu komunikace je zobrazen na obrázku (viz Obr. 5.8). Nejprve se odešle v pollu PACKET1, obratem okamžitě přijde prázdné ack (malé ack je z Enhanced ShocBurst™, velké ACK je z RFSP2), nyní je 5ms čas na zpracování paketu a nachystání odeslání ACK, senzor znovu odesílá PACKET1 (ošetření proti ztrátám a vyváženosti komunikace kvůli pseudopohybu). Na něj mu v ack přichází ACK a může odeslat další paket. U klíče je situace podobná. Takže na odeslání jednoho paketu je zapotřebí dvou pollů.



Obr. 5.8 Očekávaný průběh komunikace

Ve skutečnosti takhle komunikace probíhá pouze když se spustí první klíč a až poté senzor. A stejně se po určitém čase (v řádu desítek sekund) ustálí do následující podoby (viz Obr. 5.9). Jak je vidět z obrázku, tak pro odeslání jednoho paketu je zapotřebí čtyř pollů. Problém byl mnohokrát analyzován a řešen, ale žádný ze zásahů nepřinesl výsledek. Proto bylo pojato podezření, že je způsoben nižší vrstvou (Enhanced ShockBurst™). Všechna další měření byla prováděno pro tento způsob komunikace jakožto nejhorší případ. Odstraněním tohoto problému by se dala zdvojnásobit přenosová rychlost a latence snížit na polovinu.



Obr. 5.9 Skutečný průběh komunikace

Z měření je patrné (viz Tab. 5.2), že krom výše uvedeného problému odpovídá teoretickým předpokladům. Latence s počtem paketů roste lineárně.

Tab. 5.2 Měření latence

Počet paketů	**t [ms]**
před ustalenim na 4 polly	
1	5,5 - 8
po ustálení na 4 polly	
1	15,5 - 20
2	35,1 - 40
4	73,4 - 80
8	152 - 159

5.3.2 Měření propustnosti

Měření probíhalo podmínkách v jakých bude systém pracovat následujícím způsobem: na obou stranách komunikačního řetězce byl protokol v určitých časových intervalech zahlcován požadavky na odeslání tak, aby byla kapacita přenosu plně využita. Délka časového intervalu, cca 36s, byla dána pouze smyčkou ve vyšší vrstvě, která pro každé měření odešle 500000 žádostí o odeslání. Čas byl sledován pomocí osciloskopu změnou úrovně GPIO pinu na každé straně. Na obou stranách byly počítány přijaté pakety. Měření proběhlo pro zprávy složené z 1 paketu, 4 paketů a 8 paketů. V každém paketu byl využit maximální možný počet užitečných datových bytů.

Z měření je patrné (viz Tab. 5.3), že k odeslání jednoho paketu a vyřízení acknowledge je potřeba čtyř pollů od senzoru. Polly jsou vysílány vždy jednou za 5 ms a jeden poll s užitečným payloadem může obsahovat maximálně 5 bytů dat, tj. 40 bitů za 20ms. Za 1 sekundu se tedy odešle 2000 bitů. Dále je vidět, že delší zprávy nemají na rychlost komunikace výraznější vliv, drobné snížení rychlosti je dáno nutností při odeslání a příjmu zpracovat větší objem dat.

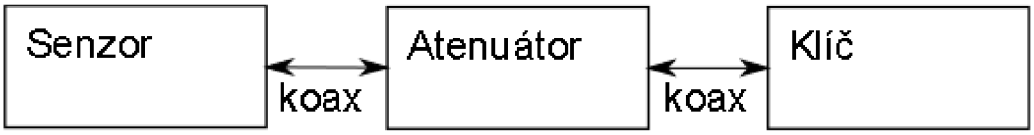
Tab. 5.3 Měření propustnosti

1 paket ve zprávě	**Měření**	**t [s]**	**Přijatých paketů**	**Přijatých byte**	**kbps**
Tx	1	36,4	1822	9110	2002,2
	2	36,4	1823	9115	2003,3
	3	36,4	1822	9110	2002,2
	4	36,4	1823	9115	2003,3
	5	36,4	1820	9100	2000,0
Průměr					**2002,2**
Rx	1	36,0	1804	9020	2004,4
	2	36,0	1805	9025	2005,6
	3	36,0	1805	9025	2005,6
	4	36,0	1802	9010	2002,2
	5	36,0	1804	9020	2004,4
Průměr					**2004,4**
4 pakety ve zprávě					
Tx	1	36,4	1816	9080	1995,6
	2	36,4	1816	9080	1995,6
	3	36,4	1812	9060	1991,2
	4	36,4	1816	9080	1995,6
	5	36,4	1820	9100	2000,0
Průměr					**1995,6**
Rx	1	36,0	1800	9000	2000,0
	2	36,0	1796	8980	1995,6
	3	36,0	1800	9000	2000,0
	4	36,0	1800	9000	2000,0
	5	36,0	1796	8980	1995,6
Průměr					**1998,2**
8 paketů ve zprávě					
Tx	1	36,4	1816	9080	1995,6
	2	36,4	1816	9080	1995,6
	3	36,4	1808	9040	1986,8
	4	36,4	1816	9080	1995,6

	5	36,4	1816	9080	1995,6
Průměr					**1993,8**
Rx	1	36,0	1800	9000	2000,0
	2	36,0	1800	9000	2000,0
	3	36,0	1792	8960	1991,1
	4	36,0	1800	9000	2000,0
	5	36,0	1792	8960	1991,1
Průměr					**1996,4**
Celkový průměr					**1998,5**

5.3.3 Počet přenesených paketů v závislosti na útlumu

Měření bylo uskutečněno přenosem po koaxiálním kabelu s SMA konektory s vloženým programovatelným atenuátorem Marconi Instruments 20GHz programmable attenuator 2187, který má rozsah útlumu od 3dB do 130dB s krokem 1dB, pro měření je také potřeba nastavit frekvenci v rozsahu 0,01GHz az 20GHz. Zapojení při měření je patrné z následujícího obrázku (viz Obr. 5.10), útlum koaxiálních kabelů je uvažován 0,5 dB pro každý kabel.



Obr. 5.10 Zapojení měření počtu přenesených paketů v závislosti na útlumu

Na jedné straně bylo odesláno vždy 1000 paketů, na druhé straně se počítaly přijaté pakety. Celou dobu probíhala obousměrná komunikace. Na odeslání zprávy byl ponechán dostatek času, aby nemohlo dojít k zahození paketu z důvodu zaplnění odesílacího bufferu. Na senzoru bylo také vypnuto automatické resetování při nepřijmutí ACK. Jak je vidět z následujících tabulek (viz Tab. 5.4 – Tab. 5.6) a grafu (viz Obr. 5.11) počet přenesených paketů nezávisí na směru komunikace, ani na počtu odesílaných paketů ve zprávě, ale jen na útlumu. Z rovnice (viz $L = \left(\frac{4\pi r}{\lambda}\right)^2$ (5.1) [5]) pro výpočet útlumu na rádiové trase bylo spočítáno, že v ideálním případě isotropické antény se dá očekávat dosah datové komunikace dosah cca 100m.

$$L = \left(\frac{4\pi r}{\lambda}\right)^2 \tag{5.1}$$

$$L = 10^{\frac{L_{[dB]}}{10}} = 10^8$$

$$\lambda = \frac{c}{f} = \frac{3 \cdot 10^8}{2{,}4 \cdot 10^9} = 0{,}125m$$

$$r = \sqrt{\frac{L\lambda}{4\pi}} = \sqrt{\frac{10^8 \cdot 0{,}125}{4\pi}} = 99{,}47184m$$

Tab. 5.4 Tx – Rx jeden paket

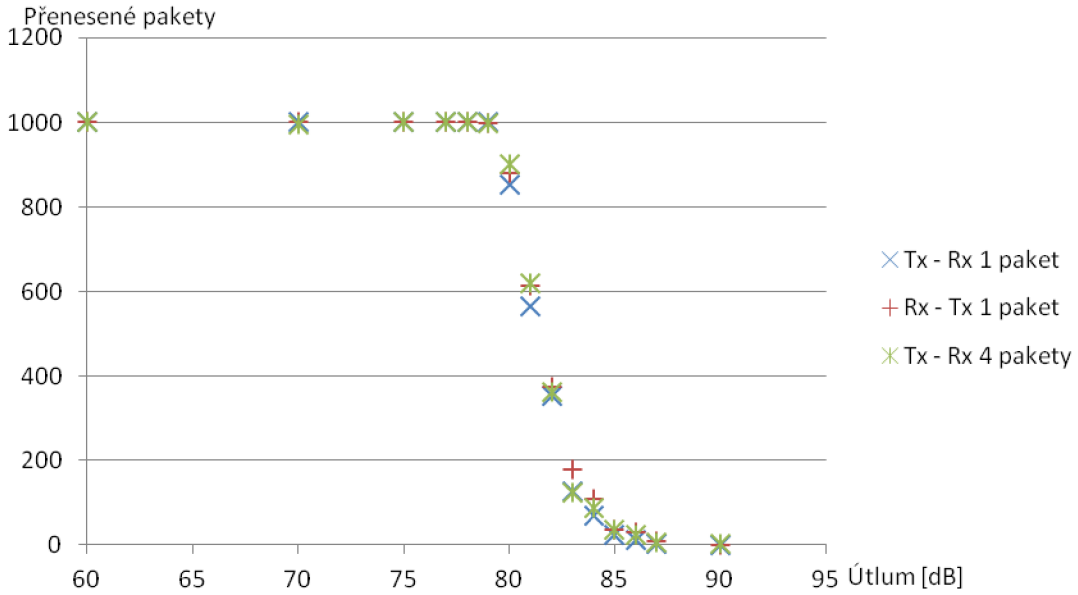
Útlum [dB]	60	70	75	77	78	79	80	81
Odeslané pakety	1000	1000	1000	1000	1000	1000	1000	1000
Přenesené pakety	1000	1000	1000	1000	1000	1000	852	564
Útlum [dB]	82	83	84	85	86	87	90	
Odeslané pakety	1000	1000	1000	1000	1000	1000	1000	
Přenesené pakety	354	125	70	22	12	1	0	

Tab. 5.5 Rx – Tx jeden paket

Útlum [dB]	60	70	75	77	78	79	80	81
Odeslané pakety	1000	1000	1000	1000	1000	1000	1000	1000
Přenesené pakety	1000	1000	1000	1000	1000	997	880	613
Útlum [dB]	82	83	84	85	86	87	90	
Odeslané pakety	1000	1000	1000	1000	1000	1000	1000	
Přenesené pakety	375	179	108	36	28	7	0	

Tab. 5.6 Tx – Rx čtyři pakety

Útlum [dB]	60	70	75	77	78	79	80	81
Odeslané pakety	1000	1000	1000	1000	1000	1000	1000	1000
Přenesené pakety	1000	996	1000	1000	1000	999	902	620
Útlum [dB]	82	83	84	85	86	87	90	
Odeslané pakety	1000	1000	1000	1000	1000	1000	1000	
Přenesené pakety	363	123	87	35	22	5	3	



Obr. 5.11 Závislost přenesených paketů na útlumu

5.3.4 Test spolehlivosti

Pro zjištění spolehlivosti byl zrealizován test obsahující 14 hodin nepřetržité rádiové komunikace. Test byl uskutečněn v podmínkách v jakých bude systém pracovat, tj. přenos vzduchem bez žádných uměle vložených útlumů. Komunikace probíhala opět simultánně oběma směry. Na obou stranách se počítaly odeslané a přijaté pakety. Komunikace nebyla rušena žádnými výpisy na UART, výsledky se vypisovaly až po ukončení testu. Jak je vidět z tabulky (viz Tab. 5.7) úspěšnost byla 100%, nedošlo ke ztrátě jediného paketu, nedošlo tedy k žádnému selhání, které by se dalo zdůvodnit chybou v návrhu protokolu RFSP2, případně chybou v implementaci.

Tab. 5.7 Test spolehlivosti

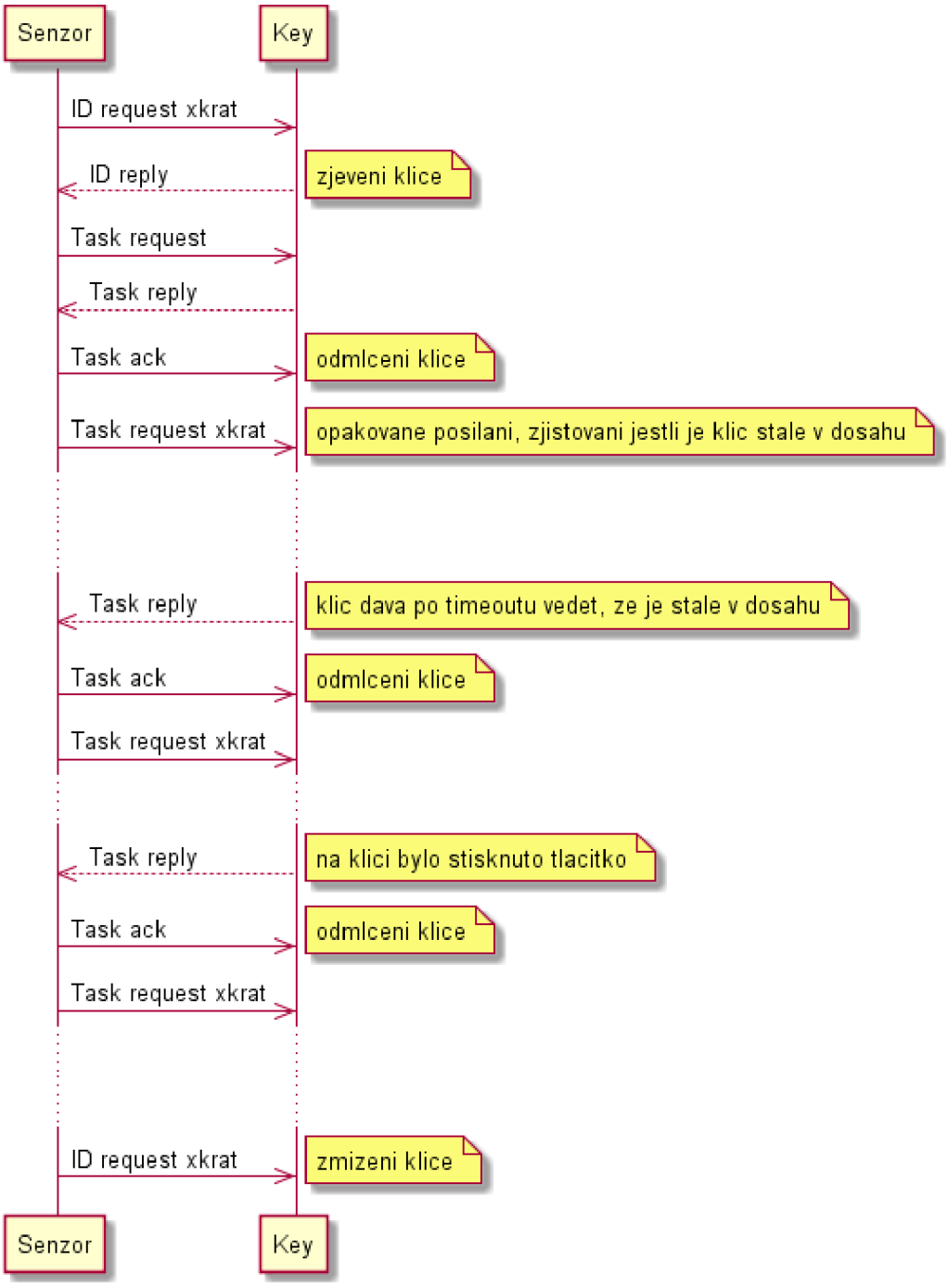
Směr komunikace	Odeslané	Přijaté
Tx – Rx	1281864	1281864
Rx – Tx	1247464	1247464

6 APLIKAČNÍ VRSTVA

Aplikační vrstva slouží k výměně zpráv pro identifikaci klíče a následném příjmu informace z klíče, podle které senzor vykoná určitou činnost. Zároveň na senzoru obsluhuje ADC převodník a vyhodnocuje informaci o pohybu v jeho okolí. Model konečného stavového automatu (finite-state machine, FSM) komunikačního protokolu byl vytvořen a simulován v open-source software Ptolemy II, vyvinutém na University of California, Berkeley [3].

Na obrázku (viz Obr. 6.1) je vyobrazené komunikační schéma pro ideální případ komunikace, nedochází k žádným ztrátám paketů. Komunikaci vždy zahajuje senzor,

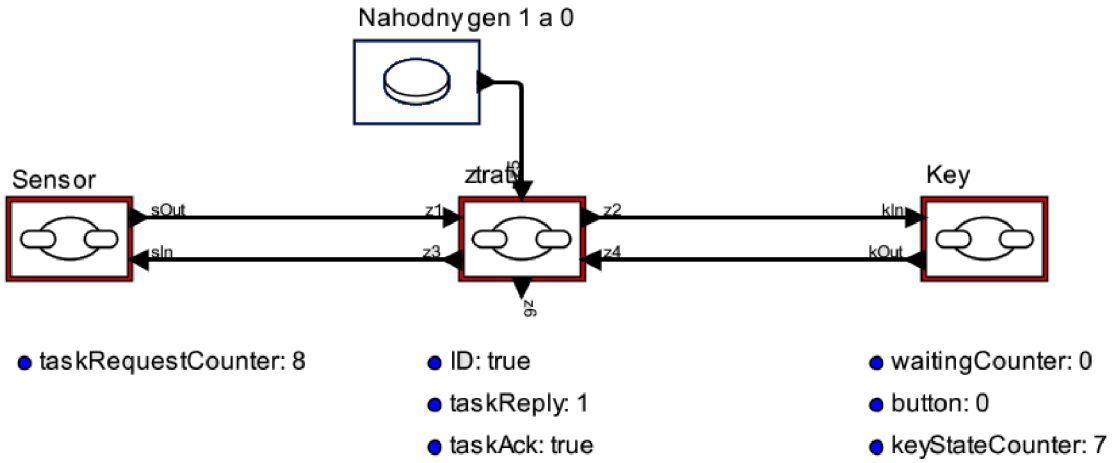
který neustále vysílá žádost o identifikaci. Jakmile se objeví v dosahu senzoru klíč a přijme ID request, odpoví svým ID. Pokud je ověření ID v pořádku, odešle senzor žádost o přidělení úkolu. Klíč odpoví informací o úkolu a senzor potvrdí přijetí úkolu a znovu posílá žádosti o úlohu. Klíč se odmlčí. Pokud je na klíči zmáčknuto tlačítko, aby se vykonala jiná úloha, odešle klíč okamžitě (je probuzen z odmlčení) úlohu senzoru, přijme potvrzení a opět se odmlčí. V určitém časovém intervalu dává klíč vědět, že je stále v dosahu senzoru. Pokud se ale senzor v tomto čase nedočká odpovědi, tak klíč zmizel z dosahu a senzor znovu začne vysílat žádosti o identifikaci.



Obr. 6.1 Komunikační schéma

6.1 Simulace FSM v Ptolemy II

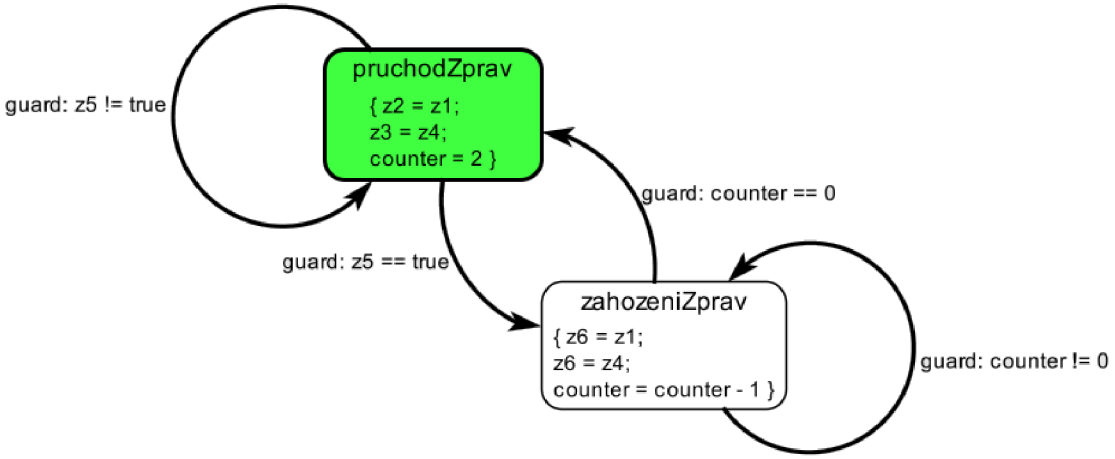
Na stavových diagramech je rozebrána komunikace podrobněji. Je řešeno chování při ztrátě paketu a dalších možných situacích. Na obrázku (viz Obr. 6.2) je zobrazena simulace komunikačního řetězce v programu Ptolemy II. [3] Jedná se o čtyři faktory: Senzor, ztrátové prostředí, klíč a náhodný generátor jedniček a nul, který rozhoduje o ztrátě paketu. O stavových diagramech senzoru a klíče je pojednáno v samostatných podkapitolách níže. Simulace komunikace proběhla v pořádku.



Obr. 6.2 Komunikační řetězec

6.1.1 FSM bloku náhodných ztrát paketů

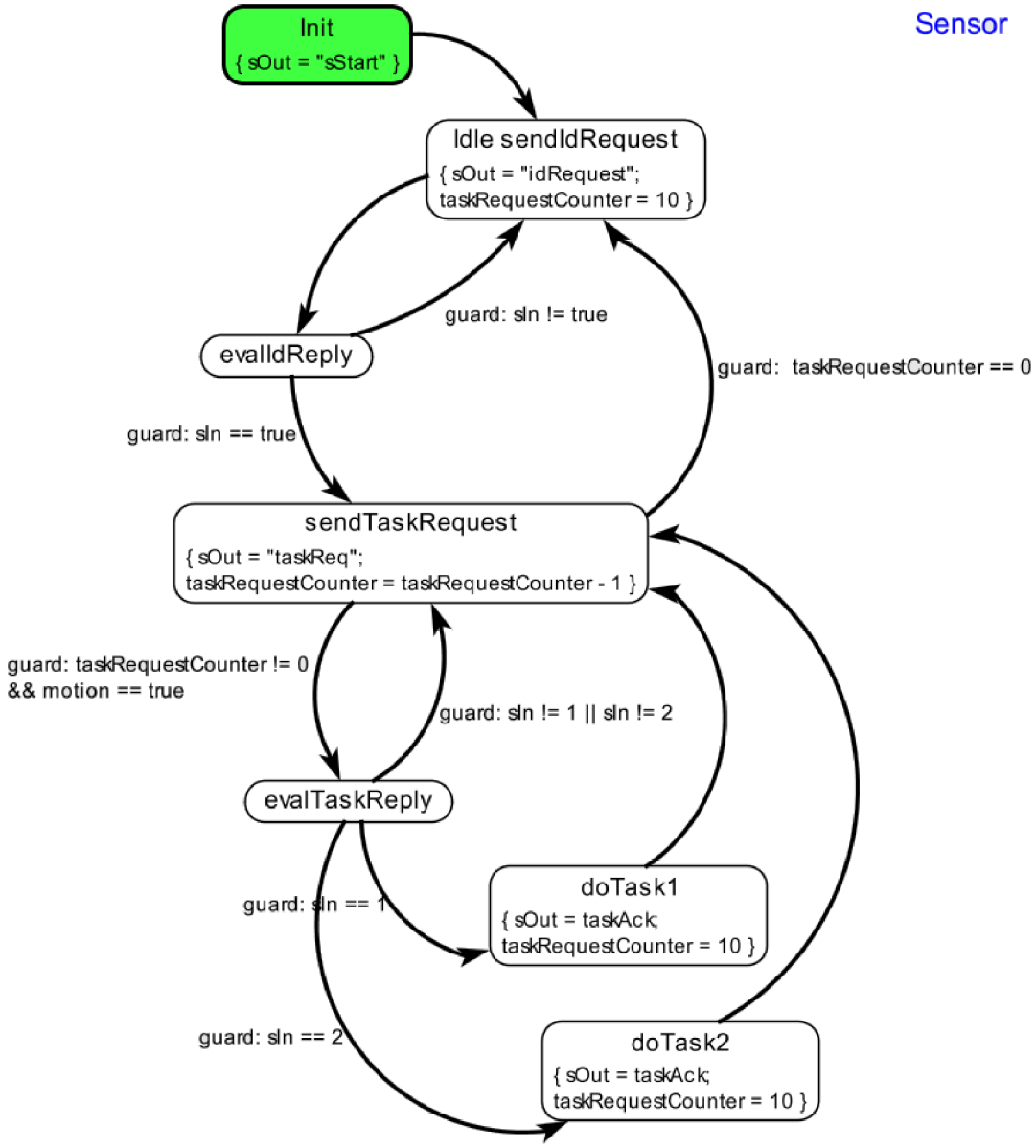
Ztráty na přenosové cestě jsou simulovány na základě náhodného generování jedniček a nul na vstup z5 z aktoru již implementovaného v Ptolemy II. Pokud přichází nuly, tak jsou data předávány mezi klíčem a senzorem. Jakmile přijde jedna, tak dojde k zahození dat na výstup z6 (viz Obr. 6.3).



Obr. 6.3 FSM bloku náhodných ztrát paketů

FSM senzor

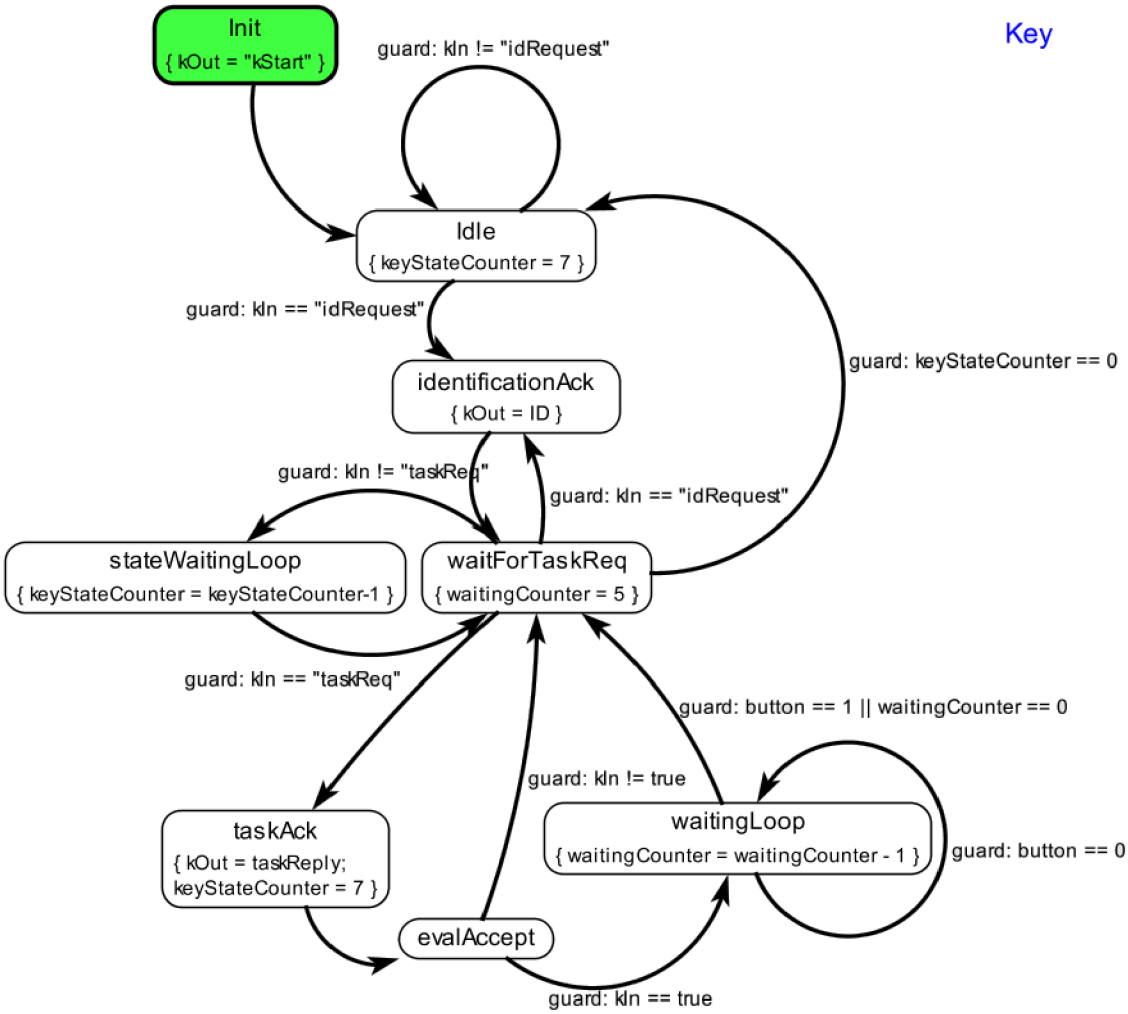
Stavový diagram senzoru (viz Obr. 6.4) ukazuje chování senzoru v průběhu všech částí komunikace. Po zapnutí začne senzor vysílat ID request, pokud nedostane odpověď, nebo odpověď nesouhlasí s očekávaným ID, tak zůstává v tomto stavu. Jakmile dostane senzor relevantní odpověď (správné ID), přesune se do stavu, kdy žádá o informaci o úkonu, který má senzor vykonat. Tato zpráva je pro klíč zároveň informací o tom, že identifikace proběhla v pořádku. Pokud klíč hned neodpoví senzor opakuje odesílání zprávy, po určitém počtu znovuodeslání (přesný počet opakování odesílání je ještě předmětem dalšího ladění) bez odpovědi se senzor vrátí do stavu Idle a opět odesílá žádosti o identifikaci. Jakmile senzor obdrží od klíče odpověď, odešle potvrzení o příjmu a vykoná daný úkol. Následně se senzor vrátí do stavu, kdy si žádá přidělení úkolu. Pokud v době opakování tohoto volání po úkolu nedostane žádnou odpověď, vrací se do stavu Idle.



Obr. 6.4 FSM Senzor

6.1.2 FSM klíč

Stavový diagram senzoru (viz Obr. 6.5) ukazuje chování senzoru v průběhu všech částí komunikace. Po zapnutí je klíč v Idle módu a čeká na příjem zprávy ID request. Jakmile zprávu přijme, tak zapíše do ACK své identifikační číslo a přesune se do stavu, kdy čeká na žádost o přidělení úlohy. Pokud identifikační číslo nedorazilo k senzoru v pořádku, dostane opět ID request a číslo pošle znovu. Jestliže nějakou dobu (opět předmět pozdějšího ladění) nic nepřijme, vrací se do stavu Idle. V případě, že dorazí žádost o úlohu, tak v ACK odpoví zrovna vybranou úlohou. Pokud obratem přijde potvrzení o přijmutí úkolu, tak se klíč dostane do čekací smyčky, ze které se vrátí do stavu, kdy čeká na žádost o úkol. Doba čekání je kratší než doba, kterou je senzor ve stavu posílání task request, aby nedocházelo k opakované identifikaci a zbytečnému vybíjení klíče. Klíč ze smyčky čekání vyskočí také v případě, že se změní úkol, který požaduje od senzoru, ve stavovém diagramu to reflektuje stisk tlačítka. Pokud klíč neobdrží nějakou dobu žádné zprávy, tak se vrací do stavu Idle.



Obr. 6.5 FSM Klíč

6.2 Realizace

Jedná o nejvyšší vrstvu systému, pro kterou byla implementována samostatná knihovna pro klíč a pro senzor. Obstarává jak identifikaci klíče, zadávání úkolů k vykonání (tyto rutiny jsou volány v hlavní smyčce programu), tak AD převod ze senzoru pohybu, tato rutina je spouštěna pomocí callback funkce časovačem T2 z nižší vrstvy. Tento spouštěč je zaveden proto, aby zabránil zahlcení procesoru neustálým voláním AD převodu. Pro správný chod aplikace byly na klíči i senzoru implementovány různé funkce, které jsou popsány níže

6.2.1 Senzor

Jedná se o komplikovanější část komunikačního řetězce, protože musí zvládat komunikovat i snímat pohyb. Nejprve, před hlavní smyčkou, je v `main` volána inicializační funkce `init`, která vyčistí prostor pro přijatou zprávu, buffer běžícího průměru pro AD převod, inicializuje AD převod a nižší vrstvu, nastaví vstupní a výstupní piny a zaregistruje callback funkci. Následuje samotná smyčka hlavního programu, ve které jsou volány tři funkce `messageInHandle`, `sendHandle` a `taskHandle`, pro testovací účely zde byla také funkce `vypisZpravuNaUart`, která vypisovala buď aktuálně přijímané zprávy nebo údaje z AD převodníku.

Funkce `messageInHandle` kontroluje, zdali byla přijata nějaká zpráva, pokud ne, tak vrací `NO_MESSAGE_IN`. Pokud zpráva přijata byla, tak se podle čísla ve zprávě rozhoduje, jestli se jedná o zprávu s ID – číslo 4, nebo o zprávu s úkolem – číslo 5. Jedná-li se o zprávu s ID následuje jeho kontrola a buď přepnutí do stavu `SEND_ID_ACK` nebo setrvání ve stavu `SEND_ID_REQ`. Když je přijata zpráva s úkolem, tak je přepnuto do stavu `DO_TASK`.

V `sendHandle` se na základě stavů odesílají různé zprávy, zároveň je zde měřen interval odmlky klíče a v případě delší odmlky se systém přepne opět do stavu žádosti o ID. Samotné odesílání je řízeno pomocí callback funkce časovačem T2 z nižší vrstvy. V callback funkci se počítá počet jednotlivých pollů. Před samotným odsláním je ponechán čas na zvládnutí všech úkonů pro zpracování jednotlivých zpráv. Pro bezproblémovou funkci bez zbytečného znovuodesílání zpráv byla stanovena frekvence 1 zpráva na 20 pollů. Jsou odesílány 4 typy zpráv: `idReqMes` – žádost o ID, `idAckMes` – potvrzení ID, následuje přepnutí na stav `SEND_TASK_REQ`, `taskReqMes` – žádost o úkol, následuje zvýšení počítadla pro měření doby odmlčneí, `taskAckMes` – potvrzení úkolu, následuje přepnutí na stav `SEND_TASK_REQ`. Pokud žádná z výše popsaných možností neproběhne, tak je vrácen `SEND_HANDLE_ERROR`.

Funkce taskHandle pouze vykoná podle čísla úkolu požadovaný úkol a přepne stav na `SEND_TASK_ACK`.

Snímání pohybu je díky řízení callback funkcí prováděno vždy jednou za 5 ms. Rozlišení AD převodníku je 10 bitů. Pro vyčítání dat byl proto vytvořen typ union se dvěma byty. Kvůli vyhlazení výstupu z AD převodníku byl použit filtr typu dolní propust, implementován jako běžící průměr. Celý algoritmus zpracování signálu ze senzoru pohybu je následující. Protože mikroprocesor nezvládá násobení čísel s výsledkem větším než 16 bitů, je nejprve hodnota z AD převodníku vydělena 10. Tím sice dojde ke ztrátě přesnosti, ale pro detekci pohybu větších objektů (člověk, auto) je

stále dostačující. Následuje vyrovnání na nulovou úroveň. Klidový signál ze senzoru pohybu se nachází zhruba v polovině rozsahu AD převodníku. Teoretická hodnota od které by se mělo odečítat je tedy 51,2, toto číslo je zaokrouhleno na 51, s tím že na nulové hodnotě, která značí žádný pohyb je ve většině případů vrácena hodnota 1. Signál nicméně mírně kolísá, takže hodnota osciluje mezi 0 a 4. Identifikace pohybu je založena na analýze energie signálu ze senzoru, takže pro další kroky se již používá druhá mocnina hodnoty vzorků. Následuje vyhlazení v průměrovacím kruhovém bufferu, kde se vždy odečte nejdříve zapsaný prvek a na místo tohoto prvku se zapíše nový prvek, přičte se k celkovému součtu a ten se vydělí celkovým počtem prvků v průměrovacím bufferu. Nakonec se o jedno posune ukazatel na prvek v bufferu a vrátí se výsledek.

6.2.2 Klíč

Klíč se stará pouze o datovou komunikaci, obsluhu dvou tlačítka, dvou LED a proces odmlčování (úspora energie). Opět je před hlavní smyčkou inicializační funkce, ve které je vyčištěn prostor pro přijatou zprávu, inicializováno RFSP2, nastaveny vstupní a výstupní piny, zhasnuty LED a zapsáno ID do zprávy s ID. Poté je už volána hlavní smyčka, která obsahuje funkce: `LEDs`, `buttons`, `timeoutLoop` a `messageInAndSendHandle`. Kvůli testování byla také dočasně implementována funkce `vypisZpravuNaUart`.

Ve funkci `LEDs` jsou pouze kontrolovány návěstí, zdali je klíč v komunikačním dosahu senzoru a jestli proběhla úspěšná identifikace. Pokud klíč není v dosahu, tak nesvítí žádná LED, pokud je v dosahu a není identifikován, tak svítí pravá LED (z horního pohledu pro konektory dole) a v případě že je v dosahu a je identifikován, tak svítí levá LED a pravá LED blikne při příjmu potvrzení od senzoru o přijetí úkolu.

Funkce `buttons` se stará o obsluhu dvou tlačítek. Tlačítka jsou skenována pouze tehdy, pokud je potvrzeno správné ID. Stisk tlačítka se kontroluje podle toho, zdali je na pin, na kterém je tlačítko připojeno přivedena vysoká úroveň. Aby nedocházelo k nechtěnému opakovanému stisku tlačítka, tak je zároveň kontrolován flag, který se nahodí při prvním stisku a nuluje až při puštění. Toto řeší většinu zákmitů, nicméně občas se ještě některý zákmit při puštění tlačítka vyhodnotí jako nové stisknutí, řešení tohoto problému by mohlo být zohledněno v hardwarovém návrhu dalších prototypů klíče. Po každém stisku tlačítka je nejprve zapnuto rádio (pro případ, že by se klíč zrovna nacházel ve fázi odmlčení), následuje přiřazení čísla zvoleného úkolu a vynulování časovacích počítadel.

Funkce `timeoutAndRangeChecking` zajišťuje počítadlo času odmlčení klíče a počítadlo času pro kontrolu, jestli je klíč v dosahu. Frekvence pollů od senzoru podle kterých se ověřuje zdali je klíč v dosahu, je jednou za 5ms, nemá proto cenu ptát se častěji. Klíč to tedy kontroluje jednou za dva polly. Pokud se klíč dostane mimo dosah senzoru, dojde k nastavení příznaku, že identifikace proběhla v pořádku a do okamžiku další identifikace již se senzorem nekomunikuje.

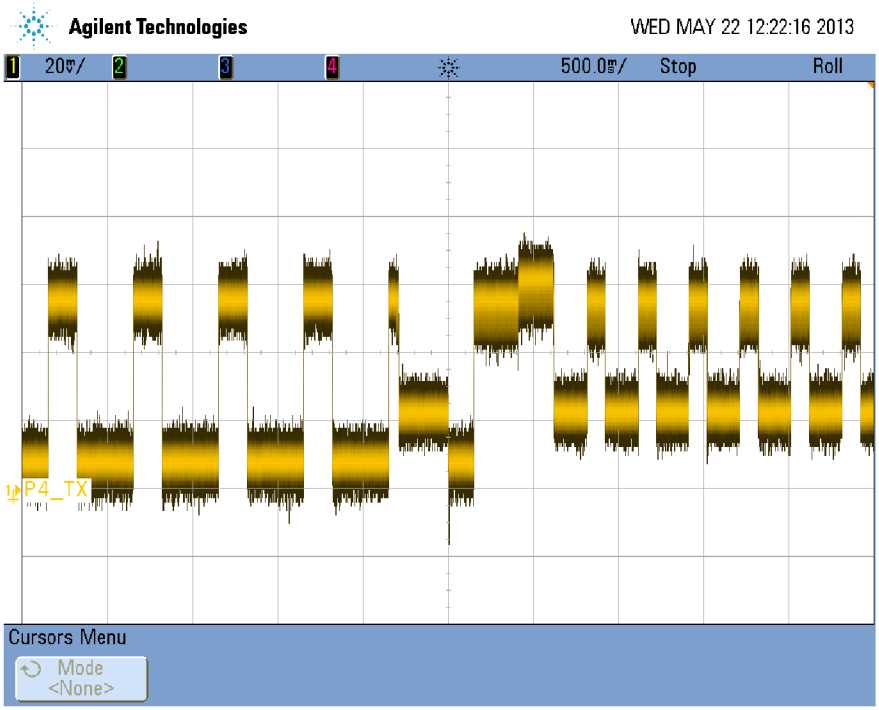
Správu datových zpráv zajišťuje metoda `messageInAndSendHandle`. Nejprve se zkontroluje, jestli je přijata zpráva, pokud ne, tak je vráceno `NO_MESSAGE_IN`, pokud ano, tak se o dalším postupu rozhoduje podle čísla v přijaté zprávě. Když přijde zpráva se žádostí o ID – 1, je odesláno ID. Pokud přijde zpráva s potvrzením správného ID – 2,

tak se nahodí `IDOkFlag` a nastaví se číslo úkolu na úkol, který má být vykonán ihned po identifikaci. Pokud přijde zpráva se žádostí o úkol – 3 a zároveň je potvrzené správné ID, tak se přiřadí číslo úkolu a odešle se zpráva s číslem úkolu. Vpřípadě že přijde zpráva s potvrzením o přijmutí úkolu – 4 a zároveň je potvrzené správné ID, tak blikne LED, vynuluje se číslo úkolu (nula v čísle úkolu je pouze pro potvrzení, když se senzor ptá zdali je klíč v dosahu, žádný úkol vykonán není) a vypne se rádio – klíč se odmlčí.

Odmlčování klíče probíhá ve dvou různých situacích. Jedná se o situace kdy klíč je a kdy není v dosahu senzoru. Pokud je v dosahu, tak se odmlčuje pouze přepínáním rádia do standby-I módu, protože musí skenovat tlačítka, vykonávat identifikační rutinu atp. V případě že není v dosahu senzoru, tak nic z toho dělat nemusí, proto je klíč přepínán do tzv. register retention, timers on módu (viz [1]). V tomto módu je zapnutý pouze oscilátor, ze kterého je syntetizován timer RTC2 od kterého přichází přerušení a znovu nahazuje systém a rádio. Intervaly odmlčování patrné z tabulky (viz Tab. 6.1) a obrázku (viz Obr. 6.6) na kterém je v polovině vidět přechod do dosahu senzoru.

Tab. 6.1 Odmlčování klíče

	V dosahu	**Mimo dosah**
Perioda [ms]	400	500
Aktivní [ms]	100	150
Neaktivní [ms]	300	350



Obr. 6.6 Odmlčování klíče

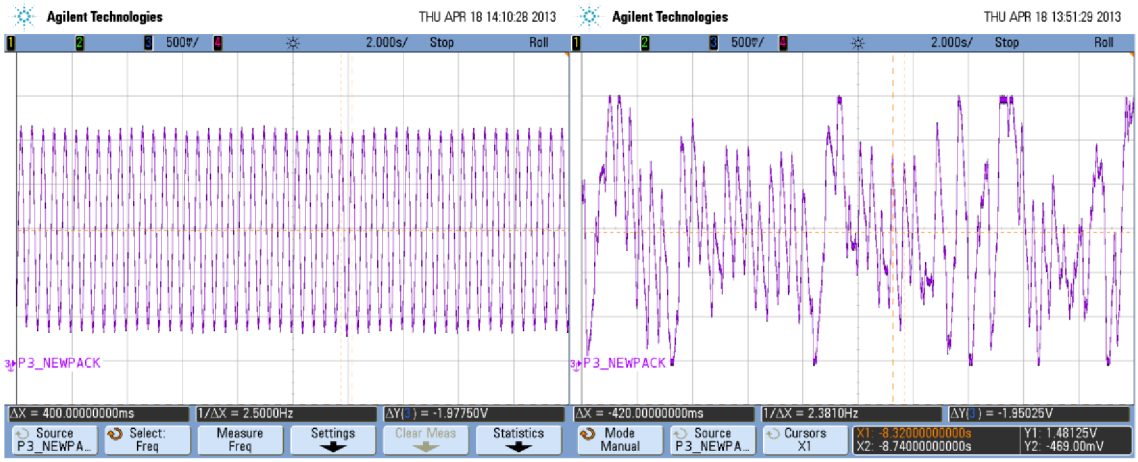
6.3 Test implementace

6.3.1 Komunikace

Pro ověření chování při zmizení a objevení klíče v dosahu senzoru bylo použito stejné zapojení testu jako pro měření počtu přenesených paketů na velikosti vloženého útlumu pomocí atenuátoru. Test potvrdil předchozí měření. Do útlumu 80dB se neobjevil žádný problém, jak v signalizaci o správné identifikaci, tak v odesílání úkolů. Se stoupajícím útlumem se začaly vlivem větší ztrátovosti paketů projevovat problémy s potvrzením identifikace, takže klíč signalizoval pouze to, že je v dosahu senzoru. Při ještě větším útlumu, zhruba od 83dB a výše, už klíč úplně vypadl z dosahu senzoru. Při snížení útlumu pod 80dB se klíč opět v pořádku identifikoval a byl schopen odesílat žádosti o úkol.

6.3.2 Senzor

Měření výstupu průměrování AD převodníku pro orientační hodnoty vrácené při různé intenzitě pohybu. Měřící aparatura byla stejná jako v případě měření pseudopohybu (viz výše). Pohyb byl opět simulován pomocí napěťovým generátorem řízeným blokem fázového posuvu. Pro srovnání je uveden pohyb simulovaný (viz Obr. 6.7 vlevo) u kterého se frekvence přibližně blížila skutečnému pohybu a skutečný náhodný pohyb v blízkosti senzoru (viz Obr. 6.7 vpravo).

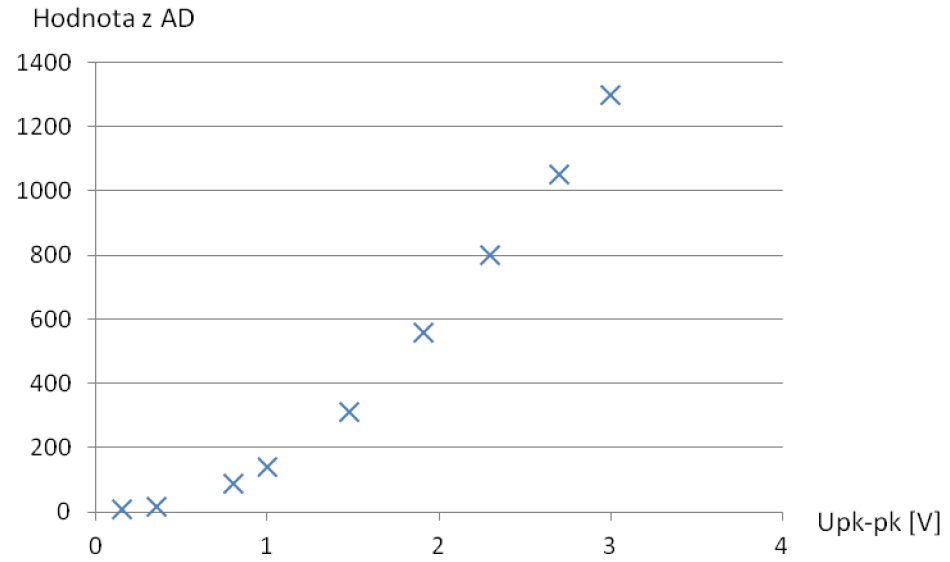


Obr. 6.7 Srovnání simulovaného a skutečného pohybu

Hodnoty (viz Tab. 6.2 a Obr. 6.8) jsou jen velmi orientační, protože nedochází k dokonalému vyhlazení signálu v průměrovacím bufferu, takže hodnoty kolísají cca 15% na obě strany od uváděné hodnoty. Maximální hodnota, kdy je senzor absolutně saturován – stálá krajní hodnota, žádné kolísání nebo harmonický děj, nezáleží na kterou stranu od hodnoty bez pohybu – je absolutních 2601. Změřené napětí je napětí na výstupu senzoru, 3V je maximální hodnota, kterou výstup generuje. Hodnota je průměrovaná hodnota z AD převodníku.

Tab. 6.2 Tabulka průměrovaných hodnot z AD převodníku

U_{pk-pk} [V]	0,15	0,35	0,80	1,00	1,48	1,91	2,30	2,70	3,00
hodnota	7	17	90	140	310	560	800	1050	1300

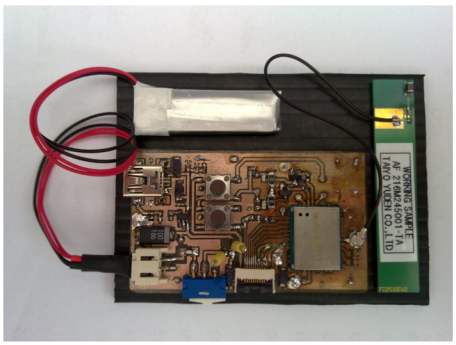


Obr. 6.8 Závislost hodnot z AD převodníku na napětí

7 HW REALIZACE KLÍČE

Pro sérii testů byl vytvořen prototyp aktivního klíče.

Klíč byl realizován na oboustranné desce plošných spojů. Srdcem návrhu je čip společnosti ALPS UMRM1 X001B obsahující nRF24LE1 a senzor pohybu. Klíč dále obsahuje dvě LED ovládané přes tranzistory, dvě tlačítka spínaná vysokou úrovní napájecího napětí, dvoustavový vypínač a programovací konektor FFC s osmi vývody. Napájení je řešeno pomocí li-lon akumulátoru s kapacitou 120 mAh, který lze nabíjet přes miniUSB konektor. Jelikož realizace antény a jejího přizpůsobovacího obvodu přímo na DPS nebyla v rámci této práce možná, byl použit již přizpůsobený vzorek čipové antény, připojený přes U.FL konektor. Realizace je vidět na obrázku (viz Obr. 7.1). Všechny použité součástky kromě vypínače jsou typu SMD. Schéma, seznam součástek, návrh desky a osazovací plán jsou v příloze.



Obr. 7.1 Realizace klíče

7.1 Měření spotřeby

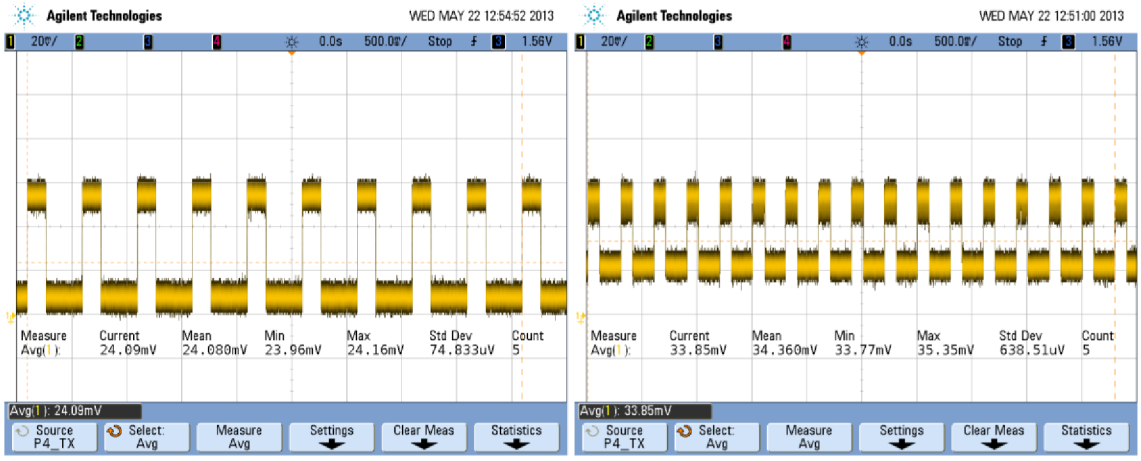
Rezistory R4 a R10 slouží k měření odběru proudu klíče. V normálním provozu je na místě R10 rezistor s nulovým odporem, R4 zůstává celou dobu neosazen. Pro měření se vymění R10 za rezistor s malou hodnotou, v případě tohoto měření 2,7Ω a na ploškách R4 se měří napětí. Bylo měřeno průměrné napětí pomocí osciloskopu Agilent DSO7014B měření průměru. Pro případ mimo dosah na 9 periodách odmlčení, pro případ v dosahu na 14 periodách odmlčení (viz Obr. 7.2). Pro oba případy bylo provedeno pět měření a vzala se průměrná hodnota. Výsledky měření jsou vidět v tabulce (viz Tab. 7.1)

Tab. 7.1 Měření spotřeby

V dosahu	Napětí [mV]	Měřicí Odpor [Ω]	Proud [mA]
Ne	24,08	2,7	8,92
Ano	33,85	2,7	12,54

Proud byl počítán podle rovnice $I = \frac{U}{R} = \frac{0{,}02408}{2{,}7} = 8{,}92mA$ (7.1) [5]:

$$I = \frac{U}{R} = \frac{0{,}02408}{2{,}7} = 8{,}92mA \qquad (7.1)$$



Obr. 7.2 Měření spotřeby

8 TEST SYSTÉMU

V závěrečném testu jsou testovány základní parametry systému, dosah senzoru, dosah datové komunikace, schopnost senzoru rozlišit různé úkoly zadané na klíči tlačítky a

schopnost rozlišit obecný pohyb od pohybu s klíčem. Foto měřícího pracoviště a použitého prototypu senzoru a jeho vyzařovacích charakteristik je v příloze. [4]

Pro potřeby testování bylo v aplikační vrstvě senzoru přidáno periodické odesílání jednoho měřícího informačního bytu na UART. Podoba měřícího informačního bytu je patrná z následující tabulky (viz Tab. 8.1).

Tab. 8.1 Informační byte pro měření

Bit	Využití
7 (MSB)	Identifikovaný klíč v dosahu
6	Přijat úkol 1
5	Přijat úkol 2
4	Výrazný pohyb
3	Pohyb
2	Pohyb
1	Pohyb
0 (LSB)	Malý pohyb

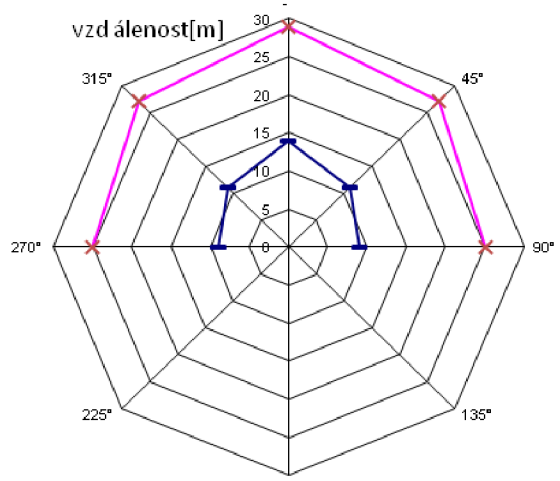
Při měření byl senzor umístěn na stativ a měřilo se pro dvě výšky umístění senzoru 0,95 m a 2,1 m. Pokusná osoba s klíčem i bez klíče se pohybovala v různé vzdálenosti a úhlu od senzoru. Foto měřícího pracoviště je v příloze. Dosah datové komunikace je patrný z tabulky a grafu vlevo (viz Tab. 8.2 a Obr. 8.1) pro 2,1 m a z tabulky a grafu vpravo (viz Tab. 8.3 a Obr. 8.2) pro 0,95 m. Oblast „Komunikace“ znamená, že klíč komunikoval bez výpadku identifikace, oblast „Problémy“ značí, že klíč sice komunikoval, ale docházelo k občasným selháním identifikace či celkové ztrátě signálu. Vždy, ale došlo k obnovení komunikace. Oblast „Mimo dosah“ značí úplný výpadek komunikace bez obnovení.

Tab. 8.2 Měření vzdálenosti komunikace pro výšku senzoru 2,10 m

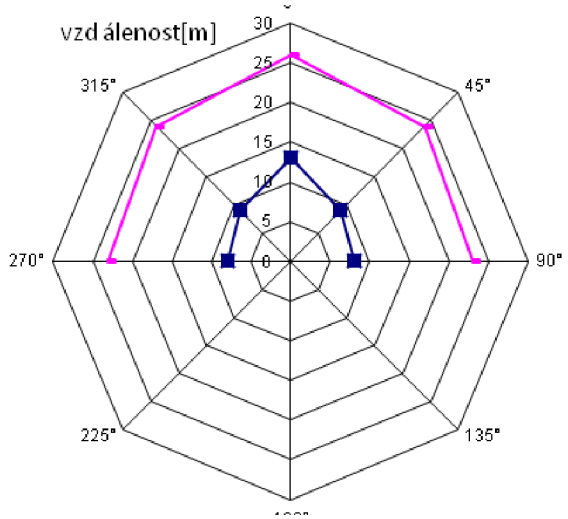
Úhel	Komunikace [m]	Problémy [m]	Mimo dosah [m]
0°	0-14	14-29	29+
45°	0-11	11-27	27+
90°	0-9	9-25	25+

Tab. 8.3 Měření vzdálenosti komunikace pro výšku senzoru 0,95 m

Úhel	Komunikace [m]	Problémy [m]	Mimo dosah [m]
0°	0-13	13-26	26+
45°	0-9	9-24	24+
90°	0-8	8-23	23+



Obr. 8.1 Měření vzdálenostsi komunikace pro výšku senzoru 2,1 m



Obr. 8.2 Měření vzdálenostsi komunikace pro výšku senzoru 0,95 m

Dosah snímání pohybu je patrný z tabulky (viz Tab. 8.4), vzhledem k nízkým hodnotám bylo provedeno pouze měření v úhlu 0° od senzoru.

Tab. 8.4 Měření vzdálenosti snímání pohybu

Výška senzoru [m]	**Dosah senzoru [m]**
2,10	1,5
0,95	2,3

Z měření je patrné, že dosah datové komunikace mnohonásobně přesahuje dosah snímání pohybu. Pro měření byl bohužel k dispozici pouze modul senzoru HW konfigurovaný pro měření malých biologických pohybů (pohyb srdečního svalu, respirace). Dle dostupných informací je možné s HW úpravou interferometru dosáhnout snímání pohybu až 6 m. [4] Nicméně v krátké vzdálenosti senzor bezpečně rozeznal pohyb s klíčem a bez něj (signalizace 0. bitem). Také přijímání úkolů z klíče proběhlo bez problémů. Rozdílné větší hodnoty snímání pohybu a datové komunikace při

různých výškách jsou způsobeny tím, že při nižší výšce se testovací osoba pohybovala bezprostředně před senzorem.

9 NÁVRHY DALŠÍHO POSTUP

Provedená práce se dá považovat za stavební kámen pro složitější a komplexnější systémy. V této kapitole jsou nastíněny body, kterými by bylo možno dále postupovat ve zlepšování tohoto řešení. Tam, kde je to možné, jsou zároveň jsou uvedeny problémy a omezení zjištěné v průběhu zpracování tohoto projektu.

9.1 Odstranění zbytečně dlouhého odesílání

V prvé řadě by se měl odstranit problém s odesíláním během čtyř pollů místo dvou (pokud je to v této konfiguraci možné). Tím se zajistí dvojnásobná přenosová rychlost a poloviční latence mezi odesláním a příjmem.

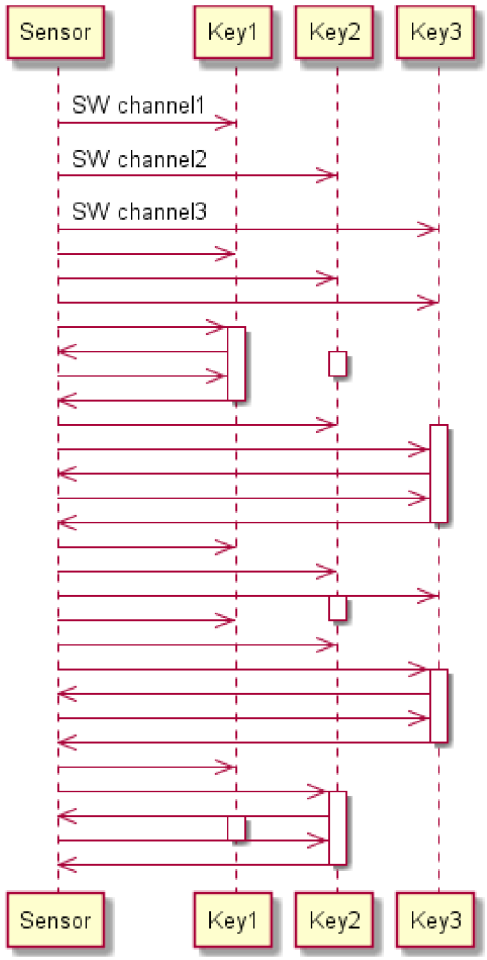
9.2 Kooperace více klíčů

Další fází vývoje je kooperace více klíčů zároveň. Nabízí se několik možných řešení tohoto problému.

Například by mohl každý klíč vysílat na jiné frekvenci a senzor by si v pravidelných intervalech přepínal nosnou. Toto řešení by ovšem znemožnilo důvěryhodné snímání pohybu, protože přepínání nosné na senzoru způsobuje pseudopohyb (viz výše).

Další možností by bylo využití vícekanálové komunikace Enhanced ShockBurst™. Toto řešení je nevhodné proto, že se jedná o komunikace více PTX (senzorů) a jedním PRX (klíč), zde by to bylo přesně naopak. Navíc přepínání kanálů také způsobuje pseudopohyb.

Jediným schůdným řešením se zdá náhodné odmlčování klíčů, tento přístup se ukázal jako bezkonfliktní se snímáním pohybu při řešení úspory energie klíče. Pro rozlišení klíčů by bylo použito softwarové oddělení kanálů. Klíč by každý paket začínal svým identifikačním bytem, to by vedlo k malému snížení přenosové rychlosti – paket by přišel o jeden datový byte, nicméně toto není systém pro přenos větších objemů dat. Senzor by také musel uvádět každý paket příslušným identifikačním bytem. Pakety jiných klíčů by klíče zahazovaly. Proti zabránění opakování kolizí, by se klíče odmlčovali v náhodných intervalech za použití generátoru náhodných čísel umístěného na čipu (viz [1]). Možný průběh komunikace je znázorněn na sekvenčním diagramu (viz Obr. 9.1). Senzor by pravidelně střídal softwarové kanály a pokud by dostal odpověď od klíče, tak by si mezi sebou vyměnili jeden paket (minimálně dva polly). Pokud by klíč přijal cizí paket, tak by se okamžitě vypnul, aby nerušil případnou komunikaci.



Obr. 9.1 Kooperace více klíčů

9.3 Bezdrátová konfigurace senzoru

Dalším rozšířením by mohla být implementace bezdrátové konfigurace senzoru. V praxi bývají senzory pohybu umístěny na místech, se ztíženým přístupem (ve výškách apod.) Existence bezpečného a oboustranného komunikačního protokolu umožňuje vytvořit softwarové a hardwarové nástroje, pomocí kterých by bylo možné senzor nakonfigurovat vzdáleně. (Například z PC přes UART s lehce modifikovaným klíčem).

9.4 Snižování spotřeby klíče

Při současné spotřebě cca 9mA mimo dosah senzoru a 12,5 mA v dosahu senzoru by klíč s použitou baterií s kapacitou 120mAh vydržel zhruba pouhých dvanáct hodin. Proto je vhodné se při dalším vývoji soustředit také na snižování spotřeby klíče.

Jednak se mohou zvyšovat intervaly mezi probouzením klíče a prodlužovat doba odmlčení. Také by se mohl snížit vysílací výkon klíče, což by mělo za následek i srovnání vzdáleností mezi dosahem datové komunikace a dosahem snímání pohybu na senzoru.

ZÁVĚR

V diplomové práci jsem se zabýval problematikou řešení detekce pohybu doplněného o rádiovou identifikaci. V úvodu práce je seznámení s možnostmi RF senzoru firmy ALPS Electric, který obsahuje interferometr pro měření pohybu a integrovaný transciever Nordic nRF24LE1. Následuje rozbor systému z hlediska softwaru a je popsán způsob měření pohybu a osvětlena problematika pseudopohybu. Po rozsáhlém zmapování výskytu pseudopohybu a vyzkoušení možností datového protokolu na nRF24LE1 Enhanced ShockBurst™ bylo přistoupeno k samotnému návrhu systému. Systém sestává ze dvou, respektive tří vrstev, nejnižší vrstvu tvoří Enhanced ShockBurst™, na něm byl implementován komunikační protokol RFSP2, na kterém byla postavena aplikační vrstva s identifikačním protokolem.

Paketový komunikační protokol slouží k oboustranně rovnocenné výměně zpráv mezi senzorem a klíčem. Při řešení tohoto protokolu musel být brán zásadní zřetel na možný výskyt pseudopohybu, jinak by správně nefungoval pohybový senzor. Tento problém se podařilo vyřešit a výsledkem je paketový komunikační protokol, který zvládá datovou komunikaci při současném snímání pohybu na jednom rádiovém kanálu aniž by se vzájemně ovlivňovali. Byla vytvořena samostatná knihovna pro využití protokolu v dalších aplikací s tímto senzorem. V diplomové práci je popsán princip a série testů protokolu.

Vývoj aplikační vrstvy byl proveden v systémovém simulačním programu Ptolemy II. V této vrstvě je řešeno snímání pohybu senzorem, identifikační protokol mezi senzorem a klíčem a zadávání úkolů z klíče na senzor. Všechny tyto funkce byly implementovány ve dvou knihovnách zvlášť pro klíč a zvlášť pro senzor. Opět bylo provedeno několik testů.

Následoval návrh a implementace HW prototypu klíče. Byl kladen důraz na malé rozměry (proto byly použity SMD součástky) a nízkou spotřebu. Té se docílilo systémem odmlčování klíče přidaným do aplikační vrstvy. Nicméně další snižování spotřeby a rozměrů by mohlo být námětem dalšího vývoje.

Při ověření funkčnosti celého systému bylo zjištěno, že datová komunikace má mnohem větší dosah. To bylo dáno senzorem, který byl dostupný pouze ve verzi pro snímání malých pohybů, při použití správného senzoru se rozdíl zmenší. Systém dokázal rozlišit obecný pohyb od pohybu osoby s klíčem.

Na závěr práce je navržen další možný postup při návrhu komplexnějších systémů. Byla rozpracovaná teorie pro kooperaci více klíčů, bezdrátovou konfiguraci senzorů a navrženy podmínky pro snížení spotřeby klíče a zmírnění rozdílů mezi dosahem snímání pohybu a datové komunikace.

.

LITERATURA

[1] *Nordic Semicoductors nRF24LE1*[online]. Datasheet, 196 s., 2010. Dostupné na www: <http://www.nordicsemi.com/>.

[2] *nAN24-15 Creating Application with the Keil™ C51 C Compiler* [online]. Application Note, 37 s., 2010. Dostupné na www: <http://www.nordicsemi.com/>.

[3] UNIVERSITY OF CALIFORNIA, BERKELEY, *Heterogeneous Concurrent Modeling and Design in Java (Volume 1: Introduction to Ptolemy II)* [online]. Technical Report 304 s., 2008. Dostupné na www: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-28.pdf>

[4] Interní materiály ALPS Electric

[5] ĎUROVIČ, S.- WILFERT,O.: Radioreléové spoje. Knižnice odborných a vědeckých spisů VUT v Brně – sv. A-53 1991

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

F_0	Frekvence nosné

MCU	Micro Controllel Unit
RF	Radiofrekvenční
GFSK	Gaussian Frequency-Shift Keying
HW	Hardware
RAM	Random Access Memory
AES	Advanced Encryption Standard
MDU	Mains Distribution Unit
ADC	Analog Digital Converter
IO	Input/Output
QFN	Quad-flat no-leads package
ISM	Industrial, Scientific and Medical band
ACK	Acknowledgment
TX	Transmission
RX	Recieve
FIFO	First In, First Out
PTX	Primary Transmission
PRX	Primary Recieve
PLL	Phase Lock Loop
CRC	Cyclic Redundancy Check
RPD	Received Power Detector
DO	Dákové ovládání
LED	Light Emitting Diode
GPIO	General Purpose Input Output
RFSP2	RF Sensor protocol 2
AD	Analog Digital
FW	Firmware
FSM	Finite State Machine
ID	Identification

PŘÍLOHA1: KNIHOVNY SYSTÉMU

```
/**
 * \file rfsp2.h
 * \brief Communication protocol based on Enhanced ShockBurstTM
 * Author: Vaclav Halbich
 * Date: 23.05.2013
 */

#include <stdint.h>
#include <stdbool.h>

/**
 * Initialize Master (TX) or Slave (RX).
 * Depends on #define MASTER_TX or #define SLAVE_RX.
 * Must be set in rfsp2.c!!!
 */
void rfsp2_init(void);

/**
 * Send the message to buffer.
 * Max 50 chars.
 * \param   txPay pointer to source array
 * \param   length      length of the source array
 * \return  TRUE  message is in buffer
 * \return  FALSE buffer is full
 */
uint8_t send(uint8_t *txPay, uint8_t length) small reentrant;

/**
 * Assembly the message from RXbuffer.
 * \param   message     pointer to finish array
 * \return  length      returns message length
 */
uint8_t assemblyMessage(uint8_t *message) small reentrant;

/**
 * Info about the incoming message.
 * \return  length      returns number of messages in buffer
 */
uint8_t incomeMessage(void) small reentrant;


/**
 * Info about the reset.
 * \return  TRUE  returns if the rfsp2 was reseted
 * \return  FALSE returns if the rfsp2 wasn't reseted
 */
uint8_t resetInfo(void) small reentrant;

/**
 * Info if comunnication running.
 * Used in SLAVE_RX.
 * \return  TRUE  returns if comunnication running
 * \return  FALSE returns if comunnication not running
 */
uint8_t communicationCheck(void);
```

```
/**
 * Info about polling triggered by TIMER2.
 * Used in MASTER_TX.
 * \return  TRUE  returns if there was new poll
 * \return  FALSE returns if there wasnt new poll
 */
uint8_t wasTherePoll(void);

/**
 * Disable RF transceiver.
 * Transciever is set to standby-I mode.
 * Used in SLAVE_RX.
 */
void disableRF(void);

/**
 * Enable RF transceiver.
 * Transciever is set to RX mode.
 * Used in SLAVE_RX.
 */
void enableRF(void);

/**
 *  Copmletely  flushes  all  Tx  buffers  (in  RFSP2  and  in  Enhanced
ShockBurst)
 */
void flushTx(void);

/**
 *  Copmletely  flushes  all  Rx  buffers  (in  RFSP2  and  in  Enhanced
ShockBurst)
 */
void flushRx(void);

/**
 * Registering callback for info about polls.
 * Used in MASTER_TX
 * \param   Function which you want to call after poll
 */
void registerSendTrigger(uint8_t (*sendTriggerCallback)(void *));



/**
 * \file appSensor.h
 * \brief Application layer of the sensor
 * Author: Vaclav Halbich
 * Date: 23.05.2013
 */

#include <stdint.h>
#include <stdbool.h>

/**
 * Initialize oscillator, radio, set GPIO etc.
 */
void initSensor(void);
```

```
/**
 * Handle incoming messages, operating necessary.
 * \return  info byte about past action
 */
uint8_t messageInHandle(void);

/**
 * Sending info byte to UART.
 */
void infoUART(void);

/**
 * Send messages, operating necessary.
 * \return  info byte about past action
 */
uint8_t sendHandle(void);

/**
 * Handle tasks, operating necessary.
 * \return  info byte about past action
 */
uint8_t taskHandle(void);
```

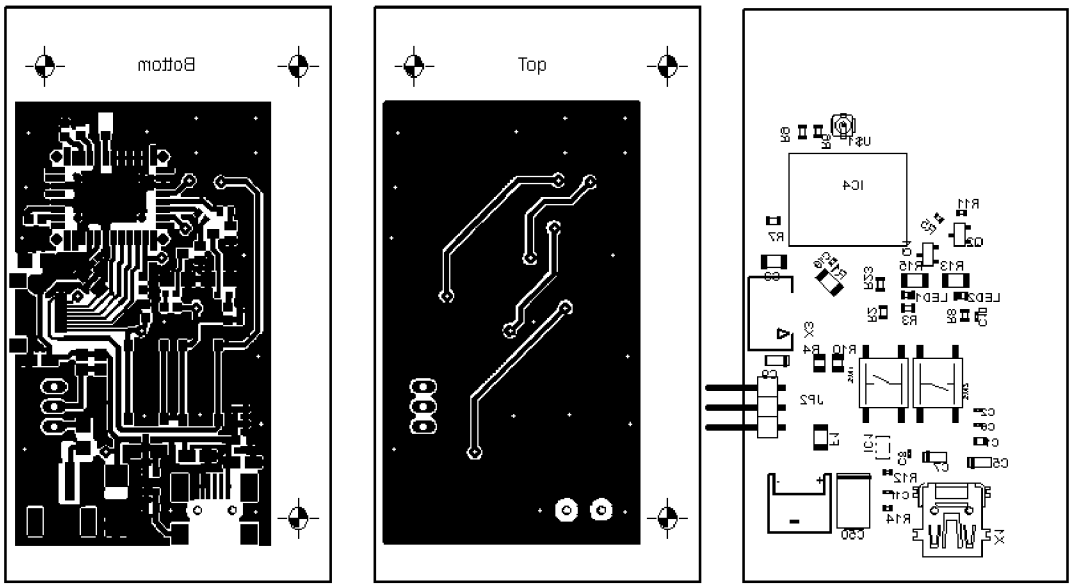
```
/**
 * \file appKey.h
 * \brief Application layer of the active key
 * Author: Vaclav Halbich
 * Date: 23.05.2013
 */

#include <stdint.h>
#include <stdbool.h>

/**
 * Initialize oscillator, radio, set GPIO etc.
 */
void initKey(void);

/**
 * Info about communication and identification.
 */
void LED(void);

/**
 * Handle buttons.
 */
void buttons(void);

/**
 * Timing for lull, operating necessary.
 * \param   time for lull if sensor is in range (10000 - 100000)
 * \param   time for lull if sensor is out range (1 - 10)
 * \return  info byte about past action
 */
uint8_t   timeoutAndRangeChecking(uint32_t   maxCountInRange,   uint8_t
maxCountOutRange);
```

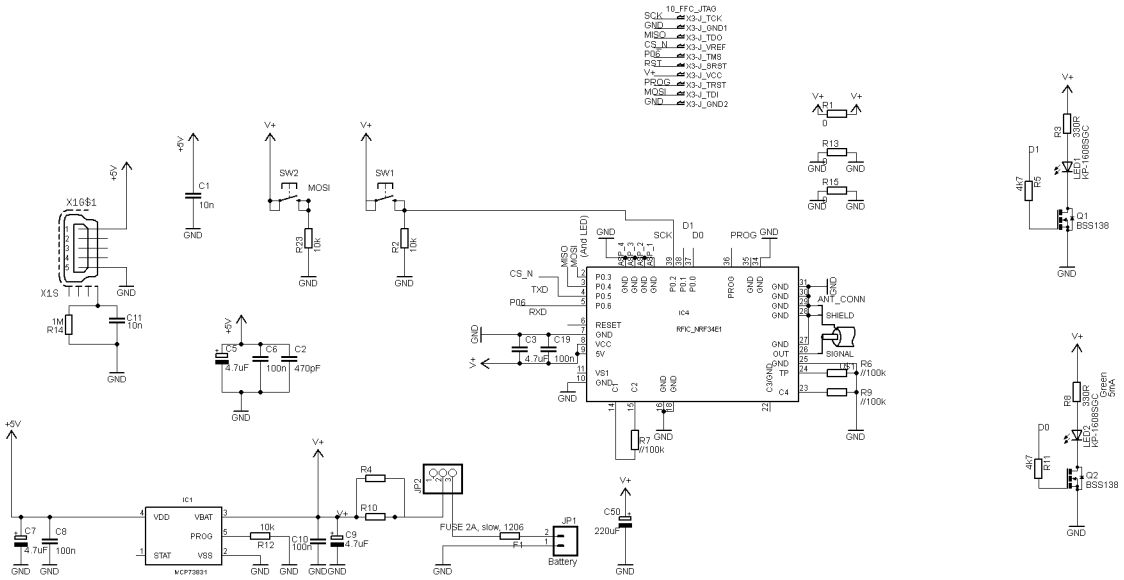
```
/**
 * Handle messages, operating necessary.
 * \return  info byte about past action
 */
uint8_t messageInAndSendHandle(void);
```

PŘÍLOHA2: INFORMACE O KOMPILÁTORU

uVision		V4.24.00.0
Toolchain	DK51	
C Compiler	C51.exe	V9.06
Assembler	A51.exe	V8.02a
Linker/Locator	BL51.exe	V6.22
Librarian	LIB51.exe	V4.29
Hex Converter	OH51.exe	V2.6
CPU DLL	S8051.DLL	V3.79.0.1
Dialog DLL	DP51.DLL	V2.62.0.1
Target DLL	nrfKeil515.DLL	
Dialog DLL	TP51.DLL	V2.58

PŘÍLOHA3: DOKUMENTACE KLÍČE



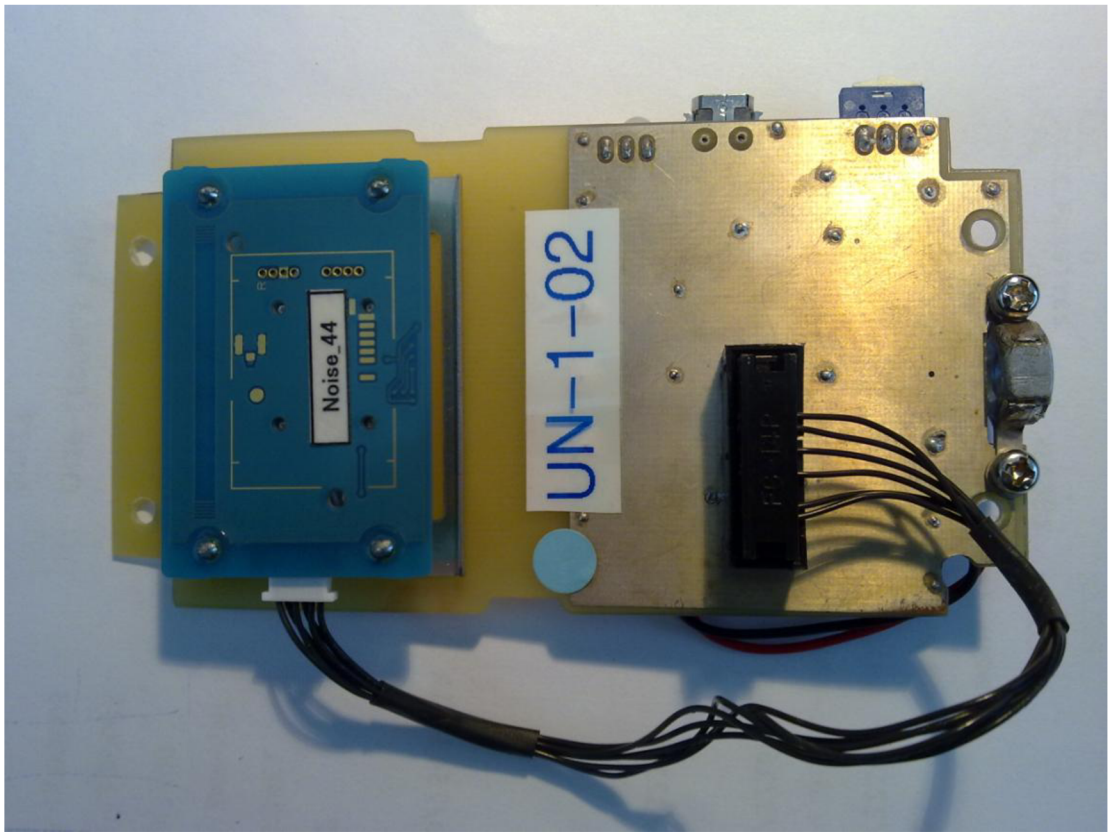


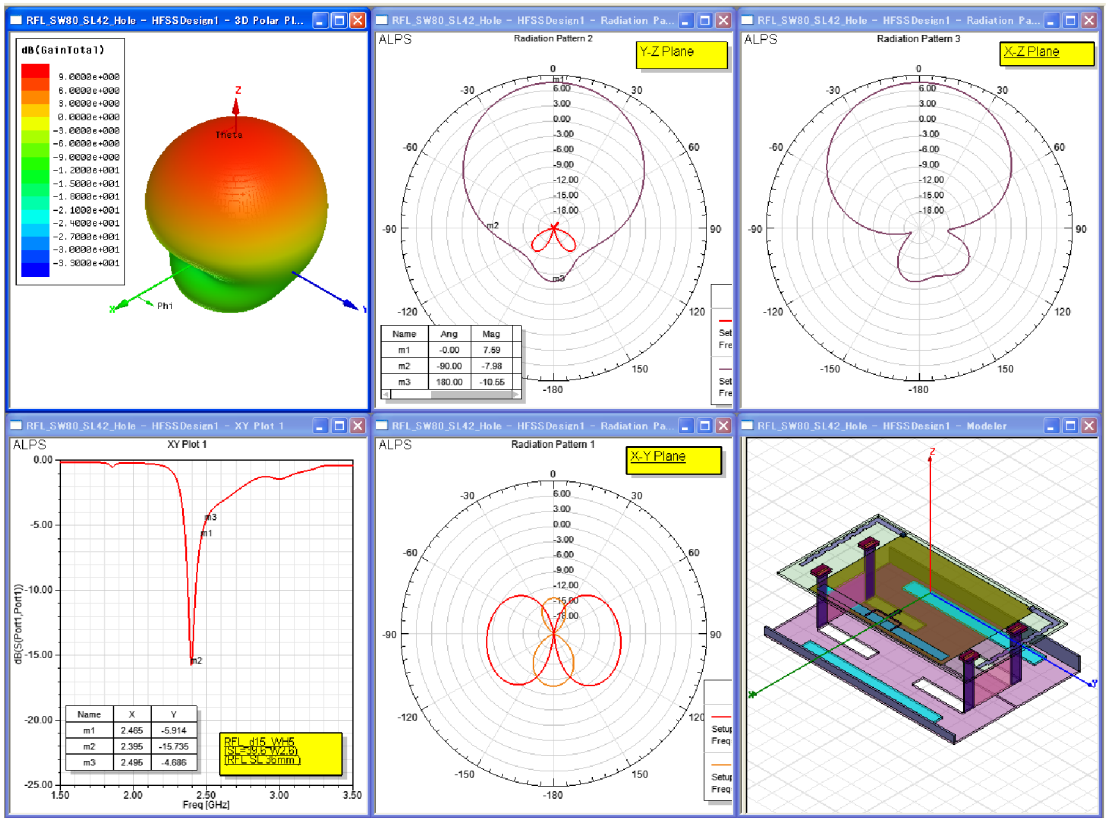
Part	Value	Device	Package	
C1	10n	C0603	C0603K	
C2	470pF	C0402R	C0402R	
C3	4.7uF	C-EUC1206	C1206	
C5	4.7uF	CPOL-EUA/3216R	A/3216R	
C6	100n	C0402R	C0402R	
C7	4.7uF	CPOL-EUA/3216R	A/3216R	
C8	100n	C0402R	C0402R	
C9	4.7uF	CPOL-EUA/3216R	A/3216R	
C10	100n	C0402	C0402K	
C11	10n	C0402	C0402K	
C19	100n	C0402R	C0402R	
C50	220uF	CPOL-EUD/7343-31R	D/7343-31R	
F1	FUSE 2A, slow, 1206	R1206	R1206	
IC1	MCP73831	MCP73832	SOT-23-5	
MCP73832T	BATT CHARGER,	LI-ION/LI-POLY, 5SOT23		
IC4	RFIC_NRF34E1	RFIC_NRF34E1	UMRM1X001B	
JP1	Battery	M02-JST-2MM-SMT	S2B-PH	
JP2		PINHD-1X3/90	1X03/90	
LED1	KP-1608SGC	LEDCHIP-LED0603	CHIP-LED0603	
LED2	KP-1608SGC	LEDCHIP-LED0603	CHIP-LED0603	L
Q1	BSS138	BSS123	SOT23	
Q2	BSS138	BSS123	SOT23	
R1	0	R1206	R1206	
R2	10k	R0603	R0603	
R3	330R	R0603	R0603	
R4		R0805	R0805	
R5	4k7	R0402	R0402	
R6	//100k	R0603	R0603	
R7	//100k	R0603	R0603	
R8	330R	R0603	R0603	
R9	//100k	R0603	R0603	
R10		R0805	R0805	
R11	4k7	R0402	R0402	
R12	10k	R0402R	R0402R	
R13	0	R1206	R1206	
R14	1M	R0402	R0402	
R15	0	R1206	R1206	

R23	10k	R0603	R0603
SW1		B1720SMD	B1720/SMD
SW2		B1720SMD	B1720/SMD
U$1	ANT_CONN	ANT_CONN	ANT_CONN
X1		MINI-USB-SCHIELD-32005-201	32005-201
X3	10_FFC_JTAG	10_FFC_JTAG	10_FFC_KYOCERA FFC

PŘÍLOHA4: FOTO MĚŘÍCÍHO PRACOVIŠTĚ

PŘÍLOHA5: SENZOR





PŘÍLOHA6: FOTO KLÍČE