ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

KATEDRA APLIKOVANÉ ELEKTRONIKY A TELEKOMUNIKACÍ

BAKALÁŘSKÁ PRÁCE

Bezdrátový systém přenosu dat

Jan Rada **2020**

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta elektrotechnická

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jan RADA**
Osobní číslo:	**E17B0092P**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronika a telekomunikace**
Téma práce:	**Bezdrátový systém přenosu dat**
Zadávající katedra:	**Katedra aplikované elektroniky a telekomunikací**

Zásady pro vypracování

Návrh a ověření systému pro bezdrátový přenos dat mezi mikrokontrolérem a osobním počítačem s obvody RFM02/868S2 (vysílač) a RFM01/868S2 (přijímač).

1. Popište možnosti bezdrátového přenosu dat mezi periferním zařízením a osobním počítačem.
2. Proveďte rozbor vlastností obvodů RFM01 a RFM02.
3. Navrhněte systém pro přenos dat mezi mikrokontrolérem a osobním počítačem.
4. Realizujte přenosový systém a ověřte jeho vlastnosti.

Rozsah bakalářské práce:	**30 – 40 stran**
Rozsah grafických prací:	**podle doporučení vedoucího**
Forma zpracování bakalářské práce:	**tištěná/elektronická**

Seznam doporučené literatury:

1. http://www.tme.eu/cz/details/rfm02_868s2/komunikacni-moduly-rf/hope-microelectronics/rfm02-868s2/
2. http://www.tme.eu/cz/details/rfm01_868s2/komunikacni-moduly-rf/hope-microelectronics/rfm01-868s2/

Vedoucí bakalářské práce:	**Prof. Ing. Milan Štork, CSc.** Katedra aplikované elektroniky a telekomunikací
Datum zadání bakalářské práce:	**4. října 2019**
Termín odevzdání bakalářské práce:	**11. června 2020**

Západočeská univerzita v Plzni Fakulta elektrotechnická

L.S.

Prof. Ing. Zdeněk Peroutka, Ph.D.
děkan

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 4. října 2019

Abstrakt

Práce se zabývá problematikou bezdrátového přenosu dat mezi dvěma mikrokontrolery a osobním počítačem, návrhem bezdrátového přenosového systému pro komunikaci mezi dvěma mikrokontrolery a osobním počítačem a jeho realizací. Důraz byl kladen na možnost budoucího použití v lékařství, konkrétně pro přenos dat z mobilního zařízení umístěném na pacientovi, které obsahuje tříosý akcelerometr a měřič EKG. V práci jsou porovnány různé metody bezdrátového přenosu dat z periferií do osobního počítače. Dále jsou blíže popsány vlastnosti obvodů RFM01 a RFM02 a dalších bezdrátových modulů.

Klíčová slova

komunikace mikrokontroleru a počítače, bezdrátový přenos dat, rádiový modul, klíčování frekvenčním posuvem (modulace FSK), klíčování amplitudovým posuvem (modulace ASK), Bluetooth, Wifi, CWUSB (certifikovaná bezdrátová univerzální sériová sběrnice), DASH7

Abstract

The task deals with issue about wireless data transfer between two microcontrollers and personal computer, design of wireless transmission system between two microcontrollers and personal computer and its implementation. Emphasis was placed on possibility to use in medicine in the future, specifically for a data transfer from a device placed on patient. This device should contain from three-axes accelerometer and ECG meter. In thesis are compared different methods of data transfer from peripheries to personal computer. The properties of RFM01 and RFM02 circuits and other modules are described in detail.

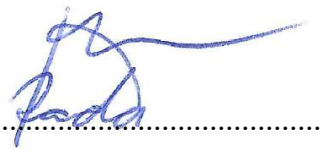
Key words

microcontroller and computer communication, data wireless transfer, radio module, Frequency-shift keying (FSK), Amplitude-shift keying (ASK), Bluetooth, Wi-Fi, Certified Wireless Universal Serial Bus (CWUSB), DASH7

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této bakalářské práce.

Dále prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.



.....

podpis

V Plzni dne 8. 7. 2020 Jan Rada

Obsah

Úvod

Předkládaná práce se zabývá problematikou bezdrátového přenosu dat mezi periferním zařízením a osobním počítačem. V první části práce jsou popsány různé možnosti bezdrátové komunikace – Wifi, Bluetooth, CW USB, DASH7. Dále jsou blíže popsány vlastnosti obvodů RFM01, RFM02, XY-FST a XY-MK-5V. V druhé části se práce zabývá návrhem bezdrátového přenosového systému pro komunikaci mezi dvěma mikrokontrolery a osobním počítačem s obvody RFM01 a RFM02. Tento systém byl navrhován tak, aby bylo možné jej v budoucnu použít pro přenos dat z mobilního zařízení umístěném na pacientovi, obsahující tříosý akcelerometr a měřič EKG. Soustava byla realizována a ověřena různými měřeními, které jsou součástí této práce. Celý systém je funkční a může být použit pro výše uvedené účely.

Seznam symbolů a zkratek

apod.	…	a podobně
tzv.	…	takzvaně
tzn.	…	to znamená
tj.	…	to jest
resp.	…	respektive
atd.	…	a tak dále
např.	…	například
log.	…	logická
PC	…	personal computer, tj. osobní počítač
ISM	…	Industrial Scientific Medicine, tj. nelicencované frekvenční pásmo
FHSS	…	Frequency Hopping Spread Spectrum, tj. klíčování kmitočtovým skákáním nosné vlny
FSK	…	Frequency Shift Keying, tj. klíčování fázovým posuvem
GFSK	…	Gaussian FSK, tj. Gaussovské FSK
GMSK	…	Gaussian Minimum Shift Keying, tj. Gaussovské klíčování minimálním frekvenčním posuvem
TDD	…	Time Division Duplex, tj. duplex s časovým dělením
IEEE	…	Institute of Electrical and Electronics Engineers
LTE	…	Long Term Evolution, tj. mobilní telefonní síť 4. generace
MAN	…	Metropolitan Area Network, tj. metropolitní síť
LAN	…	Local Area Network, tj. lokální síť
WLAN	…	Wireless LAN, tj. bezdrátová lokální síť
DSSS	…	Direct Sequence Spread Spectrum, tj. technika přímého rozprostřeného spektra
OFDM	…	Orthogonal Frequency Division Multiplexing, tj. ortogonální multiplex s frekvenčním dělením
MIMO	…	Multiple-input multiple-output, tj. více vstupů více výstupů
MU-MIMO	…	multi-user MIMO, tj. víceuživatelský MIMO
BSS	…	Base Service Station
IoT	…	Internet of things, tj. internet věcí
TWT	…	Target Wake Time

USB	…	Universal Serial Bus, tj. univerzální sériová sběrnice
CWUSB	…	Certified Wireless USB, tj. certifikované bezdrátové USB
UWB	…	Ultra Wide Band, tj. ultra široké frekvenční pásmo
HWA	…	Wire Host Adapter
DWA	…	Device Wire Adapter
ISO	…	International Organization for Standardization, tj. Mezinárodní organizace pro normalizaci
D7A	…	DASH7 Alliance Protocol
FCC	…	Federal Communications Commission, tj. Federální komunikační komise (USA)
ETSI	…	European Telecommunications Standards Institute, tj. Evropský ústav pro telekomunikační normy
SPI	…	Serial Peripheral Interface, tj. sériové periferní rozhraní
FIFO	…	First in, First out, tj. první dovnitř, první ven
RSSI	…	Received Signal Strength Indication, tj. indikátor síly přijímaného signálu
DQD	…	Data quality detection, tj. detekce kvality přijatých dat
AFC	…	Automatic frequency control, tj. automatické řízení frekvence
ASK	…	Amplitude-shift keying, tj. klíčování amplitudovým posuvem
VCP	…	Virtual COM Port, tj. virtuální komunikační port
COM	…	Communication port, tj. komunikační port
CPU	…	Central Processing Unit, tj. centrální procesorová jednotka
I^2S	…	Inter-IC Sound
USART	…	Universal Synchronous / Asynchronous Receiver and Transmitter, tj. Univerzální synchronní / asynchronní sériové rozhraní
CAN	…	Controller Area Network
RTC	…	Real-time clock, tj. hodiny reálného času
GPIO	…	General-purpose input / output, tj. Univerzální vstupní/výstupní pin
PLL	…	phase-locked loop, tj. fázový závěs
HSI	…	high-speed internal, tj. vnitřní vysokorychlostní hodiny

1 Možnosti bezdrátového přenosu dat mezi periferiemi a osobním počítačem

1.1 Bluetooth

Bluetooth je jeden z nejrozšířenějších způsobů propojení periferie a osobního počítače. Jedná se o open source standart pro bezdrátovou komunikaci s krátkým dosahem a nízkou spotřebou, propojující dvě nebo více elektronických zařízení [1].

1.1.1 Popis technologie Bluetooth

Bluetooth technologie měla za cíl nahradit sériové rozhraní RS-232 [1]. Využívá pásmo 2,4 GHz. To je v tzv. frekvenčním pásmu ISM (Industrial Scientific Medicine) tzv. nelicencované pásmo [2]. Největší výhodou tohoto pásma je, že není třeba žádat o přidělení frekvenčního pásma, ani platit poplatky za užívání s tím spojené [3]. Nevýhody tohoto pásma jsou především silná rušení způsobená jinými technologiemi pracujícími v tomto pásmu, jako například sítě Wifi 802.11 standardů b, g a n nebo mikrovlnné trouby [3, 4].

Aby se zmírnily vlivy tohoto nežádoucího rušení, a zároveň aby Bluetooth nerušilo jiné technologie pracující ve stejném frekvenčním pásmu, používá se pro komunikaci na fyzické vrstvě metoda s kmitočtovým skákáním nosné vlny FHSS (Frequency Hopping Spread Spectrum) [5]. Nosný kmitočet se za jednu sekundu změní celkem 1600krát v rámci pásma 2,4 GHz [5].

Spektrum Bluetooth tvoří celkem 79 rádiových kanálů od 2400 MHz do 2483,5 MHz [5]. Pro modulaci na každou nosnou frekvenci se používá šířka kanálu 1 MHz [5]. Dolní ochranné pásmo má šířku 2 MHz, horní 3,5 MHz [5]. Pro nosné kmitočty těchto kanálů platí vztah (1.1) [5]:

$$f = 2402 + 1000k \text{ [MHz]}, \text{kde } k = 0, \dots, 78 \tag{1.1}$$

Všechny kanály se ale využívají jen zemích Evropy a dalších státech, a to kvůli národní legislativě. Pro ostatní státy se využívá druhá varianta systému. Šířka pásma Bluetooth je zde od 2446,5 do 2483,5 MHz [5]. Šířka kanálu je také 1 MHz a ochranná pásma (horní i dolní) jsou 7,5 MHz. Zde platí pro nosný kmitočet vztah (1.2) [5]:

$$f = 2454 + 1000k \text{ [MHz]}, \text{kde } k = 0, \dots, 22 \quad (1.2)$$

Kódování jednotlivých symbolů se dělá modulací GMSK. GMSK je speciální případ GFSK s modulačním indexem 0,5 [4]. Logické úrovně jedničky nebo nuly se určují podle kladné, resp. záporné kmitočtové odchylky. Tato odchylka bývá v rozmezí 140 až 175 kHz, typicky 157,5 kHz [4].

Pakety přenášených dat se posílají v krátkých časových intervalech tzv. time slotech [5]. Využívají při tom metodu časového duplexu (TDD) [5]. Jak už bylo zmíněno výše, nosný kmitočet se mění 1600krát za sekundu. Z toho vyplývá, že doba vysílání na jedné nosné je 625 μs ($1/1600 = 625 \cdot 10^{-6}$) [4].

1.1.2 Topologie

Bluetooth zařízení se propojují do malých sítí nazývaných piconet [2]. Síť je zapojená tzv. do hvězdy (Star) a pracuje v režimu tzv. ad-hoc [2]. Síť se skládá z jednoho hlavního zařízení (Master) a z jednoho až sedmi aktivních podřízených zařízení (Slave) [2]. V rámci piconetu je fyzický rádiový kanál sdílen touto skupinou zařízení a zařízení jsou synchronizována společnými hodinami a vzorem pro FHSS, přičemž Master zařízení poskytuje synchronizační reference [2]. Například master zařízení může být smartphone a Slave zařízení mohou být chytré hodinky, sluchátka, autorádio apod.

Jedno zařízení může být Master pouze v jedné síti, ale není vyloučeno, aby bylo Slave v nějaké další [4]. Také je možné aby jedno zařízení bylo Slave ve více sítí zároveň [4]. Tímto způsobem se mohou sítě zvětšovat a řetězit. Takovým sítím se říká scatternet [4].

1.1.3 Struktura paketu

První část paketu je 72 bitů dlouhý přístupový kód, který je dán kombinací MAC adresy zařízení a hodinového taktu mater zařízení [6]. Dále je zde 54bitové záhlaví. Přístupový kód je unikátní pro každou síť piconet a kromě synchronizace slouží i k autorizovanému přístupu do sítě [6]. Celková délka paketu tak může být od 126 do 2871 bitů.

Obr. 1.1 Znázornění paketu v síti Bluetooth [7]

1.1.4 Výkonové třídy

Další možností, jak snížit rušení v pásmu 2,4 GHz, je vhodná volba výkonnostní třídy zařízení. Tyto výkonové třídy jsou celkem tři a jsou uvedeny v *Tab. 1.1*. Výkonové úrovně jsou vztaženy k anténnímu konektoru Bluetooth zařízení.

Výkonové třídy	**Maximální výstupní výkon**	**Nominální výstupní výkon**	**Minimální výstupní výkon**
1	100 mW (20 dBm)	-	1 mW (0 dBm)
2	2,5 mW (4 dBm)	1 mW (0 dBm)	0,25 mW (-6 dBm)
3	1 mW (0 dBm)	-	-

Tab. 1.1 Výkonové třídy systému Bluetooth

1.1.5 Vývoj a verze Bluetooth

Vývojem technologie Bluetooth se zabývá soukromý podnik *Bluetooth Special Interest Group* [8]. Vznikl v roce 1998 ve spolupráci firem *Ericsson*, *IBM*, *Intel*, *Toshiba* a *Nokia* [1, 8]. V tabulce *Tab. 1.2* jsou přehledně rozepsány jednotlivé verze Bluetooth a jejich základní vlastnosti. Bližší popis jednotlivých verzí viz níže.

Verze	**Rok vydání**	**Rychlost**	**Dosah**
Bluetooth 1.0	1999	721 kb/s	?
Bluetooth 2.0	2007	2,2 Mb/s	10 m
Bluetooth 3.0	2009	24 Mb/s	10 m
Bluetooth 4.0	2010	1 Mb/s	50 m (10 m)
Bluetooth 5.0	2016	125 kb/s, 500 kb/s, 1Mb/s, 2Mb/s	240 m (40 m)

Tab. 1.2 Přehled verzí Bluetooth [8]

1.1.5.1 Bluetooth 1.0

První verze měly spousty problémů především kvůli nekompatibilitě různých zařízení [1, 8]. U mobilních telefonů se Bluetooth vyskytovalo pouze u dražších zařízení střední a vyšší třídy [8].

1.1.5.2 Bluetooth 2.0

U druhé verze se zbezpečnilo a zjednodušilo párování zařízení, snížila se spotřeba energie [1, 8]. Bluetooth zařízení verze 2.0 byly ovšem použitelné u přenosu menších souborů, kvůli častým výpadkům spojení [8].

1.1.5.3 Bluetooth 3.0

U verze 3.0 bylo zmenšené riziko přerušení spojení, a tak už byla spolehlivě použitelná pro přenos objemnějších dat jako např. videí nebo hudby [8]. Byla také vybudována spolupráce se sítěmi Wifi, kdy se Bluetooth používalo pro navázání spojení mezi zařízeními a vysokorychlostní přenos dat se prováděl souběžně právě přes standart 802.11 [1, 8]. Teoretická hodnota rychlosti přenosu 24 Mb/s byla dosažitelná pouze právě v kombinaci se standardem Wifi, jinak byla přenosová rychlost značně nižší [1, 8].

1.1.5.4 Bluetooth 4.0

Cílem této verze nebylo nehradit předchozí verzi, ale vzájemně společně spolupracovat [1, 8]. Hlavním úkolem bylo snížit energetickou náročnost pro nenáročné datové přenosy (např. chytrých hodinek, handsfree apod.), čímž se mohla značně prodloužit výdrž těchto zařízení na jedno nabití [1, 8]. Naopak pro potřebu rychlého přenosu objemných dat se využívá předchozí verze Bluetooth 3.0 [1]. Dále byla v této verzi prodloužena přenosová vzdálenost ve volném prostranství až na 50 metrů a byl vyřešen problém s rušením LTE sítě [8].

1.1.5.5 Bluetooth 5.0

V této verzi byla znovu prodloužen dosah, a to až čtyřikrát. Ve volném prostranství se dosahuje vzdálenosti až 240 metrů a uvnitř budov až 40 metrů [1, 8]. Pro větší přenosové vzdálenosti se použije nižší přenosová rychlost a naopak. Bohužel se ale ztrácí zpětná kompatibilita s verzemi nižšími než 4.0 [8].

Ve verzi Bluetooth 5.1 z roku 2019 se zpřesnila navigace zařízení až na centimetry, a to i v uzavřených prostorech a byla opět snížena energetická náročnost [8].

Ve verzi Bluetooth 5.2 z roku 2020 byla ještě více snížena energetická náročnost a byla zvýšena bezpečnost, na kterou byl v této verzi kladen důraz [8].

1.2 Wifi

Wifi je nejčastěji používaná bezdrátová komunikační technologie, kterou v současnosti celosvětově využívá více než 13 miliard zařízení [9]. Jde o skupinu standardů založenou na IEEE 802.11 normě [10].

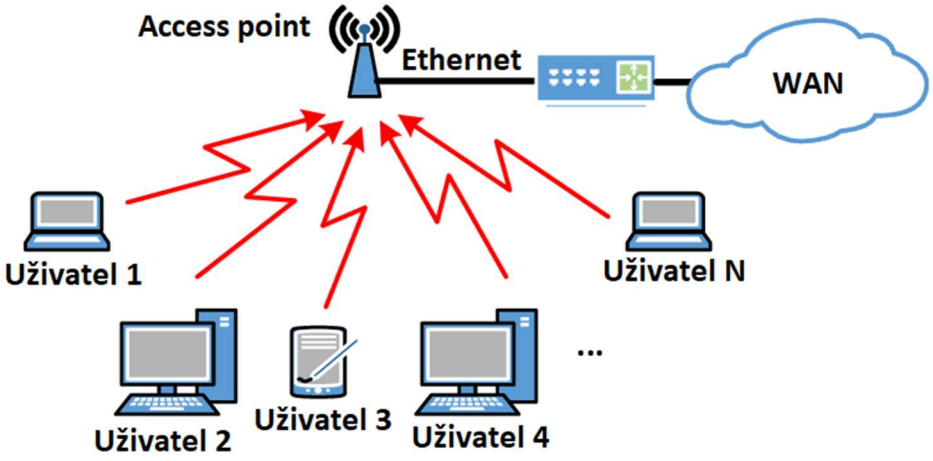
1.2.1 Popis Wifi technologie

Stejně jako technologie Bluetooth využívá i technologie Wifi tzv. nelicencované pásmo ISM. Výhody a nevýhody využití tohoto pásma jsou popsány v kapitole 1.1.1. Technologie Wifi je spravována neziskovou organizací *Wi-Fi Alliance*, která se skládá ze stovek světových společností [9]. Wifi měla za cíl nahradit klasickou kroucenou dvoulinku využívanou na propojování počítačových sítí LAN a MAN ve standardu Ethernet a propojit jednotlivé stanice bezdrátově [11]. Proto je také někdy označována jako WLAN (Wireless LAN) [11].

1.2.2 Topologie Wifi

Stejně jako v technologii Bluetooth spolu mohou zařízení komunikovat přímo (ad-hoc). Takto se dá například propojit tiskárna s osobním počítačem, aniž bychom museli mít doma vybudovanou Wifi síť. Takový způsob se ale používá jen zřídka.

Daleko častěji se Wifi používá v režimu infrastruktury, tj. připojení k síti přes přístupový bod tzv. Access point [12]. Některá zařízení dokonce umí pracovat jen v tomto režimu [12]. Síť je zapojená tzv. do hvězdy (Star) [12]. Access point může být pak připojen k routeru (v domácích sítích bývá AP přímo součástí routeru) a dále do sítě internet (WAN). Takovéto zapojení je znázorněno na *Obr. 1.2*.



Obr. 1.2 Znázornění Wifi sítě s Access pointem

1.2.3 Verze Wifi

Wifi technologie nepracuje pouze v pásmu 2,4 GHz jako technologie Bluetooth, kde samozřejmě může docházek ke vzájemnému rušení těchto dvou a dalších technologií, ale i v dalších pásmech, např. 3,7 GHz, 5 GHz nebo nejnověji i 60 GHz (standard *IEEE 802.11ad* s názvem *WiGig*) [13, 14]. Standard *802.11* je soubor celé řady různých specifikací verzí Wifi, které se liší nejen frekvenčním pásmem, ale i použitou modulací a samozřejmě i vývojem této technologie, a kvůli odlišnostem národních legislativ je někdy nutné upravit i využití jednotlivých kanálů nebo vyzařovaný výkon. [13, 14]. Následuje popis některých variant Wifi. Parametry některých verzí jsou shrnuty v *Tab. 1.3*.

1.2.3.1 IEEE 802.11

Byl vytvořen v roce 1997 jako vůbec první WLAN standard [14]. Pro přenos na fyzické vrstvě se používala modulace DSSS (Direct Sequence Spread Spectrum, tj. technika přímého rozprostřeného spektra) a FHSS v pásmu 2,4 GHz [13, 14]. Kvůli své nízké datové propustnosti maximálně 2 Mb/s se dlouho nepoužíval a v současné době se nevyrábí žádná zařízení s tímto standardem [14]. Položil ale základ, ze kterého se vyvíjely všechny další specifikace Wifi [14].

1.2.3.2 IEEE 802.11b

IEEE 802.11b byl vydán v roce 1999 a je založen na původním standardu *IEEE 802.11* [14]. Pracuje ve stejném pásmu 2,4 GHz a jeho přenosová rychlost se zvedla na teoretických 11 Mb/s [14]. Využívá modulaci DSSS [13]. Ve volném prostranství má dosah až 140 m [11].

1.2.3.3 IEEE 802.11a

Byl vydán ve stejné době jako *IEEE 802.11b* (v roce 1999) [14]. Na rozdíl od tohoto standartu pracuje v kmitočtovém pásmu 5 GHz a je modulován OFDM (Orthogonal Frequency Division Multiplexing, tj. ortogonální multiplex s frekvenčním dělením) [13]. Kvůli vyšší frekvenci než *802.11b* se signál šíří hůře uvnitř budov, a tím má horší dosah (120 metrů), naopak má vyšší datovou propustnost – až 54 Mb/s [11, 14].

1.2.3.4 IEEE 802.11g

Tato verze v roce 2003 zkombinovala výhody standardů *802.11a* a *802.11b*. Využívá stejnou modulaci signálu jako *802.11a* a dosahuje stejné přenosové rychlosti 54 Mb/s, ale vysílá stejně jako *802.11b* na kmitočtu 2,4 GHz, což zlepšuje šíření signálu v budovách a zvětšuje dosah signálu [14]. Této specifikaci se také říká *Wi-Fi 3* [14].

1.2.3.5 IEEE 802.11n

Klíčovou vlastností dodatku *802.11n* je použití technologie MIMO (Multiple-input multiple-output) [14]. MIMO nevyužívá k přenosu dat pouze jednu anténu, jako předchozí verze, ale pro větší propustnost dat hned několik vysílacích a několik přijímacích antén [14]. Dále byla rozšířena šířka kanálu na 40 MHz [15]. Teoretická přenosová rychlost na fyzické vrstvě dosahuje 300 Mb/s, reálná rychlost se pohybuje okolo 130 Mb/s [15]. Ve standardech *802.11a/b/g* bylo vícecestné šíření vln bráno jako nežádoucí interference a snižovalo propustnost kanálu, kdežto u standartu *802.11n* se právě vícecestného šíření využívá ke zvýšení schopnosti přijímače obnovení původní informace v přeneseném signálu [15]. Využívá buď pásmo 2,4 GHz, kde může opět docházet k rušení od technologie Bluetooth, mikrovlnných trub, či bezdrátových telefonů, nebo 5 GHz [15]. Tato verze se také označuje jako *Wi-Fi 4* [14]. Je zpětně kompatibilní s předchozími verzemi *802.11b/g*, ovšem za snížené přenosové rychlosti [14].

1.2.3.6 IEEE 802.11ac

Tato verze se také označuje jako *Wi-Fi 5* [9]. *802.11ac* pracuje jak na frekvenci 2,4 GHz, tak zároveň i na 5 GHz, kdežto *802.11n* mohl pracovat jen na jedné z nich [9]. Na 5 GHz kmitočtu probíhá komunikace pro aplikace náročné na datový tok a na 2,4 GHz probíhá základní a rychlostně nenáročná komunikace [9]. Zvětšila se šířka kanálu na 80 MHz [9].

Využívá se zde technologie multi-user MIMO (MU-MIMO), kdy může jedno zařízení komunikovat s více zařízení současně, což u MIMO u *Wi-Fi 4* nebylo možné [9]. Díky tomu se také zvedl datový tok. Maximální teoretická přenosová rychlost na kmitočtu 5 GHz dosahuje 1,3 Gb/s, na kmitočtu 2,4 GHz 450Mb/s, reálná maximální rychlost je 720 resp. 240 Mb/s [16].

1.2.3.7 IEEE 802.11ax

Nejnovější standard označovaný jako *Wi-Fi 6* byl vydán v roce 2019 a s prvními výrobky na světovém trhu se počítá na začátku roku 2021 [9]. Stejně jako předchozí verze využívá technologii MU-MIMO [9]. Technologie pracuje nově i na 6 GHz, což umožnila změna legislativy v USA (tento kmitočet je tedy možné zatím využít pouze zde) [9]. Opět se zvětšila šířka kanálu, a to na 160 MHz [17]. Technologií Base Service Station Color (BSS Color) se minimalizují kolize resp. rušení více různých sítí *Wi-Fi 6* v jedné lokalitě, a tím se zvyšuje datová propustnost [17]. TWT (Target Wake Time) technologií se řídí, jak často mají zařízení probudit pro komunikaci, prodlužuje dobu spánku zařízení a podstatně zlepšují výdrž baterie mobilních zařízení a IoT zařízení [17]. Tato všechna vylepšení Wifi technologie zvýšila teoretickou maximální přenosovou rychlost až na 11 Gb/s [18].

IEEE standard	Označení	Rok vydání	Maximální rychlost [Mb/s]	Pásmo [GHz]	Fyzická vrstva
802.11	-	1997	2	2,4	DSSS a FHSS
802.11b	-	1999	11	2,4	DSSS
802.11a	-	1999	54	5	OFDM
802.11g	Wi-Fi 3	2003	54	2,4	OFDM
802.11n	Wi-Fi 4	2009	600	2,4 / 5	MIMO OFDM
802.11ac	Wi-Fi 5	2013	1750	2,4 a 5	MU-MIMO OFDM
802.11ax	Wi-Fi 6	2019	11000	2,4 a 5 / 6	MU-MIMO OFDM

Tab. 1.3 Přehled Wifi verzí [9, 13, 14, 17, 18]

1.3 Bezdrátové USB

Certified Wireless Universal Serial Bus (CWUSB), neboli certifikované bezdrátové USB, má za cíl, jak je z názvu zřejmé, nahradit klasické drátové rozhraní USB. První verze této technologie byla oficiálně vydána v květnu roku 2005 společností *Wireless USB Promoter Group* [19]. CWUSB je určeno pro připojení periferií stejně jako USB, např. tiskárny,

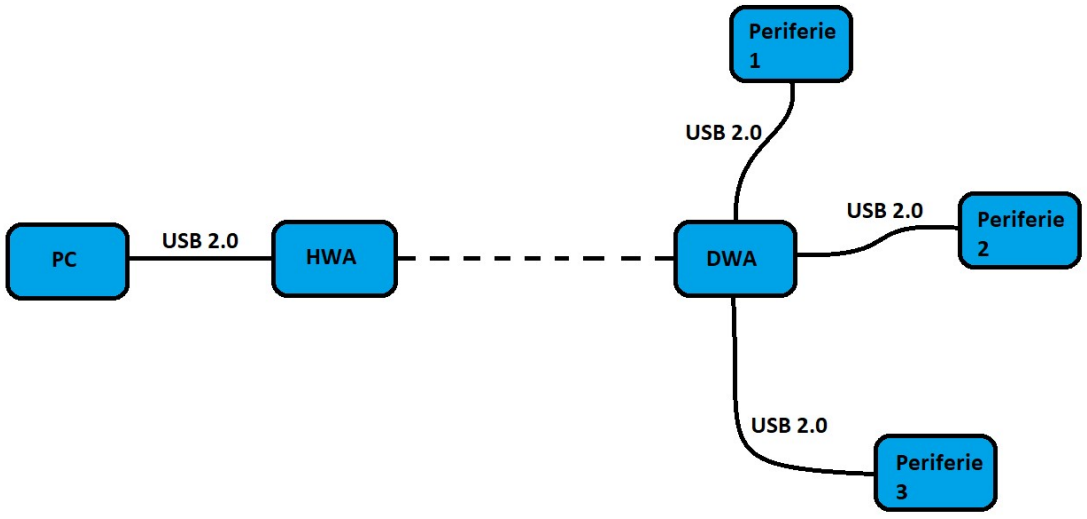
klávesnice, myši, externí grafické karty, datového uložiště apod. do vzdálenosti 10 metrů [19]. Není určeno pro vytváření počítačových sítí, i když je to teoreticky možné [20].

Na rozdíl od Wifi nebo Bluetooth využívá CWUSB tzv. Ultra-wideband (UWB) [19]. Tato metoda bezdrátového přenosu dat na krátkou vzdálenost s velkou šířkou pásma a nízkým příkonem umožňuje bezpečné a vysokorychlostní připojení potřebné pro uživatelské prostředí podobné USB [19]. Na fyzické vrstvě je použito klíčování OFDM [19]. Komunikace používá kmitočty od 3,1 GHz do 10,6 GHz [21]. Vlastnosti přenosu jsou shodné jako u klasického USB 2.0, teoretická maximální přenosová rychlost dosahuje 480 Mb/s, pokud jsou zařízení vzdáleny do 3 metrů, nebo 110 Mb/s při vzdálenosti do 10 metrů [21].

Pomocí CWUSB je možné propojit až 127 zařízení stejně jako USB, s tím rozdílem, že CWUSB samozřejmě nepotřebuje žádné rozbočovací huby [20]. Hlavní zařízení (Master), většinou PC, řídí celou komunikaci, ostatní klientská zařízení pouze odpovídají na požadavky [20].

Pro připojení periferií, které mají pouze USB, nikoliv CWUSB, slouží Wire Host Adapter (HWA) a Device Wire Adapter (DWA) [20]. HWA slouží jako bezdrátový modul pro Master zařízení, DWA slouží jako hub, do kterého se mohou připojit klientská zařízení [20]. Tyto moduly se připojují pomocí klasického USB. Tento způsob je znázorněn na *Obr. 1.3*.

Co se týče protokolů komunikace a ovladačů zařízení se od klasického USB nijak neliší, CWUSB řeší pouze fyzickou vrstvu [20].



Obr. 1.3 Připojení periferií pomocí HWA a DWA k PC

1.4 DASH7

Technologie DASH7 se zcela liší od výše zmíněných způsobů bezdrátového přenosu dat mezi zařízeními. DASH7 Alliance Protocol (D7A) je otevřený standart pro obousměrnou sub-gigahertzovou bezdrátovou komunikaci v ISM pásmu se středním dosahem [22]. Byl vydán v květnu roku 2015 [23]. Podporu existence a rozvoj dalších specifikací D7A zajišťuje nezisková vzájemně prospěšná organizace *The DASH7 Alliance* [22]. Záměrem této aliance je vylepšit technologii nad rámec svých současných schopností a fyzických hranic a umožnit tak zabezpečení, automatizaci a řídicí systémy pro mnoho prostředí [22]. DASH7 protokol je založen na normě *ISO 18000-7* [22].

D7A definuje komunikační mechanismus mezi koncovými uzly, senzory, alarmy, bránami, aktuátory apod. [22, 23]. Typické použití je takové, že senzory komunikují s uzly, a ty pak předávají data do brány pro přístup na internet [23]. Koncové uzly spolu mohou také komunikovat napřímo [23].

Komunikace probíhá na frekvencích 433, 868 nebo 915 MHz [23]. Šířka kanálu je 25 nebo 200 kHz [23]. Přenosová rychlost je 9,6 kb/s, 55,55 kb/s nebo 166,667 kb/s [23]. Maximální velikost paketu 256 bajtů [23]. Dosah ve volném prostředí bez překážek dosahuje až 5 km [23]. D7A vyplňuje mezeru mezi sítěmi s krátkým a dlouhým rozsahem [22]. D7A vyniká při použití v zastavěných městských částech a průmyslových oblastí s dosahem až 500 m [22]. Vyznačuje se malou latencí propojení pohybujících se zařízení a komunikací s velmi malou spotřebou. Senzory budou bezpečně vysílat hlášení událostí a akční členy mohou přijímat řídicí příkazy s typickou latencí 1 sekundy, přičemž v průměru odebírají pouze 30 μA [22].

2 Vlastnosti bezdrátových modulů

2.1 Obvod RFM02

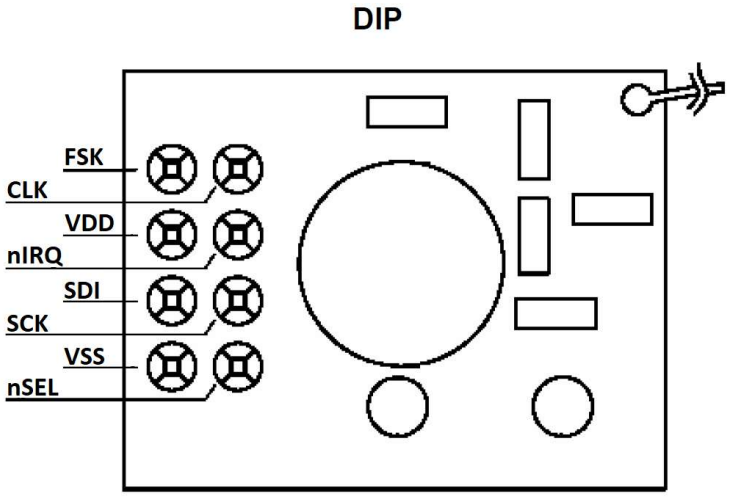
Obvod RFM02 je levný, ale zároveň vysoce funkční, vysoce integrovaný vysílací modul komunikující v nelicencovaném ISM pásmu [24]. Tento vysílací modul RFM02 pracuje s přijímacím modulem RFM01 [24]. Pracuje s FSK modulovaným signálem na frekvencích 433, 868 nebo 915 MHz v souladu s vyhláškami FCC i ETSI [24]. Pomocí rozhraní SPI lze komunikovat s mikrokontrolerem pro nastavení parametrů obvodu [24]. Ve volném prostranství může být dosah až 300 metrů na frekvenci 433 MHz, na frekvencích 868 a 915 MHz je dosah do 200 metrů [24]. Aby se minimalizovaly náklady, může čip poskytnout mikropočítači hodinový signál, čímž se vyhne nutnosti použití dvou krystalů [24]. Dvojice obvodů RFM02 a RFM01 mohou být použity pro dálkové ovládání hraček, bezdrátové alarmy, bezdrátové senzory, domácí automatizace nebo bezdrátové klávesnice a myši [24].

Další funkce RFM02 [24, 25]:

- stabilní a přesná modulace FSK s programovatelnou odchylkou
- fázový závěs s vysokým rozlišením
- napájecí napětí 2,2 až 5,4 V
- datový přenos až 115,2 kb/s
- programovatelný výstupní výkon až 8 dBm (6.3 mW)
- SPI rozhraní
- hodinový výstup pro mikrokontroler
- obvod s automatickým laděním antény
- programovatelná kapacita pro krystalový oscilátor
- detekce vybité baterie
- časovač probuzení
- nízká spotřeba
- nízký proud během stand-by režimu (0,3 μA)

Jednotlivé specifikace modulů se značí: typ modulu – frekvence – typ pouzdra, např. RFM02-433-D znamená vysílací modul RFM02 operující na frekvenci 433 MHz v pouzdře DIP [24].

2.1.1 Popis vývodů RFM02



Obr. 2.1 Vývody modulu RFM02-433D [24]

Název	Typ	Funkce
FSK	vstup	vstup FSK dat
CLK	výstup	hodinový výstup pro externí mikrokontroler (1 MHz – 10 MHz)
VDD	zdroj	kladný pól napájení
nIRQ	výstup	interrupts request – příznak přerušení (aktivní v log. 0)
SDI	vstup	vstup SPI dat
SCK	vstup	hodiny SPI komunikace
VSS	zdroj	záporný pól napájení
nSEL	vstup	chip select – výběr obvodu (aktivní v log. 0)

Tab. 2.1 Popis vývodů modulu RFM02-433D [24]

2.2 Obvod RFM01

Obvod RFM01 je levný, ale zároveň vysoce funkční, vysoce integrovaný přijímací protějšek pro obvod RFM02 [26]. Pracuje s FSK modulovaným signálem na frekvencích 315, 433, 868 nebo 915 MHz v souladu s vyhláškami FCC i ETSI [26]. Pomocí rozhraní SPI lze komunikovat s mikrokontrolerem pro nastavení parametrů obvodu [26]. Společně s obvodem RFM02 může být použit pro dálkové ovládání hraček, bezdrátové alarmy, bezdrátové senzory, domácí automatizace nebo bezdrátové klávesnice a myši [26].

Modul dramaticky snižuje zatížení mikrokontroleru pomocí integrovaného digitálního zpracování dat: filtrování dat, obnovování hodin, rozpoznávání datových vzorů a integrovaná paměť FIFO [26]. Aby se minimalizovaly náklady celého zařízení, může čip poskytnout

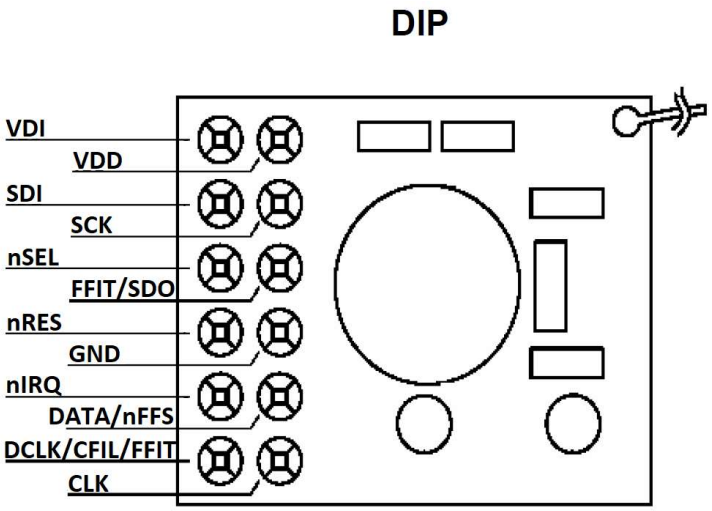
mikrokontroleru hodinový signál, čímž se vyhne nutnosti použití dvou krystalů [26]. Pro jednoduché aplikace může být přijímač RFM01 použit v samostatném provozním režimu a může tak fungovat bez mikrokontroleru [26]. Takto lze použít 12 nebo více předdefinovaných frekvenčních kanálů v kterémkoliv ze čtyř pásem [26].

Další funkce RFM01 [26, 27]:

- fázový závěs s vysokým rozlišením
- napájecí napětí 2,2 až 5,4 V
- vysoká přenosová rychlost (až 115,2 kb/s v digitálním režimu a 256 kb/s v analogovém režimu)
- citlivost -109 dBm
- programovatelná šířka základního pásma (67 až 400 kHz)
- analogové a digitální výstupy RSSI
- SPI rozhraní pro nastavení parametrů obvodu
- výstupní 16bitová FIFO
- detekce kvality přijatých dat (DQD)
- automatické řízení frekvence (AFC)
- hodinový výstup pro mikrokontroler
- programovatelná kapacita pro krystalový oscilátor
- detekce vybité baterie
- časovač probuzení
- nízká spotřeba (přibližně 9 mA v nízkých pásmech)
- provozní cyklus s nízkým příkonem (průměrný napájecí proud menší než 0,5 mA)
- nízký proud během stand-by režimu (0,3 μA)

Jednotlivé specifikace modulů se značí: typ modulu – frekvence – typ pouzdra, např. RFM01-433-D znamená přijímací modul RFM01 operující na frekvenci 433 MHz v pouzdře DIP [26].

2.2.1 Popis vývodů RFM01

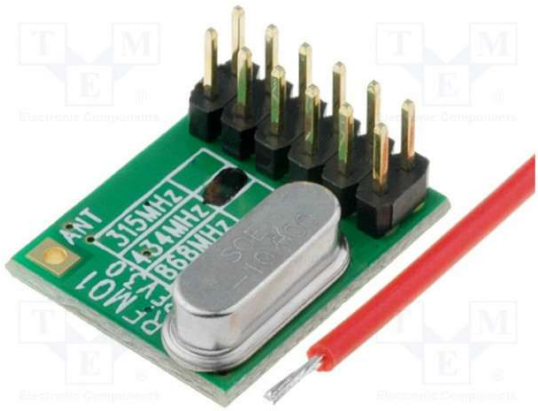


Obr. 2.2 Vývody modulu RFM01-433D [26]

Název	Typ	Funkce
VDI	výstup	valid data indicator – indikátor platných dat
VDD	zdroj	kladný pól napájení
SDI	vstup	vstup SPI dat
SCK	vstup	hodiny SPI komunikace
nSEL	vstup	chip select – výběr obvodu (aktivní v log. 0)
FFIT	výstup	přerušení naplnění FIFO (aktivní v log. 0)
SDO	výstup	čtení dat
nRES	výstup	reset (aktivní v log. 0)
GND	zdroj	zem
nIRQ	výstup	interrupts request – příznak přerušení (aktivní v log. 0)
DATA	výstup	data (režim bez FIFO)
nFFS	vstup	FIFO select – volba FIFO (aktivní v log. 0)
DCLK	výstup	hodinový výstup (režim bez FIFO)
CFIL	analog	externí filtrační kondenzátor (analogový režim)
FFIT	výstup	FIFO přerušení (aktivní v log. 1)
CLK	výstup	hodinový výstup pro externí mikrokontroler (1 MHz – 10 MHz)

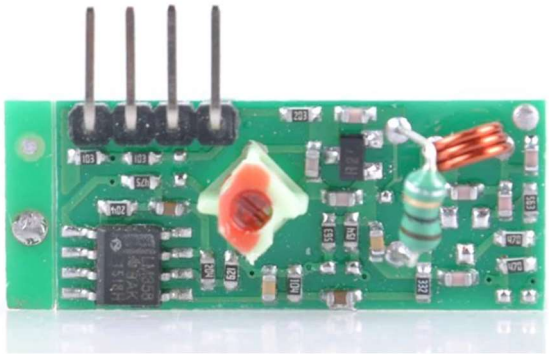
Obr. 2.3 Popis vývodů modulu RFM01-433D [26]

Obr. 2.4 Obvod RFM02-433-D [25]



Obr. 2.5 Obvod RFM01-433-D [27]

Obr. 2.6 Obvod XY-FST [30]



Obr. 2.7 Obvod XY-MK-5V [30]

2.3 Dvojice modulů XY-FST a XY-MK-5V

Sada vysílacího modulu XY-FST a přijímacího modulu XY-MK-5V je jedna z vůbec nejlevnějších variant bezdrátového přenosu dat [28]. Je určena pro použití v nejjednodušších aplikacích např. bezdrátový zvonek, bezdrátový teploměr apod. Každý modul má pouze tři piny: napájení, zem a data [29]. Pracuje v režimu modulace ASK [28]. Dvojice modulů komunikuje na frekvenci 315 nebo 434 MHz [29]. Dosah je 20 až 200 metrů [29]. Maximální přenosová rychlost dosahuje 9,6 kb/s [28, 29].

Další funkce XY-FST [29]:

- napájení 3 až 12 V
- vysílací výkon 14 dBm (25 mW)
- šířka pásma 2 MHz
- spotřeba 9 až 40 mA

Další funkce XY-MK-5V [29]:

- napájení 5 V
- spotřeba do 6 mA
- citlivost -100 dBm

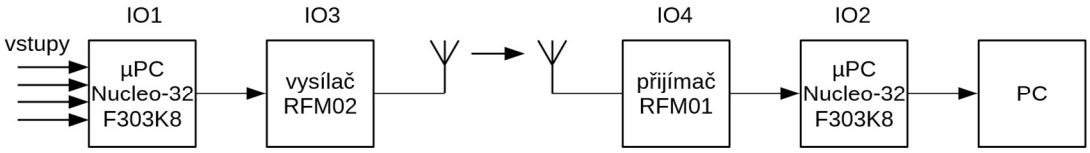
3 Návrh, realizace a ověření funkce bezdrátového systému pro přenos dat mezi mikrokontrolerem a osobním počítačem

3.1 Cíle realizace bezdrátového systému

Sestava bezdrátového přenosového systému byla navrhována tak, aby bylo možné její použití v lékařství. Zaměření bylo na možnost bezdrátového přenosu dat z mobilního zařízení, umístěném na pacientovi. Toto zařízení by mělo obsahovat tříosý akcelerometr a měřič EKG. Akcelerometr proto, aby mohl lékař vzdáleně ze svého PC sledovat, zda se pacient např. prochází, leží, či upadl apod. Měřič EKG pro sledování činnosti pacientova srdce. Celkem je tedy potřeba přenášet 4 soubory informací: EKG a tři osy akcelerometru. Akcelerometr ani měřič EKG není součástí této realizace, ani neproběhla žádná měření s těmito komponenty.

Pro vzorkování elektrokardiogramu pro analýzu ve frekvenční oblasti je možné použít vzorkovací frekvence 500 Hz i 250 Hz s výbornými výsledky [31]. Vzorkování s frekvencí 100 Hz už není přijatelné pro analýzu ve frekvenční oblasti, ale stále je použitelné pro analýzu v časové oblasti a pro Poincaréovy grafy [31]. Vzorkovací kmitočet 50 Hz už není přijatelný [31]. Požadavek na periodu měření akcelerometrem určitě není náročnější než na EKG.

3.2 Popis bezdrátové soustavy

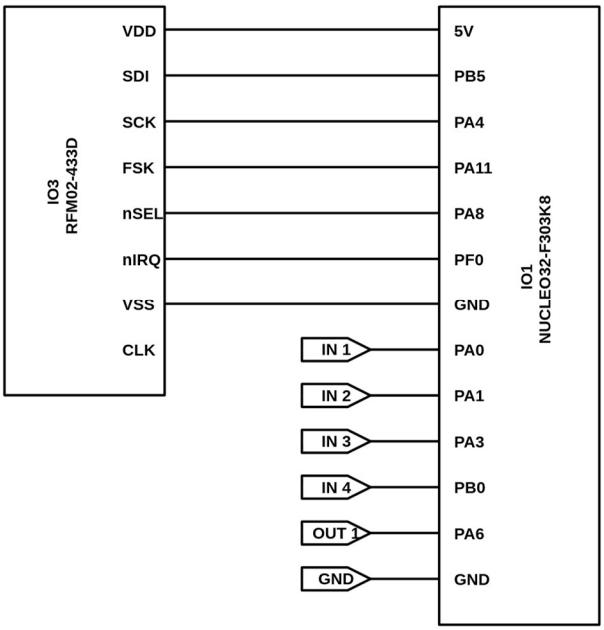


Obr. 3.1 Blokové schéma navrhovaného bezdrátového systému přenosu dat

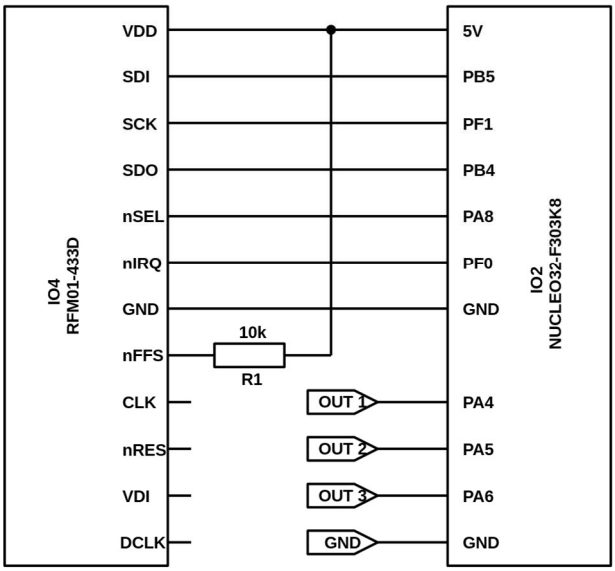
Navrhovaný bezdrátový systém přenosu dat je znázorněn blokovým schématem na *Obr. 3.1*. Pro ověření funkce systému a simulaci EKG a akcelerometru jsou na 4 vstupy přiváděny analogové signály. Tyto signály jsou převedeny A/D převodníkem na číslicové hodnoty a dále zpracovány v mikropočítači *Nucleo32-F303K8*. Tato deska s mikrokontrolerem se stará jak o zpracovávání vstupních signálů, tak i o komunikaci s bezdrátovým modulem *RFM02-433D*.

Mikrokontroler posílá data do vysílače, který je vysílá v ISM pásmu 433 MHz. Až potud byla popsána vysílací část systému (v budoucnu umístěná na pacientovi).

V přijímací části u lékařského PC pak řetězec pokračuje přijímacím modulem *RFM01-433D*. Tento modul je ovládán druhým mikropočítačem stejného typu *Nucleo32-F303K8*, jako ten první. Mikrokontroler čte přijatá data z přijímacího bezdrátového modulu, zpracovává je a odesílá je po USART rozhraní a dále přes virtuální COM port (VCP) po USB rozhraní do PC. Po připojení k VCP se přijatá data v PC mohou nechat zobrazit v terminálu, případně se mohou vykreslit do grafů pomocí dalších systémových aplikací.



Obr. 3.2 Schéma zapojení vysílací části



Obr. 3.3 Schéma zapojení přijímací části

Kompletní schéma zapojení viz *Příloha A*, resp. *Příloha B*.

3.2.1 Použití Nucleo32-F303K8

Tato vývojová deska *Nucleo32-F303K8* od společnosti *STMicroelectronics* byla vybrána především pro svoji jednoduchost ovládání, komunikace a připojení, svoje malé rozměry, postačující výkon a relativně nízkou cenu. Srdcem desky je mikrokontroler *STM32F303K8T6* v pouzdře *LQFP32* [32].

Hlavními parametry tohoto jednočipového počítače [32]:

- ARM®32-bit Cortex®-M4 CPU s FPU
- maximální taktovací frekvence 72 MHz
- 64 kB Flash paměti
- 12 kB SRAM paměti
- 12bitový A/D převodník s 9 kanály
- 12bitový D/A převodník se 3 kanály
- SPI a I^2S rozhraní
- USART rozhraní
- CAN rozhraní
- 25 GPIO portů s možností externího přerušení
- RTC krystal
- Časovače

Desku je možné připojit k PC pomocí USB-mikro B konektoru [32]. Zařízení se pak pro počítač tváří jako 3 zařízení: VCP, velkokapacitní uložiště a debug port [32]. Součástí desky je i druhý mikrokontroler, který řídí komunikaci po USB, debug, nahrávání programu apod [32].

Obr. 3.4 Nucleo32-F303K8 [32]

3.2.2 Použití RFM02 a RFM01

Jak již bylo zmíněno výše, systém je určen pro použití v lékařství. Dá se tedy očekávat, že se zařízení bude provozovat uvnitř budov na vzdálenosti 10 až 100 metrů. Společně s relativně nízkými nároky na datový tok se rádiové moduly *RFM02-433D* a *RFM01-433D* jeví jako nejlepší řešení. Kvůli nedostatečnému dosahu a zbytečné předimenzovanosti z pohledu datového toku je nevhodné používat technologii *Bluetooth* například s modulem *HC-05* [33]. Použití levných ASK modulů *XY-FST* a *XY-MK-5V* je nevhodné kvůli velmi nízkému datovému toku, dosahu a téměř žádné možnosti úpravy parametrů. RFM moduly se zde proto jeví jako ideální řešení. Parametry těchto modulů byly popsány v kapitole *2.1 Obvod RFM02* a v kapitole *2.2 Obvod RFM01*.

K těmto obvodům byla použita anténa *SW433-TH22*. Je to anténa určená pro frekvence 433 (± 8) MHz, vinutá z pozlaceného měděného drátu o průměru 0,5 mm, výškou 22 mm, maximálním výkonem 10 W, ziskem 2,15 dBi a vstupní impedancí 50 Ω [34, 35].

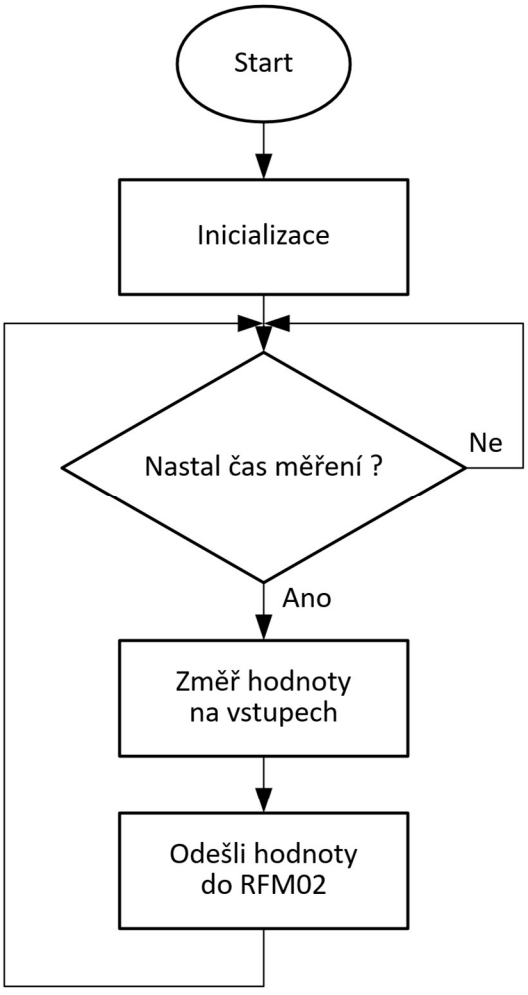
Obr. 3.5 Anténa SW433-TH22 [35]

3.3 Popis fungování programů

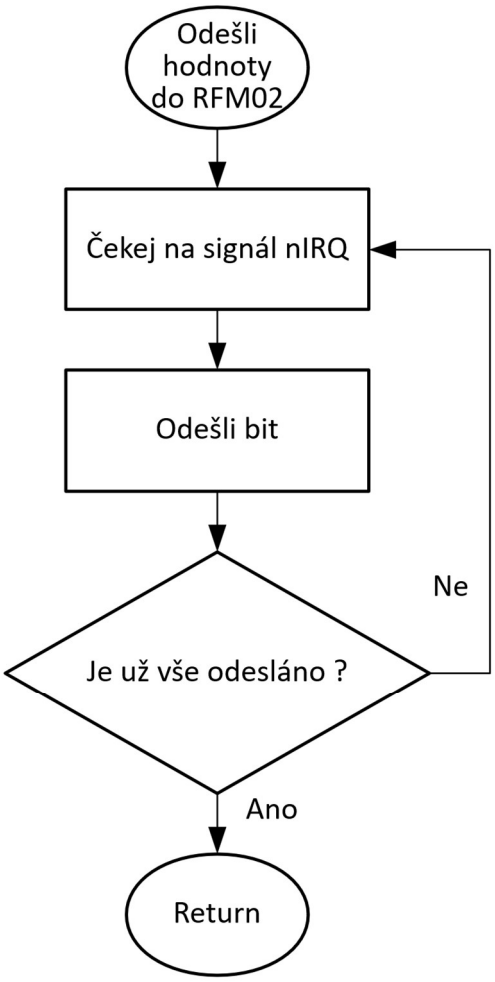
Programový kód byl prán v editoru *Atollic TrueSTUDIO for STM32 9.3.0* jazykem C. Celý kód je součástí *Příloha C* resp. *Příloha D*.

3.3.1 Programový kód pro vysílací část systému

Na *Obr. 3.6* je znázorněn vývojový diagram, podle kterého mikropočítač 1 vykonává svoji činnost. Na *Obr. 3.7* je blíže znázorněna funkce *Odešli hodnoty do RFM02*. Kompletní programový kód viz *Příloha C*.



Obr. 3.6 Vývojový diagram pro vysílání dat



Obr. 3.7 Vývojový diagram pro blok Odešli hodnoty do RFM02

V bloku Inicializace se provádí následující funkce. Funkce `PLLInit()` přenastavuje taktovací frekvenci mikrokontroleru z 8 MHz na 64 MHz pomocí integrovaného fázového závěsu (PLL) a vnitřního RC oscilátoru (HSI). Funkce `SysTick_Config()` nastavuje obsluhu vnitřního přerušení od SysTicku, které se používá k časování. Funkce `EXTI0Init()` nastavuje externí přerušení na lince 0 portu F. Funkce `USART2Init()` nastavuje sériovou komunikaci USART po VCP s PC. Rychlost komunikace je nastavena na 460800 Bd. Další funkce `SPI1Init()` nastavuje sériovou komunikaci SPI pro komunikaci s RFM02. Funkce `ADC1Init()` povoluje a nastavuje 12-bitový A/D převodník pro měření vstupních signálů. A/D převodník funguje v tzv. injected režimu. `DAC2Init()` je funkce, která nastaví 12-bitový D/A převodník pro vyvedení kontrolního pilového signálu. `TIM3Init()` inicializuje časovač

a povoluje přerušení programu. Časovač je nastaven na 1,85 ms. Během přerušení se provede A/D převod a inkrementuje se osmibitová proměnná `pila` pro generování pilového signálu. Vzorkovací frekvence je tedy 540 Hz a frekvence pilového signálu je 2,11 Hz.

Asi nejzajímavější inicializační funkcí je funkce `RFM02Init()`, která pomocí sériové komunikace SPI nastaví parametry obvodu *RFM02-433D*. Funkce je vypsána na *Obr. 3.8*. V prvním příkazu se čtou status registry obvodu RFM02. Tyto hodnoty nejsou nijak zaznamenávány. V dalším příkazu se nastavuje frekvenční pásmo, tj. 434 MHz, výstupní kmitočet hodinových pulzů CLK se nastavuje na 2 MHz, kapacita pro krystalový oscilátor se nastavuje na 11,5 pF, rozdíl značkové, resp. mezerové frekvence od nosné je nastaven na ± 60 kHz [24]. Příkazem na řádku 300 se nastavuje opět pásmo 434 MHz, tentokrát do jiného registru [24]. Některé hodnoty byly převzaty z příkladu programu uváděné výrobcem v katalogovém listu, jako např. kapacita kondenzátoru, šířka frekvenčního pásma apod. [24]. Datový tok je nastaven dalším příkazem na 57,6 kb/s. Hodnota zapsaná do registru se vypočte pomocí vztahu (3.1) [24]:

$$R = \frac{\frac{10\,000\,000}{29}}{BR} - 1 = \frac{\frac{10\,000\,000}{29}}{57\,600} - 1 \doteq 5 \tag{3.1}$$

Kde *R* je hodnota zapisovaná do registru, *BR* je datový tok. Následujícím příkazem je nastaven fázový závěs pro tuto hodnotu datového toku. Dále je příkazem povolena bitová synchronizace a zakázán Wake-up timer. Nakonec se posledním příkazem celé nastavení uzavře.

```
296 void RFM02Init(void)
297 {
298   WriteSPI(0xCC00);
299   WriteSPI(0x8B61);     // 433 band, +/- 90 kHz
300   WriteSPI(0xA640);     // 434 MHz
301   WriteSPI(0xC805);     // 57.6 kbps
302   WriteSPI(0xD200);     // 57.6 kbps
303   WriteSPI(0xC220);     // en bit sync
304   WriteSPI(0xC001);     // close all
305
306
307   Nucleo_SetPinGPIO(FSK, ioPortOutputPP);
308   GPIOWrite(FSK, false);
309 }
```

Obr. 3.8 Úryvek z programového kódu – funkce `RFM02Init()`

Odesílání dat se provádí vždy až po třech převodech A/D na všech čtyřech kanálech. Najednou se tedy odesílá celkem 12 hodnot po 16 bitech. Je to uděláno proto, aby se odesílalo

více hodnot najednou a ušetřilo se odesílání režijních znaků pro začátek a konec vysílání. Vysílání ukončuje odeslání kontrolního součtu všech 12 hodnot. O odesílání dat se stará funkce `RFMSendMessage()` viz *Obr. 3.9*.

Na začátku se pošle po SPI příkaz C038h, pro spuštění taktovacích pulzů na pinu nIRQ. Vždy při sestupné hraně taktovacího signálu nIRQ vysílač přečte logickou hodnotu na pinu FSK a vyšle ji [24]. Tyto pulzy mají kmitočet 57,6 kHz odpovídající datovému toku 57,6 kb/s. Potom se odesílají (už po pinu FSK) tři bajty AA jako preambule. Je to střídání log. 1 a log. 0 vždy po jednom taktu hodin pro synchronizaci [24]. Následují bajty 2D a D4, podle kterých přijímač pozná, že vysílaná data jsou určená právě jemu [24]. Dále, jak už bylo zmíněno výše, následují datové bajty. Těch je 24. Potom se posílá vyšší bajt kontrolního součtu počítaném v 16-bitové proměnné `ChkSum`. Nakonec se pošle dummy bajt AA po FSK pinu a vysílání je ukončeno příkazem C001h po SPI rozhraní [24]. Celkem je tedy v jedné zprávě odesláno 31 B, z toho 24 B jsou užitečná data, zbytek jsou režijní znaky. Při periodickém vysílání každých 5,55 ms je skutečný datový tok 34,56 kb/s (podle rovnice *(3.2)*). Jednotlivé bajty se odesílají funkcí `RFMSendByte()`. Funkce je vypsána na *Obr. 3.10*.

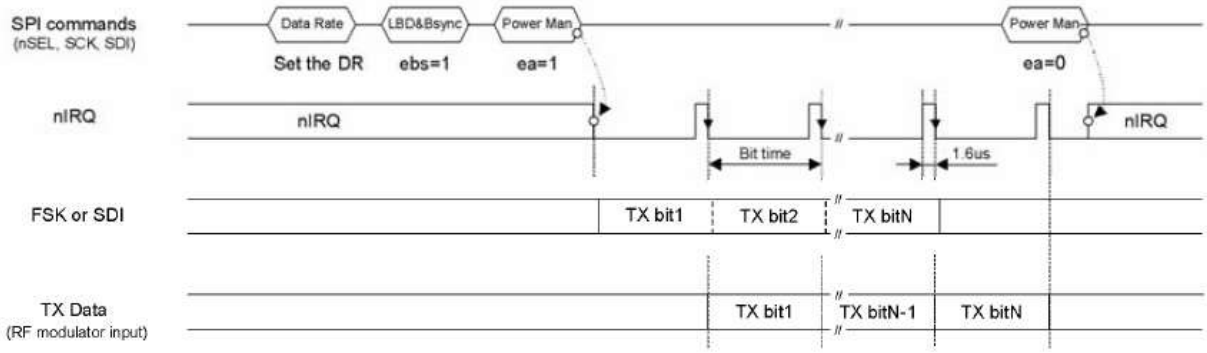
$$BR = \frac{1}{3 \cdot 1{,}85 \cdot 10^{-1}} \cdot 24 \cdot 8 = 34560 \text{ [b/s]} \tag{3.2}$$

```
325 void RFMSendMessage(uint16_t *message)
326 {
327   uint16_t ChkSum = 0;                  // kontrolni soucet
328
329   WriteSPI(0xC038);                     // start Tx po FSK pinu
330
331   uint32_t j;
332   for(j = 0; j < 1500; j++)
333     ;
334
335   RFMSendByte(0xAA);                    // Preamble
336   RFMSendByte(0xAA);                    // Preamble
337   RFMSendByte(0xAA);                    // Preamble
338
339   RFMSendByte(0x2D);                    // Synchron pattern
340   RFMSendByte(0xD4);                    // Synchron pattern
341
342   int i;
343   for(i = 0; i < 12; i++)
344   {
345     RFMSendByte(message[i] / 256);
346     RFMSendByte(message[i] % 256);
347     ChkSum += message[i];
348   }
349
350   RFMSendByte(ChkSum / 256);
351
352   RFMSendByte(0xAA);                    // konec odesilani
353
354   WriteSPI(0xC001);                     // konec Tx po FSK pinu
355 }
```

Obr. 3.9 Úryvek z programového kódu – funkce `RFMSendMessage()`

```
311 void RFMSendByte(uint8_t aByte)          // BLOKUJICI FUNKCE !!!
312 {
313   int i = 0;
314   for(i = 0; i < 8; i++)
315   {
316     while(!nIRQsignal)                    // cekani na nIRQ
317       ;
318
319     nIRQsignal = false;                   // shodit priznak nIRQ
320     GPIOWrite(FSK, (aByte & 0x80) != 0);// maska 1000 0000
321     aByte <<= 1;                          // bitovy posun
322   }
323 }
```

Obr. 3.10 Úryvek z programového kódu – funkce `RFMSendByte()`

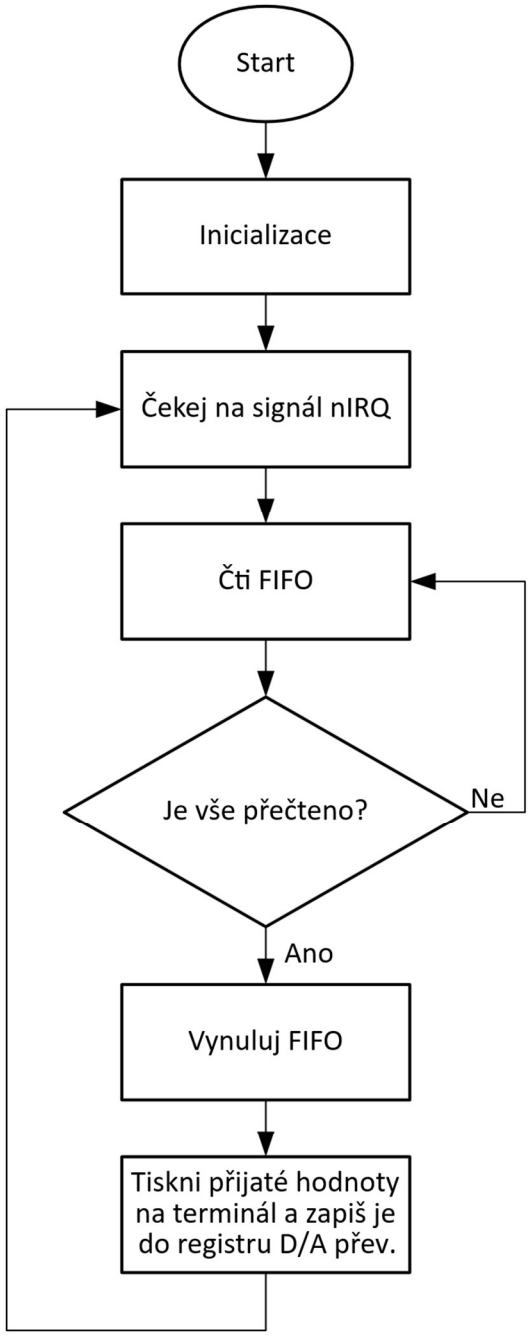


Obr. 3.11 Časový diagram pro vysílání dat [24]

3.3.2 Programový kód pro přijímací část systému

Na *Obr. 3.12* je znázorněn vývojový diagram, podle kterého mikropočítač 2 vykonává svoji činnost. Kompletní programový kód viz *Příloha D*.

V bloku Inicializace se provádějí některé funkce stejné nebo velmi podobné jako ve vysílací části (viz *kapitola 3.3.1*). Jsou to funkce `PLLInit()`, `Usart2Init()`, `SPI1Init()`, `TIM3Init()` a `DAC2Init`. Navíc zde přibyly funkce `DAC1Init()` pro nastavení dvoukanálového 12-bitového D/A převodníku a funkce `RFM01Init()`, která pomocí sériové komunikace SPI nastaví parametry obvodu *RFM01-433D* podobně jako funkce `RFM02Init()` u vysílací části.



Obr. 3.12 Vývojový diagram pro příjem dat

```
328 void RFM01Init(void)
329 {
330   WriteSPI(0x0000);
331   WriteSPI(0x898A);     // 433 band, 134 kHz
332   WriteSPI(0xA640);     // 433 MHz
333   WriteSPI(0xC805);     // 57.6 kbps
334   WriteSPI(0xC69B);     // AFC setting
335   WriteSPI(0xC42A);     // Clock recovery manual control, Digital filter DQD=4
336   WriteSPI(0xC240);     // output 1.66 MHz
337   WriteSPI(0xC080);
338   WriteSPI(0xCE88);     // use FIFO
339   WriteSPI(0xCE8B);
340   WriteSPI(0xC081);     // open Rx
341
342   Nucleo_SetPinGPIO(SDO, ioPortInputPU);
343   Nucleo_SetPinGPIO(nIRQ, ioPortInputPU);
344 }
```

Obr. 3.13 Úryvek z programového kódu – funkce `RFM01Init()`

Na *Obr. 3.13* je vypsána funkce `RFM01Init()`. Podobně jako ve vysílací části se zde nastavují řídicí registry po rozhraní SPI. Prvním příkazem se přečte status registr, ale s přečtenými hodnotami se nijak nenakládá [26]. Dále na řádku 331 se nastavuje pásmo 434 MHz, povoluje se krystalový oscilátor, kapacita je nastavena na 12,5 pF, šířka základního pásma je nastavena na 134 kHz [26]. Na dalším řádku se opět nastavuje pásmo 434 MHz [26]. Dalším příkazem se nastavuje datový tok na 57,6 kb/s [26]. Hodnota do registru se opět vypočte podle vztahu (3.1) [26]. Dále je nastaven registr automatického řízení frekvence (AFC). Nastavuje se zde řízení mikrokontrolerem, rozsah frekvenční odchylky -16 až +15 kHz, AFC bez zvýšené přesnosti, a nakonec se celé AFC povolí [26]. Další příkaz nastaví manuální obnovení hodin a digitální filtr [26]. Potom se nastavují hodiny na kmitočet 1,66 MHz [26]. Následně se nastaví indikátor platných dat (VDI), zisk předzesilovače na 0 dBm, indikátor síly přijímaného signálu DRSSI na -103 dBm, a zakáže se přijímač [26]. Dalšími dvěma příkazy se povolí FIFO, nastaví se jeho délka na 8 bitů, nastaví se začátek přijímaných dat na synchronizační bajty a vymaže se obsah FIFO [26]. Nakonec se příkazem povolí přijímač [26]. Některé hodnoty byly převzaty z příkladu programu uváděné výrobcem v katalogovém listu, jako např. kapacita kondenzátoru, šířka frekvenčního pásma, délka FIFO, rozsah apod [26].

O čtení FIFO se stará funkce `RFM01ReadFIFO()` vypsaná na *Obr. 3.14*. Tato funkce přečte vždy jeden přijmutý bajt.

Velice důležité pro správný přenos tvaru signálu je přesné časování jak na straně přijímače, tak vysílače. To je zajištěno u obou částí systému přerušením programu od časovače TIM3,

který je nastaven na 1,85 ms. Zatímco ve vysílači se měří A/D převodníkem hodnoty analogových vstupů na všech čtyřech kanálech najednou a až po třech těchto měřeních v čase jsou data odeslány, na straně přijímače se postupně v čase tyto čtveřice hodnot odesílají do D/A převodníku.

```
346 uint8_t RFM01ReadFIFO(void)
347 {
348   uint8_t i, Result;
349   GPIOWrite(SCK, false);
350   GPIOWrite(SDI, false);
351   GPIOWrite(nSEL, false);
352
353   for(i = 0; i < 16; i++)   // preskoc status bity
354   {
355     GPIOWrite(SCK, true);
356     GPIOWrite(SCK, true);
357     GPIOWrite(SCK, false);
358     GPIOWrite(SCK, false);
359   }
360
361   Result = 0;
362   for(i = 0; i < 8; i++)
363   {
364     Result <<= 1;
365     if(GPIORead(SDO))
366     {
367       Result |= 1;
368     }
369     GPIOWrite(SCK, true);
370     GPIOWrite(SCK, true);
371     GPIOWrite(SCK, false);
372     GPIOWrite(SCK, false);
373   }
374
375   GPIOWrite(nSEL, true);
376   return Result;
377 }
```

Obr. 3.14 Úryvek z programového kódu – funkce `RFM01ReadFIFO()`

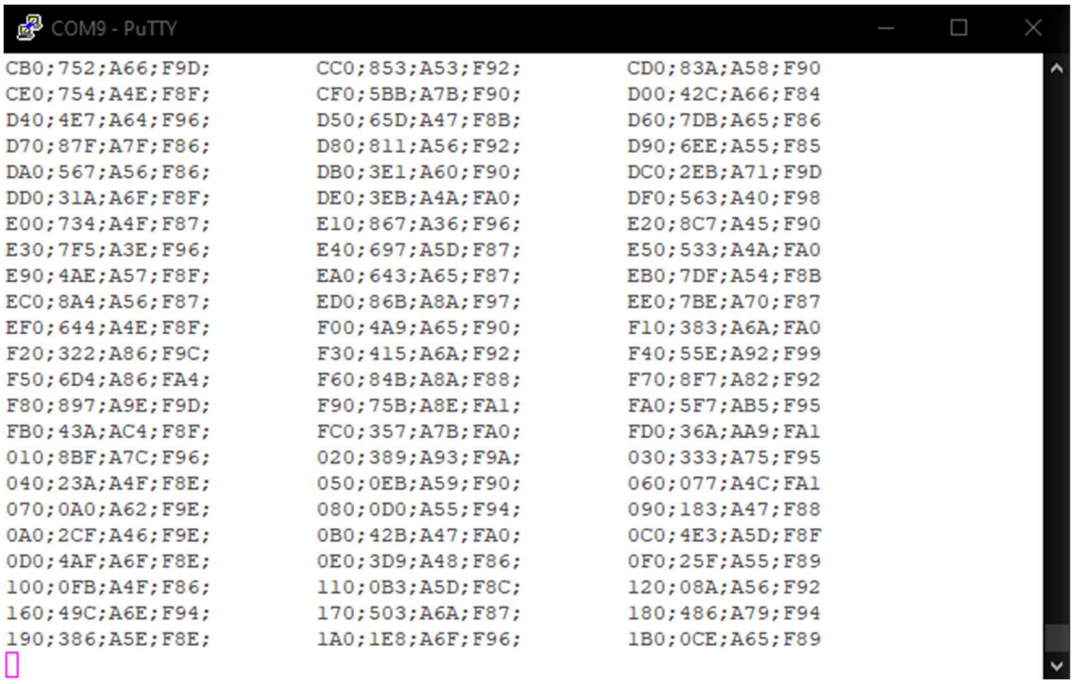
Ochrana proti příjmu nesprávných dat je řešena dvěma způsoby: jednak se porovnává kontrolní součet všech přenášených bajtů, jednak je na první čtyři nevyužité bity prvního kanálu přidána hodnota A, která se při přijetí dat kontroluje. Nevyužité proto, že A/D i D/A převodníky jsou 12-bitové, ale jsou ukládány v 16-bitové proměnné. Výhoda kontroly znaku A na začátku zprávy je taková, že jsou ihned rozpoznána špatná data, tím se ihned vymaže FIFO, aby byla připravena na další příjem dat, a nemusí se přijímat celá špatná zpráva a tím zdržovat mikropočítač.

3.4 Ověření funkce bezdrátového systému

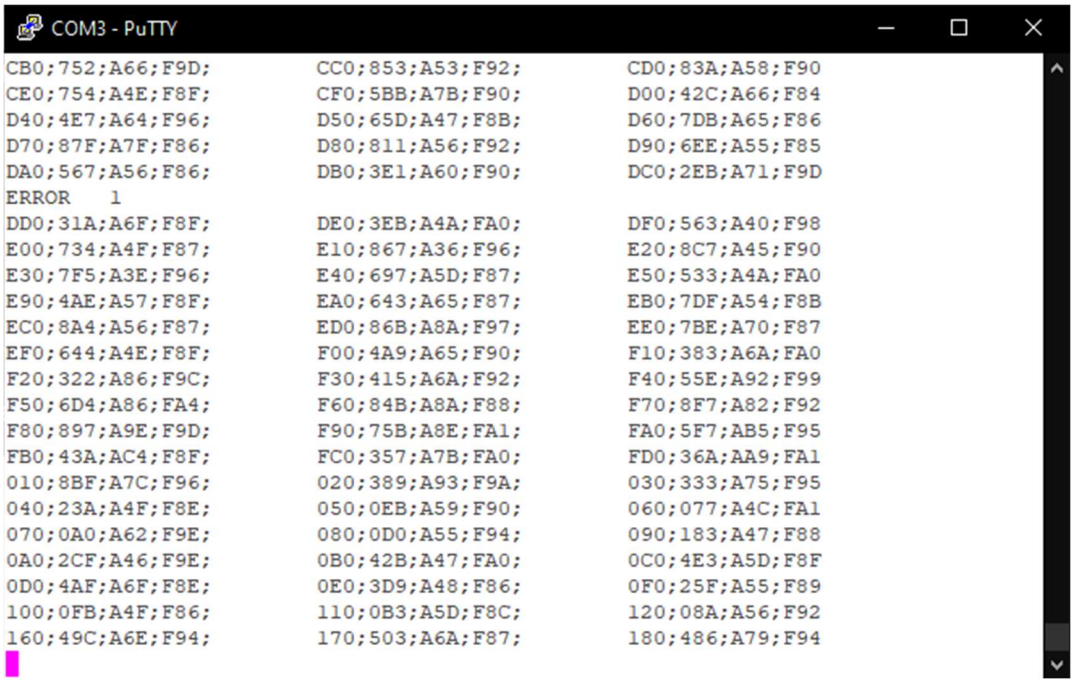
Při měřeních byly použity tyto přístroje:

- čtyřkanálový osciloskop Agilent Technologies InfiniiVision DS07034A 350 MHz 2GSa/s
- funkční generátor HEWLETT PACKARD 33120A 15 MHz

3.4.1 Ověření funkce zobrazením hodnot na obrazovce počítače



Obr. 3.15 Odeslaná data vysílačem zobrazená na terminálu PC



Obr. 3.16 Přijatá data přijímačem zobrazená na terminálu PC

První ověření funkce přenosového systému bylo pomocí výpisu hodnot do terminálu osobního počítače. Výpis hodnot je zobrazen na *Obr. 3.15* a *Obr. 3.16*. Zřejmě se hodnoty vyslané shodují s hodnotami přijatými. První hodnota řetězce v tomto případě reprezentuje pilový signál generovaný mikropočítačem na vysílací straně. Další tři hodnoty reprezentují tři analogové vstupy, na které jsou v tomto případě přivedená různá stejnosměrná napětí. Oddělení tabulátorem znázorňuje další měření (o 1,85 ms později).

Lze si všimnout chyby na přijímací straně, která je značena *ERROR 1*. Tato chyba je vyhodnocena tak, že první čtyři bity prvního kanálu nebyly přečteny jako hodnota A. Obvykle v těchto přijatých zprávách objevovalo několik hodnot 0 a následovaly hodnoty zcela nesouvisející s vysílanými. Lze si ale také všimnout, že i přes tuto chybu se žádná data neztratila a hodnoty na sebe navazují. To je dáno rychlým vymazáním FIFO, čímž se povedlo včas zachytit platná data.

Vzdálenost vysílače a přijímače při tomto měření byla do jednoho metru.

3.4.2 Ověření funkce pilovým signálem

Další ověření proběhlo kontrolou signálu na obrazovce osciloskopu viz *Obr. 3.17*. Kanál 2 osciloskopu byl připojen na výstup D/A převodníku vysílače, ve schématu označovaný jako ANALOG_OUT_CH1. Zde je vyveden softwarově generovaný pilový signál. Kanál 4 osciloskopu byl připojen na výstup D/A převodníku na přijímači, označovaný jako ANALOG_OUT_CH1.

Obr. 3.17 Snímek obrazovky osciloskopu při ověřování pilovým signálem

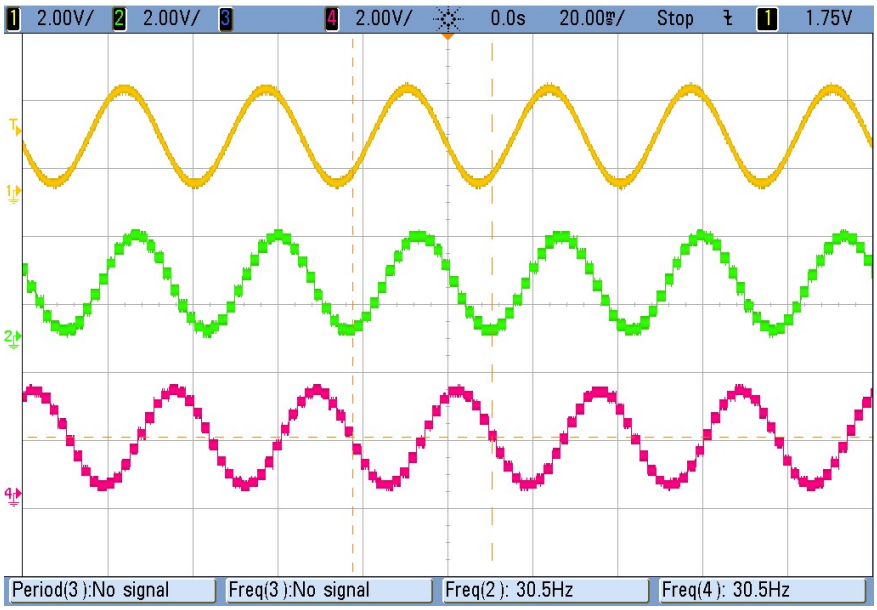
Jak je vidět, signál byl přenesen a zrekonstruován velmi věrohodně a nezkresleně, jen s malým fázovým posuvem. Tento fázový posuv byl měřen později.

Vzdálenost vysílače a přijímače při tomto měření byla do jednoho metru.

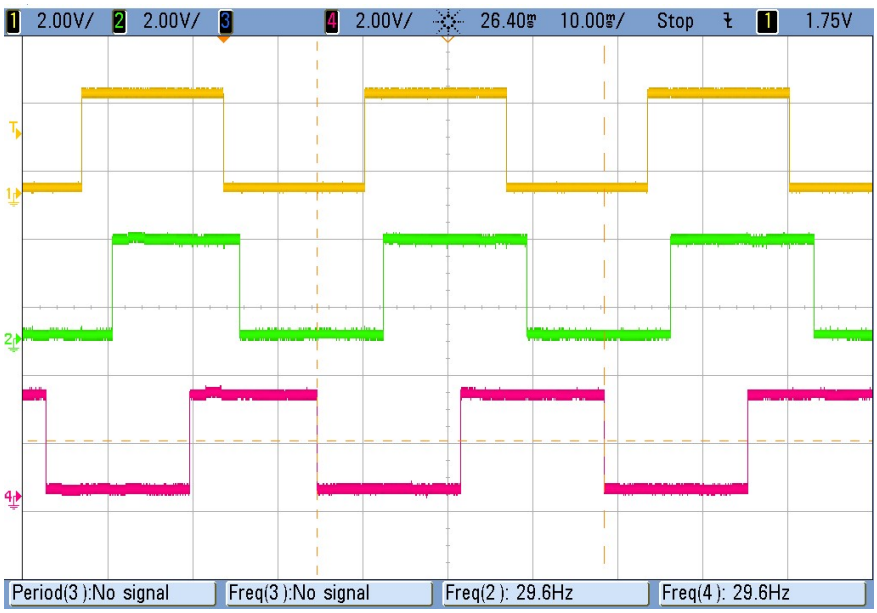
3.4.3 Ověření funkce přivedením externího signálu z funkčního generátoru

Další měření proběhlo připojením funkčního generátoru na analogový vstup vysílače ANALOG_IN_CH2. Před tímto připojením bylo nutné pečlivě nastavit napětí špička-špička a offset přiváděného signálu. Pro ochranu byl ještě mezi výstup z generátoru a vstup A/D převodníku zařazen do série rezistor o hodnotě 1 kΩ. Frekvence signálu byla 30 Hz. Na kanál 1 osciloskopu byl přiveden signál z generátoru. Na kanál 2 osciloskopu byl přiveden signál z D/A převodníku na vysílací straně, ve schématu označovaný jako ANALOG_OUT_CH1. Tentokrát byla tímto vývodem rekonstruován již zdigitalizovaný signál. Na kanál 4 osciloskopu byl přiveden signál ANALOG_OUT_CH2 z přijímače.

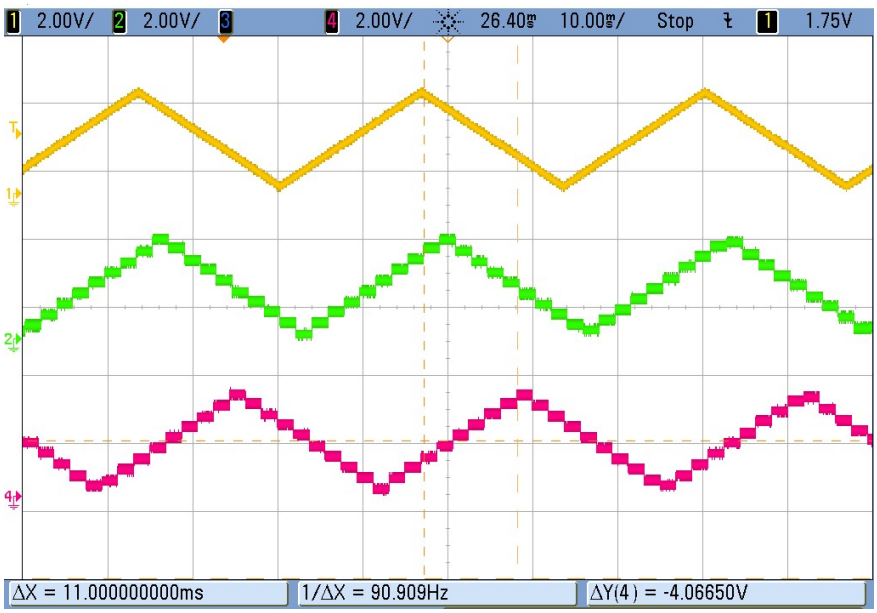
Z *Obr. 3.20* lze vyčíst zpoždění signálu mezi přiváděným signálem z generátoru a vyvedeným signálem z přijímače. Zpoždění 11 ms je způsobeno především akumulováním tří měření, než se hodnoty vysílačem odešlou a následným postupným převáděním na straně přijímače.



Obr. 3.18 Snímek obrazovky osciloskopu při ověřování sinusovým signálem



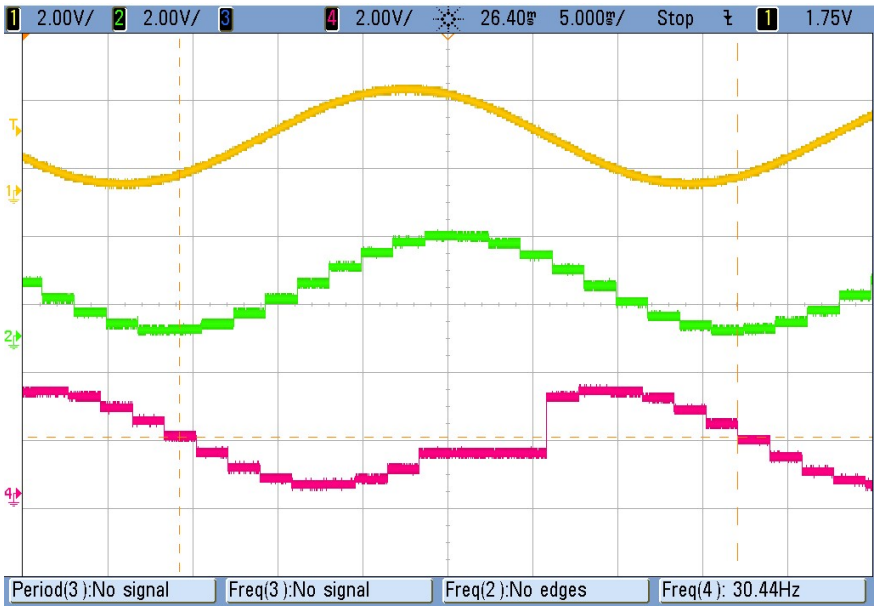
Obr. 3.19 Snímek obrazovky osciloskopu při ověřování obdélníkovým signálem



Obr. 3.20 Snímek obrazovky osciloskopu při ověřování trojúhelníkovým signálem

Na *Obr. 3.21* je příklad nesprávně přeneseného a zrekonstruovaného signálu. Chyba je zřejmě způsobená nezaznamenáním hodnot, vyhodnocením nesprávnosti hodnot kontrolním součtem nebo hodnoty A v prvních čtyřech bitech zprávy nebo rušením jiným signálem. Tato chyba se objevuje velmi nepravidelně a velmi zřídka. Ze snímku je vidět opět chyba ve třech hodnotách způsobená chybou celého řádku, tzn. tří měření v čase.

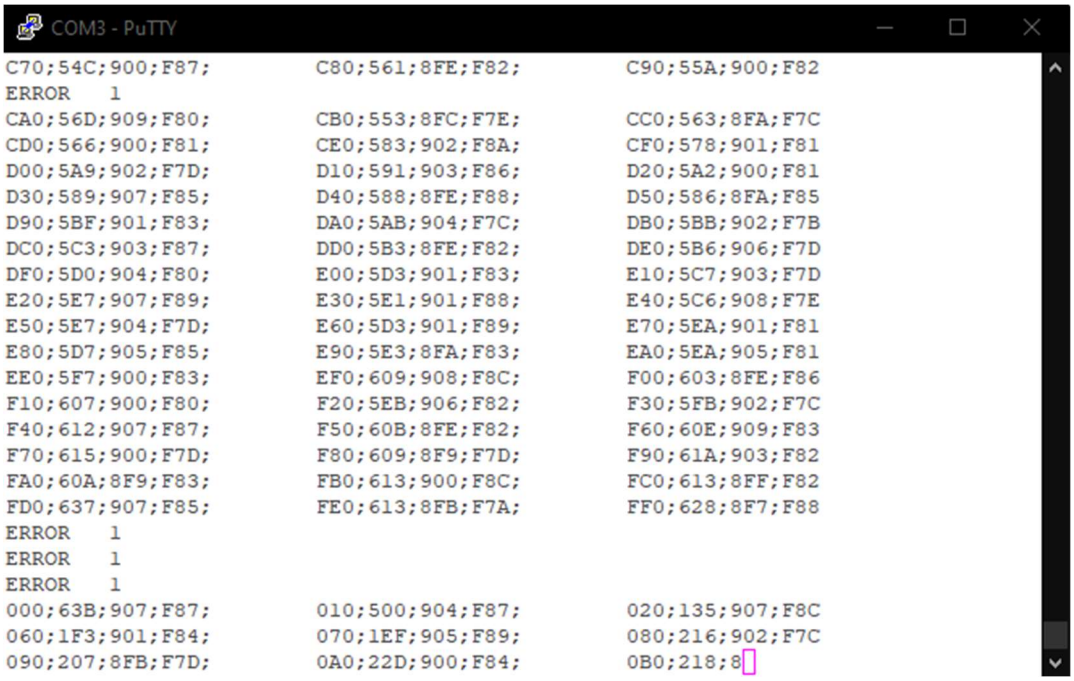
Vzdálenost vysílače a přijímače při tomto měření byla do jednoho metru.



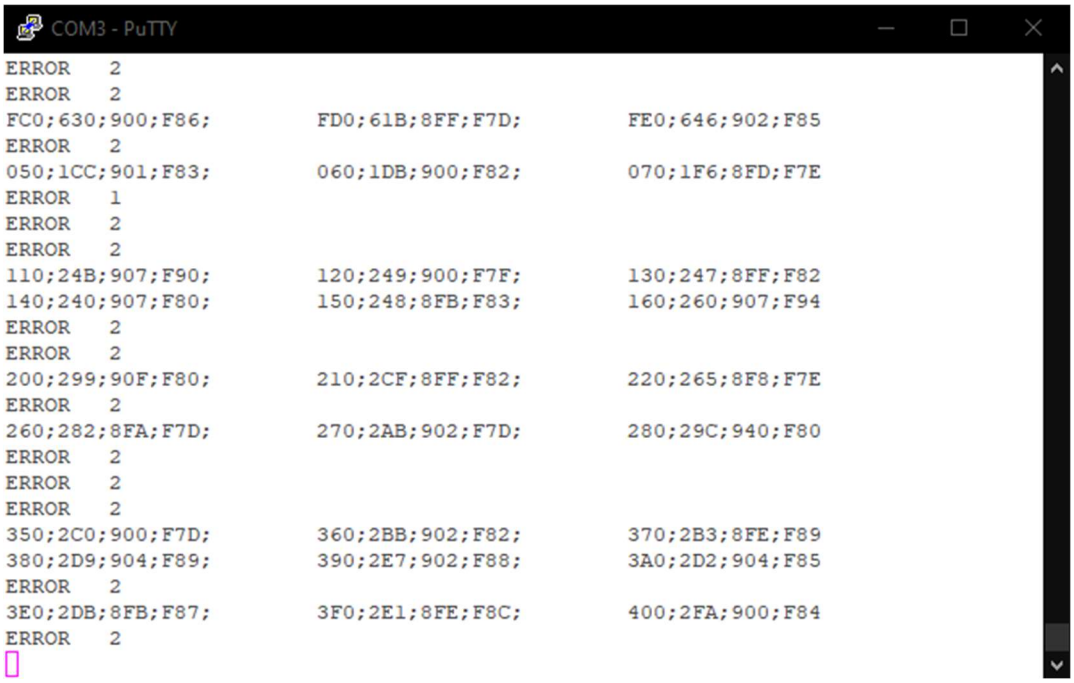
Obr. 3.21 Snímek obrazovky osciloskopu při ověřování sinusovým signálem – chyba

3.4.4 Ověření maximálního dosahu spojení

Tato zkouška proběhla orientačně metodou kontroly hodnot na terminálu PC. Ve venkovním volném prostředí bylo zjištěno bezchybné fungování na vzdálenost 120 metrů viz *Obr. 3.22*. Na tuto vzdálenost přenos probíhal stejně jako na vzdálenost do 1 metru. Na vzdálenost větší docházelo už k častým chybám viz *Obr. 3.23* a na vzdálenost větší než 150 metrů se žádné hodnoty nepřenášely bezchybně viz *Obr. 3.24*.



Obr. 3.22 Zkouška dosahu bezdrátového systému – vzdálenost do 120 metrů



Obr. 3.23 Zkouška dosahu bezdrátového systému – vzdálenost od 120 do 150 metrů

Obr. 3.24 Zkouška dosahu bezdrátového systému – vzdálenost nad 150 metrů

Uvnitř budovy s velkým množstvím zdí, nábytku a elektrospotřebičů proběhla zkouška následovně: obdobně jako na *Obr. 3.22* probíhal přenos na vzdálenost do 20 metrů na stejném podlaží. Nad vzdálenost 20 metrů rozdílném podlaží už byl přenos chybový podobně jako na *Obr. 3.24*.

Závěr

V první části práce byly porovnány a popsány různé metody bezdrátového propojení periferního zařízení a osobního počítače. Byly zde stručně představeny vlastnosti a funkce technologií *Bluetooth*, *Wifi*, *Certified Wireless USB* a *DASH7*.

V druhé kapitole byl proveden rozbor bezdrátových modulů *RFM01*, *RFM02*, *XY-MK-5V* a *XY-FST*. Moduly *RFM01* a *RFM02* byly použity pro realizaci a následné ověření bezdrátového přenosu dat, popsané v kapitole třetí.

Ve třetí části byly uvedeny priority pro návrh bezdrátového systému pro přenos dat mezi dvěma mikrokontrolery, byly rozebrány jednotlivé komponenty systému a v neposlední řadě byl popsán celý algoritmus a program, podle kterého mikrokontrolery vykonávali své činnosti jak na vysílací straně, tak na přijímací.

Bylo ověřeno různými měřeními, že navržený a realizovaný systém je funkční. Systémem lze přenášet čtyři kanály se vzorkovací frekvencí 540 Hz. Aby byl dodržen Shannonův–Nyquistův–Kotělnikovův teorém musí spektrum signálu přenášené systémem obsahovat jen složky kmitočtů do 270 Hz, aby nedošlo k aliasingu. Systém lze tedy použít pro měření a bezdrátový přenos EKG. Skutečný datový tok (očištěný o synchronní bajty apod.) byl 34,56 kb/s. Toto číslo by se mohlo daleko více přiblížit nastaveným 57,6 kb/s zrychlením komunikace SPI, či kontinuálním vysíláním bez ukončování vysílání.

Vzdálenost, na kterou byla zjištěna spolehlivá komunikace, byla do 120 metrů ve volném otevřeném prostředí. Tento údaj se od výrobcem udávaného dosahu liší. Výrobce udává dosah až 300 metrů, ovšem při rychlosti komunikace 1,2 kb/s. Kvůli rozdílnému datovému toku se nejspíš tyto hodnoty liší. Uvnitř budovy byl zjištěn maximální dosah bezchybné komunikace na 20 metrů na stejném patře. Dosah by byl možný prodloužit volbou lepší antény.

Celé zařízení bylo stavěno jako prototyp, kdy jednotlivé komponenty byly spojeny rozebíratelnými spojeními vodičů. Miniaturizace pro praktické použití by byla možná volbou SMD modulů místo vývodových, použitím plošného spoje s paticí pro vývojovou desku s mikrokontrolerem, případně návrhem kompletní desky pro připojení samotného mikrokontroleru s SMD pouzdrem. Dále by bylo nutné v praxi vyřešit napájení z baterie.

Seznam použité literatury

[1] Bluetooth. *Wikipedie* [online]. [vid. 2020-04-15]. Dostupné z: https://cs.wikipedia.org/wiki/Bluetooth

[2] How does Bluetooth work? *Scientific American* [online]. 2007 [vid. 2020-04-15]. Dostupné z: https://www.scientificamerican.com/article/experts-how-does-bluetooth-work/

[3] Všeobecné oprávnění č. VO-R/10/12.2019-9 k využívání rádiových kmitočtů a k provozování zařízení krátkého dosahu. *Český telekomunikační úřad* [online]. 2019 [vid. 2020-04-15]. Dostupné z: https://www.ctu.cz/sites/default/files/obsah/ctu/vseobecne-opravneni-c.vo-r/10/12.2019-9/obrazky/vo-r10-122019-9.pdf

[4] Bluetooth Connection GFSK. *UK Essays* [online]. 2017 [vid. 2020-04-18]. Dostupné z: https://www.ukessays.com/essays/communications/bluetooth-connection-gfsk.php

[5] MIKÉSKA, Zdeněk. Specifikace rádiové části systému Bluetooth. *Ústav radioelektroniky, Fakulta elektrotechniky a komunikačních technologií, Vysoké učení Technické v Brně* [online]. 2004 [vid. 2020-04-15]. Dostupné z: http://www.elektrorevue.cz/clanky/04003/index.html#Obsah

[6] BHAGWAT, P. a A. SEGALL. A routing vector method (RVM) for routing in Bluetooth scatternets. In: *1999 IEEE International Workshop on Mobile Multimedia Communications (MoMuC'99) (Cat. No.99EX384)* [online]. B.m.: IEEE, 2003, s. 375–379. ISBN 0-7803-5904-6. Dostupné z: doi:10.1109/MOMUC.1999.819514

[7] Radio Versions. *Bluetooth®* [online]. [vid. 2020-04-15]. Dostupné z: https://www.bluetooth.com/learn-about-bluetooth/bluetooth-technology/radio-versions/

[8] Bluetooth (INFORMACE): verze, dosah, frekvence a protokoly. *Alza.cz* [online]. 2019 [vid. 2020-04-18]. Dostupné z: https://www.alza.cz/slovnik/bluetooth-art12370.htm#verze

[9] *Wi-Fi Alliance* [online]. [vid. 2020-04-23]. Dostupné z: https://www.wi-fi.org/

[10] *What Does WiFi Stands for and How It Works* [online]. [vid. 2020-04-22]. Dostupné z: https://www.netspotapp.com/what-is-wifi.html

[11] TJENSVOLD, Jan Magne. *Comparison of the IEEE 802.11, 802.15.1, 802.15.4 and 802.15.6 wireless standards* [online]. 2007 [vid. 2020-04-15]. Dostupné z: http://lars.mec.ua.pt/public/LAR Projects/SystemDevelopment/2016_RaquelAlves/wireless/comparison-ieee-802-standards.pdf

[12] HOFFMAN, Chris. What's the Difference Between Ad-Hoc and Infrastructure Mode Wi-Fi? *How-To Geek* [online]. 2016 [vid. 2020-04-24]. Dostupné z: https://www.howtogeek.com/180649/htg-explains-whats-the-difference-between-ad-hoc-and-infrastructure-mode/

[13] IEEE 802.11. *Wikipedie* [online]. 2020 [vid. 2020-04-24]. Dostupné z: https://cs.wikipedia.org/wiki/IEEE_802.11#cite_note-80211ad-1

[14] MITCHELL, Bradley. Wireless Standards Explained: 802.11ax, 802.11ac, 802.11b/g/n. *Lifewire* [online]. 2020 [vid. 2020-04-24]. Dostupné z: https://www.lifewire.com/wireless-standards-802-11a-802-11b-g-n-and-802-11ac-816553

[15] IEEE 802.11n - Zrychlete a rozšiřte svou bezdrátovou síť. *Intelek* [online]. 2009 [vid. 2020-04-27]. Dostupné z: https://www.intelek.cz/art_doc-5C56A0147621A13AC12575510053AE3E.html

[16] KELLY, Gordon. 802.11ac vs 802.11n WiFi: What's The Difference? *Forbes* [online]. 2014 [vid. 2020-04-27]. Dostupné z: https://www.forbes.com/sites/gordonkelly/2014/12/30/802-11ac-vs-802-11n-wifi-whats-the-difference/#721e3b683957

[17] WiFi 6. *TP-Link* [online]. [vid. 2020-05-05]. Dostupné z: https://www.tp-link.com/cz/wifi6/

[18] DIGNAN, Larry. D-Link, Asus tout 802.11ax Wi-Fi routers, but you'll have to wait until later in 2018. *ZDNet* [online]. 2018 [vid. 2020-05-05]. Dostupné z: https://www.zdnet.com/article/d-link-asus-tout-802-11ax-wi-fi-routers-but-youll-have-to-wait-until-later-in-2018/

[19] Wireless USB FAQ. *Everything USB* [online]. [vid. 2020-05-05]. Dostupné z: https://www.everythingusb.com/wireless-usb.html

[20] Wireless USB. *Wikipedie* [online]. 2019 [vid. 2020-05-05]. Dostupné z: https://cs.wikipedia.org/wiki/Wireless_USB

[21] TORRES, Gabriel. Introduction to Wireless USB (WUSB). *Hardware Secrects* [online]. 2008 [vid. 2020-05-05]. Dostupné z: https://www.hardwaresecrets.com/introduction-to-wireless-usb-wusb/

[22] *DASH7 Alliance – An open specification* [online]. [vid. 2020-05-05]. Dostupné z: https://dash7-alliance.org/

[23] QUINNELL, Richard. Low power wide-area networking alternatives for the IoT. *EDN* [online]. 2015 [vid. 2020-05-06]. Dostupné z: https://www.edn.com/low-power-wide-area-networking-alternatives-for-the-iot/

[24] HOPE MICROELECTRONICS CO., LTD. *RFM02 Universal ISM Band FSK Transmitter* [online]. 2006 [vid. 2020-05-09]. Dostupné z: https://www.tme.eu/Document/648e17cf6585c8a49f7dbc75bc074706/RFM02-433-D.pdf

[25] RFM02-433D HOPE MICROELECTRONICS. *TME Czech Republic s.r.o.* [online]. [vid. 2020-05-10]. Dostupné z: https://www.tme.eu/cz/details/rfm02-433-d/moduly-rf/hope-microelectronics/rfm02-433d/

[26] HOPE MICROELECTRONICS CO., LTD. *RFM01 Universal ISM Band FSK Receiver* [online]. 2006 [vid. 2020-05-09]. Dostupné z: https://www.tme.eu/Document/12638ea34a59443d6d7a7e6951bede32/RFM01-433-D.pdf

[27] RFM01-433D HOPE MICROELECTRONICS. *TME Czech Republic s.r.o.* [online]. [vid. 2020-05-10]. Dostupné z: https://www.tme.eu/cz/details/rfm01-433-d/moduly-rf/hope-microelectronics/rfm01-433d/

[28] 433 MHz vysílač + přijímač. *Arduino-shop.cz* [online]. [vid. 2020-05-10]. Dostupné z: https://arduino-shop.cz/docs/produkty/0/32/1427821401.pdf

[29] ECLIPSERA S. R. O. *433 MHz vysílač + přijímač* [online]. 2016 [vid. 2020-05-10]. Dostupné z: https://arduino-shop.cz/arduino/1003-433-mhz-vysilac-prijimac.html

[30] XY-MK-5V / XY-FST 433 MHz / 315 MHz RF vysílač a přijímač. *ChinaExpress.cz* [online]. [vid. 2020-05-12]. Dostupné z: http://www.chinaexpress.cz/cs/item/xy-mk-5v-xy-fst-433mhz-315mhz-rf-transmitter-and-receiver-link-for-arduino-arm-mcu-raspberry-pi/32732438014.html

[31] KWON, Ohhwan, Jinwoo JEONG, Hyung Bin KIM, In Ho KWON, Song Yi PARK, Ji Eun KIM a Yuri CHOI. Electrocardiogram sampling frequency range acceptable for heart rate variability analysis. *Healthcare Informatics Research* [online]. 2018, **24**(3), 198–206 [vid. 2020-06-09]. ISSN 2093369X. Dostupné z: doi:10.4258/hir.2018.24.3.198

[32] NUCLEO-F303K8. *MBED* [online]. [vid. 2020-06-09]. Dostupné z: https://os.mbed.com/platforms/ST-Nucleo-F303K8/

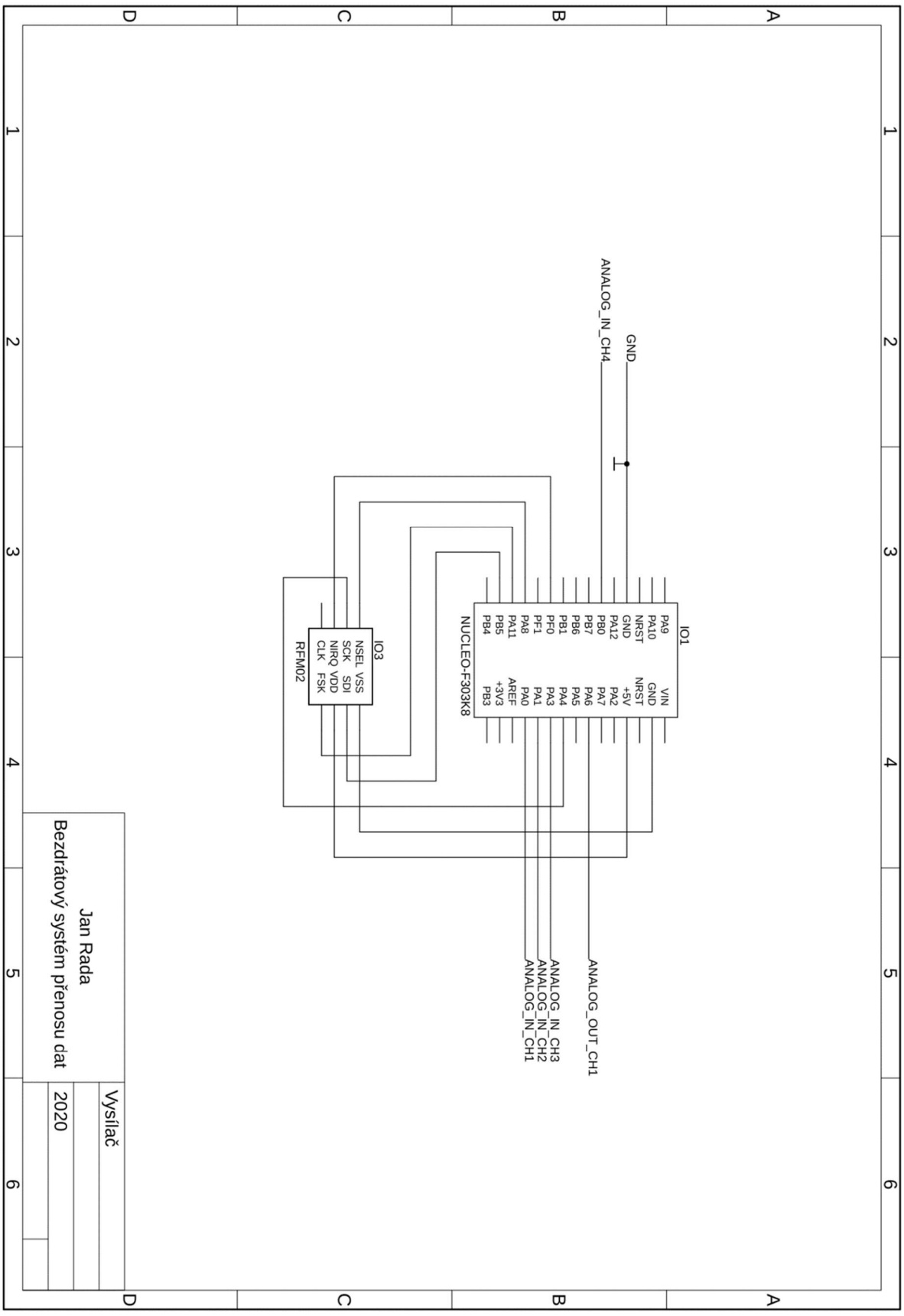
[33] HC-05. *TME Czech Republic s.r.o.* [online]. [vid. 2020-06-09]. Dostupné z: https://www.tme.eu/cz/details/hc-05/moduly-iot-wifi-bluetooth/

[34] SW433-TH22. *NiceRF Wireless Technology Co., Ltd.* [online]. [vid. 2020-06-11]. Dostupné z: https://www.gme.cz/data/attachments/dsh.756-166.1.pdf

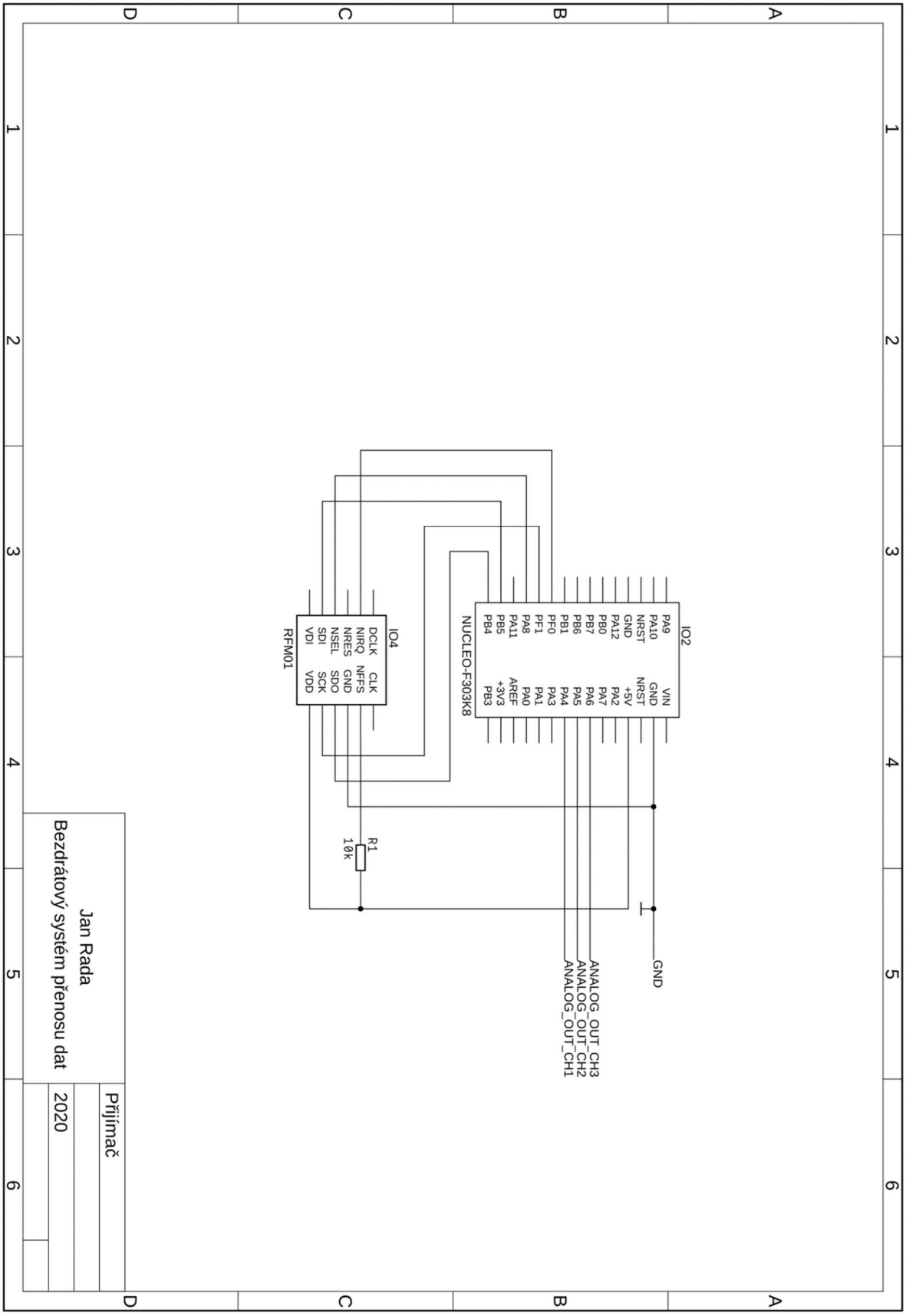
[35] Anténa SW433-TH22, 433MHz 22mm, pozlacená. *GM electronic, spol. s.r.o.* [online]. [vid. 2020-06-11]. Dostupné z: https://www.gme.cz/antena-sw433-th22-433mhz-22mm-pozlacena

Přílohy

A. Příloha – Schéma zapojení vysílací části



B. Příloha – Schéma zapojení přijímací části



C. Příloha – Programový kód pro vysílací část

```
/*****************************************************************************
 * Zapadoceska univerzita v Plzni
 * Fakulta elektrotechnicka
 * Katedra aplikovane elektroniky a telekomunikaci
 *
 * Bakalarska prace
 *
 * Bezdratovy system prenosu dat
 *
 * Jan Rada
 *
 * 2020
 *
 * Vysilac RFM02-433D
 *****************************************************************************/

#include "nucleo_core.h"
#include <stdio.h>
// #include <syscalls.c>// upravene nektere funkce pro vypis v terminalu pomoci usart2

#define SDI   GPIOB,5   // MOSI
#define SDO   GPIOB,4   // MISO
#define SCK   GPIOA,4   // CLK
#define nSEL  GPIOA,8   // nSEL (active low)
#define FSK   GPIOA,11  // FSK

volatile uint32_t _ticks = 0;           // pocitadlo stovek mikrosekund
volatile bool nIRQsignal = false;       // priznak ze se muze vysilat
volatile bool vysilej = false;          // priznak pro zahajeni vysilani
uint8_t seq = false;                    // cislo sloupce dat
uint8_t pila;                           // generovani piloveho signalu
uint16_t x[12];                         // namerene hodnoty
uint16_t message[12];                   // vysilane hodnoty

void SysTick_Handler(void);             // obsluha preruseni od SysTicku
void EXTI0_IRQHandler(void);            // externi preruseni
bool PLLInit(void);                     // nastaveni taktu procesoru na 64 MHz
void EXTI0Init(void);                   // nastaveni externiho preruseni EXTI0
void SPI1Init(void);                    // nastaveni manualniho rezimu SPI1
void WriteSPI(uint16_t val);            // drevni zpusob komunikace po SPI1, ale
  postacujici
void RFM02Init(void);                   // nastaveni vysilace RFM02
uint8_t RFMSendBit(uint8_t val);        // posle jeden bit do vysilace BLOKUJICI FUNKCE !!!
void RFMSendByte(uint8_t aByte);        // BLOKUJICI FUNKCE !!!
void RFMSendMessage(uint16_t *message); // posle zpravu 16 Bytu
void ADC1Init(void);                    // nastaveni 4 kanalu AD prevodniku 1
void DAC1Init(void);                    // nastaveni 1 kanalu DA prevodniku 1
void DAC2Init(void);                    // nastaveni 1 kanalu DA prevodniku 2
void TIM3Init(void);                    // nastaveni casovace TIM3
void TIM3_IRQHandler(void);             // obsluha preruseni od TIM3
int Usart2Send(char c);                 // vyslani znaku po USART2
void Usart2String(char *txt);           // vyslani retezce po USART2
int Usart2Recv(void);                   // prijem po USART2
bool IsUsart2Recv(void);                // priznak, ze je neco v bufferu
void Usart2Init(int baudSpeed);         // nastaveni komunikace po USART2


void TIM3_IRQHandler(void)              // obsluha preruseni od TIM3
{
  TIM3->SR &= ~TIM_SR_UIF;              // shozeni proznaku preruseni

  static uint16_t pom[8];
```

```
62
63  ADC1->CR |= ADC_CR_JADSTART;          // spusteni injected prevodu
64
65  while(!(ADC1->ISR & ADC_ISR_JEOS))    // pockej na konec prevodu
66    ;
67
68  if(seq == 0)
69  {
70  // pom[0] = ADC1->JDR1;               // na teto pozici je pila
71    pom[0] = pila << 4;
72    pom[1] = ADC1->JDR2;
73    pom[2] = ADC1->JDR3;
74    pom[3] = ADC1->JDR4;
75
76    DAC2->DHR12R1 = pom[0];             // DAC2/1 prevod
77    seq++;
78  }
79
80  else   if(seq == 1)
81  {
82  // pom[4] = ADC1->JDR1;               // na teto pozici je pila
83    pom[4] = pila << 4;
84    pom[5] = ADC1->JDR2;
85    pom[6] = ADC1->JDR3;
86    pom[7] = ADC1->JDR4;
87
88    DAC2->DHR12R1 = pom[4];             // DAC2/1 prevod
89    seq++;
90  }
91
92  else // if(seq == 2)
93  {
94    // x[8] = ADC1->JDR1;               // na teto pozici je pila
95    x[8] = pila << 4;
96    x[9] = ADC1->JDR2;
97    x[10]= ADC1->JDR3;
98    x[11]= ADC1->JDR4;
99
100   DAC2->DHR12R1 = x[8];               // DAC2/1 prevod
101
102   int i;
103   for(i = 0; i < 8; i++)
104   {
105     x[i] = pom[i];
106   }
107   seq = 0;
108   vysilej = true;   // vsechny potrebne hodnoty jsou zmereny, muzes vysilat
109 }
110
111 pila++;
112}
113
114int main(void)
115{
116 PLLInit();
117 uint32_t tm = 0;
118 bool ledOn = false;
119
120
121 SystemCoreClockUpdate();    // aktualizuje globalni promennou SystemCoreClock
122
123 SysTick_Config(SystemCoreClock / 64000);   // nastav 100 us (64M / 8 / 8000)
```

```
124
125   Nucleo_SetPinGPIO(LED_BOARD, ioPortOutputPP);
126
127   EXTI0Init();
128   Usart2Init(460800);
129   SPI1Init();
130   RFM02Init();
131   ADC1Init();
132   DAC2Init();
133   TIM3Init();
134
135   printf("\r\n\r\nStart APP\r\n");
136
137   while (1)
138   {
139     if(_ticks >= tm)
140     {
141       tm = _ticks + (ledOn ? 500 : 9500);   // 50 ms sviti a 950 ms nesviti = perioda 1 s
142
143       GPIOWrite(LED_BOARD, ledOn);
144       ledOn = !ledOn;
145     }
146
147     if(vysilej)
148     {
149       vysilej = false;
150
151       int i;
152       for(i = 0; i < 12; i++)
153       {
154         message[i] = x[i];
155       }
156
157 /*
158       printf("%03X;%03X;%03X;%03X;\t%03X;%03X;%03X;%03X;\t%03X;%03X;%03X;%03X\n\r",
159         message[0], message[1], message[2], message[3],
160         message[4], message[5], message[6], message[7],
161         message[8], message[9], message[10],message[11]);
162 */
163
164       message[0] |= 0xA << 12;
165       message[4] |= 0xA << 12;
166       message[8] |= 0xA << 12;
167
168       RFMSendMessage(message);            // vyslani po RFM02
169     }
170
171   }
172   return 0;
173 }
174
175 void SysTick_Handler(void)                // obsluha preruseni od SysTicku
176 {
177   _ticks++;                               // priznak dopocitani neni nutne nulovat
178 }
179
180 void EXTI0_IRQHandler(void)               // externi preruseni
181 {
182   if((EXTI->PR & EXTI_PR_PR0) != 0  )
183   {
184     nIRQsignal = true;
185   }
```

```
186   EXTI->PR |= EXTI_PR_PR0;                // shozeni priznaku
187 }
188
189 bool PLLInit(void)
190 {
191   uint32_t t;
192
193   if(!(RCC->CR & RCC_CR_HSION))           // HSI not running ?
194   {
195     RCC->CR |= RCC_CR_HSION;              // enable
196
197     t = 100;
198     while(!(RCC->CR & RCC_CR_HSION) && t) // cekej na ON
199       t--;
200     if(!t)
201       return false;                       // chybovy priznak
202   }
203
204   if(RCC->CR & RCC_CR_PLLON)              // bezi PLL ?
205   {
206     RCC->CR &= ~RCC_CR_PLLON;             // zakazani PLL
207   }
208
209   RCC->CFGR &= ~RCC_CFGR_SW;              // nastav HSI jako zdroj HCLK
210
211   RCC->CFGR = 0;                          // RESET vseho
212
213   RCC->CFGR |= RCC_CFGR_PPRE1_DIV2;       // PRE1 = 100 = /2 pro APB1 (32 MHz)
214   RCC->CFGR |= RCC_CFGR_PLLSRC_HSI_DIV2;  // HSI / 2 jako zdroj pro PLL
215   RCC->CFGR |= RCC_CFGR_PLLMUL16;         // PLLMUL = 1111 = nastav nasobeni x16
216
217   RCC->CR |= RCC_CR_PLLON;                // spust PLL
218
219   t = 100;
220   while(!(RCC->CR & RCC_CR_PLLRDY) && t)  // cekej na ON
221     t--;
222
223   if (!t)
224     return false;                         // chybovy priznak
225
226   // Nastavení waitState... pokud dám rychlejší frekvenci než zvládá flashka vydávat
227   FLASH->ACR &= ~FLASH_ACR_LATENCY;       // vymazani LATENCY
228   FLASH->ACR |= FLASH_ACR_LATENCY_1;      // 2 cekaci cykly (48 < HCLK < 72 MHz)
229
230   RCC->CFGR |= RCC_CFGR_SW_PLL;           // nastav PLL jako zdroj HCLK
231
232   t = 100;
233   while(!((RCC->CFGR & RCC_CFGR_SWS) == RCC_CFGR_SWS_PLL) && t)   // cekej na overeni SWS
234     t--;
235
236   if (!t)
237     return false;
238
239   return true;
240 }
241
242 void EXTI0Init(void)
243 {
244   Nucleo_SetPinGPIO(GPIOF, 0, ioPortInputPU);   // EXTI0 pin
245
246   if(!(RCC->APB2ENR & RCC_APB2ENR_SYSCFGEN))    // povoluje hodiny pro EXTI
247   {
```

```
248     RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN;
249     RCC->APB2RSTR |= RCC_APB2RSTR_SYSCFGRST;
250     RCC->APB2RSTR &= ~RCC_APB2RSTR_SYSCFGRST;
251   }
252
253   NVIC_EnableIRQ(EXTI0_IRQn);                    // povoleni NVICu pro preruseni EXTI0
254   NVIC_SetPriority(EXTI0_IRQn, 2);               // priorita 2
255
256   SYSCFG->EXTICR[0] &= ~SYSCFG_EXTICR1_EXTI0;    // zakazany vsechny preruseni na Line0
257   SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI0_PF; // povlen Port F na Line0 (PF0)
258
259   // EXTI->RTSR |= EXTI_RTSR_TR0;                // nabezna hrana povolena
260   EXTI->FTSR |= EXTI_RTSR_TR0;                   // dobezna hrana zakazana
261   EXTI->IMR |= EXTI_IMR_MR0;                     // prerusni povoleno
262 }
263
264 void SPI1Init(void)                              // manualni rezim SPI1
265 {
266   Nucleo_SetPinGPIO(SDI, ioPortOutputPP);        // SPI1_MOSI
267   Nucleo_SetPinGPIO(SCK, ioPortOutputPP);        // SPI1_CLK
268   Nucleo_SetPinGPIO(nSEL, ioPortOutputPP);       // SPI1_CS
269
270   GPIOWrite(SDI, false);
271   // GPIOWrite(SDO, false);
272   GPIOWrite(SCK, false);
273   GPIOWrite(nSEL, true);
274 }
275
276 void WriteSPI(uint16_t val) // drevni zpusob, ale postacujici
277 {
278   GPIOWrite(nSEL, 0);       // nizka uroven na nSEL (aktivni uroven) (t_MIN = 10 ns)
279
280   int i;
281   for(i = 0; i < 16; i++)
282   {
283     GPIOWrite(SDI, (val & 0x8000) != 0);    // maska 1000 0000 0000 0000
284
285     GPIOWrite(SCK, 1);                      // pulz na SCK (hodiny)
286     GPIOWrite(SCK, 1);                      // 2 takty (t_MIN = 25 ns)
287     GPIOWrite(SCK, 0);
288     GPIOWrite(SCK, 0);
289
290     val <<= 1;
291   }
292
293   GPIOWrite(nSEL, 1);   // vysoka uroven na nSEL (neaktivni uroven) (t_MIN = 10 ns)
294 }
295
296 void RFM02Init(void)
297 {
298   WriteSPI(0xCC00);
299   WriteSPI(0x8B61);     // 433 band, +/- 90 kHz
300   WriteSPI(0xA640);     // 434 MHz
301   WriteSPI(0xC805);     // 57.6 kbps
302   WriteSPI(0xD200);     // 57.6 kbps
303   WriteSPI(0xC220);     // en bit sync
304   WriteSPI(0xC001);     // close all
305
306
307   Nucleo_SetPinGPIO(FSK, ioPortOutputPP);
308   GPIOWrite(FSK, false);
309 }
```

```
310
311 void RFMSendByte(uint8_t aByte)         // BLOKUJICI FUNKCE !!!
312 {
313   int i = 0;
314   for(i = 0; i < 8; i++)
315   {
316     while(!nIRQsignal)                  // cekani na nIRQ
317       ;
318
319     nIRQsignal = false;                 // shodit priznak nIRQ
320     GPIOWrite(FSK, (aByte & 0x80) != 0);// maska 1000 0000
321     aByte <<= 1;                        // bitovy posun
322   }
323 }
324
325 void RFMSendMessage(uint16_t *message)
326 {
327   uint16_t ChkSum = 0;                  // kontrolni soucet
328
329   WriteSPI(0xC038);                     // start Tx po FSK pinu
330
331   uint32_t j;
332   for(j = 0; j < 1500; j++)
333     ;
334
335   RFMSendByte(0xAA);                    // Preamble
336   RFMSendByte(0xAA);                    // Preamble
337   RFMSendByte(0xAA);                    // Preamble
338
339   RFMSendByte(0x2D);                    // Synchron pattern
340   RFMSendByte(0xD4);                    // Synchron pattern
341
342   int i;
343   for(i = 0; i < 12; i++)
344   {
345     RFMSendByte(message[i] / 256);
346     RFMSendByte(message[i] % 256);
347     ChkSum += message[i];
348   }
349
350   RFMSendByte(ChkSum / 256);
351
352   RFMSendByte(0xAA);                    // konec odesilani
353
354   WriteSPI(0xC001);                     // konec Tx po FSK pinu
355 }
356
357 void ADC1Init(void)
358 {
359   Nucleo_SetPinGPIO(GPIOA, 0, ioPortAnalog);    // ADC1/1
360   Nucleo_SetPinGPIO(GPIOA, 1, ioPortAnalog);    // ADC1/2
361   Nucleo_SetPinGPIO(GPIOA, 3, ioPortAnalog);    // ADC1/4
362   Nucleo_SetPinGPIO(GPIOB, 0, ioPortAnalog);    // ADC1/11
363
364   if(!(RCC->AHBENR & RCC_AHBENR_ADC12EN))
365   {
366     RCC->AHBENR |= RCC_AHBENR_ADC12EN;
367     RCC->AHBRSTR |= RCC_AHBRSTR_ADC12RST;
368     RCC->AHBRSTR &= ~RCC_AHBRSTR_ADC12RST;
369   }
370
371   ADC1->CFGR = 0;           // RES = 00 = 12 bit, ALIGN = 0 = right align
```

```
372
373   ADC1->SMPR1 = 0x4120;     // 100 na SMP1, SMP2 a SMP4
374   ADC1->SMPR2 = 4 << 3;     // 100 = 19.5 clock cycles on SMP11
375
376   ADC1->JSQR |= ADC_JSQR_JL_1 | ADC_JSQR_JL_0                          // JL = 11 = 4 kanaly
377               | ADC_JSQR_JSQ1_0                                        // JSQ1 = ADC1/1
378               | ADC_JSQR_JSQ2_1                                        // JSQ2 = ADC1/2
379               | ADC_JSQR_JSQ3_2                                        // JSQ3 = ADC1/4
380               | ADC_JSQR_JSQ4_3 | ADC_JSQR_JSQ4_1 |ADC_JSQR_JSQ4_0;   // JSQ4 = ADC1/11
381
382   ADC1_2_COMMON->CCR |= ADC12_CCR_CKMODE_1;   // CLK = HCLK/2 (prescaler), neni teplotni
    senzor, neni Vbat
383
384   ADC1->CR |= ADC_CR_ADEN;    // povol ADC
385 }
386
387 void DAC1Init(void)
388 {
389   Nucleo_SetPinGPIO(GPIOA, 4, ioPortAnalog);    // DAC1/1
390
391   if(!(RCC->APB1ENR & RCC_APB1ENR_DAC1EN))
392   {
393     RCC->APB1ENR |= RCC_APB1ENR_DAC1EN;
394     RCC->APB1RSTR |= RCC_APB1RSTR_DAC1RST;
395     RCC->APB1RSTR &= ~RCC_APB1RSTR_DAC1RST;
396   }
397
398   DAC1->CR = 0;               // clear all
399   DAC1->CR |= DAC_CR_BOFF1;   // = OUTEN1 = DAC ch1 output switch enabled
400   DAC1->CR |= DAC_CR_EN1;     // DAC1/1 en.
401 }
402
403 void DAC2Init(void)
404 {
405   Nucleo_SetPinGPIO(GPIOA, 6, ioPortAnalog);    // DAC2/1
406
407   if(!(RCC->APB1ENR & RCC_APB1ENR_DAC2EN))
408   {
409     RCC->APB1ENR |= RCC_APB1ENR_DAC2EN;
410     RCC->APB1RSTR |= RCC_APB1RSTR_DAC2RST;
411     RCC->APB1RSTR &= ~RCC_APB1RSTR_DAC2RST;
412   }
413
414   DAC2->CR = 0;               // clear all
415   DAC2->CR |= DAC_CR_BOFF1;   // = OUTEN1 = DAC ch1 output switch enabled
416   DAC2->CR |= DAC_CR_EN1;     // DAC2/1 en.
417 }
418
419 void TIM3Init(void)
420 {
421   if(!(RCC->APB1ENR & RCC_APB1ENR_TIM3EN))    // TIM3 nutno povolit hodiny ?
422   {
423     RCC->APB1ENR |= RCC_APB1ENR_TIM3EN;
424     RCC->APB1RSTR |= RCC_APB1RSTR_TIM3RST;
425     RCC->APB1RSTR &= ~RCC_APB1RSTR_TIM3RST;
426   }
427
428   TIM3->CR1 = TIM_CR1_DIR;                  // DIR = 1 - count-down
429   TIM3->CR2 = 0;                            // nic spec.
430   TIM3->PSC = 64;                           // 1 us
431   TIM3->ARR = 1850 - 1;                     // 1.85 ms
432
```

```
433   TIM3->DIER |= TIM_DIER_UIE;                 // povol vznik preruseni z periferie
434   NVIC_EnableIRQ(TIM3_IRQn);                  // povol zpracovani v NVICu
435
436   TIM3->CR1 |= TIM_CR1_CEN;                   // spusteni TIM3
437 }
438
439 int Usart2Send(char c)
440 {
441   while(!(USART2->ISR & USART_ISR_TXE))       // cekej dokud neni volny TDR
442     ;
443   USART2->TDR = c;                            // zapis do Transmit Data Registru k odeslani
444   return c;
445 }
446
447 void Usart2String(char *txt)
448 {
449   while(*txt)
450   {
451     Usart2Send(*txt);
452     txt++;
453   }
454 }
455
456 int Usart2Recv(void)
457 {
458   while(!(USART2->ISR & USART_ISR_RXNE))      // cekej dokud neco neprijde do RDR
459     ;
460   return USART2->RDR;                         // vycti a vrat jako hodnotu
461 }
462
463 bool IsUsart2Recv(void)                       // priznak, ze je neco v bufferu
464 {
465   return (USART2->ISR & USART_ISR_RXNE) != 0; // podminka vynuti true / false
466 }
467
468 void Usart2Init(int baudSpeed)
469 {
470   Nucleo_SetPinGPIO(GPIOA, 2, ioPortAlternatePP);
471   Nucleo_SetPinAFGPIO(GPIOA, 2, 7);           // AF7 je USART2
472   Nucleo_SetPinGPIO(GPIOA, 15, ioPortAlternatePP);
473   Nucleo_SetPinAFGPIO(GPIOA, 15, 7);          // AF7 je USART2
474
475
476   if(!(RCC->APB1ENR & RCC_APB1ENR_USART2EN))  // neni povolen USART2
477   {
478     RCC->APB1ENR |= RCC_APB1ENR_USART2EN;
479     RCC->APB1RSTR |= RCC_APB1RSTR_USART2RST;
480     RCC->APB1RSTR &= ~RCC_APB1RSTR_USART2RST;
481   }
482
483   USART2->CR1 = USART_CR1_RE | USART_CR1_TE;  // staci povoleni prijmu a vysilani
484   USART2->CR2 = 0;                            // nic specialniho
485   USART2->CR3 = 0;                            // nic specialniho
486
487   // TODO doplnit vypocet BRR podle pozadovaneho Baud-rate a podle clocku
488   // POZOR USART2 je pripojen na APB1 sbernici => pro tak procesoru 64MHz je 32MHz
489   USART2->BRR = (4 << 4)    // USARTDIV = 4.340277778 pro 64MHz a 460800Bd, 8.680555556 pro
    64MHz a 230400Bd, 13.02083333 pro 8MHz a 38400Bd
490                 | 5;        // USARTDIV = fCLK / (baud x 8 x (2 - OVER8))
491                             // mantisa = 4 (posunuta o 4 bity vlevo)
492                             // fraction = 5 (zaoktouhleno na 5/16)
493
```

```
494   USART2->CR1 |= USART_CR1_UE;    // povolen blok USARTu
495
496   // zrusit bufferovani vystupu i vstupu
497   setvbuf(stdout, NULL, _IONBF, 0);
498   setvbuf(stdin, NULL, _IONBF, 0);
499 }
500
```

D. Příloha – Programový kód pro přijímací část

```
/*****************************************************************************
 * Zapadoceska univerzita v Plzni
 * Fakulta elektrotechnicka
 * Katedra aplikovane elektroniky a telekomunikaci
 *
 * Bakalarska prace
 *
 * Bezdratovy system prenosu dat
 *
 * Jan Rada
 *
 * 2020
 *
 * Prijimac RFM01-433D
 *****************************************************************************/

#include "nucleo_core.h"
#include <stdio.h>
// #include <syscalls.c>// upravene nektere funkce pro vypis v terminalu pomoci usart2

#define SDI   GPIOB,5   // MISO
#define SDO   GPIOB,4   // MOSI
#define SCK   GPIOF,1   // CLK
#define nSEL  GPIOA,8   // nSEL (active low)
#define nIRQ  GPIOF,0   // nIRQ (active low)

volatile uint32_t _ticks = 0;         // pocitadlo stovek mikrosekund
volatile bool nIRQsignal = false;     // priznak ze se muze vysilat
volatile bool rec = false;            // priznak ze prisla nova data

uint16_t message[13];
uint8_t seq = 0;

void SysTick_Handler(void);           // obsluha preruseni od SysTicku
void EXTI0_IRQHandler(void);          // externi preruseni
bool PLLInit(void);                   // nastaveni taktu procesoru na 64 MHz
void EXTI0Init(void);                 // nastaveni externiho preruseni EXTI0
void SPI1Init(void);                  // nastaveni manualniho rezimu SPI1
void WriteSPI(uint16_t val);          // drevni zpusob komunikace po SPI1, ale postacujici
void RFM01Init(void);                 // nastaveni prijimace RFM01
uint8_t RFM01ReadFIFO(void);          // prijmi Byte z RFM01
void DAC1Init(void);                  // nastavi DAC1/1
void DAC2Init(void);                  // nastavi DAC2/1
void TIM3Init(void);                  // nastaveni casovace TIM3
void TIM3_IRQHandler(void);           // obsluha preruseni od TIM3
int Usart2Send(char c);               // vyslani znaku po USART2
void Usart2String(char *txt);         // vyslani retezce po USART2
int Usart2Recv(void);                 // prijem po USART2
bool IsUsart2Recv(void);              // priznak, ze je neco v bufferu
void Usart2Init(int baudSpeed);       // nastaveni komunikace po USART2

void TIM3_IRQHandler(void)            // obsluha preruseni od TIM3
{
  TIM3->SR &= ~TIM_SR_UIF;            // shozeni proznaku preruseni

  if(rec)
  {
    seq = 0;
    rec = false;
  }
  static uint16_t pom[12];

```

```
63  if(seq == 0)
64  {
65
66    int i;
67    for(i = 0; i < 12; i++)
68      pom[i] = message[i];
69
70    DAC2->DHR12R1 = pom[0];
71    DAC1->DHR12R1 = pom[1];
72    DAC1->DHR12R2 = pom[2];
73  }
74
75  else if(seq == 1)
76  {
77    DAC2->DHR12R1 = pom[4];
78    DAC1->DHR12R1 = pom[5];
79    DAC1->DHR12R2 = pom[6];
80  }
81
82  else if(seq == 2)
83  {
84    DAC2->DHR12R1 = pom[8];
85    DAC1->DHR12R1 = pom[9];
86    DAC1->DHR12R2 = pom[10];
87    TIM3->CR1 &= ~TIM_CR1_CEN;              // zastaveni TIM3
88  }
89
90  seq++;
91}
92
93int main(void)
94{
95  PLLInit();
96
97  uint32_t k;
98  for(k = 0; k < 10000; k++)
99    ;
100
101 uint32_t tm = 0;
102 bool ledOn = false;
103 uint8_t i = 0;
104 uint16_t ChkSum = 0;
105 uint8_t recChkSum = 0;
106 uint16_t x[13];
107
108 SystemCoreClockUpdate();    // aktualizuje globalni promennou SystemCoreClock
109
110 SysTick_Config(SystemCoreClock / 64000);   // nastav 100 us (64M / 8 / 8000)
111
112 Nucleo_SetPinGPIO(LED_BOARD, ioPortOutputPP);
113
114 Usart2Init(460800);
115 SPI1Init();
116 RFM01Init();
117 DAC1Init();
118 DAC2Init();
119 TIM3Init();
120
121 printf("\r\n\r\nStart APP\r\n");
122
123 while (1)
124 {
```

```
125
126     if(_ticks >= tm)
127     {
128       tm = _ticks + (ledOn ? 500 : 9500);   // 50 ms sviti a 950 ms nesviti = perioda 1 s
129
130       GPIOWrite(LED_BOARD, ledOn);
131       ledOn = !ledOn;
132     }
133
134     while(!(GPIORead(nIRQ)))
135     {
136       if(!(i%2))
137       {
138         x[i/2] = RFM01ReadFIFO() << 8;  // horni Byte
139         if((i == 0) || (i == 8) || (i == 16))
140         {
141           if((x[i/2] & ~0x0FFF) != 0xA000)
142           {
143             WriteSPI(0xCE88);           // reset FIFO
144             WriteSPI(0xCE8B);
145
146             printf("ERROR\t1\n\r");
147             break;
148           }
149           else
150           {
151             x[i/2] &= 0x0FFF;           // maskovani A
152           }
153         }
154         i++;
155       }
156       else
157       {
158         x[i/2] |= RFM01ReadFIFO();      // spodni Byte , MASKOVAT!
159         i++;
160       }
161       if (i == 24)
162       {
163         recChkSum = RFM01ReadFIFO();
164         i = 0;
165
166         WriteSPI(0xCE88);               // reset FIFO
167         WriteSPI(0xCE8B);
168
169         ChkSum = 0;
170
171         int j;
172         for(j = 0; j < 12; j++)         // kopirovani pole
173         {
174           message[j] = x[j];
175           ChkSum += x[j];
176         }
177
178         ChkSum += 0xE000;               // kompenzace 3x 0xA000
179         ChkSum = ChkSum / 256;          // jen horni Byte
180
181         if(recChkSum == ChkSum)
182         {
183           rec = true;
184           TIM3_IRQHandler();
185           TIM3->CR1 |= TIM_CR1_CEN;     // spusteni TIM3
186
```

```
187           printf("%03X;%03X;%03X;%03X;\t%03X;%03X;%03X;%03X;\t%03X;%03X;%03X;%03X\n\r",
188             message[0], message[1], message[2], message[3],
189             message[4], message[5], message[6], message[7],
190             message[8], message[9], message[10],message[11]);
191
192         }
193
194         else
195         {
196           printf("ERROR\t2\n\r");
197         }
198
199       }
200
201     }
202
203   }
204   return 0;
205 }
206
207 void SysTick_Handler(void)    // obsluha preruseni od SysTicku
208 {
209   _ticks++;                   // priznak dopocitani neni nutne nulovat
210 }
211
212 void EXTI0_IRQHandler(void)   // externi preruseni
213 {
214   if((EXTI->PR & EXTI_PR_PR0) != 0  )
215   {
216     nIRQsignal = true;
217   }
218   EXTI->PR |= EXTI_PR_PR0;    // shozeni priznaku
219 }
220
221 bool PLLInit(void)
222 {
223   uint32_t t;
224
225   if(!(RCC->CR & RCC_CR_HSION))           // HSI not running ?
226   {
227     RCC->CR |= RCC_CR_HSION;              // enable
228
229     t = 100;
230     while(!(RCC->CR & RCC_CR_HSION) && t) // cekej na ON
231       t--;
232     if(!t)
233       return false;                       // chybovy priznak
234   }
235
236   if(RCC->CR & RCC_CR_PLLON)              // bezi PLL ?
237   {
238     RCC->CR &= ~RCC_CR_PLLON;             // zakazani PLL
239   }
240
241   RCC->CFGR &= ~RCC_CFGR_SW;              // nastav HSI jako zdroj HCLK
242
243   RCC->CFGR = 0;                          // RESET vseho
244
245   RCC->CFGR |= RCC_CFGR_PPRE1_DIV2;       // PRE1 = 100 = /2 pro APB1 (32 MHz)
246   RCC->CFGR |= RCC_CFGR_PLLSRC_HSI_DIV2;  // HSI / 2 jako zdroj pro PLL
247   RCC->CFGR |= RCC_CFGR_PLLMUL16;         // PLLMUL = 1111 = nastav nasobeni x16
248
```

```
249   RCC->CR |= RCC_CR_PLLON;                // spust PLL
250
251   t = 100;
252   while(!(RCC->CR & RCC_CR_PLLRDY) && t)   // cekej na ON
253     t--;
254
255   if (!t)
256     return false;                          // chybovy priznak
257
258   // Nastavení waitState... pokud dám rychlejší frekvenci než zvládá flashka vydávat
259   FLASH->ACR &= ~FLASH_ACR_LATENCY;         // vymazani LATENCY
260   FLASH->ACR |= FLASH_ACR_LATENCY_1;        // 2 cekaci cykly (48 < HCLK < 72 MHz)
261
262   RCC->CFGR |= RCC_CFGR_SW_PLL;             // nastav PLL jako zdroj HCLK
263
264   t = 100;
265   while(!((RCC->CFGR & RCC_CFGR_SWS) == RCC_CFGR_SWS_PLL) && t)   // cekej na overeni SWS
266     t--;
267
268   if (!t)
269     return false;
270
271   return true;
272 }
273
274 void EXTI0Init(void)
275 {
276   Nucleo_SetPinGPIO(GPIOF, 0, ioPortInputPU);   // EXTI0, pozdeji prepnout na Float
277
278   if(!(RCC->APB2ENR & RCC_APB2ENR_SYSCFGEN))    // povoluje hodiny pro EXTI
279   {
280     RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN;
281     RCC->APB2RSTR |= RCC_APB2RSTR_SYSCFGRST;
282     RCC->APB2RSTR &= ~RCC_APB2RSTR_SYSCFGRST;
283   }
284
285   NVIC_EnableIRQ(EXTI0_IRQn);                   // povoleni NVICu pro preruseni EXTI0
286
287   SYSCFG->EXTICR[0] &= ~SYSCFG_EXTICR1_EXTI0;   // zakazany vsechny preruseni na Line0
288   SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI0_PF; // povlen Port F na Line0 (PF0)
289
290   // EXTI->RTSR |= EXTI_RTSR_TR0;               // nabezna hrana povolena
291   EXTI->FTSR |= EXTI_RTSR_TR0;                  // dobezna hrana zakazana
292   EXTI->IMR |= EXTI_IMR_MR0;                    // prerusni povoleno
293 }
294
295 void SPI1Init(void)                             // manualni rezim SPI1
296 {
297   Nucleo_SetPinGPIO(SDI, ioPortOutputPP);       // SPI1_MOSI
298   Nucleo_SetPinGPIO(SDO, ioPortInputPU);        // SPI1_MISO
299   Nucleo_SetPinGPIO(SCK, ioPortOutputPP);       // SPI1_CLK
300   Nucleo_SetPinGPIO(nSEL, ioPortOutputPP);
301
302   GPIOWrite(SDI, false);
303   GPIOWrite(SCK, false);
304   GPIOWrite(nSEL, true);
305 }
306
307 void WriteSPI(uint16_t val)                   // drevni zpusob, ale postacujici
308 {
309   GPIOWrite(nSEL, 0);     // nizka uroven na nSEL (aktivni uroven) (t_MIN = 10 ns)
310
```

```
311   int i;
312   for(i = 0; i < 16; i++)
313   {
314     GPIOWrite(SDI, (val & 0x8000) != 0);    // maska 1000 0000 0000 0000
315
316     GPIOWrite(SCK, 1);                      // pulz na SCK (hodiny)
317     GPIOWrite(SCK, 1);                      // 2 takty (t_MIN = 25 ns)
318     GPIOWrite(SCK, 0);
319     GPIOWrite(SCK, 0);
320
321     val <<= 1;
322   }
323
324   GPIOWrite(nSEL, 1);   // vysoka uroven na nSEL (neaktivni uroven) (t_MIN = 10 ns)
325   GPIOWrite(nSEL, 1);   // vysoka uroven na nSEL (neaktivni uroven) (t_MIN = 10 ns)
326 }
327
328 void RFM01Init(void)
329 {
330   WriteSPI(0x0000);
331   WriteSPI(0x898A);     // 433 band, 134 kHz
332   WriteSPI(0xA640);     // 433 MHz
333   WriteSPI(0xC805);     // 57.6 kbps
334   WriteSPI(0xC69B);     // AFC setting
335   WriteSPI(0xC42A);     // Clock recovery manual control, Digital filter DQD=4
336   WriteSPI(0xC240);     // output 1.66 MHz
337   WriteSPI(0xC080);
338   WriteSPI(0xCE88);     // use FIFO
339   WriteSPI(0xCE8B);
340   WriteSPI(0xC081);     // open Rx
341
342   Nucleo_SetPinGPIO(SDO, ioPortInputPU);
343   Nucleo_SetPinGPIO(nIRQ, ioPortInputPU);
344 }
345
346 uint8_t RFM01ReadFIFO(void)
347 {
348   uint8_t i, Result;
349   GPIOWrite(SCK, false);
350   GPIOWrite(SDI, false);
351   GPIOWrite(nSEL, false);
352
353   for(i = 0; i < 16; i++)   // preskoc status bity
354   {
355     GPIOWrite(SCK, true);
356     GPIOWrite(SCK, true);
357     GPIOWrite(SCK, false);
358     GPIOWrite(SCK, false);
359   }
360
361   Result = 0;
362   for(i = 0; i < 8; i++)
363   {
364     Result <<= 1;
365     if(GPIORead(SDO))
366     {
367       Result |= 1;
368     }
369     GPIOWrite(SCK, true);
370     GPIOWrite(SCK, true);
371     GPIOWrite(SCK, false);
372     GPIOWrite(SCK, false);
```

```
373   }
374
375   GPIOWrite(nSEL, true);
376   return Result;
377 }
378
379 void DAC1Init(void)
380 {
381   Nucleo_SetPinGPIO(GPIOA, 4, ioPortAnalog);    // DAC1/1
382   Nucleo_SetPinGPIO(GPIOA, 5, ioPortAnalog);    // DAC1/2
383
384   if(!(RCC->APB1ENR & RCC_APB1ENR_DAC1EN))
385   {
386     RCC->APB1ENR |= RCC_APB1ENR_DAC1EN;
387     RCC->APB1RSTR |= RCC_APB1RSTR_DAC1RST;
388     RCC->APB1RSTR &= ~RCC_APB1RSTR_DAC1RST;
389   }
390
391   DAC1->CR = 0;               // clear all
392   DAC1->CR |= DAC_CR_BOFF1;   // output buffer enabled
393   DAC1->CR |= DAC_CR_BOFF2;   // output buffer enabled
394   DAC1->CR |= DAC_CR_EN1;     // DAC1/1 en.
395   DAC1->CR |= DAC_CR_EN2;     // DAC1/2 en.
396 }
397
398 void DAC2Init(void)
399 {
400   Nucleo_SetPinGPIO(GPIOA, 6, ioPortAnalog);    // DAC2/1
401
402   if(!(RCC->APB1ENR & RCC_APB1ENR_DAC2EN))
403   {
404     RCC->APB1ENR |= RCC_APB1ENR_DAC2EN;
405     RCC->APB1RSTR |= RCC_APB1RSTR_DAC2RST;
406     RCC->APB1RSTR &= ~RCC_APB1RSTR_DAC2RST;
407   }
408
409   DAC2->CR = 0;               // clear all
410   DAC2->CR |= DAC_CR_BOFF1;   // = OUTEN1 = DAC ch1 output switch enabled
411   DAC2->CR |= DAC_CR_EN1;     // DAC2/1 en.
412 }
413
414 void TIM3Init(void)
415 {
416   if(!(RCC->APB1ENR & RCC_APB1ENR_TIM3EN))    // TIM3 nutno povolit hodiny ?
417   {
418     RCC->APB1ENR |= RCC_APB1ENR_TIM3EN;
419     RCC->APB1RSTR |= RCC_APB1RSTR_TIM3RST;
420     RCC->APB1RSTR &= ~RCC_APB1RSTR_TIM3RST;
421   }
422
423   TIM3->CR1 = TIM_CR1_DIR;                // DIR = 1 - count-down
424   TIM3->CR2 = 0;                          // nic spec.
425   TIM3->PSC = 64;                         // 1 us
426   TIM3->ARR = 1850 - 1;                   // 1.85 ms
427
428   TIM3->DIER |= TIM_DIER_UIE;             // povol vznik preruseni z periferie
429   NVIC_EnableIRQ(TIM3_IRQn);              // povol zpracovani v NVICu
430
431   TIM3->CR1 |= TIM_CR1_CEN;               // spusteni TIM3
432 }
433
434 int Usart2Send(char c)
```

```
435 {
436   while(!(USART2->ISR & USART_ISR_TXE))   // cekej dokud neni volny TDR
437     ;
438   USART2->TDR = c;    // zapis do Transmit Data Registru k odeslani
439   return c;
440 }
441
442 void Usart2String(char *txt)
443 {
444   while(*txt)
445   {
446     Usart2Send(*txt);
447     txt++;
448   }
449 }
450
451 int Usart2Recv(void)
452 {
453   while(!(USART2->ISR & USART_ISR_RXNE))      // cekej dokud neco neprijde do RDR
454     ;
455   return USART2->RDR;                         // vycti a vrat jako hodnotu
456 }
457
458 bool IsUsart2Recv(void)                       // priznak, ze je neco v bufferu
459 {
460   return (USART2->ISR & USART_ISR_RXNE) != 0; // podminka vynuti true / false
461 }
462
463 void Usart2Init(int baudSpeed)
464 {
465   Nucleo_SetPinGPIO(GPIOA, 2, ioPortAlternatePP);
466   Nucleo_SetPinAFGPIO(GPIOA, 2, 7);           // AF7 je USART2
467   Nucleo_SetPinGPIO(GPIOA, 15, ioPortAlternatePP);
468   Nucleo_SetPinAFGPIO(GPIOA, 15, 7);          // AF7 je USART2
469
470
471   if(!(RCC->APB1ENR & RCC_APB1ENR_USART2EN))  // neni povolen USART2
472   {
473     RCC->APB1ENR |= RCC_APB1ENR_USART2EN;
474     RCC->APB1RSTR |= RCC_APB1RSTR_USART2RST;
475     RCC->APB1RSTR &= ~RCC_APB1RSTR_USART2RST;
476   }
477
478   USART2->CR1 = USART_CR1_RE | USART_CR1_TE;  // staci povoleni prijmu a vysilani
479   USART2->CR2 = 0;    // nic specialniho
480   USART2->CR3 = 0;    // nic specialniho
481
482   // TODO doplnit vypocet BRR podle pozadovaneho Baud-rate a podle clocku
483   // POZOR USART2 je pripojen na APB1 sbernici => pro tak procesoru 64MHz je 32MHz
484   USART2->BRR = (4 << 4)    // USARTDIV = 4.340277778 pro 64MHz a 460800Bd, 8.680555556 pro
    64MHz a 230400Bd, 13.02083333 pro 8MHz a 38400Bd
485                 | 5;        // USARTDIV = fCLK / (baud x 8 x (2 - OVER8))
486                             // mantisa = 4 (posunuta o 4 bity vlevo)
487                             // fraction = 5 (zaoktouhleno na 5/16)
488
489   USART2->CR1 |= USART_CR1_UE;    // povolen blok USARTu
490
491   // zrusit bufferovani vystupu i vstupu
492   setvbuf(stdout, NULL, _IONBF, 0);
493   setvbuf(stdin, NULL, _IONBF, 0);
494 }
495
```