**ZÁPADOČESKÁ UNIVERZITA V PLZNI**
**FAKULTA ELEKTROTECHNICKÁ**

**Katedra aplikované elektroniky a telekomunikací**

# BAKALÁŘSKÁ PRÁCE

**Pavel Mühlbacher** **2019**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
**Fakulta elektrotechnická**
Akademický rok: **2018/2019**

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

| | |
|---|---|
| Jméno a příjmení: | **Pavel MÜHLBACHER** |
| Osobní číslo: | **E15B0301P** |
| Studijní program: | **B2612 Elektrotechnika a informatika** |
| Studijní obor: | **Elektronika a telekomunikace** |
| Název tématu: | **Přenos dat pomocí Bluetooth** |
| Zadávající katedra: | **Katedra aplikované elektroniky a telekomunikací** |

Zásady pro vypracování:

Návrh a realizace zařízení umožňující sériový přenos dat mezi mikrokontrolérem a osobním počítačem na bázi Bluetooth.

1. Popište možnosti bezdrátového přenosu dat mezi periferním zařízením a osobním počítačem.
2. Rozeberte princip a základní vlastností přenosu dat pomocí Bluetooth.
3. Navrhněte systém pro přenos dat mezi mikrokontrolérem a osobním počítačem na bázi systému Bluetooth.
4. Realizujte přenosový systém a ověřte jeho vlastnosti.

Rozsah grafických prací: **podle doporučení vedoucího**

Rozsah kvalifikační práce: **30 - 40 stran**

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

Seznam odborné literatury:


1. **http://www.egr.msu.edu/classes/ece480/capstone/spring14/group01/docs/appno SendingAndReceivingDataViaBluetoothWithAnAndroidDevice.pdf**
2. **http://epubl.ltu.se/1402-1617/2004/088/LTU-EX-04088-SE.pdf**
3. **http://www.ijarcsse.com/docs/papers/Volume_3/3_March2013/V3I3-0170.pdf**


Vedoucí bakalářské práce: **Prof. Ing. Milan Štork, CSc.**
Katedra aplikované elektroniky a telekomunikací

Datum zadání bakalářské práce: **5. října 2018**

Termín odevzdání bakalářské práce: **13. června 2019**

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

L.S.

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 5. října 2018

## Abstrakt

Tato práce je zaměřena na bezdrátový přenos dat. Jako zdroj dat byl vybrán akcelerometr resp. gyroskop. Jsou zde prakticky realizovány přenosy dat pomocí rádiových modulů NRF24L01+ a jejich různé konfigurace jak pro maximální rychlost přenosu, tak po maximální dosah resp. maximální spolehlivost přenosu dat. Je zde také realizován přenos pomocí Bluetooth modulu HC-05. Jedna kapitola je věnována jak lze tento modul programovat. Jako mikrokontrolér bylo zvoleno Arduino Nano. Je zde rozebrán princip sběrnic IIC a SPI.



## Klíčová slova

Bluetooth modul HC-05,bezdrátový přenos dat, moduly NRF24L01+, Arduino Nano, Bluetooth,Wi-fi, SPI, IIC, RN42

## Abstract

This work is focused on wireless data transmission. As a data source, the accelerometer or gyroscope was chosen. There are practically realized data transmissions by means of radio modules NRF24L01 + and their various configurations for both the maximum transmission speed and the maximum range respectively. maximum data transfer reliability. There is also a transmission via the HC-05 Bluetooth module. One chapter is devoted to how to program this module. Arduino Nano was chosen as a microcontroller. The principle of IIC and SPI buses is discussed.



## Keywords

Bluetooth module HC-05, wireless data transfer, modules NRF24L01+, Arduino Nano, Bluetooth, Wi-fi, RN42

# 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, je legální.

.....

Podpis

V Plzni dne 13.6.2019 Pavel Mühlbacher

## Poděkování

Rád bych poděkoval Prof. Ing. Milanu Štorkovi, CSc. za věnovaný čas, trpělivost a cenné rady při psaní této bakalářské práce.

# Obsah

# Seznam obrázků

# 1. Úvod

Tato práce je zaměřena na bezdrátový přenos dat, je zde vyzkoušen Bluetooth modul HC-05 přímo spojený s počítačem, tím je realizován přenos mikrokontrolér-počítač. Dále jsou zde různé konfigurace pro moduly NRF24L01+, které komunikují mezi sebou a realizují tedy přenos mikrokontrolér-mikrokontorlér. Jako mikrokontrolér byla zvolena deska Arduino Nano, která má mikroprocesor ATmega328. Hlavní důraz v této práci je vedený na moduly NRF24L01+. Dále byly vyzkoušeny moduly RN42 od firmy Microchip. Jako vstupní periferie byl zvolen akcelerometr a gyroskop MPU-6050 a mezi výstupní byl zvolen display LCD1602. Jsou zde popsány možnosti bezdrátového přenosu dat mezi počítačem a mikrokontolérem. Je zde popsán princip SPI sběrnice, pomocí které komunikují moduly NRF24L01+ . Také princip I2C sběrnice pomocí které komunikuje akcelerometr resp. gyroskop a LCD display.

# 2. Možnosti bezdrátového přenosu mezi počítačem a periferním zařízením

Tato kapitola je zaměřena na možnosti přenosu dat mezi periferním zařízením a počítačem. V úvahu připadají 3 možnosti přenosu dat Bluetooth, Wifi a infraport (u starších PC) tyto možnosti přenosu má většina PC integrované. Infraport zkráceně IrDa z anglického slova *Infrared Data Asssociation* vysílá a přijímá modulované infračervené světlo o vlnové délce 875nm. Vysílačem mohou být infračervené Led diody nebo laserové diody, jako přijímače se využívají fotodiody. Infraport se již tolik nepoužívá z důvodu nutnosti přímé viditelnosti obou zařízení, a krátkého dosahu.

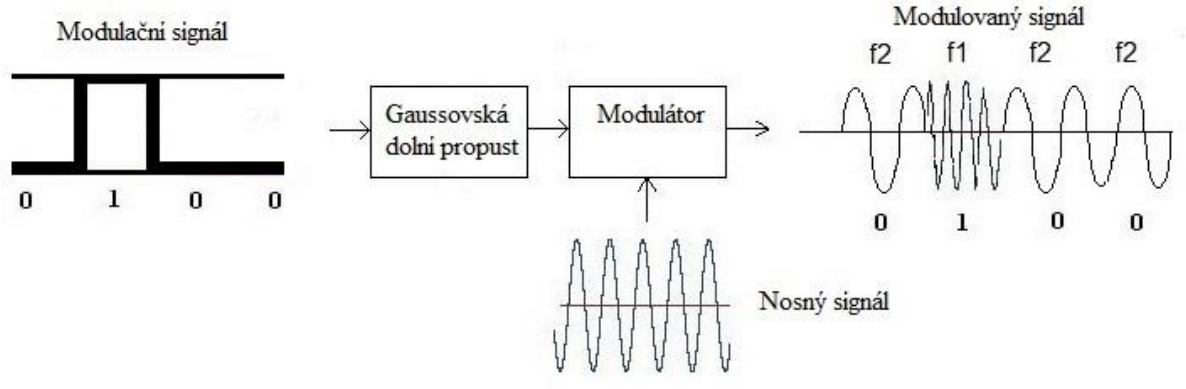
## 2.1 Bluetooth

Bluetooth slouží pro bezdrátovou komunikaci dvou a více zařízení, byl vyvinut jako náhrada za kabelové rozhraní RS-232 a je definován standardem IEEE 802.15.1. Pracuje v nelicencovaném pásmu 2,4000GHz až 2,4835 GHz. Není nutné žádat o přidělení frekvenčního pásma a případně platit poplatky za jeho vyžívání. Hlavním účelem Bluetooth je přenos dat. Pro přenos dat je nutné modulovat signál. Modulace má dva vstupní signály nosný signál a modulační signál. Výstupem je pak modulovaný signál.

Modulace, které využívá technologie Bluetooth jsou GFSK, $\pi$ /4DQPSK a 8DPSK. Bluetooth lze rozdělit do 3 tříd dle výstupního výkonu viz tabulka číslo 1. FHSS je metoda

pro přenos v rozprostřeném spektru. Princip spočívá v přeskakování nosné vlny. V Evropě se využívá šířka 80MHz. Celkem je tedy v pásmu definováno 79 kmitočtů s šířkou 1MHz. (1)

Modulace GFSK (Graussin Frequency Shift Keying) se řadí mezi digitální modulace. Nosný signál je signál s harmonickým průběhem a modulačním signálem je digitální signál viz obrázek číslo 1. V modulaci GFSK na rozdíl od frekvenční modulace FSK jsou impulzy základního pásma, obsahující 0 a 1 před vstupem do modulátoru procházeny gaussovskou dolní propustí. Původní pravoúhlé změny impulzů jsou tak omezením vysokých kmitočtů zaobleny. Filtrace způsobí potlačení nežádoucích postranních složek spektra modulovaného GFSK signálu. (2)




*Obrázek 1 - Průběh GFSK modulace*

| Verze Bluetooth | Teoretické maximální rychlosti přenosu dat |
|---|---|
| Verze 1.2 | 1 Mbit/s |
| Verze 2.0+ EDR | 3 Mbit/s |
| Verze 3.0 + HS | 24 Mbit/s |
| Verze 4.0 | 24 Mbit/s |
| Verze 5.0 | 255Mbit/s |

*Tabulka 1 - Rychlosti jednotlivých verzí Bluetooth*

| Třídy výkonu | Třída 1 | Třída 2 | Třída 3 |
|---|---|---|---|
| Dosah[m] | 100 | 10 | 1 |
| Výkon[mW] | 100 | 2,5 | 1 |
| Šířka kanálu [MHz] | 1 | | |
| Počet kanálů | 79 | | |
| Frekvenční rozsah [MHz] | 2400 – 2483,5 ± 75kHz | | |
| Modulace | GFSK, π/4-DQPSK,8DPSK | | |
| Metoda přenosu | FHSS | | |

*Tabulka 2 - Rozdělení Bluetooth do tříd*

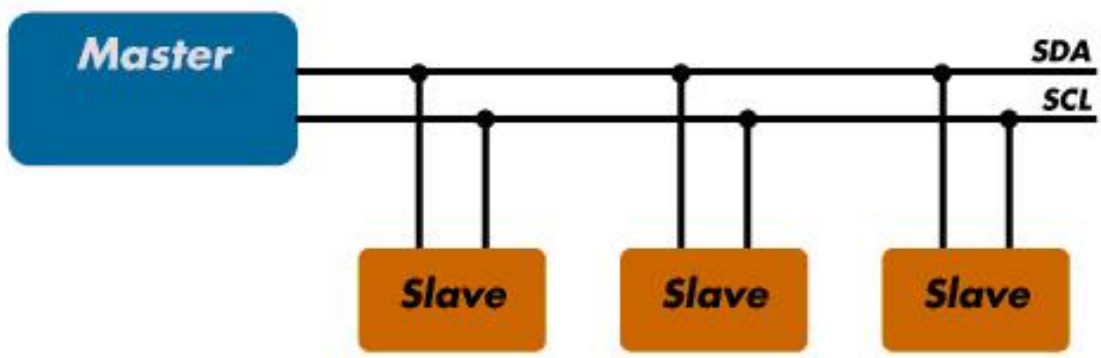
## 2.2 Wi-fi

Wi-fi je definována standartem 802.11. Nejčastěji se vyskytuje Wi-fi pracující v pásmu 2,4GHz, takže stejné frekvenční pásmo jako Bluetooth, ale také lze najít Wifi pracující 5 GHz. Hlavním účelem Wi-fi je připojení mobilu, počítače či jiného zařízení k internetu. K připojení k internetu se musí nastavit IP adresa, maska pod sítě, DNS (*Domain name server*) a bránu. DNS sever slouží k překladu doménové adresy na IP adresu. To znamená, že pokud do vyhledávače napíšeme například www.seznam.cz DNS sever to přeloží na IP adresu 77.75.79.53. Zařízení má danou ještě MAC adresu ta je dána výrobcem a je unikátní. Například pokud máme na počítači jak Wi-fi tak kabelové rozhraní Ethernet každé má svojí MAC adresu. MAC adresa je 48 bitová z toho vyplívá, že lze dosáhnout více jak 281 biliard různých adres. IP adresu, maska podsítě a bránu lze nastavit ručně nebo nechat nastavit automaticky to bývá nejčastěji. Automaticky znamená, že zařízení po připojení si zažádá o tyto adresy z DHCP (*Dynamic Host Configuration Protocol*) serveru. IP adresa se nejčastěji vyskytuje verze IPv4, ale postupně je nahrazována IPv6. IPv4 má velikost 4 bajty to znamená 4,2 miliardy adres. IPv6 umožňuje $2^{128}$ kombinací různých adres.

# 3. Sběrnice IIC

IIC sběrnice využívá topologii master/slave. To znamená, že jedno zařízení se chová jako hlavní nebo-li master a řídí komunikaci. Zařízení typu slave, které se řídí pokyny zařízení master. V topologii nedochází ke kolizím, protože master zařízení určuje, které zařízení bude

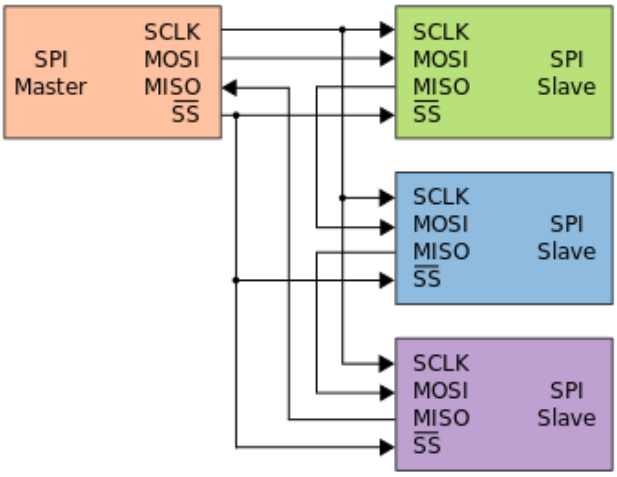
v danou chvíli vysílat data. Pro IIC komunikaci se používají pouze dva signálové vodiče. První SDA ten slouží pro přenos dat oběma směry. Druhý SCL, můžeme najít i označení SCK, přes tento vodič vysíláme hodinový signál. IIC je synchronní komunikace. Každé zařízení musí mít svojí jedinečnou adresu. Adresa se vysílá po datovém vodiči. IIC umožňuje komunikaci rychlostí až 5MHz. (5)



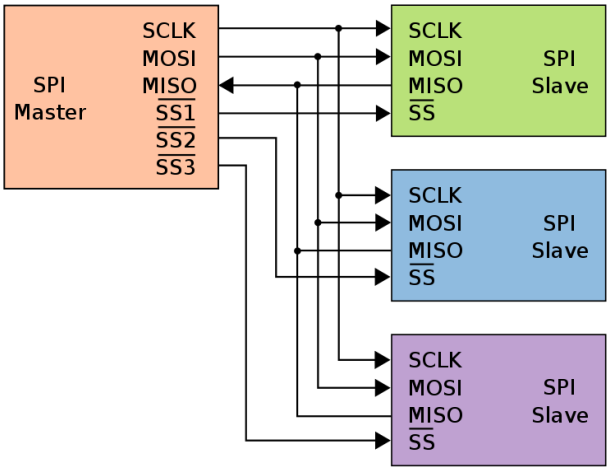

*Obrázek 2- Schéma zapojení IIC sběrnice*

## 4. Sběrnice SPI

Sběrnice SPI (*Serial Peripheral Interface*) se používá k synchronní komunikaci na krátké vzdálenosti, také využívá topologii Mater/Slave Podporuje komunikaci FULL duplex, to znamená, že zařízení může zároveň, přijímat i vysílat data až do rychlosti 10MHz. Ke komunikaci se využívají 4 vodiče: SCLK – pomocí tohoto vodiče vysílá zařízení master hodinový signál pro zařízením slave, MOSI – slouží k vysílání dat ze zařízení master do zařízení slave, MISO k přenosu dat, ze zařízení typu slave do zařízení typu master, SS slouží k výběru konkrétního zařízení. Zařízení slave nemohou komunikovat mezi sebou. Rychlost komunikace řídí master. Přijetí zpráv se nepotvrzuje. Zařízení typu slave lze připojit i na jeden SS kanál viz. obrázek číslo 3. Při poslání zprávy poslednímu zařízení slave musí zpráva projít všemi zařízením i typu slave. Při výpadku prvního zařízení slave dojde k nefunkčnosti celé topologie, při zapojení přes jeden SS kanál. (5)




*Obrázek 3- Zapojení SPI přes jeden SS kanál*




*Obrázek 4 - Zapojení SPI, kde každý slave má svůj SS kanál*

# 5. Přenos dat mezi počítačem a mikrokontrolérem pomocí Bluetooth modulu HC-05

Pro tyto přenosy přímého přenosu mezi počítačem a mikrokontrolérem lze uvažovat již zmíněné tří možnosti přenosu: infraport, Bluetooth a Wifi. Mezi Bluetooth moduly se řadí například moduly HC-05, či HC-06. Moduly HC-06 a HC-05 jsou téměř totožné, hlavní rozdíl je v tom, že moduly HC-06 neumějí pracovat jako master.

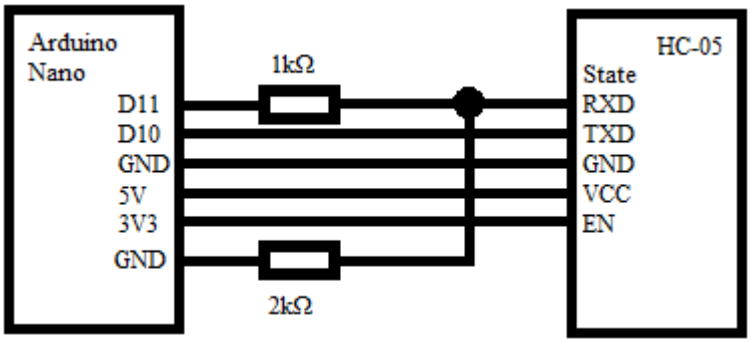
Bluetooth modul HC05 podporuje Bluetooth verze V2.0+EDR, tedy rychlost přenosu dat až 3Mbit/s, může fungovat v režimech master i slave . Modul má integrovanou anténu. Lze ho připojit k počítači, mobilu, tabletu apod. Pro připojení k mobilu je nutné mít stažený příslušný software, který umožnuje komunikaci. Vyzkoušený byl Serial Bluetooth Terminal z Google store. LED dioda na modulu HC-05 má tři režimy svícení viz tabulka číslo 3.

| Režim svícení | Popis |
|---|---|
| Dvojité rychlé zabliknutí každé 2 vteřiny | Modul je spárovaný |
| Bliká cca 5x za vteřinu | Modul je zapnutý a čeká na spárování |
| 2 vteřiny svítí, 2 vteřiny nesvítí | Modul je v AT režimu |

*Tabulka 3 - Režimy svícení Led diody Bluetooth modulu HC-05*

## 5.1 Programování pomocí AT příkazů

Programování se modulů, se provádí pomocí tzv. AT příkazů viz tabulka číslo 4 důležitých příkazů. Uvedení do programovacího režimu se provádí, přivedením pinu EN do 1 a až poté přivedení napájení. Proto při zapojování je tedy nutné zapojit nejdříve pin EN do 3,3V a až poté připojení napájení do 5V. Zda je režim programování se pozná blikáním LED diody, tak že 2 vteřiny svítí a 2 vteřiny nesvítí. Pro pin RX je nutné snížit napětí, proto byl zvolen odporový dělič, který vytvoří úbytek napět, jelikož piny mají logickou úroveň 5V. Po otevření COM portu je nutné mít nastavenou stejnou sériovou rychlost jako v programu, ale je nutné mít nastaveno zakončení řádku na ‚*Obojí NL & CR*'. Pro komunikaci je využita knihovna SoftwareSerial.h. Výchozí rychlost pro komunikaci v programovacím režimu 38400 baudů. Z programovacího AT režimu se lze dostat tak, že nejdříve odpojíme pin EN, poté zadáme příkaz AT+RESET, nejedná se o reset do továrního nastavení ale o klasický reset zařízení.




*Obrázek 5- Schéma zapojení Arduino Nano a Bluetooth modulu HC-05 pro programování pomocí AT příkazů*

| Příkaz | Odpověď | Parametry | Funkce |
|---|---|---|---|
| AT | OK | Žádné | Zkouška spojení |
| AT+ORGL | OK | Žádné | Obnovení do továrního nastavení |
| AT+ADDR? | ADDR+<parametr><br>OK | Parametr je adresa modulu například 98d3:32:11422a | Vrátí adresu Bluetooth modulu |
| AT+RESET | OK | Žádné | Reset modulu |
| AT+UART? | UART=<parametr1, parametr2, parametr3><br>OK | Parametr1 je sériová komunikační rychlost 4800 až 1 382 400<br>Parametr2= stop bity 0-1 bit, 1-bity<br>Parametr 3 paritní bit 0- bez parity, 1-lichá parita<br>2-sudá parita | Vrátí nastavenou sériovou rychlost, nastavení ohledně parity a stop bit |
| AT+UART<parametr1, parametr2, parametr3> | OK | | Změna nastavení sérové rychlost, stop bitu a parity |

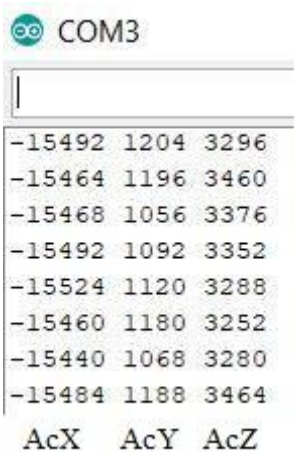
*Tabulka 4- Tabulka AT důležitých příkazů Bluetooth modulu HC-05*

## 5.2 Spárování a přenos dat z akcelerometru pomocí Bluetooth modulu HC-05

Spárování s počítačem je velmi jednoduché stačí zapnout Bluetooth a vyhledat zařízení. Výchozí název je *H-C-2010-06-01* a výchozí heslo je 1234. Zda je zařízení spárováno lze poznat indikační Led diodou tak, že dioda 2 krát rychle problikne každé 2 vteřiny. Pro spárování není nutné měnit nastavení modulu pomocí AT příkazů postačí ponechat nastavení ve výchozím stavu. Po spárování s počítačem se objeví dva COM porty jeden z nich je správný. V mém případě se objevili jako COM3 a COM4. COM3 byl správný. Pro tento přenos jsou čtena data z akcelerometru následně jsou vypisovány. Výpis je realizován pomocí *print()* resp. *println()* viz obrázek číslo 7. Schéma zapojení je dle obrázku

číslo 8. Při výpisu pomocí jsou data při čtení akcelerometru ukládány do proměnné typu integer. Při výpisu pomocí *print()*, jsou data čitelná přímo na sériovém monitoru případně plotru. Při výpisu pomocí *print()* resp *println()* byla změřena doba trvání výpisu 5,6ms plus je nutné připočítat dobu čtení akcelerometru, to bylo změřeno, že trvá 300us při hodinovém signálu 400 000. Výpisy, byly realizovány při rychlosti 38 400 baudů. Takže celá perioda T je pak 5,9ms z toho lze vypočítat frekvenci vzorkování. Při zvětšování vzdálenosti mezi zařízeními bylo zjištěno, že při vzdálenosti 10 metrů dochází ke zpomalování přenosu. Frekvenci vzorkování tedy lze uvažovat pouze na velmi krátké vzdálenosti cca do 5 metrů a při přímé viditelnosti zařízení.

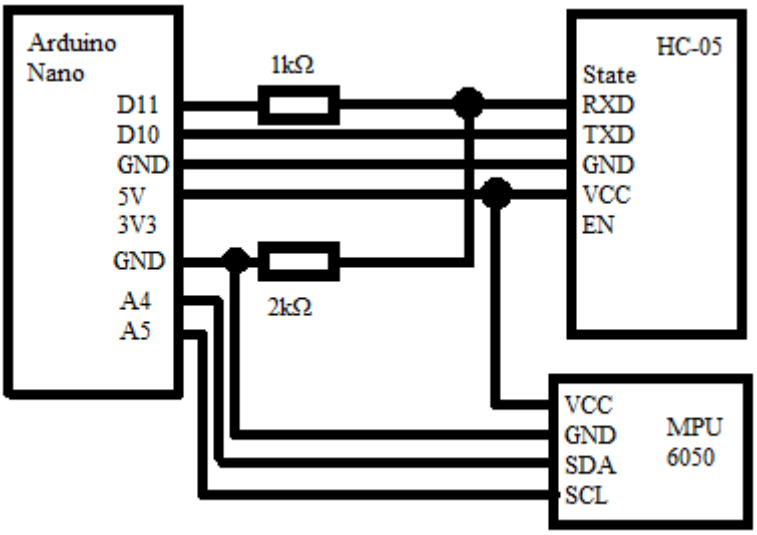
$$f = \frac{1}{T} = \frac{1}{5{,}9*10^{-3}} = 169Hz \quad (1)$$




*Obrázek 6 - Výpis na COM port*

```
BT.print(AcX);
BT.print(" ");
BT.print(AcY);
BT.print(" ");
BT.println(AcZ);
```

*Obrázek 7 - Programová realizace výpisu*




*Obrázek 8 - Schéma zapojení Bluetooth modulu HC-05 a MPU-6050 s Arduino Nano*

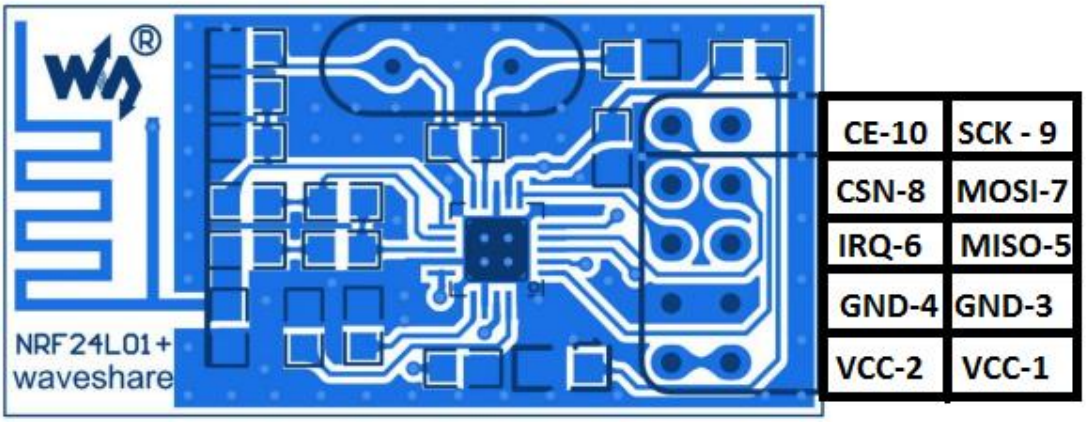
# 6. Přenos mikrokontrolér – mikrokontrolér realizovaný pomocí rádiových modulů NRF24L01+

Moduly jsou nejčastěji označovány jako Wifi moduly se zdůvodněním, že pracují v pásmu 2,4 GHz, toto označení je ale není vhodné tyto moduly nelze připojit přímo k Wi-fi routeru. Pro připojení k internetu by bylo nutné využít tzv. RFX (Radio for Duino) například Raspberry pi (3). Moduly využívají GFSK modulaci. Pracují na frekvenci 2400 – 2525 GHz záleží na zvoleném kanálu, který je uložený v registru 0x05 Šířka pásma je při rychlostech 250kbps a 1Mbps 1 MHz a při rychlosti 2Mbps 2MHz. Moduly komunikují s Arduinem pomocí SPI sběrnice a napájení je 3,3V. Pro datové piny není nutné převádět napěťové úrovně, datové piny pracují s 5V logikou. Schéma zapojení je dle obrázku číslo 10, kde v programu lze změnit připojení pinu CE a CSN při inicializaci, ostatní piny jsou pevně dány zvoleným mikrokontrolérem. Na trhu lze najít moduly NRF24L01+ a NRF24L01 hlavní rozdíl je v tom že verze NRF24L01 neumožňují přenosovou rychlost 250kbps. Deska, která byla k dispozici byla od výrobce WaweShare a lze ji zakoupit v třech různých provedeních (A, B, C) byla k dispozici verze B ta má na rozdíl od zbylých dvou desek dva piny napájecí piny a dva zemnící piny, zbytek pinů je stejný. Rozmístění pinů je znázorněno na obrázku 9. Výrobce uvádí, že rezistory a kapacitory jsou lepší kvality než u desky A a mají jiný package. Dosah uvedený výrobcem je proto desky B je 55metrů u desky A 50metrů, při nastavení 250kbps,0dBm a měření v otevřeném prostoru. Pro programování modulů je využívána knihovna RF24.h a nRF24L01.h.

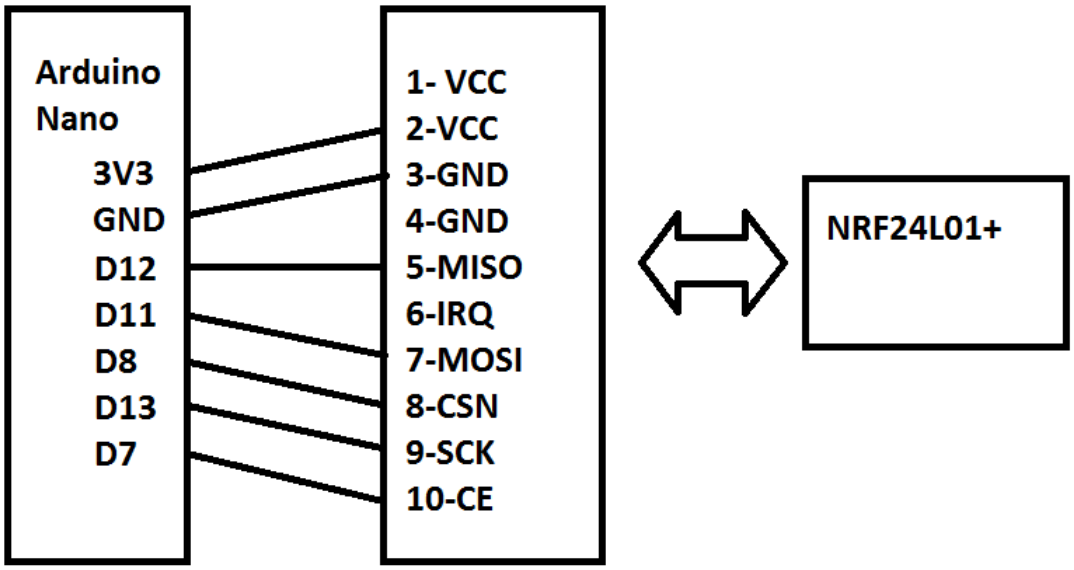
Moduly lze konfigurovat pomocí různých příkazů například *setDataRate()* – nastavení přenosové rychlosti, *setPALevel()* – nastavení vysílacího výkonu nebo konfigurace modulůNRF24L01+ byla zvolena pomocí příkazu *write_register(číslo registru, hodnota),* kde hodnota znamená dekadické číslo uložené do registru. V tabulce číslo 5 jsou popsány důležité bity jednotlivých registrů. Jelikož byla funkce *write_register()* mezi privátními funkcemi je nutné jí přesunout mezi public nebo-li veřejné funkce. Kontrolu nastavení registrů lze provést pomocí příkazu *read_register(číslo registru),* která vrátí hodnotu uloženou v registru, je také nutné tuto funkci přesunout mezi veřejné. Jelikož v knihovně se nachází mezi privátními funkcemi

| Číslo registru | Bit | Popis |
|---|---|---|
| 0x00 | 3 | Využít CRC pokud je alespoň jeden bit v registru 0x01 jedna |
| | 2 | CRC kódování 0-1 bajty 1-2 bajty |
| | 1 | Zapnout/ vypnout modul 1- zapnout 0- vypnout |
| | 0 | Nastavení režimu 1- Přijímač 0-Vysílač |
| 0x01 | 5 až 0 | Povolit potvrzení přijmutí dat na potrubí 5 až 0 |
| 0x02 | 5 až 0 | Využít datové potrubí 5 až 0 |
| 0x03 | 1 až 0 | Velikost adresního pole 00 - nedovoleno 01-3 byty 10-4 byty 11-5 bytů |
| 0x04 | 7 až 4 | Zpoždění definované od konce přenosu do začátku nového vysílaní 0000 -zpoždění 250 us 0001-500us ……. 1111-4000us |
| | 3 až 0 | Počet znovu odeslání 0000-opakovaný přenos zakázán 0001-odeslat 1x znovu ….. 1111 odeslat 15x znovu |
| 0x05 | 6 až 0 | Nastavení frekvenčního kanálu |
| 0x06 | 5 a 3 | Nastavení datové rychlosti 00-1 Mbps 01-2Mbps 10-250kbps 11-Rezervováno |
| | 2 a 1 | Nastavení vysílacího výkonu 00 - -18dBm 01 - -12dBm 10 - -6dBm 00 - 0dBm |
| 0x11 | 5 až 0 | Nastavení užitečného zatížení u přijímače u datového potrubí číslo 0 0 pipline nevyužit 1 - 1bajt ….32 - 32bajtů |

*Tabulka 5 - Důležité bity registrů a jejich funkce*

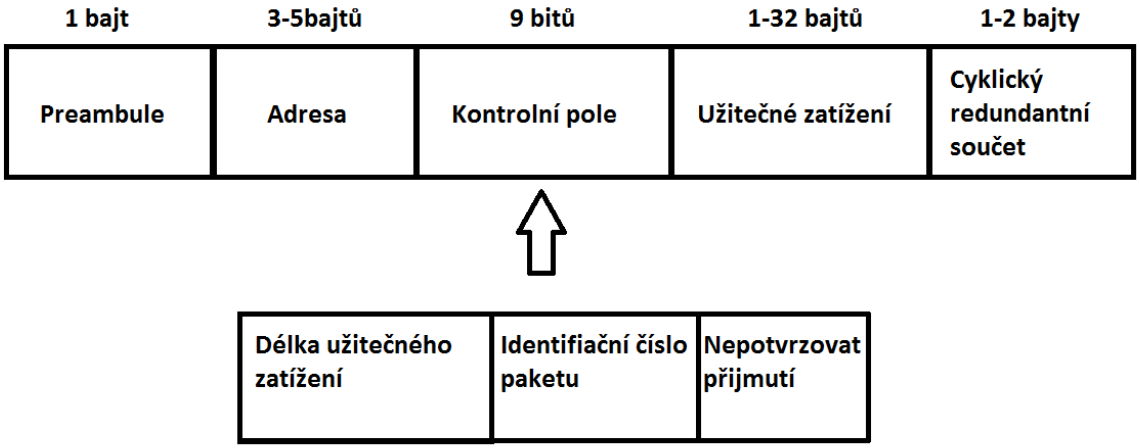



*Obrázek 9- Schéma rozmístění vývodů modulu NRF24L01+*




*Obrázek 10 - Schéma zapojení Arduino Nano a modulu NRF24L01+*

Moduly vysílají po paketech viz. obrázek číslo 11, kde paket se skládá z preambule, adresy, kontrolního pole, užitečného zatížení a cyklického redundantního součtu zkráceně CRC. Kontrolní pole obsahuje délku užitečného zatížení identifikační číslo paketu a jeden bit je vyhrazen pro to, zda se mají potvrzovat pakety nebo ne. Užitečné zatížení lze nastavit mezi 1 až 32 bajty. To záleží na množství odesílaných bajtů.




*Obrázek 11 - Schéma paketu modulu NRF24L01+*

## 4.1. Měření maximální rychlosti přenosu

Měření maximální rychlosti přenosu probíhalo při nastavení sériové rychlosti mezi počítačem a mikrokontrolérem 1 000 000 baudů. Při této rychlosti je, ale nutné využívat sériový monitor nebo plotter z Arduina prostředí, protože z jinými programy, bylo zjištěno, že to nefunguje příliš dobře. Na straně vysílače byla vytvořena 2 pole a velikosti 32 bajtů. První pole obsahovalo čísla od 100 do 131, druhé pole bylo vytvořeno číslicemi 200 až 231. Tyto čísla byla náhodně zvolena. Hlavním cílem bylo, aby se tyto dvě pole lišila a byla stejně velká. Odeslání pole je realizováno pomocí příkazu *writeFast(ukazatel na pole, délka pole),* který oproti klasickému *write(ukazatel na pole, délka pole)* nevyužívá cyklický redutantivní součet a zakazuje potvrzování paketů. Na straně příjemce bylo vytvořeno pouze jedno pole a v případě, že vysílač vyslal nové pole, původní pole se přepíše. U přijímače lze vypisovat buď celé pole, ale to bylo změřeno, že trvá 5,3ms nebo pouze číslo na zvolené pozici pole to trvá 165us,výpisy byli realizovány pomocí příkazu *println()*. Na straně vysílače je pak nutné zvolit zpoždění pomocí příkazu *delayMicroseconds()* mezi vysláním pole a odesláním nového pole. Zpoždění musí trvat nejméně dobu, kterou trvá výpis na straně přijímače nebo větší. Pro toto měření bylo zvoleno zpoždění 200us na straně vysílače po odeslání 32 bajtového pole. Zpoždění bylo nastaveno z důvodu výpisu jednoho prvku pole na straně přijímače proto, aby nedocházelo ke ztrátě dat na straně vysílače, byla změřena perioda 355us. Z toho zpoždění je

200us, takže samotný přenos celého paketu trvá 155us z toho 32 užitečných bajtů. Perioda byla změřena nastavením pinu D4 do 0 na začátku cyklu ve smyčce *loop()* a na konci smyčky nastavením D4 do 1. Poté byla připojena sonda osciloskopu k D4. Po dobu trvání 0 lze pak uvažovat periodu T. Pro maximální rychlost, bylo nastavení registrů následující dle tabulky číslo 6. Pro přijímač a vysílač je nastavení stejné, liší se pouze v nultém registru, kde se nastavuje režim a v registru 0x11 ten není nutné nastavovat pro vysílač, protože vysílač zná počet vysílaných bajtů. Rovnice číslo 2 uvádí výpočet rychlosti přenosu vzduchem. Z výpočtu vyplívá, že moduly přenášejí rychlostí 1,97Mbit/s, tato odchylka může být dána nepřesností měření.

$$\frac{Celkový\ počet\ vysílaných\ bitů}{doba\ vysílaní} \tag{2}$$

$$= \frac{1x8(preambule) + 3 * 8\ (adresa) + 9(kontrolní\ pole) + 32 * 8(užitečná\ data) + 1 * 8(CRC)}{155x10^{-6}}$$

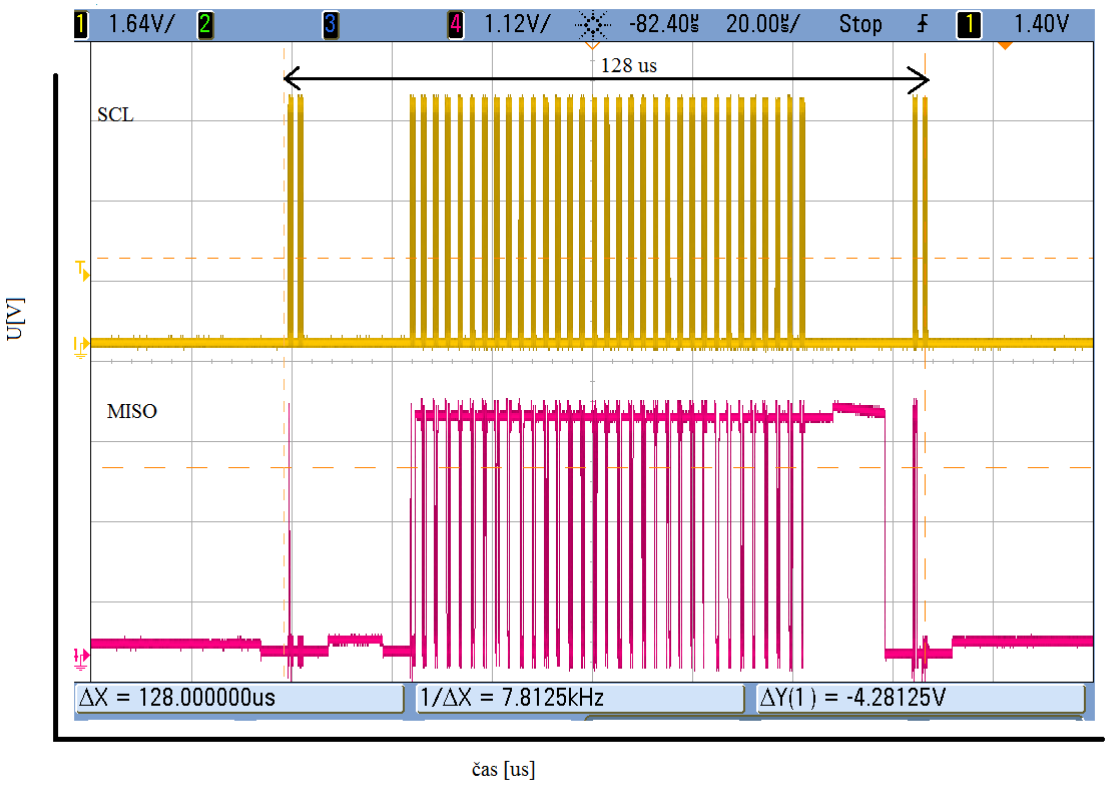
$$= \frac{305}{155x10^{-6}} = 1{,}97\text{Mbit/s}$$

| Číslo registru | Uložená hodnota dekadicky | Popis |
|---|---|---|
| 0x00 | Vysilač-2 přijímač -3 | Nevyužívat CRC, zapnout, režim vysílač/přijímač |
| 0x01 | 0 | Nepotvrzovat přijmutí paketu |
| 0x02 | 1 | Použít datový pipline 0 |
| 0x03 | 1 | Velikost adresního pole 3 bajty |
| 0x04 | 0 | Zakázat znovu odesílaní paketů |
| 0x05 | 76 | Nastavení frekvenčního kanálu (například 76) |
| 0x06 | 14 | Rychlost 2Mbps, Vysílací výkon 0dBm |
| 0x11 | 32 | Velikost užitečného zatížení 32 bajtů na piplinu 0 |

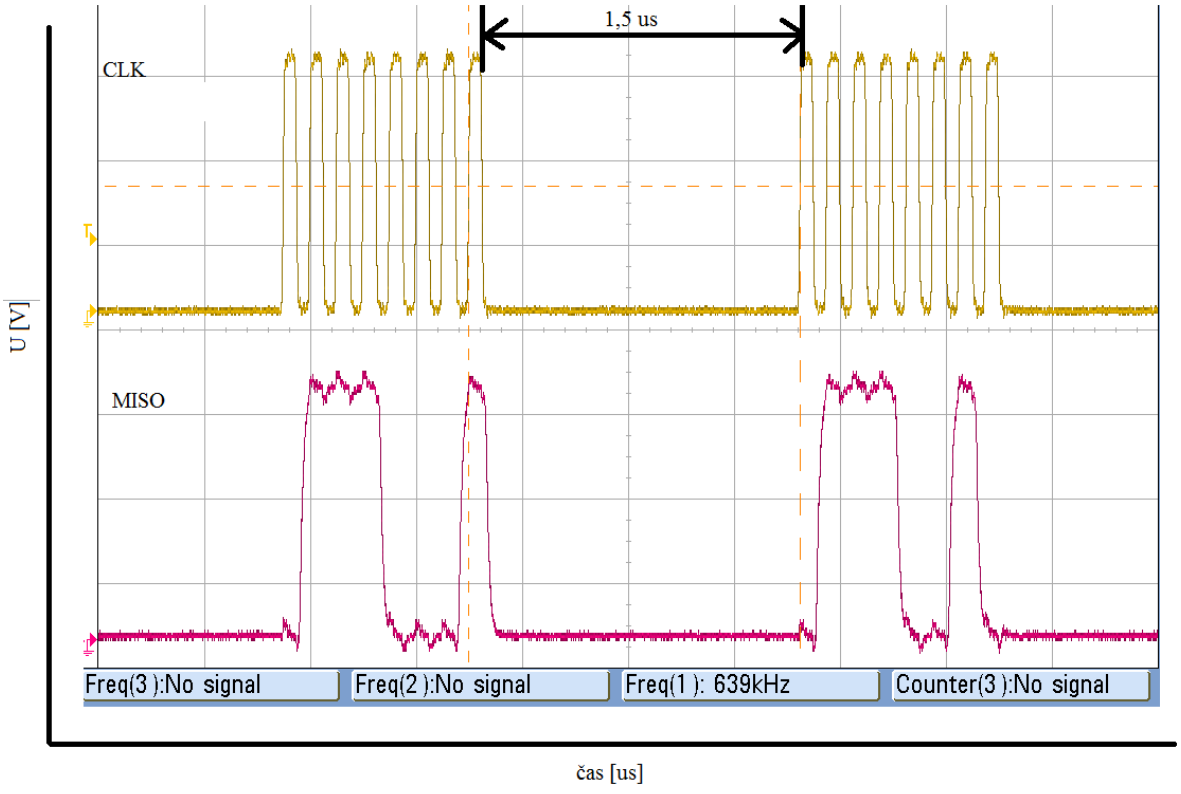
*Tabulka 6 - Nastavení registrů modulu NRF24L01+ pro měření maximální rychlosti přenosu*

Toto nastavení má výhodu, že vysílač vysílá rychle a pravidelně data, jelikož pakety nejsou potvrzovány a případně odesílány znovu. Nevýhoda nastává, že data nejsou potvrzována a při větší vzdálenosti může dojít ke ztrátě spojení a vysílač to ani nepozná.

Obrázek číslo 12 zachycuje průběh hodin a přenosu čísla 113 a následně čísla 114 měřený na pinu MISO na straně přijímače resp. D12 mikrokontroléru. Mezera mezi jednotlivými užitečnými bajty byla změřena 1,5 us. Obrázek číslo 12 zachycuje průběh jednoho paketu doba trvání paketu, byla změřena 128us.




*Obrázek 12 - Průběh signálu na MISO resp. D12*




*Obrázek 13 - Průběh užitečných bajtů čísel 113 a 114*

## 4.2 Měření maximálního dosahu

Měření dosahu probíhalo v otevřeném prostoru na kraji města, kde by se nemělo vyskytovat příliš mnoha rušení změřený dosah byl 65 metrů při nastavení dle tabulky číslo 7. Vysílána byla čísla od 0 do 255. V každém paketu byl jeden užitečný bajt. Při vzdálenosti 60 metru začal být přenos velmi zpomalený a pří vzdálenosti 65 metrů se přenos zastavil při návratu do vzdálenosti menší, jak 65 metrů se přenos obnovil. Zaručení maximální spolehlivosti přenosu dat je dáno registrem 4, kde pokud není paket potvrzen, je paket až 15 krát odeslán znovu, pokud není dříve potvrzeno přijmutí. Mezi jednotlivými pakety při znovu odeslání je pauza 4000us. Pro přenos byla zvolena minimální přenosová rychlost dle doporučení výrobce tedy 250kbps a nastavení maximálního vysílacího výkonu 0dBm.

| Číslo registru | Uložená hodnota dekadicky | Popis |
|---|---|---|
| 0x00 | Vysilač-14 přijímač - 15 | Využívat CRC, zapnout, režim vysílač/přijímač |
| 0x01 | 1 | Potvrzovat přijmutí paketu na datovém piplinu číslo 0 |
| 0x02 | 1 | Použít datový pipline číslo 0 |
| 0x03 | 1 | Velikost adresního pole 3 bajty |
| 0x04 | 255 | Posílat paket znovu pokud není dostáno potvrzení maximálně 15x zpoždění mezi jednotlivými odesláními 4000us |
| 0x05 | 76 | Nastavení frekvenčního kanálu (například 76) |
| 0x06 | 38 | Rychlost 250kbps, Vysílací výkon 0dBm |
| 0x11 | 1 | Velikost užitečného zatížení 1 bajtů na piplinu 0 |

*Tabulka 7 - Nastavení registrů pro měření maximálního dostahu*

## 4.3 Měření maximální rychlosti čteni znaků a následného přenosu

Měření maximální sériové rychlosti mezi počítačem a mikrokontrolérem při přenosu souborů o různých velikostech. Na straně vysílače byla čtena data pomocí funkce *Serial.Read()* a uložena do proměnné typu byte, proměnná byla následně vyslána. Užitečná šířka byla nastavena na 1, jinak bylo nastavení stejné jako pro maximální přenosovou rychlost. Pokud není vyslán žádný znak, vrátí funkce *Serial.Read()* -1, jelikož je ukládáno do proměnné typu byte, která má rozsah 0 až 255 dojde k podtečení proměnné, takže proměnná je pak 255. Proto byly odeslány všechna ostatní čísla, kromě 255. Soubory byly vytvořeny jako textové a začínali písmenem Z následovali znaky U binárně 0101 0101 a poslední znak byl K. Počet znaků odpovídá celkovému počtu znaků včetně Z i K na straně přijímače byly vytvořeny 2 proměnné typu unsigned long a do nich byl uložen čas v pomocí funkce micros(). Funkce *micros()* vrací čas od spuštění programu v mikrosekundách, pokud je ukládáno do proměnné typu unsigned long proměnná přeteče po 72 minutách. Do první proměnné po přijmutí znaku Z byl uložen čas. Do druhé proměnné po přijmutí znaku K. Rozdíl těchto proměnných udává čas přenosu daného počtu znaků mínus jeden znak. Důvodem je, první čas je uložen, až po přijmutí prvního znaku. Na straně přijímače byla vytvořena proměnná pro počítání přijímutích znaků., též na straně vysílače pro počet odeslaných znaků. Maximální sériová rychlost při čtení znaků je 57 600 baudů. Při vyšších rychlostech je například místo 10 000 znaků odesláno pouze 3 000 znaků. Důvodem je, že vysílač vysílá na přijímač a nestihne přečíst znaky na sériové lince s počítačem. Při tomto měřený byli moduly od sebe 1 metr. Tabulka uvádí počet znaků a čas přenosu v milisekundách. Nastavení sériové rychlosti bylo 57600 baudů. Na straně vysílače jsou znaky vypisovány pomocí příkazu *write()*. Pro toto měření byl využit program Tera Term, který umožňuje odesílání souborů.

| Počet znaků | Čas přenosu v [ms] |
|---|---|
| 9 | 1,444 |
| 99 | 16,984 |
| 499 | 86,176 |
| 999 | 172,708 |
| 2499 | 432,260 |
| 4999 | 864,884 |
| 7499 | 1297,316 |
| 9999 | 1730,044 |

*Tabulka 8 - Počet přenášených znaků a doba trvání přenosu*

## 4.4 Čtení dat z akcelerometru a gyroskopu ,následný přenos dat a výpis os akcelerometru na display

Užitečné využití může být například přenos dat z akcelerometru. Odesílány budou všechny hodnoty, tedy akcelerometr osy x, y a z, teplota a gyroskop osy x, y a z pro kontrolu je vysílán ještě jeden bajt, který se po každém odeslaní zvětší o jedna. Přenášeno bude dohromady 15 bajtů., jelikož každá osa zabírá dva bajty. Teplota je převáděna na stupně až na straně přijímače při výpisu, takže zabírá při přenosu také 2 bajty. Poslední bajt zabírá kontrolní proměnná, která nabývá hodnoty od 0 do 255. Pro přenášení dat byla definována struktura o 8 proměnných 7 krát typ integer a 1 krát typ byte. Struktura má oproti poli výhodu takovou, že všechny proměnné nemusí být stejně velké. Na straně přijímače jsou hodnoty vypisovány přes sériový plotter případně monitor a také budou vypisovány na display. Na display budou vypisovány hodnoty pouze, pokud bude kontrolní proměnná 128 nebo 255 a pouze hodnoty z akcelerometru. Z důvodu že lidské oko je velmi pomalé a výpis na display i při maximální možném kmitočtu hodin tedy 400kHz tvá mezi 230 až 260 ms to záleží na číslech, která jsou vypisovány. Ostatní hodnoty byly vypisovány pomocí vlastní funkce. Při výpisu všech hodnot byla změřena doba trvání výpisu mezi 1540–1590us. Čtení akcelerometru, gyroskopu a teploty na straně vysílače při maximálním kmitočtu tedy 400 kHz trvá 544-552 us. Na straně vysílače je tedy nutné nastavit zpoždění o velikosti rozdílu a výpisu a doby čtení akcelerometru. Na straně vysílače tedy bylo nastaveno po každém odeslání zpoždění 1,1ms a pokud byl kontrolní bajt o velikosti 128 nebo 255, bylo nastaveno zpoždění o velikosti 265ms, protože víme, že přijímač bude vypisovat na LCD obrazovku. Schéma připojení displeje je dle obrázku číslo 23. Schéma zapojení MPU6050 dle obrázku 21

## 4.5 Přenos dat z akcelerometru umístěného na testovacím vibrátoru

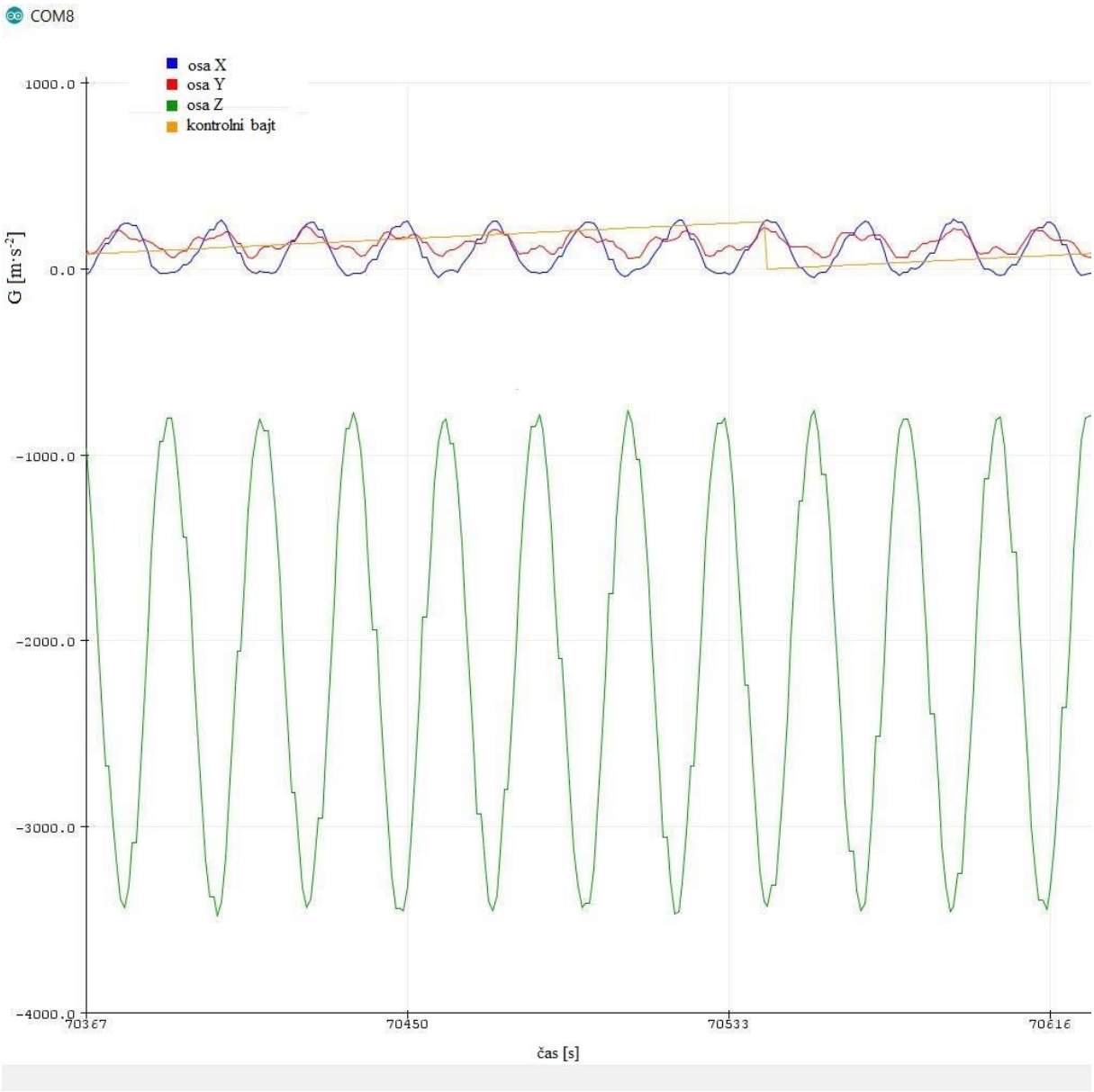
Při tomto měření byl akcelerometr umístěn na testovací vibrátor, který umožnuje po připojení generátoru vytvořit například sinusový signál o dané frekvenci. Data byla zpracována a následně vyslána. Nastavení registrů bylo stejné jako při měření maximální rychlsti přenosu. Byly vytvořeny dva programy. V prvním programu je výpis realizován pomocí print() resp println() a načtená data z akcelerometru jsou převedena do proměnné typu integer. V druhém programu je výpis realizován pomocí write();, zde jsou ponechána původní data z akcelerometru tedy 2 bajty na jednu osu. Poté při výpisu jsou přidány 2 bajty 00 a FF, z důvodu lepší orientace v souboru.

Níže uvedený obrázek zobrazuje výpis pomoci vlastní funkce Při frekvenci 50 Hz. Největší rozkmit je v ose z, protože v ose z se konaly vibrace z testovacího vibrátoru. Pokud by kontrolní byte neměl pravidelný průběh, docházelo by ke ztrátě dat. Odeslání bylo realizováno pomocí funkce *writeFast()*. Výpis byl realizován pomocí vlastní funkce a změřená doba trvání výpisu byla 688 us až 700us je nutné připočítat dobu přijmutí paketu dohromady s přijmutí paketu byla změřena doba 800us až 812us. Výpis byl realizován rychlosti 1 000 000 baudů. Proto, aby nedocházelo ke ztrátě dat tak bylo nutné nastavit na příjmači zpozdění 400us cele doba periody je tedy 832us 20us byla zvolena rezerva. Pomocí vzorce pak lze spočítat frekvenci vzorkování

$$f = \frac{1}{T} = \frac{1}{832*10^{-6}} = 1{,}20kHz \quad (3)$$

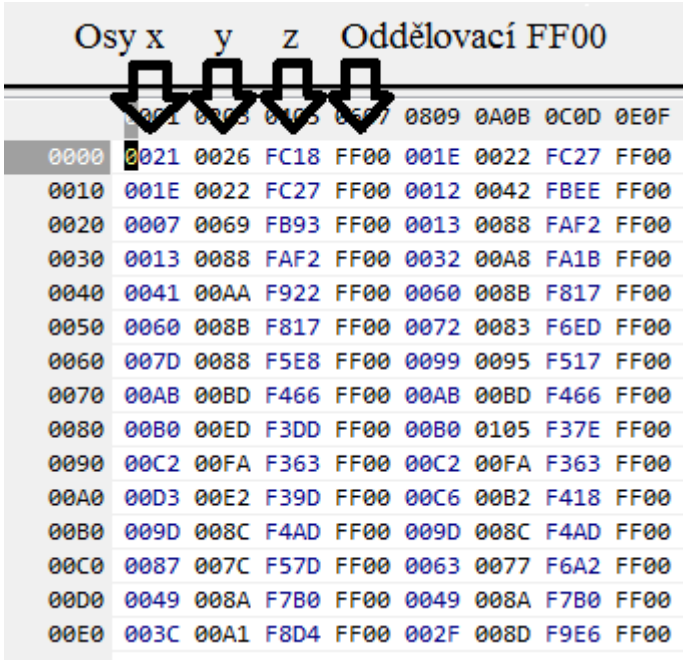
Při výpisu pomocí funkce *write()* byla využita rychlost 230 400 baudů z důvodu využití Tera Termu pro ukládání dat do souboru. Jelikož doba výpisu byla změřena 68us při výpisu 8 bajtů Není nutné pak nastavovat na straně přijímače zpoždění, jelikož doba trvání čtení dat z akcelerometru je větší. Frekvence vzorkování pak byla omezena na straně vysílače změřená doba trvání čtení dat z akcelerometru a vyslání byla 440 us. Pak lze uvažovat frekvenci vzorkování

$$f = \frac{1}{T} = \frac{1}{440*10^{-6}} = 2{,}27kHz \quad (4)$$




*Obrázek 14 - Průběh na sériovém ploteru, při výpisu pomocí print() resp. println()*

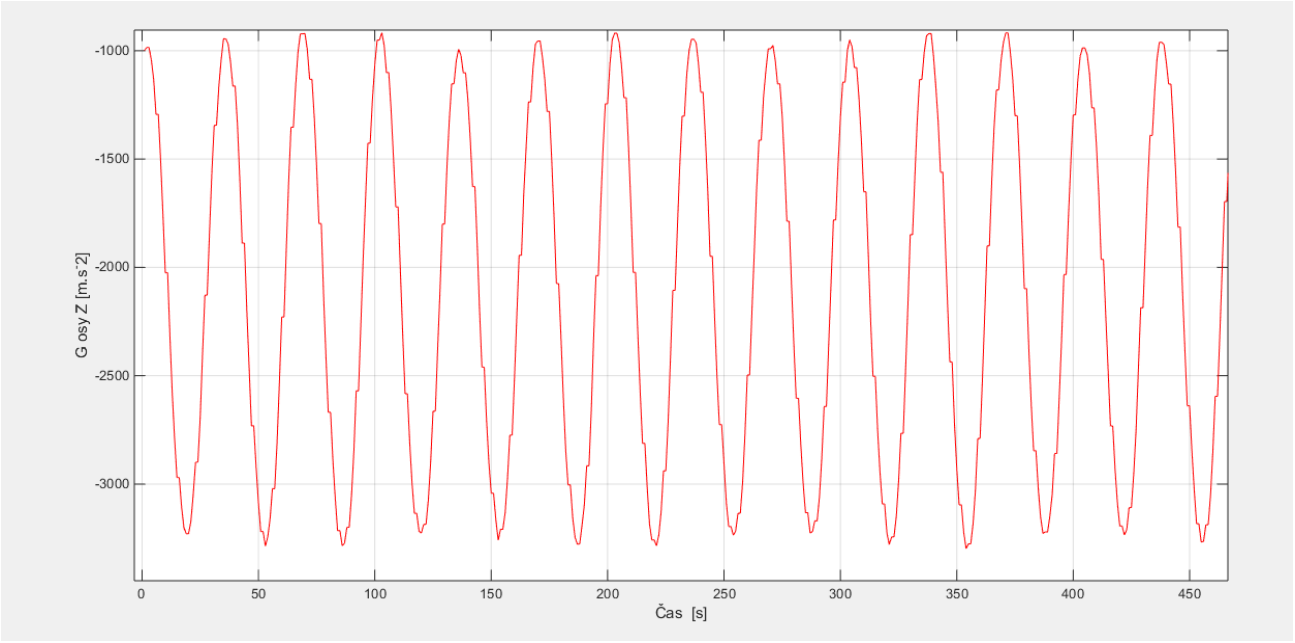
Při výpisu pomocí funkce *write()* je nutné data ukládat souboru například pomocí programu Tera Term a poté zpracovat například Matlabem. Data jsou ukládána ve formátu viz obrázek 15, kde bity jsou v pořadí od nejvyššího po nejnižší. Každá osa se skládá z 2 bajtů. Měření probíhalo při frekvenci 40 Hz. Maximální komunikační rychlost mezi mikrokontrolérem a počítačem při ukládání do souboru pomocí programu Tera Term je 230 400 baudů.




| | 0809 | 0A0B | 0C0D | 0E0F |
|---|---|---|---|---|
| 0000 | 0021 0026 FC18 FF00 | 001E | 0022 | FC27 FF00 |
| 0010 | 001E 0022 FC27 FF00 | 0012 | 0042 | FBEE FF00 |
| 0020 | 0007 0069 FB93 FF00 | 0013 | 0088 | FAF2 FF00 |
| 0030 | 0013 0088 FAF2 FF00 | 0032 | 00A8 | FA1B FF00 |
| 0040 | 0041 00AA F922 FF00 | 0060 | 008B | F817 FF00 |
| 0050 | 0060 008B F817 FF00 | 0072 | 0083 | F6ED FF00 |
| 0060 | 007D 0088 F5E8 FF00 | 0099 | 0095 | F517 FF00 |
| 0070 | 00AB 00BD F466 FF00 | 00AB | 00BD | F466 FF00 |
| 0080 | 00B0 00ED F3DD FF00 | 00B0 | 0105 | F37E FF00 |
| 0090 | 00C2 00FA F363 FF00 | 00C2 | 00FA | F363 FF00 |
| 00A0 | 00D3 00E2 F39D FF00 | 00C6 | 00B2 | F418 FF00 |
| 00B0 | 009D 008C F4AD FF00 | 009D | 008C | F4AD FF00 |
| 00C0 | 0087 007C F57D FF00 | 0063 | 0077 | F6A2 FF00 |
| 00D0 | 0049 008A F7B0 FF00 | 0049 | 008A | F7B0 FF00 |
| 00E0 | 003C 00A1 F8D4 FF00 | 002F | 008D | F9E6 FF00 |

*Obrázek 15 - Uklání bajtů pomocí funkce write()*

Níže uvedený obrázek číslo 16 uvádí průběh osy z po zpracování Matlabem. Ostatní osy byli také zaznamenány do souboru, ale hlavní pohyb se koná v ose z. Na dalším obrázku je vidět kód v Matlabu, pro dekódování souboru.




*Obrázek 16 - Průběh osy z po zpracování Matlabem*

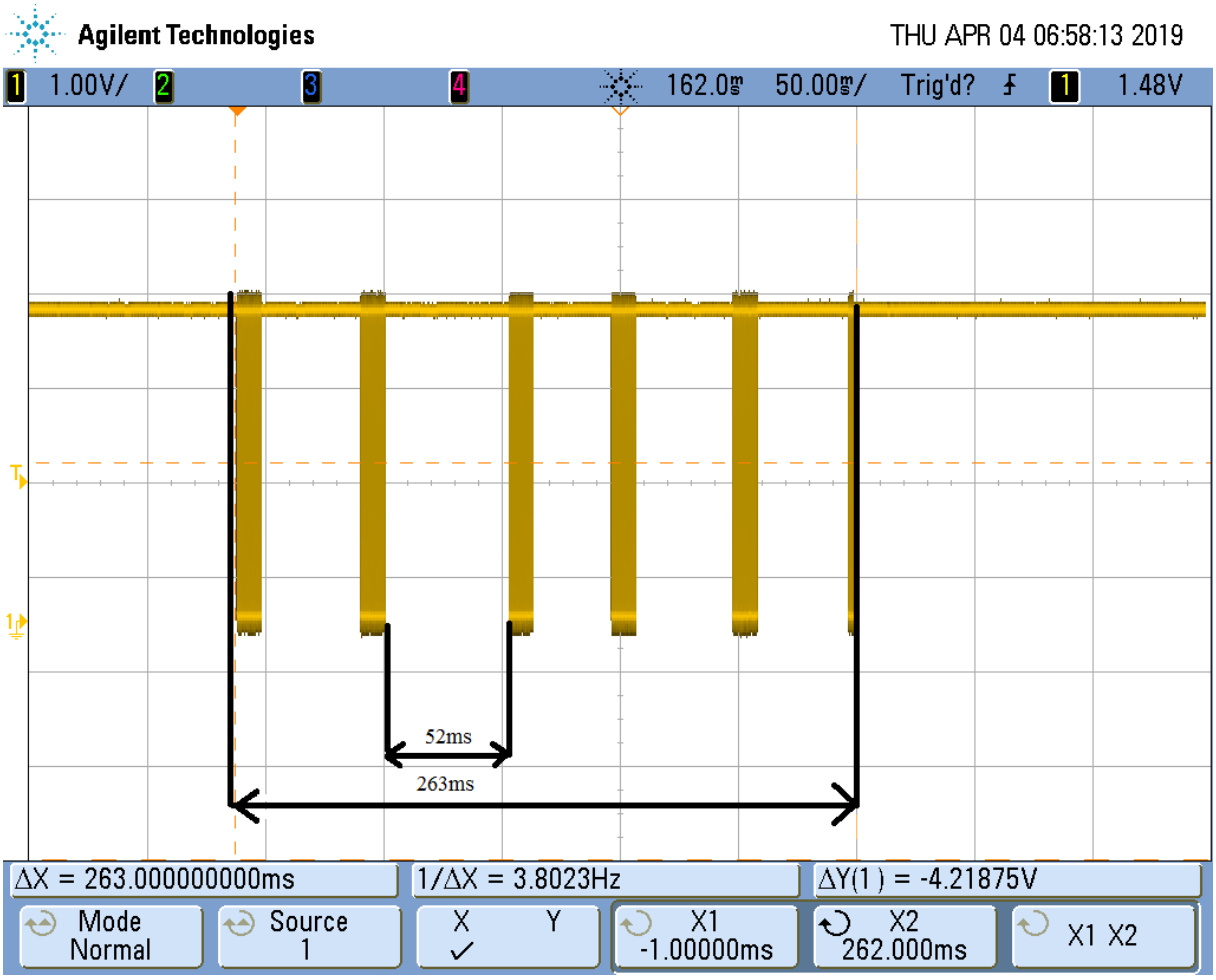
```
clear all; close all;
fid = fopen('data40Hz.txt');
xx = fread(fid,'int16','b');
fclose(fid);
N=4;  % číst každé 4 číslo
offset=3; %zacatek od 3 (osa Z)
x1= downsample(xx,N,offset);
figure;  plot(x1,'r');grid on;ylabel('G osy Z [m.s^-2]');xlabel('Čas [s]');
```

*Obrázek 17 - Kód v Matlabu pro dekódovaní souboru*

# 7. Přenos znaku pomocí Bluetooth modulů RN42

Moduly RN42 jsou od firmy Microchip a jsou typu Class 2. Využívají Bluetooth verze 2.1+ EDR Moduly se programují pomocí vysláním třech znaků dolarů za sebou bez mezery ($$$). Tyto znaky lze zaslat přes UART rozhraní, kde zařízení po připojení k počítači se objeví jako COM port.

Měření probíhalo ve škole, tedy v prostředí, kde se vyskytuje velké množství rušení.. Moduly byly naprogramovány na sériovou rychlost 921 600 baudů vysílaný znak byl U to odpovídá v ASCII tabulce dekadickému číslu 85, tedy binárně 0101 0101. Změřený dosah byl 23 metrů ± 0,5 metru, při vzdálenosti 23 metrů byl přenos zpomalený, ale bezchybný. Při vzdálenosti větší než 23 metrů byl přenos přerušen. Po návratu do vzdálenosti menší než 23 metrů byl přenos znovu obnoven. Moduly se automaticky znovu spojili. Níže uvedený obrázek uvádí průběh vysílání.




*Obrázek 18 - Průběh signálu na Bluetooth modulech RN42*

# 8. Periferie

Mikrokontrolér bez připojených periferií nemá téměř žádný přínos. Až s připojením nějaké periferie poskytuje mikrokontrolér přidanou hodnotu v podobě inteligentního zařízení. Periferie lze rozdělit na vstupní, výstupní nebo vstupně/výstupní. Vstupní periferie mohou být například akcelerometry, snímače teploty, senzory tlaku, senzory pro detekci pohybu, senzory pro měření vzdálenosti nebo snímače srdeční frekvence. Jako výstupní periferie pak lze označovat například led diody, displeje nebo elektromotor. Vstupně výstupní periferie může být například čtečka SD karet.

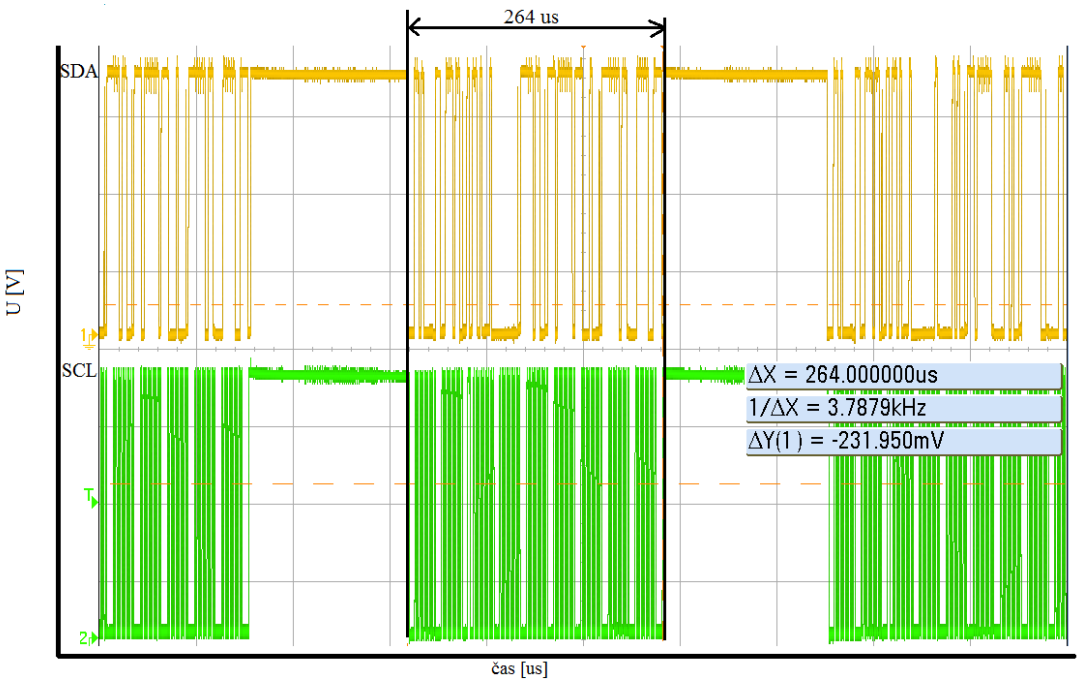
## 8.1 Akcelerometr a gyroskop MPU-6050

Akcelerometr, který byl zvolen je typ MPU-6050 Tento modul se skládá z integrovaného obvodu, který má 3 funkce: akcelerometr, gyroskop a teploměr. Akcelerometr měří pohybovou akceleraci, tedy zrychlení v $m/s^2$. Akcelerometr může měřit náklon či vibrace modulu. Gyroskop obsahuje setrvačník, který zachovává polohu osy své rotace. Lze s ním tedy poznat jaká je poloha celého modulu vůči ploše země (4). Modul komunikuje s miktrokontrolerem pomocí I2C sběrnice. Je tak nutné znát adresu., tu lze zjistit od výrobce. Vybraný akcelerometr měl adresu je 0x68. Pro programování je využívána knihovna Wire.h . Data jsou uložena v registrech viz tabulka číslo 9, kde jsou adresy nedůležitějších registrů. Registry jsou 8 bitové. Lze číst například pouze akcelerometr, gyroskop nebo vše. Záleží na na zvolené počáteční adrese a zvolené délce. Přečtené hodnoty lze ukládat do proměnné typu byte, případně integer při bitovém posunu a sloučení dvou bajtů vybrané osy. Rychlost vzorkování lze ovlivnit příkazem *Wire.setClock()*. Maximální rychlost hodinového signálu je 400 000Hz. MPU-6050 obsahuje 16 bitové analogově digitální převodníky. Pomocí příkazu *Wire.requestFrom(MPU_addr, x, true)*, kde x značí počet registrů, lze zvolit počet čtených registrů. Například 6 pro čtení os x,y a z akcelerometru nebo 14 při čtení všech os akcelerometru, gyroskopu a teploty. Při zvolená počáteční adrese pomocí příkazu *Wire.write(0x3B),* kde adresa 0x3B uvádí adresu prvního čteného registru Nastavení citlivosti akcelerometru lze nastavit v registru 0x1C na bitech 4 a 3, možnosti jsou 2g,4g, 8g a 16g Nastavení pro 2g je 0, pro 4g je 8, pro 8g a 16 a pro 16g 24, jedná se o dekadické hodnoty uložené do registru.

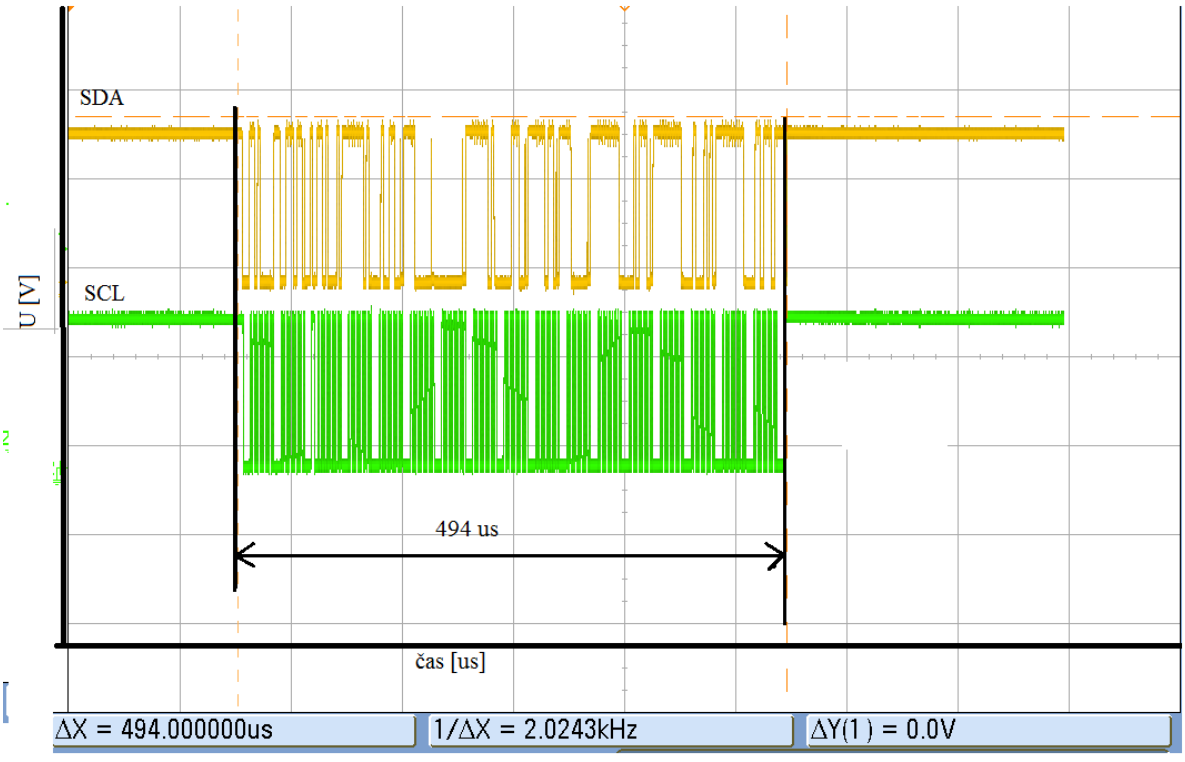
| Adresa (HEX) | Význam |
|---|---|
| 3B | Akcelerometr osa x bity 15-8 |
| 3C | Akcelerometr osa x bity 7-0 |
| 3D | Akcelerometr osa y bity 15-8 |
| 3E | Akcelerometr osa y bity 7-0 |
| 3F | Akcelerometr osa z bity 15-8 |
| 40 | Akcelerometr osa z bity 7-0 |
| 41 | Teplota bity 15-8 |
| 42 | Teplota bity 7-0 |
| 43 | Gyroskop osa x bity 15-8 |
| 44 | Gyroskop osa x bity 7-0 |
| 45 | Gyroskop osa y bity 15-8 |
| 46 | Gyroskop osa y bity 7-0 |
| 47 | Gyroskop osa z bity 15-8 |
| 48 | Gyroskop osa z bity 7-0 |

*Tabulka 9 - Registry MPU-6050*

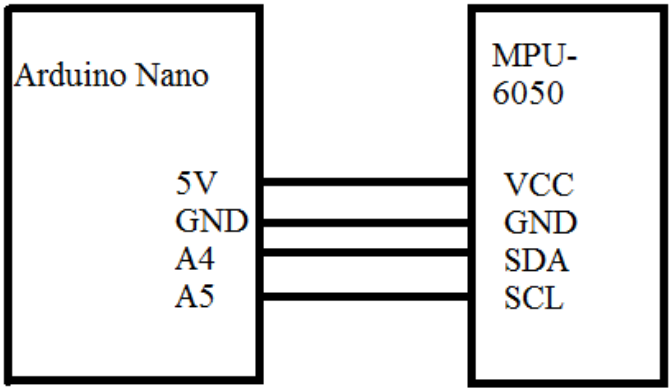
Na následujícím obrázku číslo 19 je zobrazen průběh čtení akcelerometru, kde sondy byli umístěny na hodinový vodič a datový vodič, kde byly čteny hodnoty z akcelerometru osy x,y a z. Dohromady tak bylo čteno 6 bajtů změřená doba trvání byla 264us. Pauza mezi jednotlivými čteními dat z akcelerometru byla při měření dána tím, že data byla ještě vysílána pomocí modulů NRF24L01+. Na obrázku číslo 20 je zobrazen průběh při čtení všech os akcelerometru, gyroskopu a teploty. Doba trvání byla změřena 494us




*Obrázek 19 - Průběh čtení akcelerometru os x, y a z*



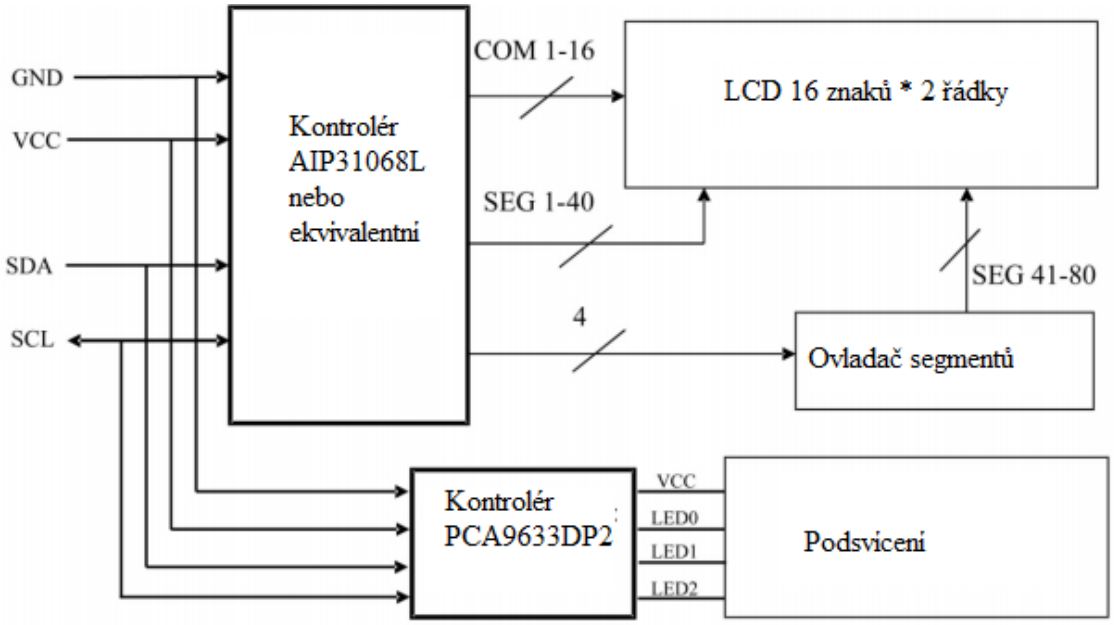

*Obrázek 20- Průběh čtení všech os akcelerometru, gyroskopu a teploty*



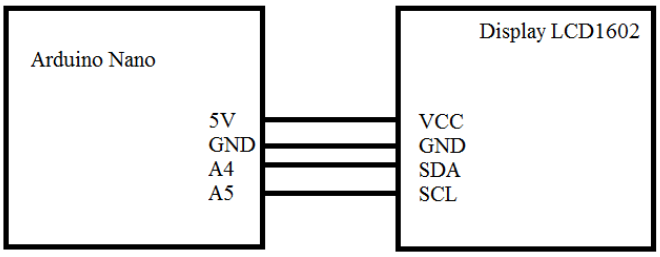

*Obrázek 21- Schéma zapojení Arduino Nano a MPU-6050*

## 8.2 Display

Displeje obecně slouží k zobrazení dat. Displeje lze rozdělit dle několika kritérií, dle technologie mezi nejrozšířenější technologie pak patří LCD, LED, TFT a OLED displeje, dále lze rozdělit podle barevnosti na monochromatické a barevné. Zvolený display byl sériový a připojený k miktrokontroléru pomocí I2C sběrnice. Display byl typu LCD 1602 od výrobce Gravity. Pro logické úrovně není nutné převádět úroveň napětí display pracuje s 5V logikou. Napájení lze 3,0V – 5.2V,jedná se o krajní hodnoty. Výrobce uvádí spotřebu menší než 60mA. LCD celým názvem *Liquid Crystal Display* se skládá z řady malých segmentů. Nejmenší zobrazitelný bod se označuje jako pixel. Krystaly mohou být upraveny tak, aby prezentovaly požadovanou informaci, například znak nebo obrázek Pro programování byly využity knihovny DFRobot_LCD.h a Wire.h. Na display lze zapsat 16 znaků na řádek, display má 2 řádky. Na obrázku číslo 21 je ukázán blokový diagram displeje. Maximální frekvence hodinového signálu dle datasheetu je 410 000 Hz. Zapojení je pak následují dle obrázku obrázek číslo 22 uvádí blokové schéma displeje.




*Obrázek 22- Blokové schéma LCD1602*




*Obrázek 23 - Schéma zapojení Arduino Nano - LCD1602*

# 9. Závěr

Cílem této práce bylo realizovat přenos na bázi Bluetooth. Realizace tohoto přenosu byla realizována pomocí rádiových modulů NRF24L01+. Data z akcelerometru lze využít pro lékařské účely například pro sledování pohybu osoby. V kapitole, kde je realizován výpis na LCD display lze využít například pro ležící osoby, kde displeji lze sledovat pohyb osoby, jelikož frekvence vzorkování není velká, ale není zde nutné připojení počítače, protože data jsou vypisovány na display. Postačí pouze napájení dvou mikrokontrolerů Pro rychlejší pohyby lze využít příklad z měření na testovacím vibrátoru. Z měření vyplynulo, že k rychlejšímu vzorkování dochází, pokud je využívána funkce *write()*. Kde jelikož data nejsou kontrolována, pouze se předpokládá, že vysílač vysílá data v delších intervalech, než trvá výpis na straně přijímače. Při výpisu pomocí *write()* není dořešeno, kdyby došlo výpadku spojení. Při výpisu pomocí vlastní funkce, by se případný výpadek projevil jako skok na kontrolním bajtu. Možné řešení při výpisu pomocí *writu()* by bylo doplnění vysílání o kontrolní 2 bajty z důvodu, že v Matlabu tvoříme dvojice bajtů. Kontrolní bajty by se po každém odeslání zvětšovali o 1. Poté lze po analýze v Matlabu uvažovat pravidelné vzorkování pokud by měli kontrolní bajty předpokládaný průběh. Pokud by došlo ke ztrátě spojení. Projeví se ztráta spojení na kontrolním bajtu jako skok. Poté tyto data kde je skok nelze použít. Přenos dat z akcelerometru pomocí Bluetooth modulů HC-05 byl velmi pomalý. Při zvýšení komunikační rychlosti na více než 38 400 baudů pomocí AT příkazů, byla vypisována na COM port nečitelná data.

# Bibliografie

(1). http://www.elektrorevue.cz/clanky/04003/index.html
(2). http://www.rfwireless-world.com/Terminology/GFSK-vs-FSK.html
(3). http://embeddedcoolness.com/home/
(4). https://navody.arduino-shop.cz/navody-k-produktum/gyroskop-a-akcelerometr.html
(5) Kniha - Arduino Uživatelská příručka Matúš Sedlecký
Datasheet MPU-6050
Datasheet NRF24L01+

# Přílohy

## A Zdrojové kódy k Bluetooth modulu HC-05

### A.1 Kód k programování pomocí AT příkazů

```
#include <SoftwareSerial.h>
SoftwareSerial BT(10, 11); // RX | TX
char c;
void setup()
{
  Serial.begin(9600);
  BT.begin(38400);
  Serial.write("Zadejte prikaz AT: \n\n");
}
void loop()
{
  if (BT.available())
    Serial.write(BT.read());
  if (Serial.available()) {
    c = Serial.read();
    Serial.write(c);
    BT.write(c);
  }
}
```

### A.2 Kód pro přenos dat akcelerometru

```
#include <SoftwareSerial.h>
#include<Wire.h>
const int MPU_addr = 0x68;
SoftwareSerial BT(10, 11); // RX | TX
void setup()
{
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
  Serial.begin(9600);
  BT.begin(38400);
  Wire.setClock(400000);
}
int AcX,AcY, AcZ;
void loop()
{
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr, 6, true);
  AcX = Wire.read() << 8 | Wire.read();
```

```
  AcY = Wire.read() << 8 | Wire.read();
  AcZ = Wire.read() << 8 | Wire.read();
  if (BT.available()) {
    BT.print(AcX);
    BT.print(" ");
    BT.print(AcY);
    BT.print(" ");
    BT.println(AcZ);
  }
}
```

# B Zdrojové kódy k rádiovým modulům NRF24L01+

## B.1 Kód pro měření maximální vzdálenosti

**Kód pro vysílač**

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include<Wire.h>
RF24 radio(7, 8); // CE, CSN
const byte address[3] = "01";
void setup() {
  Serial.begin(1000000);
  radio.begin();
  radio.openWritingPipe(address);
  radio.stopListening();
radio.write_register(0x00,14);
radio.write_register(0x01,1);
radio.write_register(0x02,1);
radio.write_register(0x03,1);
radio.write_register(0x04,255);
radio.write_register(0x05,76);
radio.write_register(0x06,38);
radio.write_register(0x11,1);
}
byte pokus = 255;
void loop() {
  pokus++;
 radio.write(&pokus, sizeof(pokus));

}
```

**Kód pro přijímač**

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Wire.h>

RF24 radio(7, 8); // CE, CSN
const byte address[3] = "01";
```

```
void setup() {
  Serial.begin(1000000);
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.startListening();

radio.write_register(0x00,15);
radio.write_register(0x01,1);
radio.write_register(0x02,1);
radio.write_register(0x03,1);
radio.write_register(0x04,255);
radio.write_register(0x05,76);
radio.write_register(0x06,38);
radio.write_register(0x11,1);
}
byte pokus=0;
void loop() {
  if (radio.available()) {
    radio.read(&pokus, sizeof(pokus));
Serial.println(pokus);
  }
}
```

## B.2 Kód pro měření maximální rychlosti přenosu znaků

**Kód pro vysílač**

```
byte prom = 0;
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include<Wire.h>
uint8_t obsah = 0;
RF24 radio(7, 8); // CE, CSN
const byte address[3] = "01";
void setup() {
  Serial.begin(57600);
  radio.begin();
  radio.openWritingPipe(address);
  radio.stopListening();
radio.write_register(0x00,2);
radio.write_register(0x01,0);
radio.write_register(0x02,1);
radio.write_register(0x03,1);
radio.write_register(0x04,0);
radio.write_register(0x05,76);
radio.write_register(0x06,14);
radio.write_register(0x11,1);
}
int pocetod=0;
```

```
byte pokus = 255;
int i=0;
void loop() {
 pokus = Serial.read();
 if(pokus!=255){
 Serial.write(pokus);
 pocetod++;
 radio.writeFast(&pokus, sizeof(pokus));
   if(pokus=='K'){
    Serial.println("pocet odeslanych znaku");
  Serial.println(pocetod);
  pocetod=0;
  }
 }
}
```

**Kód pro přijímač**

```
 #include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Wire.h>
RF24 radio(7, 8); // CE, CSN
const byte address[3] = "01";
byte pole[16];
void setup() {
Serial.begin(57600);
radio.begin();
radio.openReadingPipe(0, address);
radio.startListening();
radio.write_register(0x00,3);
radio.write_register(0x01,0);
radio.write_register(0x02,1);
radio.write_register(0x03,1);
radio.write_register(0x04,0);
radio.write_register(0x05,76);
radio.write_register(0x06,14);
radio.write_register(0x11,1);
}
int pocet = 0;
byte data = 0;
unsigned long casz;
unsigned long cask;
unsigned long casrozdil;
byte i=0;
void loop() {
  if (radio.available()) {
 radio.read(&data, sizeof(data));
 Serial.write(data);
  pocet++;
    if (data == 'Z')
      casz = micros();
```

```
    if (data == 'K') {
      cask = micros();
      casrozdil = cask - casz;
      Serial.println("cas je: ");
      Serial.println(casrozdil);
      Serial.print("pocet je: ");
      Serial.println(pocet);
      pocet = 0;
    }
  }
}
```

## B.3 Kód pro přenos dat z akcelerometru a následným výpisem na display

**Kód pro vysílač**

```
                        #include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include<Wire.h>
struct package
{
  int AcX;
  int AcY;
  int AcZ;
  byte kontrola;
  int teplota;
  int GyX;
  int GyY;
  int GyZ;
};
typedef struct package Package;
Package data;
int pole[32];
int i=0;
const int MPU_addr=0x68;
RF24 radio(7, 8); // CE, CSN
const byte address[3] = "01";
void setup() {
```

```
Wire.begin();
Wire.beginTransmission(MPU_addr);
Wire.write(0x6B);
Wire.write(0);
Wire.endTransmission(true);
Serial.begin(1000000);
radio.begin();
radio.openWritingPipe(address);
radio.stopListening();
radio.write_register(0x00,2);
radio.write_register(0x01,0);
radio.write_register(0x02,1);
radio.write_register(0x03,1);
radio.write_register(0x04,0);
radio.write_register(0x05,76);
radio.write_register(0x06,14);
radio.write_register(0x11,15);
Wire.setClock(400000);
}
void loop() {
 Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr,14,true);
  data.AcX=Wire.read()<<8|Wire.read();
  data.AcY=Wire.read()<<8|Wire.read();
  data.AcZ=Wire.read()<<8|Wire.read();
  data.teplota=Wire.read()<<8|Wire.read();
  data.GyX=Wire.read()<<8|Wire.read();
  data.GyY=Wire.read()<<8|Wire.read();
  data.GyZ=Wire.read()<<8|Wire.read();
 data.kontrola++;
 delayMicroseconds(1100) ;
  radio.writeFast(&data, sizeof(data));
```

```
 if (data.kontrola ==255 || data.kontrola ==128)
 delay(265);
}
```

**Kód pro přijímač**

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Wire.h>
#include "DFRobot_RGBLCD.h"
struct package
{
  int AcX;
  int AcY;
  int AcZ;
  byte kontrola;
  int teplota;
  int GyX;
  int GyY;
  int GyZ;
};
void vypis(int cislo){
  bool nula = false;
  byte znak;
  if(cislo<0){
    cislo = cislo *-1;
    Serial.write('-');
  }
  for(int i =10000;i>0;i=i/10){
     znak = cislo/i;
     cislo = cislo %i;
     nula = nula || znak;
     if(nula)
       Serial.write(znak + '0');
  }
```

```
  if(!nula){
  Serial.write('0');
  }
  Serial.write(' ');
}
typedef struct package Package;
Package data;
DFRobot_RGBLCD lcd(16,2);
RF24 radio(7, 8); // CE, CSN
const byte address[3] = "01";
void setup() {
  Serial.begin(1000000);
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.startListening();
  radio.write_register(0x00, 3);
  radio.write_register(0x01, 0);
  radio.write_register(0x02, 1);
  radio.write_register(0x03, 1);
  radio.write_register(0x04, 0);
  radio.write_register(0x05, 76);
  radio.write_register(0x06, 14);
  radio.write_register(0x11, 15);
 lcd.init();
 Wire.setClock(400000);
}
void loop() {
   if (radio.available()) {
    radio.read(&data, sizeof(data));
    vypis(data.AcX);
     vypis(data.AcY);
     vypis(data.AcZ);
     vypis(data.GyX);
     vypis(data.GyY);
```

```
    vypis(data.GyZ);
    vypis(data.kontrola);
    Serial.println(data.teplota/340.00+36.53);
 if(data.kontrola==255 || data.kontrola==128){
    lcd.clear();
   lcd.setCursor(0,0);
   lcd.write("X=");
   lcd.print(data.AcX);
   lcd.write(" Y=");
   lcd.print(data.AcY);
   lcd.setCursor(0,1);
   lcd.write("Z=");
   lcd.print(data.AcZ);
 }
  }
}
```

## B.4 Kód pro výpis pomocí vlastní funkce při testování na testovacím vibrátoru

**Kód pro vysílač**

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Wire.h>
struct package
{
 int AcX;
 int AcY;
 int AcZ;
 byte kontrola;
};
void vypis(int cislo){
 bool nula = false;
 byte znak;
 if(cislo<0){
   cislo = cislo *-1;
   Serial.write('-');
 }
 for(int i =10000;i>0;i=i/10){
    znak = cislo/i;
    cislo = cislo %i;
    nula = nula || znak;
```

```
    if(nula)
      Serial.write(znak + '0');
  }
  if(!nula){
  Serial.write('0');
  }
  Serial.write(' ');
}
typedef struct package Package;
Package data;
RF24 radio(7, 8); // CE, CSN
const byte address[3] = "01";
void setup() {
  Serial.begin(1000000);
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.startListening();
  radio.write_register(0x00, 3);
  radio.write_register(0x01, 0);
  radio.write_register(0x02, 1);
  radio.write_register(0x03, 1);
  radio.write_register(0x04, 0);
  radio.write_register(0x05, 78);
  radio.write_register(0x06, 14);
  radio.write_register(0x11, 7);
}
void loop() {
  if (radio.available()) {
    radio.read(&data, sizeof(data));
    vypis(data.AcX);
     vypis(data.AcY);
     vypis(data.AcZ);
     vypis(data.kontrola);
    Serial.println();
  }
   }
```

**Kód pro přijímač**

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Wire.h>
struct package
{
  int AcX;
  int AcY;
  int AcZ;
  byte kontrola;
};
void vypis(int cislo){
  bool nula = false;
```

```
  byte znak;
  if(cislo<0){
    cislo = cislo *-1;
    Serial.write('-');
  }
  for(int i =10000;i>0;i=i/10){
     znak = cislo/i;
     cislo = cislo %i;
     nula = nula || znak;
     if(nula)
       Serial.write(znak + '0');
  }
  if(!nula){
  Serial.write('0');
  }
  Serial.write(' ');
}
typedef struct package Package;
Package data;
RF24 radio(7, 8); // CE, CSN
const byte address[3] = "01";
void setup() {
  Serial.begin(1000000);
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.startListening();
  radio.write_register(0x00, 3);
  radio.write_register(0x01, 0);
  radio.write_register(0x02, 1);
  radio.write_register(0x03, 1);
  radio.write_register(0x04, 0);
  radio.write_register(0x05, 78);
  radio.write_register(0x06, 14);
  radio.write_register(0x11, 7);
}
void loop() {
  if (radio.available()) {
    radio.read(&data, sizeof(data));
    vypis(data.AcX);
     vypis(data.AcY);
     vypis(data.AcZ);
     vypis(data.kontrola);
    Serial.println();
  }
   }
```

## B.5 Kód pro výpis pomocí write() při testování na testovacím vibrátoru

**Kód pro vysílač**

```
#include <SPI.h>
#include <nRF24L01.h>
```

```
#include <RF24.h>
#include<Wire.h>
byte pole[8];
byte pom1 = 0x00;
const int MPU_addr = 0x68;
RF24 radio(7, 8); // CE, CSN
const byte address[3] = "01";
void setup() {
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
  Serial.begin(230400);
  radio.begin();
  radio.openWritingPipe(address);
  radio.stopListening();

radio.write_register(0x00,2);
radio.write_register(0x01,0);
radio.write_register(0x02,1);
radio.write_register(0x03,1);
radio.write_register(0x04,0);
radio.write_register(0x05,76);
radio.write_register(0x06,14);
radio.write_register(0x11,8);
  Wire.setClock(400000);
  pinMode(4, OUTPUT);
}
 unsigned long time1;
  unsigned long time2;
byte axd, axh, ayd, ayh, azd, azh;
int pozice = 0;
void loop() {
  digitalWrite(4, HIGH);
time1=micros();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr, 6, true);
  axh = Wire.read();
  axd = Wire.read();
  ayh = Wire.read();
  ayd = Wire.read();
  azh = Wire.read();
  azd = Wire.read();
  pole[pozice] = axh;
  pozice++;
  pole[pozice] = axd;
  pozice++;
```

```
  pole[pozice] = ayh;
  pozice++;
  pole[pozice] = ayd;
  pozice++;
  pole[pozice] = azh;
  pozice++;
   pole[pozice] = azd;
  pozice++;
   pole[pozice] = 0xFF;
  pozice++;
   pole[pozice] = 0x00;
    if (pozice ==7){
      radio.writeFast(&pole, 8);
      pozice=0;
    }
      time2=micros();
   Serial.println(time2-time1);
       }
```

**Kód pro přijímač**

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Wire.h>
byte pole[8];
RF24 radio(7, 8); // CE, CSN
const byte address[3] = "01";
void setup() {
  Serial.begin(230400);
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.startListening();
  pinMode(4, OUTPUT);
  radio.write_register(0x00, 3);
  radio.write_register(0x01, 0);
  radio.write_register(0x02, 1);
  radio.write_register(0x03, 1);
  radio.write_register(0x04, 0);
  radio.write_register(0x05, 76);
  radio.write_register(0x06, 14);
  radio.write_register(0x11, 8);
}
byte i = 0;
void loop() {
  if (radio.available()) {
    radio.read(&pole, sizeof(pole));
    for ( i = 0; i < 8; i++) {
      Serial.write(pole[i]);
      }
    }
}
```