VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

REALIZACE A TESTOVÁNÍ KOMUNIKACE MEZI ZAŘÍZENÍM S OS ANDROID A MIKROKONTROLEREM POMOCÍ PŘEVODNÍKU USB/UART

REALIZATION AND TESTING OF COMMUNICATION BUS BETWEEN DEVICE WITH OS ANDROID AND MICROCONTROLLER USING USB/RS232 CONVERTER

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Tomáš Kopčil

AUTHOR

VEDOUCÍ PRÁCE — Ing. Martin Brablc

SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Tomáš Kopčil**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**Ing. Martin Brablc**
Akademický rok:	2018/19

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Realizace a testování komunikace mezi zařízením s OS Android a mikrokontrolerem pomocí převodníku USB/UART

Stručná charakteristika problematiky úkolu:

Klesající cena a rostoucí výpočetní výkon všudypřítomných telefonů a tabletů otevírá otázku jejich využití jako výkonné HMI (Human–Machine Interface) jednotky pro nejrůznější zařízení s mikrokontrolery, kterou je možno snadno a rychle programovat. Cílem práce je seznámit se s vybraným převodníkem a dostupnou knihovnou pro jeho obsluhu v OS Android a s vytvářením jednoduchých aplikací pro tento operační systém, dále otestovat limity komunikace s mikrokontrolerem a realizovat ukázkovou aplikaci, která poslouží jako základ pro další využití v rámci projektů, které Mechatronická laboratoř realizuje s průmyslovými partnery.

Cíle bakalářské práce:

1) Proveďte stručnou rešerši všech potřebných nástrojů, knihoven a zařízení potřebných pro realizaci komunikace mezi zařízením s OS Android a vybraným mikrokontrolerem. Rešerše má v tomto případě sloužit zároveň jako stručný návod a rozcestník k podrobnějším informacím.
2) Realizujte obousměrnou komunikaci mezi vybraným zařízením s OS Android a vývojovou deskou s mikrokontrolerem PIC.
3) Navrhněte a realizujte experimenty testující limity realizované komunikace s ohledem na celkovou datovou propustnost, latenci a délku zprávy. Pokuste se odpovědět na otázku do jaké frekvence lze takovou soustavu považovat za soft real–time pro nekritické řídící aplikace (především pro použití ve výuce).
4) Realizujte jednoduchou výukovou aplikaci, která přímo řídí polohu DC motoru s enkoderem pomocí PID regulátoru, a na které lze demonstrovat jeho chování. Aplikace vykresluje všechny potřebné signály a umožňuje změnu parametrů. Pokud to dosažitelná frekvence zpětné vazby umožňuje, je regulátor realizován v OS android, pokud ne, zpětná vazba je realizována na mikrokontroleru a tato aplikace pouze mění parametry regulátoru a načítá data.
5) Vytvořte ukázkovou aplikaci – informační display pro laboratorní vozítko typu segway v Mechatronické laboratoři. Display zobrazuje základní informace (náklon, rychlost, proud, napětí baterie) a umožňuje nastavování vybraných parametrů vozítka.

Seznam doporučené literatury:

NELLES, Oliver. Nonlinear system identification: from classical approaches to neural networks and fuzzy models. Berlin: Springer, 2011. ISBN 978-364-2086-748.

LJUNG, Lennart. System identification: theory for the user. 2nd ed. Upper Saddle River, NJ: Prentice Hall PTR, 1999. ISBN 978-0136566953.

VALÁŠEK, Michael. Mechatronika. Dot. 1. vyd. Praha: České vysoké učení technické, 1996. ISBN 80-010-1276-X.

NOSKIEVIČ, Petr. Modelování a identifikace systémů. Ostrava: Montanex, 1999. ISBN 80-722-50-0-2.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2018/19

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc.
ředitel ústavu

doc. Ing. Jaroslav Katolický, Ph.D.
děkan fakulty

Abstrakt

Tato práce se zabývá komunikací mezi zařízením s OS Android a mikrokontrolerem PIC. V teoretické části práce jsou představeny potřebné nástroje a vývojová prostředí pro sestavení této komunikace. Praktická část se pak věnuje samotné realizaci komunikace a jejímu testování. Za účelem demonstrace funkčnosti realizované komunikace byly vytvořeny dvě ukázkové úlohy: aplikace umožňující regulaci polohy motoru a informační displej pro vozítko typu segway.

Summary

This thesis deals with the communication bus between Android OS-based device and the PIC microcontroller. The theoretical part presents required utilities and developement environments for setting up this communication. The practical part focuses on putting the communication bus into practice and testing its capabilities. Two applications were set up to demonstrate the data transfer: an application providing access to the regulation of motor position and an information display for a segway-type transporter.

Klíčová slova

OS Android, mikrokontroler, PIC, UART, USB, Android Studio, Java, MPLAB, Simulink Coder, HMI

Keywords

OS Android, microcontroller, PIC, UART, USB, Android Studio, Java, MPLAB, Simulink Coder, HMI

Bibliografická Citace

KOPČIL, T. *Realizace a testování komunikace mezi zařízením s OS Android a mikrokontrolerem pomocí převodníku USB/UART*. Brno, 2019. Bakalářska práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství. 49 s., Vedoucí bakalářské práce Ing. Martin Brablc.

Prohlašuji, že jsem bakalářskou práci na téma *Realizace a testování komunikace mezi zařízením s OS Android a mikrokontrolerem pomocí převodníku USB/UART* vypracoval samostatně s použitím materiálů a zdrojů uvedených v seznamu literatury.

Tomáš Kopčil

Brno . . . .

Děkuji vedoucímu Ing. Martinu Brablcovi za ochotu, trpělivost, cenné rady a vedení mé bakalářské práce správným směrem.

Tomáš Kopčil

Obsah

1 Úvod **9**

2 Rešerše **11**
2.1 UART 11
2.1.1 Popis a princip funkce 11
2.1.2 Použití UARTu v praxi 12
2.2 USB 12
2.3 Real-time systémy 13
2.3.1 Princip real-time systémů a jejich rozdělení 13
2.3.2 Soft RT systémy 13
2.3.3 Hard RT systémy 13
2.4 Vývojová prostředí pro realizaci komunikace 13
2.4.1 MPLAB 13
2.4.2 Android Studio 14
2.4.3 MATLAB & Simulink 15
2.5 Použité knihovny pro Android Studio 16
2.5.1 USBSerial 16
2.5.2 MPAndroidChart 16
2.6 Použité vývojové desky 17
2.6.1 Deska s 8-bitovým mikrokontrolerem PIC 17
2.6.2 Deska s 16-bitovým mikrokontrolerem dsPIC 17
2.6.3 Převod USB/UART čipem FT231X 18

3 Realizace komunikace **19**
3.1 Požadavky na komunikaci 19
3.2 Komunikace ze strany mikrokontroleru 20
3.2.1 Nastavení FTDI čipu 20
3.2.2 Algoritmus pro mikrokontroler 21
3.3 Komunikace ze strany OS Android 21
3.3.1 Algoritmus pro OS Android 21
3.3.2 Omezení komunikace 22
3.3.3 Ukázka realizované komunikace 22

4 Statistické testování komunikace **24**
4.1 Návrh testování 24
4.2 Naměřené hodnoty 25
4.2.1 Měření pro zprávu délky 1 25
4.2.2 Měření pro zprávu délky 5 26
4.2.3 Měření pro zprávu délky 10 26
4.2.4 Měření pro zprávu délky 50 27

4.3 Vyhodnocení testů 28

5 Demonstrační úlohy 30
5.1 Regulace polohy motoru 30
5.1.1 Mikrokontroler a připojení na motor 30
5.1.2 Realizace regulátoru v OS Android 32
5.1.3 Uživatelské prostředí 33
5.2 Informační displej vozítka Segway 34

6 Závěr 35

Seznam zkratek a symbolů 37

Seznam obrázků 39

Literatura 40

Dodatek 42
A Histogramy z naměřených dat 42
A.1 Měření pro zprávu délky 1 42
A.2 Měření pro zprávu délky 5 44
A.3 Měření pro zprávu délky 10 46
A.4 Měření pro zprávu délky 50 48
B Elektronické přílohy 49

1 Úvod

Současné masové rozšíření zařízení s OS Android spolu s jejich narůstajícím výkonem nabízejí široké možnosti využití v praxi. Výpočetní výkon těchto zařízení je opravdu obrovský – dnešním standardem vlajkových lodí mobilních zařízení jsou několika jádrové procesory s vysokým taktem a kombinací operační paměti často převyšující 4 GB. Takové konfigurace se ještě před nedávnou dobou běžně objevovaly v počítačích vyšší třídy, přitom v dnešní době nosí většina lidí zařízení o podobném výkonu v kapse.

Velkou výhodou těchto zařízení je jejich mobilita, oproti standardnímu PC lze telefon či tablet snadno přenášet, přičemž může být po poměrně dlouhou dobu napájen z baterie. To opět rozšiřuje možnosti jejich nasazení v praxi, kde jsou takové parametry často vyžadovány. Jednou z nich je připojení těchto zařízení na řídící prvek daného systému (mikrokontroler, PLC, apod.) a provozování v běžném uživatelském rozhraní pro ovládání tohoto systému. Díky dostatečnému výkonu mobilních zařízení lze získávat výsledky výpočtů z analýzy daného systému prakticky okamžitě. Příklad takového využití v praxi lze nalézt například v ovládání RC modelů či průmyslových zařízení jako jsou manipulátory, obráběcí stroje, apod. (obrázek 1.1).

Nezáleží, zda se jedná o telefony či tablety, a stejně tak nemusí jít vždy o řízení systémů. Často je potřebný pouze informační panel pro zobrazování daných hodnot a veličin uživateli. Příkladem takového zařízení může být například diagnostika automobilů nebo informační displeje obecně.

Obrázek 1.1: Tabletem řízený manipulátor [25]

Taková řešení otevírají možnosti pro ovládání komplikovanějších systémů, které by jinak neproškolený uživatel nemohl zvládnout.

Při univerzální aplikaci pro OS Android pak často není potřebný ani externí displej – v době kdy má téměř každý svůj chytrý telefon, stačí pouze rozšířit tuto aplikaci mezi koncové uživatele, kteří si ji nainstalují do svých periferních zařízení. Ovládání daného systému tak může probíhat přímo z mobilního telefonu uživatele.

Jako další se nabízí použití ve výuce – např. zpětnovazebních regulačních systémů. Při možnosti regulovat výuková zařízení z mobilních telefonů či tabletů by tak odpadla nutnost obsluhy z PC.

Dále například prezentace funkčnosti systémů na nejrůznějších konferencích, veletrzích,

a předváděcích akcích, kde by si tak snadno mohli vyzkoušet zařízení samotní účastníci.

Tato bakalářská práce si klade za cíl realizovat takovou komunikaci mezi koncovým zařízením a zmíněnými mobilními periferiemi, která by jednak umožnila konkrétní nasazení v praxi, a také zvýšila uživatelskou přívětivost ovládání daného systému. Běžný uživatel se totiž často omezuje na ovládací prvky typu *START*, *STOP*, +, −, atp., a proto by tato aplikace měla umožnit ovládání zmíněného systému bez jeho hlubší znalosti.

V dnešní době lze snadno pořídit displej, např. v podobě tabletu, za poměrně nízkou cenu. Přitom ale potenciál komunikace mezi zobrazovací technikou a průmyslovými mikrokontrolery zůstává zatím nevyužitý. To vytváří prostor k vytvoření takového prostředí, aby se ovládání zařízení spolu se zobrazováním informací mohlo stát samozřejmostí a potřebná byla pouze instalace příslušné komunikační aplikace do zařízení s displejem. Na takové zjednodušení cílí tato práce.

2 Rešerše

2.1 UART

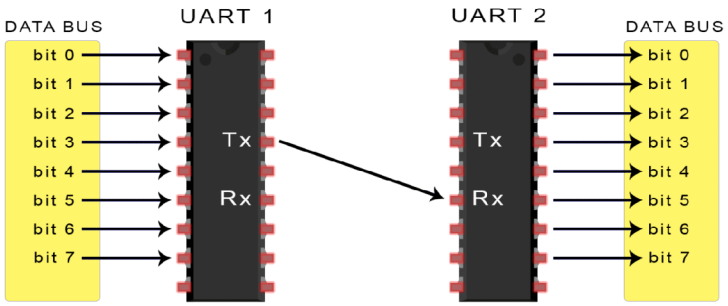
2.1.1 Popis a princip funkce

Zkratku UART lze do češtiny přeložit jako Univerzální Asynchronní Příjímač/Vysílač. Jedná se o komunikační rozhraní umožňující odesílání a příjímání dat daného zařízení.

Toto rozhraní bylo vyvinuto Gordonem Bellem již v šedesátých letech minulého století [1] a dodnes je stále hojně využíváno.

Princip fungování UARTu je následující: UART obdrží od řídící jednotky (CPU, mikrokontroler) paralelně seřazené informační bity. Ty převede do sériového pořadí, na počátek vloží tzv. start bit, a na konec za informační bity vloží tzv. stop bit. Tyto dva bity slouží pouze pro rozlišení počátku a konce odesílaných dat, nenesou žádnou jinou informaci.

Takto složenou zprávu (jednotlivé bity za sebou) následně odešle do dalšího zařízení, které může opět komunikovat pomocí UARTu. Přijatá zpráva je sestavena opět do paralelního řazení bitů, jsou odstraněny počáteční a koncový bit zprávy a zpráva je zpracována. Názorný přenos je na obrázku 2.1.



Obrázek 2.1: Příjem paralelních bitů, odeslání v sériové podobě, přijetí a sestavení do původního paralelního řazení. Tx značí vysílač (transmitter), Rx přijímač (receiver) [26].

Délka zprávy (míněny bity obsahující data) se většinou pohybuje mezi 5 až 9 bity. Tato délka pak narůstá o počáteční a koncový bit, případně ještě o paritní bit, pokud se používá [2].

Tzv. paritní bit slouží pro kontrolu přijatých dat. Z délky zprávy lze určit, zda se jedná o lichý, nebo sudý počet bitů. Tato informace o paritě je uložena do zmíněného paritního bitu, po přijetí druhým zařízením je délka zprávy přepočítána a je ověřeno, zda tato délka odpovídá informaci v paritním bitu. Tímto způsobem lze poměrně jednoduše odhalit, zda byla doručena veškerá odeslaná data a nebyla některá ztracena během přenosu.

Tato kapitola byla zpracována s využitím zdrojů [3], [4].

2.1.2 Použití UARTu v praxi

Díky jednoduchému principu funkčnosti lze UART provozovat na velkém množství zařízení. S velkou oblibou se používá zejména v embedded systémech, kde bývá základním komunikačním rozhraním. Většina mikrokontrolerů jako jsou Arduino, kontrolery PIC od firmy Microchip, nebo jednodeskové počítače Raspberry Pi už mají UART zabudovaný z výroby.

Mikrokontrolery však nejsou jediná zařízení využívající UART, ten lze nalézt také v GPS přijímačích, Bluetooth modulech, v GSM a GPRS modemech a v některých systémech využívajících RFID [5].

2.2 USB

Možností, jak mezi sebou vzájemně propojovat zařízení a koncové periferie, je celá řada. Tou nejběžnější, jsou v dnešní době konektory USB, které lze považovat za standard.

Masové rozšíření USB umožnilo propojovat typově odlišná zařízení stejným konektorem. Dříve např. tiskárny, monitory, ale i myši a klávesnice k PC používali konektory různé [6], což by dnes, vzhledem k široké škále těchto periferií, nebylo možné.

Populárním se USB stalo i na zařízeních s OS Android. Zde je nejčastěji využíván typ micro USB a v poslední době se rozšiřující nejnovější USB typu C (obrázek 2.2).

Obrázek 2.2: Konektory USB používané v telefonech a tabletech s OS Android. Zleva USB typ C, micro USB [27].

Nespornou výhodou USB je jednak podpora přenosu informací, ale také napájení připojeného zařízení. Micro USB, spadající pod USB 2.0, umožňuje napájení 5 V při maximálním proudu 500 mA [7]. To umožňuje připojení k deskám s mikrokontrolery při současném napájení, čehož využívá tato bakalářská práce.

2.3 Real-time systémy

2.3.1 Princip real-time systémů a jejich rozdělení

Zvláštním případem OS jsou tzv. real-time systémy. Na tyto je kladen nutný požadavek jejich činnosti: výstup dané operace musí být nejen správný výsledek výpočtu, ale musí být také znám v nějakém konkrétním, předem definovaném, čase (tzv. deadline) [9]. Pokud časová podmínka není splněna, dochází k chybě. Podle závažnosti této chyby rozdělujeme real-time systémy na Soft RT a Hard RT.

2.3.2 Soft RT systémy

Pokud nastane nedodržení časového limitu u systému, který označujeme jako soft RT, dochází k degradaci kvality výstupu s časem. Jinak řečeno, při překročení deadlinu v soft RT systémech ztrácí poskytovaná služba na kvalitě, ale nic zásadního není ohroženo [10].

Příkladem takového systému může být např. obyčejný počítač, kdy uživatel čeká na reakci systému. Dále např. digitální fotoaparát a mobilní telefony [11].

2.3.3 Hard RT systémy

U Hard RT systému je podmínka splnění časového limitu naprosto zásadní. Pokud je deadline překročen, selhává kompletně celý systém a důsledky mohou být až fatální. Takovým systémem může být například řízení letového provozu [12].

Stejně tak lze za Hard RT považovat řízení dynamických soustav. Například regulace rychlosti u vozítka Segway[1] nebude fungovat, pokud regulátor nezvládne včas provádět výpočet. Může tak dojít až k ohrožení jezdce.

Regulované soustavy lze obecně považovat za Hard RT. V případě obyčejného PID regulátoru může při nedodržení časové podmínky docházet k rozkmitání soustavy a tím znemožnění činnosti systému.

2.4 Vývojová prostředí pro realizaci komunikace

Pro zajištění komunikace mezi zařízením s OS Android a mikrokontrolerem PIC je nutné tato zařízení naprogramovat.

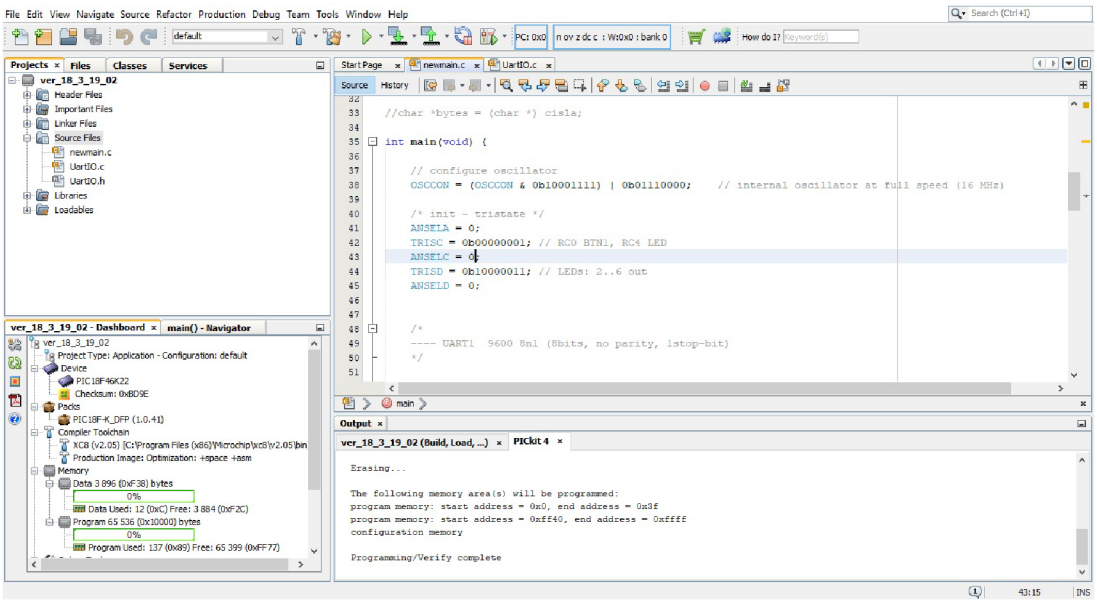
Jelikož obě zařízení používají jiný OS a jsou založena na jiné architektuře, nelze je programovat ve stejném vývojovém prostředí. Použitá prostředí jsou zmíněna v této kapitole.

2.4.1 MPLAB

Mikrokontrolery řady PIC a dsPIC výrobce Microchip, které jsou pro účely této práce využívány, je možno programovat v jazyce C. Michrochip přímo pro tyto účely vyvíjí vlastní prostředí

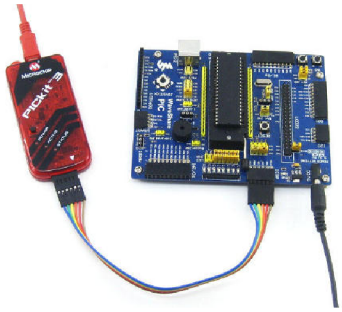
[1] Regulace rychlosti na tomto typu vozítka probíhá náklonem jezdce. Čím více se jezdec nakolní dopředu, tím rychleji vozítko pojede směrem vpřed.

MPLAB (obrázek 2.3), nabízející plnou podporu pro mikrokontrolery PIC a dsPIC.



Obrázek 2.3: Vývojové prostředí MPLAB.

Kód pro mikrokontroler je sestaven v prostředí MPLAB a následně pomocí programátoru PICkit a překladače XC8 přeložen do strojových instrukcí pro mikrokontroler (obrázek 2.4). Jedná se o komplexní řešení, jak poměrně snadno programovat řídící jednotky, které vyvinul a podporuje jejich výrobce.



Obrázek 2.4: Programování mikrokontroleru pomocí programátoru PICkit [28].

2.4.2 Android Studio

K vývoji aplikací pro OS Android je určeno prostředí Android Studio, které oficiálně poskytuje Google, hlavní vývojář tohoto operačního systému.

V tomto IDE lze programovat jazyky C++, Kotlin, nebo Java, která je v současnosti pro tyto účely nejvíce využívána.

Při tvorbě aplikace lze samozřejmě kdykoliv spustit instalaci do zařízení a kontrolovat tak, zda funguje podle očekávání. Pro tyto účely má Android Studio velice užitečnou utilitu. Aplikaci lze nainstalovat do simulátoru, kde je možno nakonfigurovat konkrétní zařízení a testovat ji zde.

Tím může vývojář vyzkoušet chování aplikace na zařízeních např. s různou velikostí displeje. Pro účely této práce však simulátor nelze použít – cílem je dosáhnout komunikace s fyzickým zařízením, které simulátor napodobit neumí.

Samotný vývoj aplikace probíhá v několika adresářích a souborech. Krátce zde budou popsány tři hlavní, které vývojář využívá.

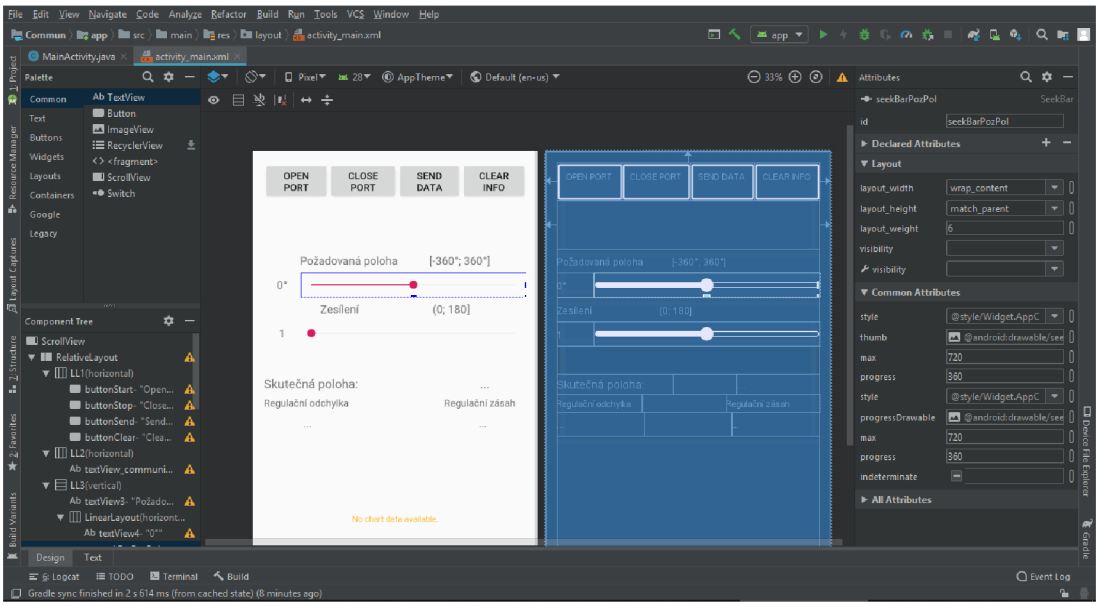
Manifest
Soubor AndroidManifest.xml obsahuje základní data pro aplikaci. Jsou zde uvedeny např. verze systému, požadavky aplikace na software i hardware, ale musí zde být deklarována i všechna povolení, která aplikace potřebuje pro přístup k chráněným částem systému, případně jiným aplikacím [14].

V tomto souboru je využíván programovací jazyk XML.

Adresář Java
Zde jsou uloženy veškeré zdrojové kódy psané v jazyku Java. Hlavní soubor *MainActivity.java* zde vytvoří Android Studio automaticky a ukládají se do něj veškeré vytvořené třídy [15]. Výpočtové části kódu jsou tedy uloženy na tomto místě.

Adresář Layout
Tento adresář obsahuje soubory, z kterých se následně sestavuje design celé aplikace. Z těchto souborů tedy následně vzniká uživatelské prostředí.

Kód je zde psán opět v jazyce XML. Uživatelské rozhraní je ale možné taky sestavovat v grafickém editoru, který je vestavěnou součástí Android Studia (obrázek 2.5).



Obrázek 2.5: Ukázka sestavování GUI v grafickém editoru vývojového prostředí Android Studio.

2.4.3 MATLAB & Simulink

Programovat náročnější algoritmy v jazyce C může být pro méně zkušeného uživatele poměrně nepohodlné (nutnost používání ukazatelů při práci s poli apod.). Pokud je ale cílové zařízení

kompatibilní, nabízí se alternativa v podobě Simulink Coder.

Uživatel zde vytvoří algoritmus v grafické podobě, který Simulink Coder přeloží do jazyka C a naprogramuje jím cílové zařízení. Nutná je zde zmíněná kompatibilita zařízení, která je pro mikrokontrolery PIC a dsPIC výrobcem zajištěna. Potřebné jsou pak některé doplňky pro MATLAB, a to Embedded Coder, MATLAB Coder a Simulink Coder. Nutné je také rozšíření bločků pro Simulink dodávané firmou Microchip, které umožní přístup k UARTu, PWM a dalším periferiím MCU. Ze strany mikrokontrolerů je pak potřebné IDE MPLAB [16].

2.5 Použité knihovny pro Android Studio

2.5.1 USBSerial

Inicializovat a správně nastavit komunikaci přes USB port je, z programátorského hlediska, na OS Android poměrně komplikované a vyžaduje relativně dobrou znalost programování pro tento systém.

Tento handicap lze řešit použitím některé z knihoven, které již tuto inicializaci obsahují. V této práci byla využita volně přístupná knihovna USBSerial dostupná z [17].

Jednou z hlavních výhod této knihovny je fakt, že autor ji stále vyvíjí (v době psaní této práce), opravuje chyby a pracuje na nových funkcích.

USBSerial umožňuje jak synchronní, tak asynchronní komunikaci, dále umožňuje připojení více zařízení současně a nově i ladění přes WIFI.

Knihovna podporuje komunikaci s relativně velkým počtem zařízení, mezi kterými jsou i FTDI čipy používané v této práci. Z přenosových rychlostí, tzv. baud rate, jsou podporovány standardní hodnoty.

Nespornou nevýhodou je však poměrně chabá dokumentace. Autor poskytuje pouze stručný návod jak s knihovnou pracovat, nicméně podrobnější informace a detaily chybí. Součástí knihovny je také několik demonstračních aplikací[2] pro ukázku funkčnosti, bohužel ale nejsou nijak podrobněji popsány a tak může být obtížné zorientovat se v nich.

2.5.2 MPAndroidChart

Vykreslování grafů není v Android Studiu nativně implementováno. Grafy je proto nutné sestavovat ručně, což není po programovací stránce vždy jednoduché ani pohodlné. V případě potřeby zahrnout do aplikace graf se tedy vyplatí sáhnout po nějaké dostupné knihovně.

Jednou z nejznámějších knihoven pro tvorbu grafů je MPAndroidChart dostupná z [18]. Tato volně přístupná knihovna zprostředkovává poměrně jednoduchou cestu, jak grafy vykreslovat a její součástí je také velmi dobrá dokumentace. I zde je výhodou také stálý vývoj autorem (v době psaní této práce).

I když knihovna oficiálně nepodporuje živé a dynamické grafy, i tak s nimi lze, do jisté míry, pracovat.

[2] Jejich zdrojový kód.

2.6 Použité vývojové desky

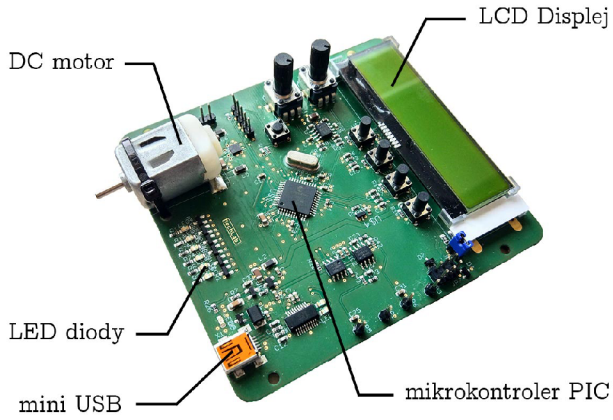
Mikrokontroler bývá v praxi osazen na desku (DPS), která je dále osazována dalšími součástmi nezbytnými pro funkčnost v závislosti na účelu použití. Jako příklad lze uvést nejrůznější konektory, displeje, tlačítka apod.

Pro tuto práci byly využity desky používané v mechatronické laboratoři.

2.6.1 Deska s 8-bitovým mikrokontrolerem PIC

Pro jednoduchý vývoj komunikace byla použita deska s mikrokontrolerem řady *PIC*, model *PIC18F46K22*, výrobce Microchip. Tento 8-bitový MCU poskytuje dostatek výkonu pro běžné, nenáročné aplikace typu práce s UARTem, standardní matematické výpočty apod.

Z prvků osazených na desce lze zmínit kontrolní LED diody, převodník UART/USB, konektor mini USB a také FTDI čip popsaný v sekci 2.6.3. Popis desky je na obrázku 2.6.

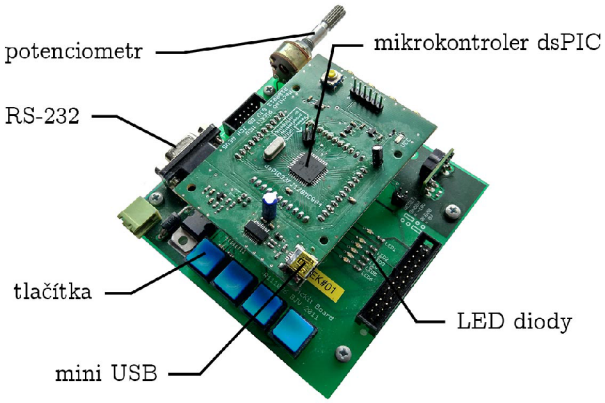


Obrázek 2.6: Popis desky s 8-bitovým mikrontrolerem PIC.

2.6.2 Deska s 16-bitovým mikrokontrolerem dsPIC

Zmíněný 8-bitový mikrokontroler není primárně určený pro řízení motoru (byť i to je do jisté míry možné), ani regulační aplikace obecně. Pro tyto aplikace byla použita deska s 16-bitovým mikrokontrolerem řady *dsPIC*, konkrétně model *dsPIC33FJ128MC804* (opět výrobce Microchip), určeným přesně pro tyto záměry.

Stejně jako předchozí deska, i tato je osazena zmíněnými součástmi, ke kterým jsou navíc přidány ještě další konektory pro širší možnosti připojení. Popis je na obrázku 2.7



Obrázek 2.7: Popis desky s 16-bitovým mikrontrolerem dsPIC.

2.6.3 Převod USB/UART čipem FT231X

Na použitých deskách probíhá komunikace zejména přes USB port. Je však nutné zajistit konverzi mezi USB a rozhraním UART mikrokontroleru. Pro tyto účely je na obou deskách použit čip firmy FTDI, model *FT231X*.

Tento čip nabízí plný komunikační handshake pro převod USB/UART, celý USB protokol je zajišťován tímto čipem, není tak potřebné další nastavování USB [19].

3 Realizace komunikace

V této kapitole je popsána samotná realizace obousměrné komunikace mezi zařízeními. Pro tyto účely je dostatečná tzv. „ozvěna“, která je založena na následujícím principu: z prvního zařízení jsou odeslána data, která druhé zařízení přijme a v nezměněné podobě je odešle zpět. V prvním zařízení jsou opět přijmuta a uživatelem porovnána s původní odeslanou zprávou.

3.1 Požadavky na komunikaci

Pro užívání komunikace v praxi je nutné, aby tato komunikace splňovala určitá specifika. Klademe na ni tedy určité požadavky v závislosti na konkrétní aplikaci.

Latence
Je vyžadováno, aby data mezi zařízeními mohla být přenášena při určité latenci[1], resp. frekvenci. Ta totiž nejvíce ovlivňuje možnosti případného nasazení komunikace na konkrétní aplikaci.

Pro demonstrační úlohy v této práci je potřebná nízká latence zejména u zpráv o relativně kratších délkách. Přenášeno bude totiž pouze pár informací v každém cyklu. S rostoucí délkou zprávy pak bude latence logicky narůstat.

Latenci také vnímá uživatel provozující daný systém. Pokud je příliš velká, provozování systému nemusí být uživatelsky pohodlné, pokud tedy bude takový systém vůbec funkční.

Datový tok
Určitým kritériem je také maximální datový tok. Ten nemusí dosahovat příliš vysokých hodnot, nicméně je potřebné znát jeho limity. Pokud je ověřeno, že daná komunikace zvládá odesílat a přijímat zprávy do určité délky, není vyloženě nutné zkoumat maximální možný tok. Záléží však opět na dané aplikaci.

Stabilita
V určitých systémech může být stabilita zásadní veličinou, avšak pro zaměření této práce není klíčová. V případě např. regulačních systémů pro výuku nebo informačních displejů, nebude kritické, pokud se během přenosu ztratí některá zpráva. K této situaci samozřejmě nesmí docházet opakovaně, aby výpadky nenarušily chod činnosti.

Zabezpečení komunikace
Pro zmíněné cílové realizace není zabezpečení komunikace vůbec potřebné. Přenášená data nebudou obsahovat žádné chráněné informace, a proto není potřebné komunikaci např. šifrovat. Pokud by však měl např. informační displej umožňovat přihlášení uživatele pod definovaným uživatelským jménem a heslem, bylo by již vhodné tento přenos nějakým způsobem zabezpečit.

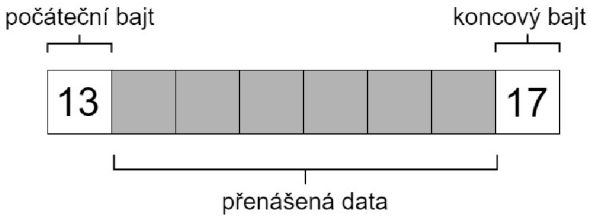
[1] Latence vyjadřuje časové zpoždění, které je nutné pro odeslání a přijetí zprávy mezi zařízeními. Někdy bývá také označována jako ping rate [20].

Komunikační protokol
Klíčové pro komunikaci obecně je rozlišení jednotlivých zpráv od sebe, rozeznání počátku a konce zprávy a následně ze zprávy převzít požadovaná data.

Pro tyto účely byl sestaven jednoduchý komunikační protokol: na počátek každé zprávy je vložen bajt signalizující počátek zprávy. Za něj je zařazena sekvence datových bajtů, po které následuje ještě jeden bajt značící konec zprávy. Schématicky znázorněná zpráva v tomto komunikačním protokolu je na obrázku 3.1.

Hodnoty počátečního a koncového bajtu, stejně jako počet datových bajtů, musí být známy oběma zařízením, aby mohla tyto bajty lokalizovat.

Komunikace pak probíhá následovně: zařízení A seřadí datové bajty, přidá počáteční a koncový bajt a zprávu odešle. Po přijetí zprávy zařízením B jsou lokalizovány počáteční a koncový bajt podle definovaných hodnot a je ověřeno, zda se mezi nimi nachází určený počet datových bajtů. Pokud je podmínka splněna jsou datové bajty vyčteny. V opačném případě je zpráva zahozena a zařízení B čeká na příjem další zprávy, kde se postup opakuje.



Obrázek 3.1: Znázornění zprávy v komunikačním protokolu.

Pro komunikaci typu „ozvěna“ samozřejmě není protokol vyžadován, stejně jako pro testování limitů komunikace. Prokol byl tedy využit následně až v demonstračních aplikacích, kde je již rozlišení dat nutné.

3.2 Komunikace ze strany mikrokontroleru

Prvním krokem ze strany MCU, resp. celé desky, by mělo být nastavení FTDI čipu. Poté lze již přistoupit k implementování konkrétního algoritmu pro mikrokontroler.

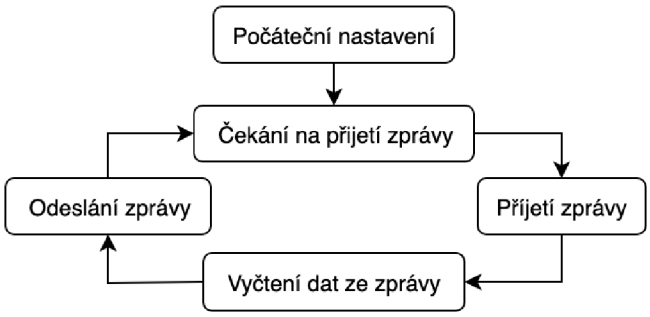
3.2.1 Nastavení FTDI čipu

Úpravu chování FTDI čipu je možné provádět pomocí programu *FT Prog*, který umožňuje programování paměti EEPROM čipu. Potřebné je nastavit vendor ID sloužící pro rozpoznání zařízení. Výrobcem je standardně nějaké VID definováno, které není nutné měnit. Je však vhodné si ho poznačit, je totiž následně potřebné při sestavování komunikace ze strany OS Android. V tomto případě bylo ponecháno VID $0x0403$ dané výrobcem.

Dalším krokem je nastavení ovladače čipu. Na výběr jsou knihovna D2XX, nebo ovladač VCP. Výchozím nastavením zde bývá VCP, nicméně byla zvola knihovna D2XX, která umožňuje otevření portu pro konkrétní zařízení na základě shody VID [21]. Měla by však být zajištěna vzájemná kompatibilita těchto ovladačů.

3.2.2 Algoritmus pro mikrokontroler

Pro účely komunikace na zmíněném principu ozvěny byl sestaven jednoduchý algoritmus pro MCU. Ten sleduje, zda nebyla přijata data a pokud tak nastane, data vyčte a odešle zpět. Algotitmus je graficky znázorněn na obrázku 3.2.



Obrázek 3.2: Algoritmus pro mikrokontroler přeposílající přijatá data.

Při programování MCU je potřebná inicializace oscilátoru, UARTu apod. Pro tyto účely byl převzat a upraven kód z výuky předmětu Aplikace embedded systémů v mechatronice na FSI dostupný z [22].

Komunikace mezi zařízeními musí být svázána stejnou baud rate pro obě zařízení, která by měla být maximální možná, aby nedocházelo k omezování. Zde bylo použito 460 800 *Bd*.

3.3 Komunikace ze strany OS Android

Zařízení s OS Android má komunikaci řídit. Implementovat je proto nutné nejen řídící algoritmus, ale i uživatelské prostředí, které umožní správu komunikace a kontrolu nad odesílanými a přijímanými daty.

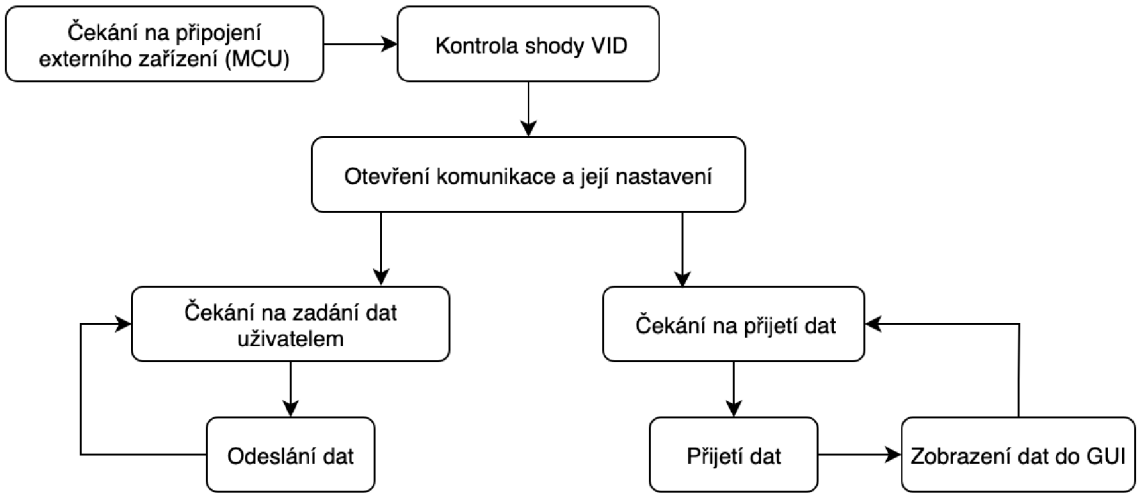
Sestavení kódu této části proběhlo podle [23]. Kód byl použit a přepracován, část kódu je následně využívána i v dalších částech této práce.

3.3.1 Algoritmus pro OS Android

Algoritmus musí provádět následující: čekat na připojení zařízení a následně ověřit shodu VID. Pokud dojde ke shodě, požádat uživatele o přístup k USB[2]. Jestliže uživatel povolení udělí, je otevřen sériový port pro komunikaci a nastaveny jeho parametry. Nastavena musí být shodná baud rate jako v mikrokontroleru, tedy 460 800 *Bd*. Algoritmus dále čeká na zadání dat uživatelem, která pak odešle do mikrokontroleru. Současně musí být schopen data přijímat. Přijatá data jsou zde vypisována do uživatelského prostředí.

Algoritmus je graficky znázorněn na obrázku 3.3. Cyklus se následně opakuje pro zadávání a odesílání dalších dat, včetně jejich přijímání.

[2]Povolení uživatelem vyžaduje Android nativně. Toto povolení je potřebné zahrnout v manifestu aplikace.



Obrázek 3.3: Algoritmus ovládající komunikaci a umožňující zadávání a odesílání dat, včetně jejich přijímání.

Uživatelské prostředí

Vytvořeno bylo také jednoduché uživatelské prostředí, skládající se z:

- informačního panelu, který zobrazuje stavy komunikace a vypisuje přijatá i odeslaná data.
- čtyř tlačítek vyvolávajících otevření komunikace, uzavření komunikace, odeslání zadaných dat a vymazání informačního panelu.
- widgetu umožňující zadávání textu.

3.3.2 Omezení komunikace

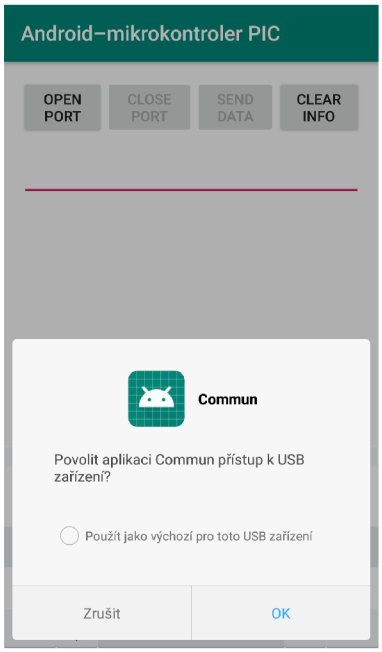
Faktem, že rozhraní UART je omezeno délkou zprávy na 8 bitů[3] je dáno i omezení komunikace na 1 bajt za zprávu. To následně omezuje odesílané a přijímané hodnoty.

Při použití znaménkového 8-bitového datového typu[4] lze přenášet hodnoty z intervalu $\langle$-128; 127$\rangle$, při neznaménkovém pak $\langle$0; 255$\rangle$. Hodnoty mimo daný interval je pak nutné rozdělit do více bajtů, odeslat jako jednotlivé bajty, a následně v koncovém zařízení opět složit do původní hodnoty. Proces rozložení a složení musí být známý oběma zařízením, aby byla konverze správná. Podobně je nutné naložit s desetinnými čísly, která 8-bitový datový typ nepojme. Zde by např. jeden bajt mohl nést informaci o desetinné čárce.

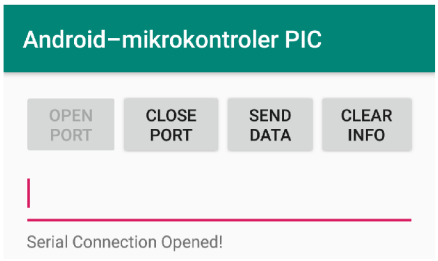
3.3.3 Ukázka realizované komunikace

Pro názornou demonstraci funkčnosti je průběh komunikace vyobrazen pomocí snímků obrazovky na obrázcích 3.4 a 3.5.

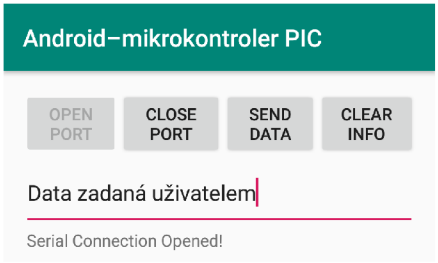
[3]Bez použití paritního bitu je možná délkaz zprávy 9 bitů.
[4]Pokud je datový typ založen na dvojkovém doplňku, tzv. two's complement.



(a) Žádání uživatele o přístup k USB probíhá pomocí vyskakovacího okna.

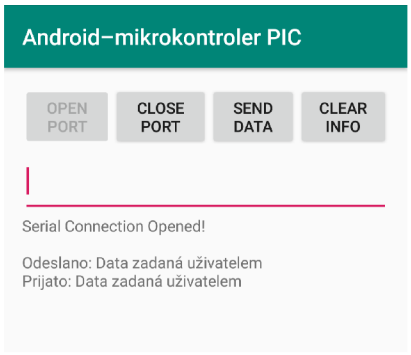


(b) Otevření komunikace na základě povolení.

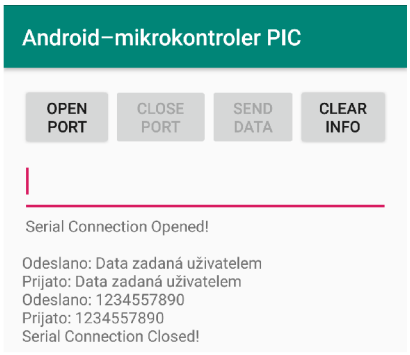


(c) Zadávání dat uživatelem.

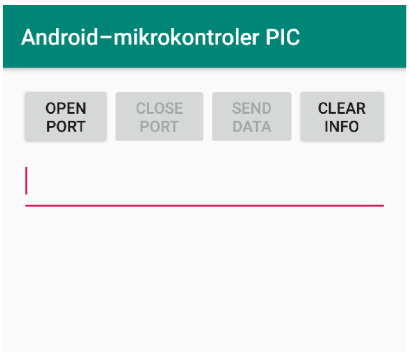
Obrázek 3.4: Ukázka realizované komunikace – otevření portu, zadání dat k odeslání, odeslání zprávy a její přijetí.



(a) Odeslání a následné přijetí zprávy.



(b) Odeslání další zprávy, její přijetí a uzavření komunikace.



(c) Vymazání informačního panelu.

Obrázek 3.5: Ukázka realizované komunikace – odeslání a přijetí druhé zprávy a vymazání informačních textů.

Toto uživatelské rozhraní má význam pouze pro testování funkčnosti komunikace mezi zařízeními a slouží tedy jako jednoduchý terminál. Konkrétní GUI bude nutné vždy vytvořit v závislosti na aplikaci.

4 Statistické testování komunikace

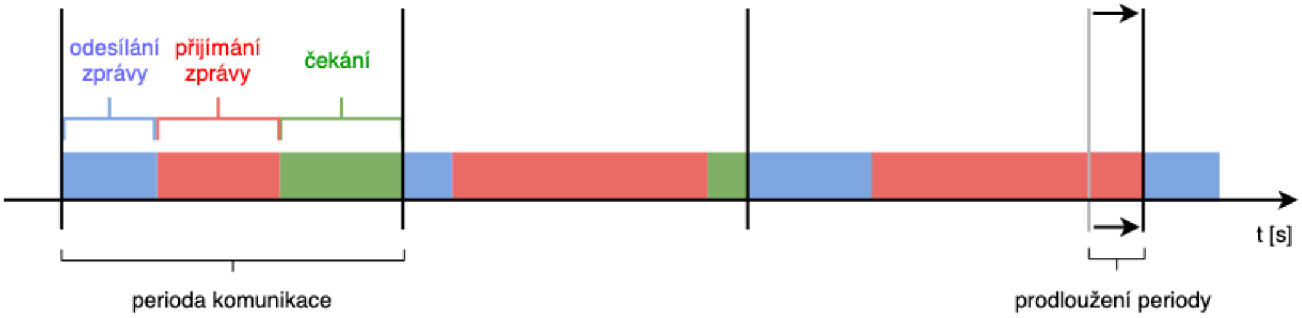
Tato práce si klade za cíl také otestovat realizovanou komunikaci, zejména její latenci. Ta totiž nejvíce ovlivňuje možnosti konkrétního nasazení komunikace na reálný systém.

4.1 Návrh testování

S ohledem na žádané nasazení komunikace na konkrétní aplikace lze konstatovat, že přenášenými daty budou obecně pouze čísla. Pro regulační systémy a informační displeje nedává příliš velký smysl přenos znaků – snahou je přenést informaci o dané veličině, která bude vyjádřena určitou hodnotou, tedy číslem.

Navrhnut byl následující test pro měření frekvence komunikace: v zařízení s OS Android bude vygenerováno pole (o předem definované délce) náhodných celých čísel z intervalu $\langle 0; 9 \rangle$ a zaznamenán čas počátku cyklu. Toto pole bude odesláno do mikrokontroleru a zaznamenán čas odeslání. Mikrokontroler přijaté pole přepošle zpět do zařízení s OS Android, kde bude opět přijato a zaznamenán čas přijetí. Z naměřeného času trvání od počátku cyklu do přijetí zprávy bude vyhodnoceno, zda je toto trvání menší jako doba, kterou má být komunikace časována[1]. Pokud je tato podmínka splněna, nastane čekání do doby, než zmíněné trvání dohromady s čekací dobou překročí zmíněné časování komunikace. Tento průběh lze zobrazit na časovou osu obrázkem 4.1. Pokud byla doba trvání větší jako časování komunikace, čekání nenastává. Tento cyklus se bude opakovat pro předem definovaný počet iterací – zvoleno bylo 100 000 iterací.

Vyhodnocena pak bude doba od odeslání zprávy po její přijetí a určeno, v kolika případech došlo k překročení doby časování komunikace dobou trvání přenosu.



Obrázek 4.1: Vyobrazení jednotlivých délek trvání na časovou osu. V některých případech se může stát, že trvání odesílání a přijímání zprávy překročí periodu komunikace, která se tímto prodlouží. Zvolením příliš krátké periody komunikace tak může docházet k jejímu prodlužování pravidelně a zvolená perioda pak není dodržována.

[1] Dobou časování je zde myšlena převrácená hodnota frekvence.

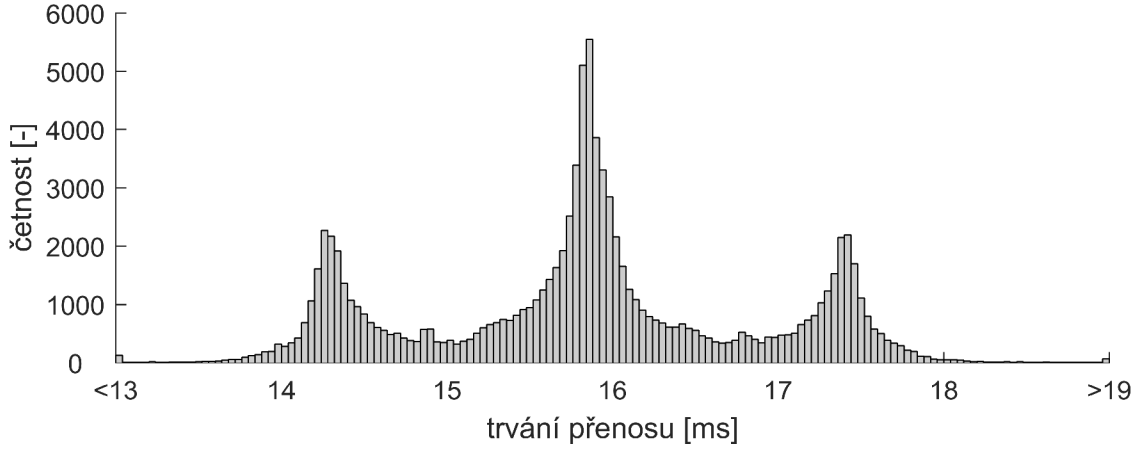
4.2 Naměřené hodnoty

Měření bylo provedeno pro tyto délky zpráv[2]: 1, 5, 10 a 50. Pro každou délku zprávy bylo měřeno několik různých dob časování. Z naměřených hodnot byly zpracovány následující histogramy.

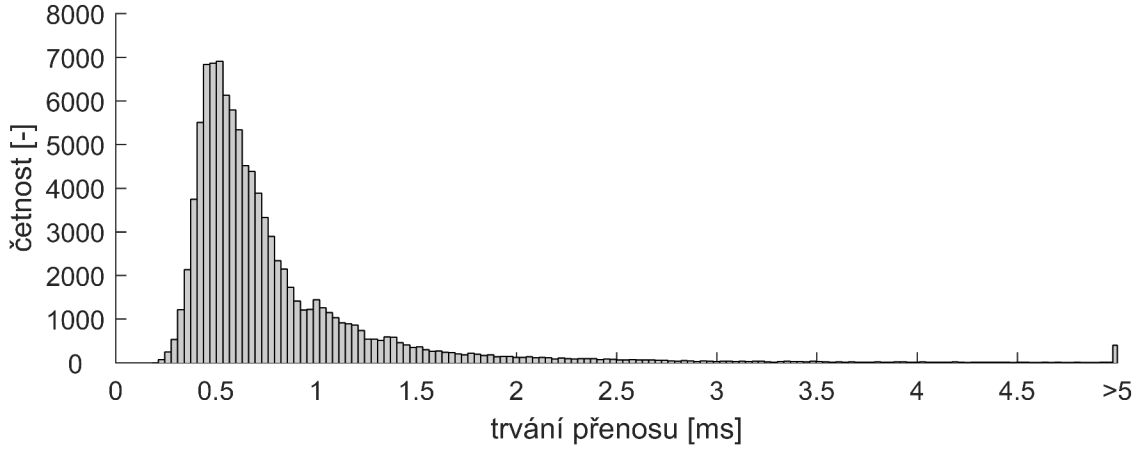
Pro názornost jsou zde uvedeny dva grafy odpovídající dvěma vybraným časováním pro každou délku zprávy. Soubor všech histogramů je přiložen v dodatku A.

4.2.1 Měření pro zprávu délky 1

Měření pro zprávu o délce 1 bylo provedeno při časování 10, 15, 16, 19 a 20 *ms*. Časováním 10 a 16 *ms* odpovídají histogramy na obrázku 4.2. Ostatní jsou uvedeny v dodatku A.1 na obrázku 6.2.



(a) časování 10 *ms*.



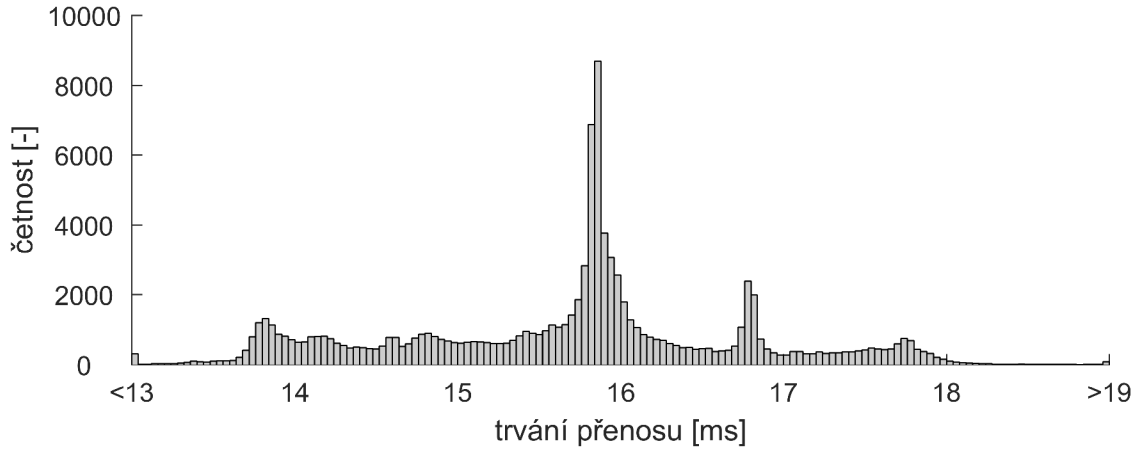
(b) časování 16 *ms*.

Obrázek 4.2: Histogramy z měření pro zprávy délky 1 při časování 10 a 16 *ms*.

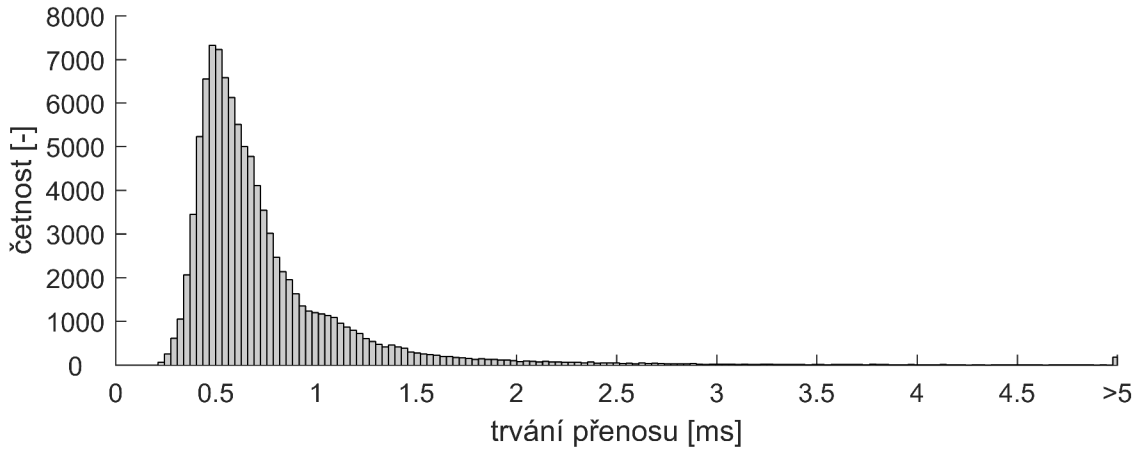
[2]Délka zprávy odpovídá počtu čísel v poli, a tedy i počtu bajtů ve zprávě.

4.2.2 Měření pro zprávu délky 5

Měření pro zprávu o délce 5 bylo provedeno při časování 15, 16, 18, 22 a 25 *ms*. Časováním 15 a 18 *ms* odpovídají histogramy na obrázku 4.3. Ostatní jsou uvedeny v dodatku A.2 na obrázku 6.3.



(a) časování 15 *ms*.

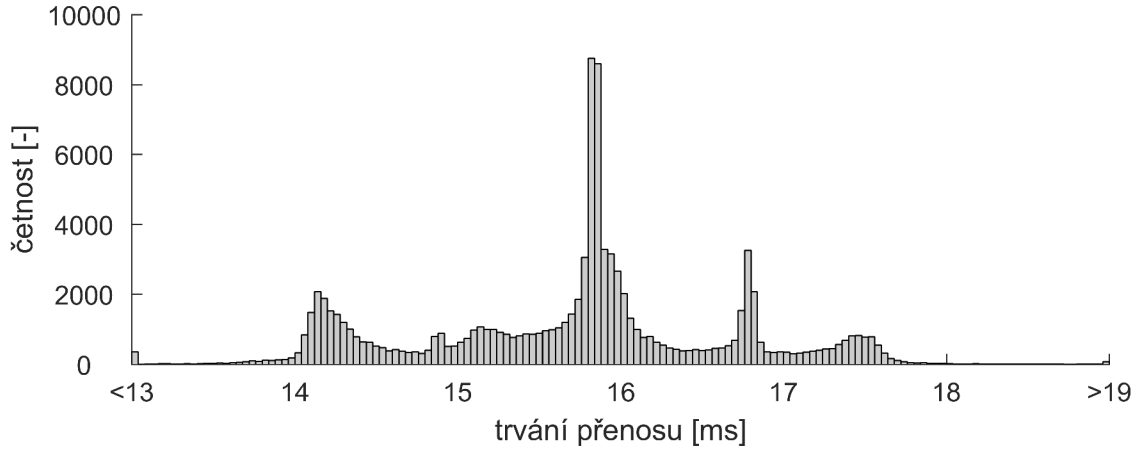


(b) časování 18 *ms*.

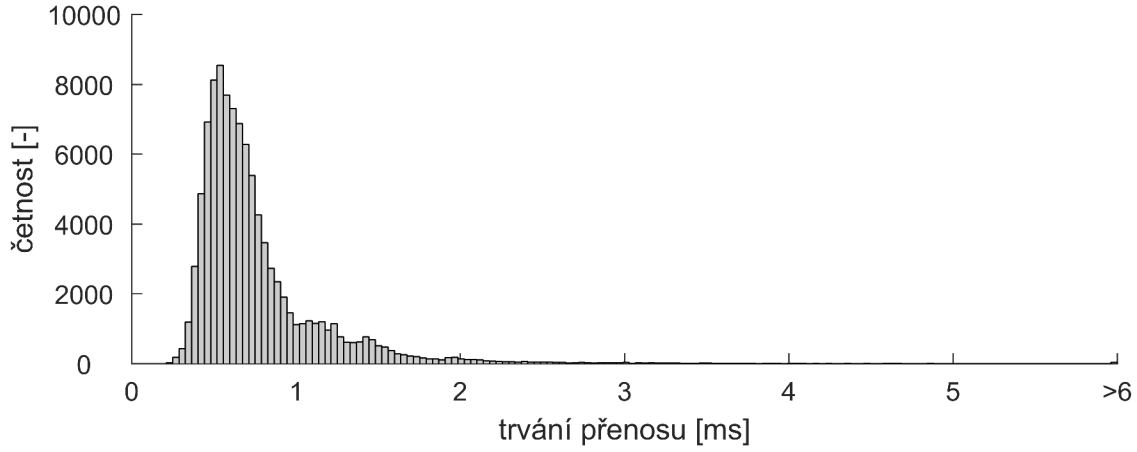
Obrázek 4.3: Histogramy z měření pro zprávy délky 5 při časování 15 a 18 *ms*.

4.2.3 Měření pro zprávu délky 10

Měření pro zprávu o délce 10 bylo provedeno při časování 15, 16, 20, 22 a 25 *ms*. Časováním 15 a 20 *ms* odpovídají histogramy na obrázku 4.4. Ostatní jsou uvedeny v dodatku A.3 na obrázku 6.4.



(a) časování 15 *ms*.

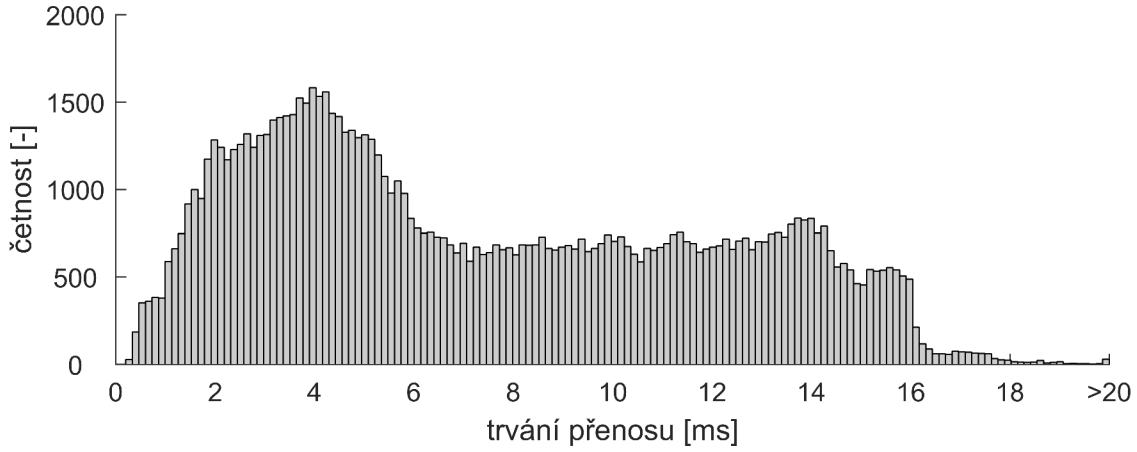


(b) časování 20 *ms*.

Obrázek 4.4: Histogramy z měření pro zprávy délky 10 při časování 15 a 20 *ms*.

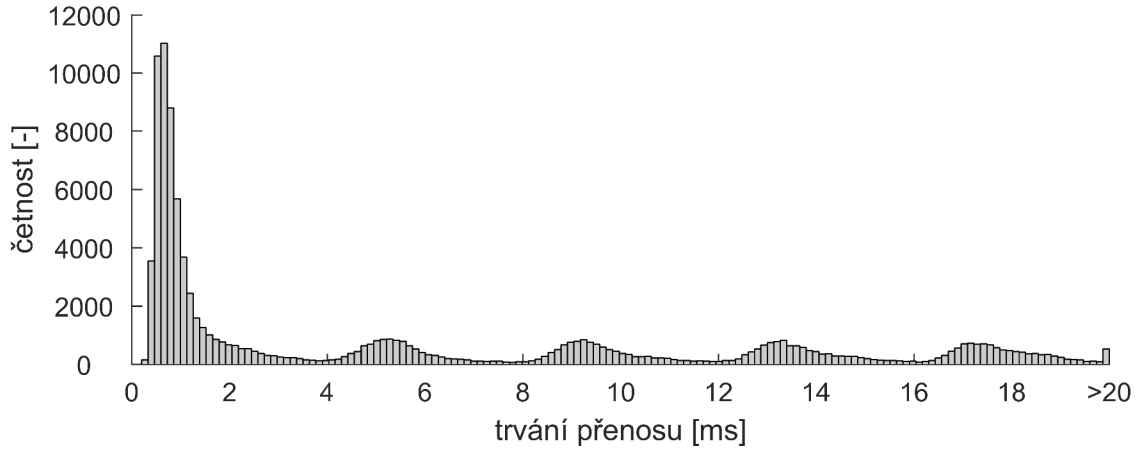
4.2.4 Měření pro zprávu délky 50

Měření pro zprávu o délce 50 bylo provedeno při časování 15, 16, 20, 25 a 35 *ms*. Časováním 16 a 20 *ms* odpovídají histogramy na obrázku 4.5. Ostatní jsou uvedeny v dodatku A.4 na obrázku 6.5.



(a) časování 16 *ms*.

Obrázek 4.5: Histogramy z měření pro zprávy délky 50 při časování 16 a 20 *ms*.



(b) časování 20 *ms*.

Obrázek 4.5: Histogramy z měření pro zprávy délky 50 při časování 16 a 20 *ms*.

4.3 Vyhodnocení testů

Ve zmíněných histogramech je podle očekávání ve většině případech vidět normální rozdělení, které se posouvá v závislosti na zvoleném časování.

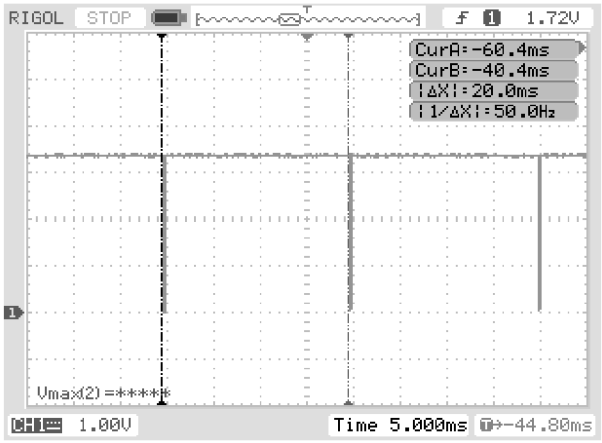
Poměrně překvapivé je rozestoupení rozdělení u všech délek zpráv časovaných pod 16 *ms*. Zde navíc dochází ke skokové změně trvání přenosu mezi časováním 15 a 16 *ms*. Tato skoková změna se opakuje u všech testovaných délek zpráv, nejedná se tedy o chybu jednoho měření. Lze se domnívat, že kritickou hranicí pro funkčnost komunikace je právě zmíněných 16 *ms*, nicméně podrobnosti tohoto přechodu se nepodařily odhalit.

Další anomálie s prokázala u měření zprávy s délkou 50. Zde se od určité frekvence objevuje jistá periodicita, která však s větším časováním klesá.

Pro zmíněné aplikace bude pravděpodobně ve většině případů dostačující délka zprávy do 10 znaků, nebo ne výrazně větší. V takovém případě by bylo vhodné využít časování do 20 *ms*, kdy je drtivá většina zpráv přijata do 3 *ms*, jak lze vidět na obrázku 4.4b. Časování 20 *ms* pak od-

povídá 50 *Hz*, což by měla být dostatečná frekvence pro regulační aplikace pro výuku apod. Do této frekvence lze tedy považovat systém za soft RT.

Toto časování bylo také ověřeno měřením na osciloskopu pro kontrolu, zda Android opravdu časuje zmíněných 20 *ms*. Ověření prokázalo, že tomu tak skutečně je, zobrazení na osciloskopu je na obrázku 4.6.



Obrázek 4.6: Ověření časování osciloskopem.

Pro větší délky zpráv je pak nutné volit delší časování, tedy nižší frekvenci, kterou je možné přibližně odhadovat z uvedených histogramů. Pro délku zprávy o délce blížící se 50 *B* je však volba frekvence problematická, kvůli poměrně vysoké četnosti ve zmíněné periodicitě. Rozhodně by toto časování nemělo klesnout pod 35 *ms*.

5 Demonstrační úlohy

V rámci této bakalářské práce byly vytvořeny dvě ukázkové aplikace – první umožňující regulovat polohu, resp. natočení daného motoru a druhá sloužící jako informační displej k vozítku segway z mechatronické laboratoře.

Tyto dvě aplikace názorně demonstrují funkčnost realizované komunikace na konkrétních zařízeních s cílem ověřit možnosti jejího nasazení v praxi. Aplikace je možné dále rozvíjet přidáváním dalších funkcionalit a rozšířit tak použitelnost nebo využít pouze některou z jejich částí pro jiné účely.

Pro obě aplikace byl použit vytvořený komunikační protokol ze sekce 3.1 – před samotným odesláním zprávy jsou mezi počáteční a koncový bajt vloženy datové bajty. Při přijetí zprávy jsou pak indentifikovány tyto datové bajty, se kterými je dále pracováno podle informace, jež přenášejí.

5.1 Výuková aplikace umožňující regulaci polohy motoru

Jak bylo zjištěno v kapitole 4, komunikace mezi zařízením s OS Android a daným mikrokontrolerem PIC může být provozována na frekvenci 50 *Hz*, která je dostatečná pro regulační systémy. Díky tomu může být regulátor polohy motoru této aplikace realizován v zařízení s OS Android, což přináší jistou výhodu při programování – hodnoty daných veličin můžou být zobrazovány přímo do uživatelského prostředí ihned po dokončení výpočtu.

Pro účely této aplikace byla vyžita výuková platforma DoubleDrive z mechatronické laboratoře. Tato platforma nabízí připojení ke dvěma stejnosměrným motorům (obrázek 5.1) s inkrementálními enkodéry, které jsou pro zmíněnou regulaci rovněž vyžadovány. Řízení motoru je prováděno pomocí PWM signálu, který na motor přivádí napětí v rozsahu $\pm 24\ V$.

Regulace zde má být realizována pomocí PID regulátoru, nicméně pro regulaci polohy je plně dostačující pouze P složka. Realizován je tedy pouze proporcionální regulátor, resp. regulátor PID s nulovou integrační a derivační složkou.

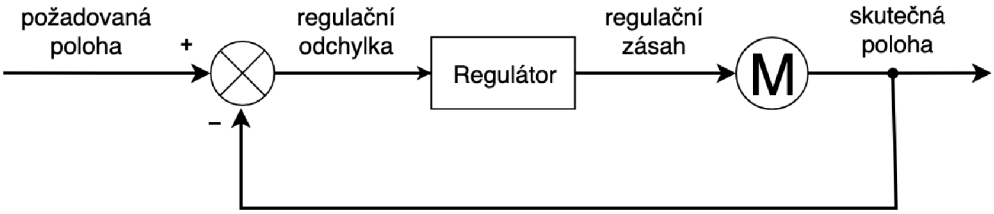
Princip regulace je následující: enkodér snímá aktuální natočení motoru, která je odečtena od požadované polohy. Tento rozdíl, označovaný jako regulační odchylka, je přenásoben zesílením regulátoru (hodnota P složky) a jako regulační zásah vstupuje do motoru ve formě napětí (pomocí střídy PWM). Tato regulační smyčka je znázorněna graficky na obrázku 5.2.

5.1.1 Mikrokontroler a připojení na motor

Mikrokontroler v této aplikaci hraje roli prostředníka mezi motorem a zařízením s OS Android, které ovládá uživatel. Úkolem použitého mikrokontroleru[1] je získávat informaci o skutečném

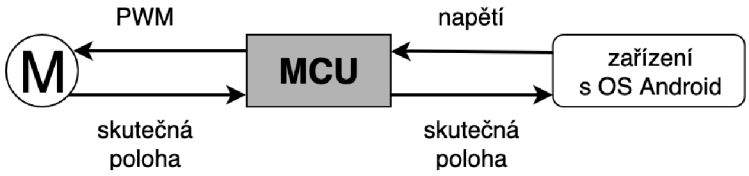
[1]Pro tuto aplikaci byl použit 16-bitový mikrokontroler *dsPIC* zmíněný v sekci 2.6.2.

Obrázek 5.1: Platforma DoubleDrive se stejnosměrnými motory.



Obrázek 5.2: Regulační smyčka pro regulaci polohy motoru.

natočení motoru a následně ji přeposílat do zařízení s OS Android. Současně musí MCU přijímat hodnotu napětí, zaslanou zařízením s OS Android, převést ji na odpovídající střídu signálu PWM a tímto signálem řídit motor. Schematicky lze přenos informací znázornit obrázkem 5.3.



Obrázek 5.3: Znázornění přenosu informací mezi motorem, mikrokontrolerem a zařízením s OS Android.

Jak bylo již zmíněno v podsekci 2.4.3, mikrokontroler dsPIC je možno programovat v prostředí Simulink s následným generováním kódu do jazyka C. Této možnosti je zde využito a programování mikrokontroleru pro řízení motoru proběhlo touto formou.

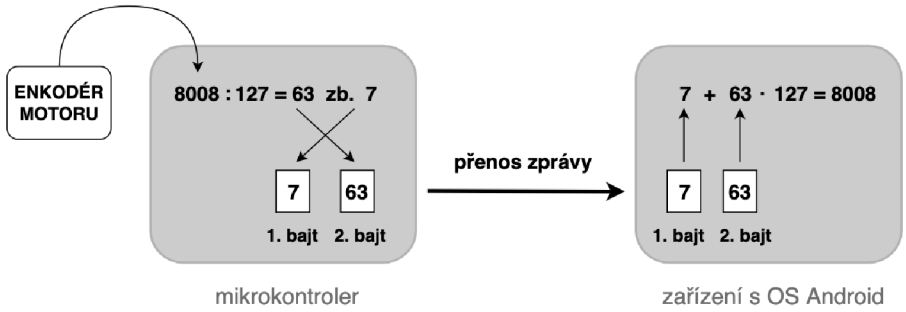
Jelikož je enkodér motoru inkrementální, dává informaci o natočení formou čísla, které se buď zvyšuje při otáčení na jednu stranu, nebo snižuje otáčením na stranu druhou. Toto číslo může dosahovat poměrně vysoké hodnoty, která s jistotou přesahuje interval $\langle$-128; 127$\rangle$, zmíněný v podsekci 3.3.2, a nelze ji tak uložit do 8-bitového datového typu, se kterým je UART schopen pracovat. Proto je nutné číslo rozdělit uložením do více bajtů – zvoleny jsou zde dva bajty a jednoduchý způsob rozdělení čísla pomocí dělení se zbytkem, popsaný níže.

Rozdělení čísla mezi dva bajty dělením se zbytkem

Číslo, které je potřebné rozdělit, je vyděleno se zbytkem číslem 127 a výsledek podělen opět číslem 127. Následně se zbytek po dělení uloží do prvního bajtu a podíl druhého dělení do bajtu druhého. Tyto dva bajty jsou přeneseny do druhého zařízení a zde složeny opačným způsobem. Tedy hodnota z druhého bajtu je vynásobena číslem 127 a k ní přičtena hodnota z bajtu prvního.

Pro názornost uvažujme následující příklad: enkodér poskytuje informaci o natočení číslem 8 008. Následuje dělení se zbytkem: $8\,008 : 127 = 63\ zb.\ 7$ a uložení čísel 7 do prvního a 63 do druhého bajtu. Bajty jsou odeslány z mikrokontroleru do zařízení s OS Android (musí být zachováno pořadí) a zde provedeno zpětné složení: $7 + 63 \cdot 127 = 8\,008$. Graficky lze operaci vyjádřit znázornit obrázkem 5.4. Důležité je také poznamenat, že nezáleží na pořadí ukládání podílu a zbytku do jednotlivých bajtů. Nutné je však dodržet stejné pořadí při rozkladu i skládání čísla.

Touto metodou lze přenášet čísla do maximální hodnoty 16 255 včetně a zároveň je použitelná i pro záporné hodnoty. Díky tomu lze posílat hodnoty z intervalu $\langle$-16 256; 16 255$\rangle$, což je již pro tuto aplikaci dostatečné.



Obrázek 5.4: Znázornění rozdělení čísla do dvou bajtů, jejich přenos do druhého zařízení a opětovné složení do původního čísla.

5.1.2 Realizace regulátoru v OS Android

Zařízení s OS Android po celou dobu chodu aplikace přijímá informaci o skutečném natočení motoru a současně získává hodnoty požadovaného natočení a zesílení regulátoru od uživatele[2]. Z těchto hodnot je vypočítáván regulační zásah, který je přepočten na napětí a to odesláno do mikrokontroleru. Přepočet na napětí je pak pouze jednoduchá trojčlenka, kde maximální regulační zásah (dán saturací) odpovídá 24 V.

Stejný problém „velkého čísla“ jako u enkodéru nastává i v případě regulačního zásahu, který bude také nabývat relativně vysokých hodnot. I zde je využita zmíněná metoda rozdělení do dvou bajtů a nastavena saturace regulačního zásahu na maximální hodnotu, kterou je možné zmíněnými dvěmi bajty přenést, tedy $\pm 16\,255$[3].

Frekvence odesílání jednotlivých zpráv je zde nastavena na zmíněných 50 Hz a je snaha tuto frekvenci dodržet oboustranně – odesílání zprávy mikrokontrolerem je vázáno na přijetí

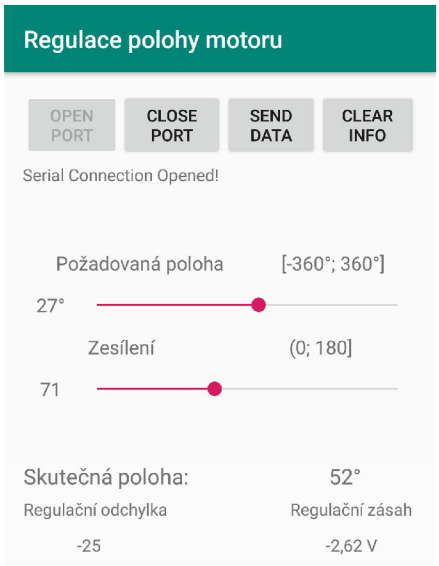
[2]Dokud uživatel hodnoty nezadá, jsou ve výchozím bodě nastaveny nulová požadovaná poloha a zesílení rovno jedné.

[3]Pro symetričnost intervalu regulačního zásahu, je saturací interval oříznut z možných $\langle$-16 256; 16 255$\rangle$ na $\langle$-16 255; 16 255$\rangle$

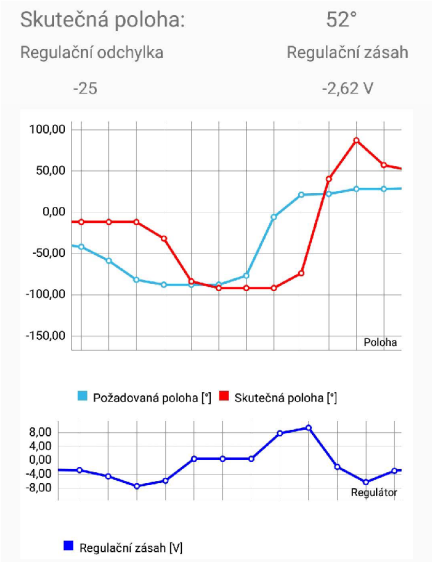
zprávy ze zařízení s OS Android. Tedy v okamžiku přijetí zprávy mikrokontrolerem a vyčtení datových bajtů z ní, je mikrokontrolerem odeslána zpráva nesoucí informaci o natočení motoru. Osciloskopem však bylo ověřeno, že tato vazba není uskutečněna a odesílání z MCU probíhá na odlišné frekvenci. Tento problém spočívá v chybě v algoritmu pro mikrokontroler (správnost časování komunikace ze zařízení s OS Android byla ověřena), konkrétní příčina však odhalena nebyla.

5.1.3 Uživatelské prostředí

Pro účely regulace musí aplikace v zařízení s OS Android poskytovat nejen změnu parametrů a ovládání komunikace, ale také zobrazovat některé veličiny a vykreslovat grafy. K tomu slouží vytvořené uživatelské prostředí, sestávající opět z tlačítek pro ovládání komunikace, informačního panelu, dvou posuvníků pro změnu parametrů požadované polohy a zesílení regulátoru, dvou grafů vykreslující průběh požadované i skutečné polohy a regulačního zásahu[4]. Dále jsou uživateli zobrazovány hodnoty skutečné polohy, regulačního odchylky a regulačního zásahu. Ukázka uživatelského prostředí aplikace je na obrázku 5.5.



(a) aplikace zobrazuje základní veličiny, změna parametrů je možná dvěma posuvníky



(b) aplikace vykresluje grafy požadované i skutečné polohy a regulačního zásahu živě

Obrázek 5.5: Ukázka z uživatelského prostředí aplikace pro regulaci polohy motoru.

Ovládání aplikace je pak následující: uživatel otevře komunikaci stisknutím tlačítka „OPEN PORT“ a poté tlačítkem „SEND DATA“ spustí výpočet regulátoru, přenos dat a vykreslování grafů. Motor uživatel spouští až nyní, kdy je komunikace otevřena, aby nedocházelo ke konfliktu při žádání o povolení k přístupu k USB apod. Poté již reguluje natočení motoru změnou parametrů požadované polohy a zesílení regulátoru, které mění implementovanými posuvníky.

[4]Při tvorbě grafů bylo postupováno podle [24].

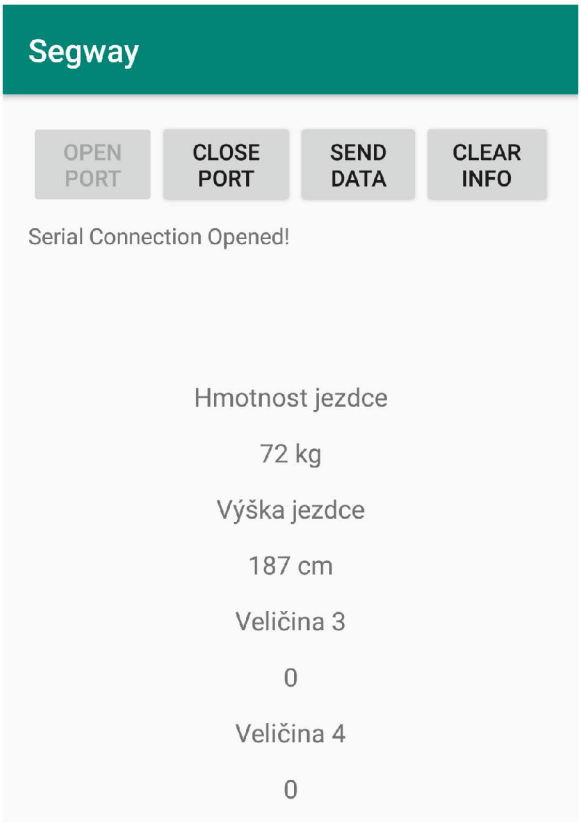
5.2 Aplikace pro informační displej vozítka Segway

Vozítko typu segway z mechatronické laboratoře je také řízeno mikrokontrolerem dsPIC, což umožnilo nasazení stejné komunikace jako v předchozí aplikaci. Rozdíl je zde pouze v počtu přenášených datových bajtů ve zprávě – zvoleny zde byly čtyři. Tento počet je samozřejmě možné kdykoliv upravit dle potřeby.

V tomto případě neslouží zařízení s OS Android jako řídící prvek systému (tím je mikrokontroler dsPIC), ale pouze jako informační displej. Proto není v aplikaci implementován žádný výpočet, dochází pouze k přijetí zprávy, vyčtení datových bajtů a zobrazení hodnot do uživatelského rozhraní.

V době psaní této práce nebylo vozítko připraveno na změnu parametrů uživatelem, proto nedochází k žádnému odesílání dat ze zařízení s OS Android, to pouze přijímá data zaslaná mikrokontrolerem. Při testování funkčnosti byly odesílány pouze hmotnost a výška jezdce, jelikož vozítko nebylo upraveno na vyhodnocování veličin jako je rychlost, nebo stav baterie.

Uživatelské prostředí aplikace je podobné jako v minulé aplikaci – opět byla implementována tlačítka pro ovládání komunikace a přidán byl nově informační panel zobrazující přijaté veličiny. Prostředí je zobrazeno na obrázku 5.6.



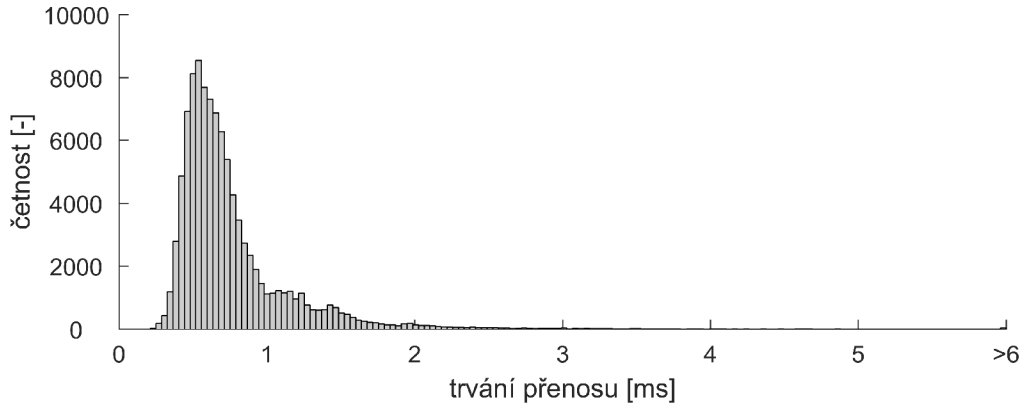
Obrázek 5.6: Ukázka uživatelského prostředí aplikace, sloužící jako informační displej vozítka segway. Uživateli se zobrazují přijaté hodnoty.

6 Závěr

Hlavním záměrem této práce bylo realizovat obousměrnou komunikaci mezi zařízením s OS Android a mikrokontrolerem PIC, otestovat její možnosti pro nasazení na konkrétní systémy a demonstrovat funkčnost tohoto přenosu na ukázkových aplikacích.

Teoretická studie této práce se zabývá jednotlivými nástroji a periferiemi potřebnými pro sestavení komunikace mezi zmíněnými zařízeními a představuje také jednotlivá vývojová prostředí, ve kterých je možné tato zařízení programovat. Dále se zaměřuje na princip činnosti rozhraní UART, pojem real-time systém a uvádí rozdělení a některé příklady takových systémů.

Po testování komunikace, které probíhalo pro různá časování a několik délek zprávy, bylo vyhodnoceno, že maximální možnou frekvencí přenosu dat je 50 Hz pro zprávy do velikosti deseti bajtů (obrázek 6.1). Poměrně překvapivé je také zjištění, že při přechodu mezi časováním 15 a 16 Hz dochází ke skokové změně měřené doby trvání přenosu a výskytu periodicity v trvání přenosu při délce zprávy 50 B. V tomto směru by bylo možné na práci navázat zaměřením se na měření časových úseků v Androidu obecně a detailním přezkoumáním tohoto skokového přechodu.



Obrázek 6.1: Ukázka z měření latence komunikace při frekvenci 50 Hz. Histogram četnosti přenesených zpráv ukazuje, že naprostá většina zpráv je přenesena do 3 ms.

Realizovaná komunikace byla využita ve dvou demonstračních úlohách – ve výukové aplikaci, umožňující regulaci natočení motoru, a také v aplikaci sloužící jako informační displej, zobrazující veličny z vozítka segway z mechatronické laboratoře.

Aplikace určená pro regulaci polohy motoru umožňuje regulaci změnou parametrů jednak požadované polohy, a také velikosti zesílení regulátoru. Do uživatelského prostředí jsou také zobrazovány hodnoty některých veličin a živě vykreslovány grafy skutečné a požadované polohy, společně s mírou regulačního zásahu.

V době psaní této práce nebylo vozítko segway připraveno na změnu parametrů ze strany uživatele. Proto aplikace aktuálně slouží pouze jako informační displej bez možnosti změny parametrů. Použitá komunikace oboustranný přenos ale umožňuje – po úpravě vozítka bude tedy změna parametrů možná. Na práci zde lze opět navázat rozšířením aplikace o tuto mož-

nost, a např. úpravou grafického uživatelského prostředí zobrazující rychlost vozítka v podobě tachometru.

Závěrem lze konstatovat, že v souladu se zpracovanou rešeršní částí této práce byla realizována obousměrná komunikace mezi zařízením s OS Android a mikrokontrolerem PIC. Bylo prokázáno, že tato komunikace může být spolehlivě provozována na frekvenci 50 Hz pro zprávy do velikosti 10 B a do této frekvence ji lze považovat za soft RT, což názorně ukazují i představené demonstrační aplikace.

Seznam zkratek a symbolů

CPU Central Processing Unit

DC Direct Current

DPS Deska Plošných Spojů

EEPROM Electrically Erasable Programmable Read-Only Memory

FSI Fakulta Strojního Inženýrství

FTDI Future Technology Devices Internatioal

GPRS General Packet Radio Service

GPS Global Positioning System

GSM Global System for Mobile Communications

GUI Graphical User Interface

ID Identifier

IDE Integrated Development Enviroment

LCD Liquid Crystal Display

LED Light-Emitting Diode

MCU Microcontroller Unit

OS Operační systém

PC Personal Computer

PID Proportional Integral Derivative

PLC Programmable logic computer

PWM Pulse Width Modulation

RC Remote controlled

RFID Radio Frequency Identification

RT Real Time

RX Receiver

TX Transmitter

UART Universal Asynchronous Receiver/Transmitter

USB Universal Serial Bus

VCP Virtual Com Port

VID Vendor ID

WIFI Wireless Fidelity

XML Extensible Markup Language

Seznam obrázků

1.1 Tabletem řízený manipulátor [25] 9

2.1 Přenos zprávy pomocí UARTu. 11
2.2 Konektory USB používané v telefonech a tabletech s OS Android. 12
2.3 Vývojové prostředí MPLAB. 14
2.4 Programování mikrokontroleru pomocí programátoru PICkit [28]. 14
2.5 Ukázka sestavování GUI v grafickém editoru vývojového prostředí Android Studio. 15
2.6 Popis desky s 8-bitovým mikrontrolerem PIC. 17
2.7 Popis desky s 16-bitovým mikrontrolerem dsPIC. 18

3.1 Znázornění zprávy v komunikačním protokolu. 20
3.2 Algoritmus pro mikrokontroler přeposílající přijatá data. 21
3.3 Agoritmus pro OS Android. 22
3.4 Ukázka prostředí realizované komunikace. 23
3.5 Ukázka funkčnosti realizované komunikce. 23

4.1 Jednotlivé délky trvání komunikace v zobrazení na časové ose. 24
4.2 Histogramy z měření pro zprávy délky 1 při časování 10 a 16 *ms*. 25
4.3 Histogramy z měření pro zprávy délky 5 při časování 15 a 18 *ms*. 26
4.4 Histogramy z měření pro zprávy délky 10 při časování 15 a 20 *ms*. 27
4.5 Histogramy z měření pro zprávy délky 50 při časování 16 a 20 *ms*. 28
4.5 Histogramy z měření pro zprávy délky 50 při časování 16 a 20 *ms*. 28
4.6 Ověření časování osciloskopem. 29

5.1 Platforma DoubleDrive se stejnosměrnými motory. 31
5.2 Regulační smyčka pro regulaci polohy motoru. 31
5.3 Znázornění přenosu indormací mezi zařízeními. 31
5.4 Ukázka rozdělení a skládání čísla do více bajtů. 32
5.5 Ukázka z uživatelského prostředí aplikace pro regulaci polohy motoru. 33
5.6 Ukázka uživatelského prostředí pro informační displej. 34

6.1 Ukázka z měření latene komunikace. 35
6.2 Histogramy z měření pro zprávy délky 1. 42
6.2 Histogramy z měření pro zprávy délky 1 (pokračování). 43
6.3 Histogramy z měření pro zprávy délky 5. 44
6.3 Histogramy z měření pro zprávy délky 5 (pokračování). 45
6.4 Histogramy z měření pro zprávy délky 10. 46
6.4 Histogramy z měření pro zprávy délky 10 (pokračování). 47
6.5 Histogramy z měření pro zprávy délky 50. 48
6.5 Histogramy z měření pro zprávy délky 50 (pokračování). 49

Literatura

[1] UART. In: DIGILENT [online]. [cit. 2019-05-13]. Dostupné z: https://reference.digilentinc.com/learn/fundamentals/communication-protocols/uart/start

[2] Basics of UART. In: OpenLab Pro [online]. [cit. 2019-05-15]. Dostupné z: https://openlabpro.com/guide/basics-of-uart/

[3] BASICS OF UART COMMUNICATION. In: Circuit Basics [online]. [cit. 2019-05-15]. Dostupné z: http://www.circuitbasics.com/basics-uart-communication/

[4] WANG, Y. a K. SONG. A new approach to realize UART. In: Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology. 2011, 5, s. 2749-2752. DOI: 10.1109/EMEIT.2011.6023602. ISBN 978-1-61284-088-8.

[5] Basics of UART Communication. In: ELECTRONICS HUB [online]. June 17, 2017 [cit. 2019-05-22]. Dostupné z: https://www.electronicshub.org/basics-uart-communication/

[6] The Universal Serial Bus: How it Works and What it Does. In: Geoffknagge [online]. [cit. 2019-05-15]. Dostupné z: https://www.geoffknagge.com/uni/elec101/essay.shtml

[7] BRAIN, Marshall. How USB Ports Work. In: How stuff works [online]. [cit. 2019-05-15]. Dostupné z: https://computer.howstuffworks.com/usb2.htm

[8] FISHER, Tim. USB: Everything You Need to Know. In: Lifewire [online]. 2019 [cit. 2019-05-15]. Dostupné z: https://www.lifewire.com/universal-serial-bus-usb-2626039

[9] JUVVA, Kanaka. Real-Time Systems. In: Electrical and Computer Engineering [online]. 1998 [cit. 2019-05-23]. Dostupné z: https://users.ece.cmu.edu/~koopman/des_s99/real_time/

[10] KOREN, Israel a C. Mani KRISHNA. CHAPTER 6 - Checkpointing. Fault-Tolerant Systems. Burlington: Morgan Kaufmann, 2007, s. 220. ISBN 978-0-12-088525-1. Dostupné také z: http://www.sciencedirect.com/science/article/pii/B9780120885251500092

[11] What is Real-Time Operating System (RTOS) and How It works?. In: ELPROCUS [online]. [cit. 2019-05-15]. Dostupné z: https://www.elprocus.com/real-time-operating-system-rtos-and-how-it-works/

[12] Operating System | Real time systems. In: Geeks for Geeks [online]. [cit. 2019-05-15]. Dostupné z: https://www.geeksforgeeks.org/operating-system-real-time-systems/

[13] Application Fundamentals. In: Android Developers [online]. [cit. 2019-05-16]. Dostupné z: https://developer.android.com/guide/components/fundamentals

[14] App Manifest Overview. In: Android Developers [online]. [cit. 2019-05-16]. Dostupné z: https://developer.android.com/guide/topics/manifest/manifest-intro

[15] Folder and File Structure. In: BrainBell [online]. [cit. 2019-05-16]. Dostupné z: https://www.brainbell.com/android/file-folder-structure.html

[16] Microchip dsPIC Microcontrollers Support from Embedded Code. In: MathWorks [online]. [cit. 2019-05-16]. Dostupné z: https://www.mathworks.com/hardware-support/microchip-dspic.html

[17] Usb serial controller for Android. In: GitHub [online]. [cit. 2019-05-16]. Dostupné z: https://github.com/felHR85/UsbSerial

[18] JAHODA, Philipp. MPAndroidChart. In: GitHub [online]. [cit. 2019-05-17]. Dostupné z: https://github.com/PhilJay/MPAndroidChart

[19] FT231X – Full Speed USB to Full Handshake UART. In: FTDI Chip [online]. [cit. 2019-05-17]. Dostupné z: https://www.ftdichip.com/Products/ICs/FT231X.html

[20] What is Latency?. In: Plug Things In [online]. [cit. 2019-05-17]. Dostupné z: http://www.plugthingsin.com/internet/speed/latency/

[21] POWRIE, Don L. VCP vs. D2XX. In: DLP Design [online]. 2011 [cit. 2019-05-18]. Dostupné z: https://www.dlpdesign.com/tnt/VCP%20Vs%20D2XX.pdf

[22] PIC18, UART + STDIO. In: SRC.ATHAJ [online]. [cit. 2019-05-20]. Dostupné z: http://src.athaj.cz/teaching/rev/part7-2018

[23] MATVAN, Hariharan. Communicate with Your Arduino Through Android. In: ALL ABOUT CIRCUITS [online]. 2015 [cit. 2019-05-19]. Dostupné z: https://www.allaboutcircuits.com/projects/communicate-with-your-arduino-through-android/

[24] SAUREL, S. Create a real time line graph in Android with MPAndroidChart. In: SSAUREL [online]. [cit. 2019-05-21]. Dostupné z: https://www.ssaurel.com/blog/create-a-real-time-line-graph-in-android-with-mpandroidchart/

[25] Tablet controlled robot. In: NEW ATLAS [online]. [cit. 2019-05-12]. Dostupné z: https://img.newatlas.com/universal-robots-arm-5.jpg?auto=format%2Ccompress&fit=max&q=60&w=1000&s=3b73f1b92eb00cf6a7b9a27035409adf

[26] BASICS OF UART COMMUNICATION. In: Circuit Basics [online]. [cit. 2019-05-13]. Dostupné z: http://www.circuitbasics.com/wp-content/uploads/2016/01/Introduction-to-UART-Data-Transmission-Diagram.png

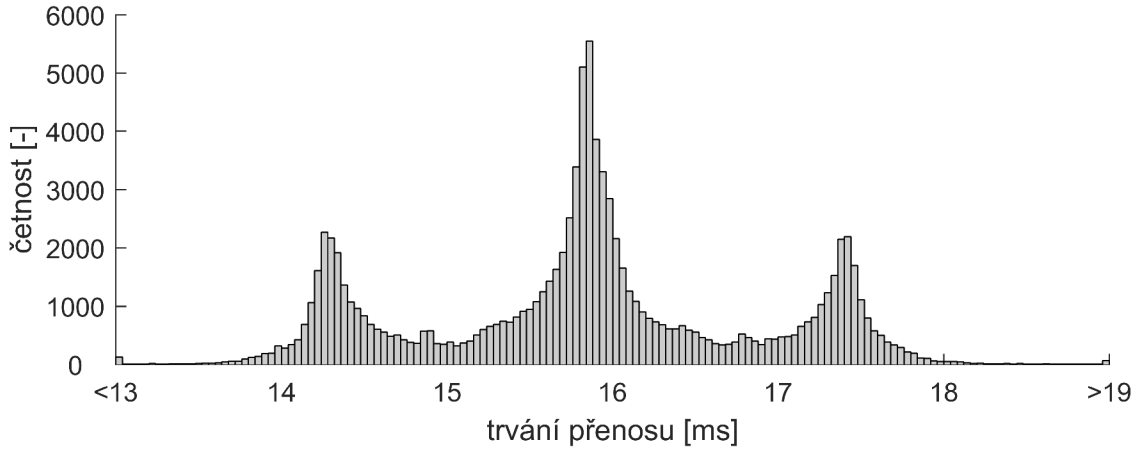
[27] USB 3.1 Type-C Male to microUSB Male Cable. In: USB Brando [online]. [cit. 2019-05-15]. Dostupné z: https://usb.brando.com/prod_img/zoom/UCABL021600_1.jpg

[28] Microcontroller In Circuit Serial Programming (ICSP) with Microchip PIC and Atmel AVR. In: Lirtex [online]. [cit. 2019-05-16]. Dostupné z: http://www.lirtex.com/wp-content/uploads/2012/02/PIC_ICSP_4801.jpg

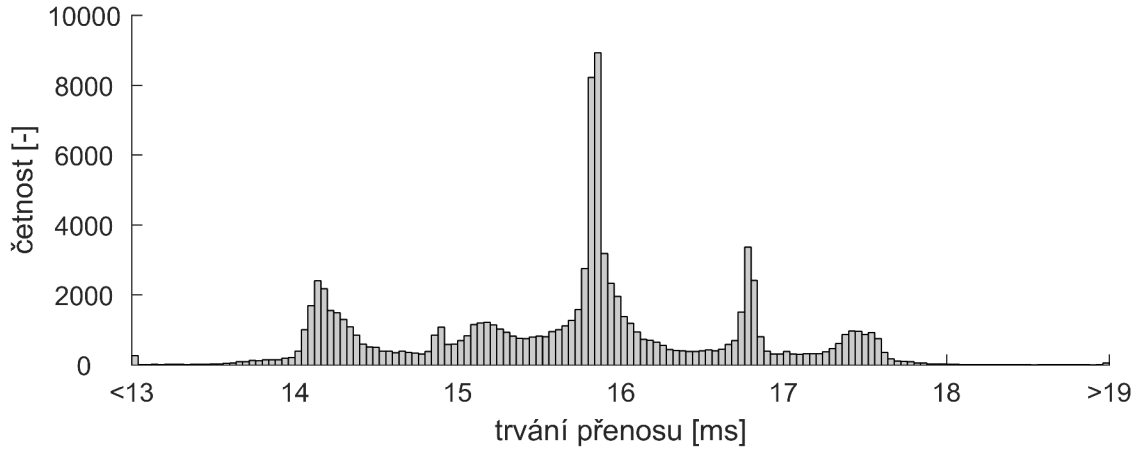
Dodatek

A Histogramy z naměřených dat

A.1 Měření pro zprávu délky 1

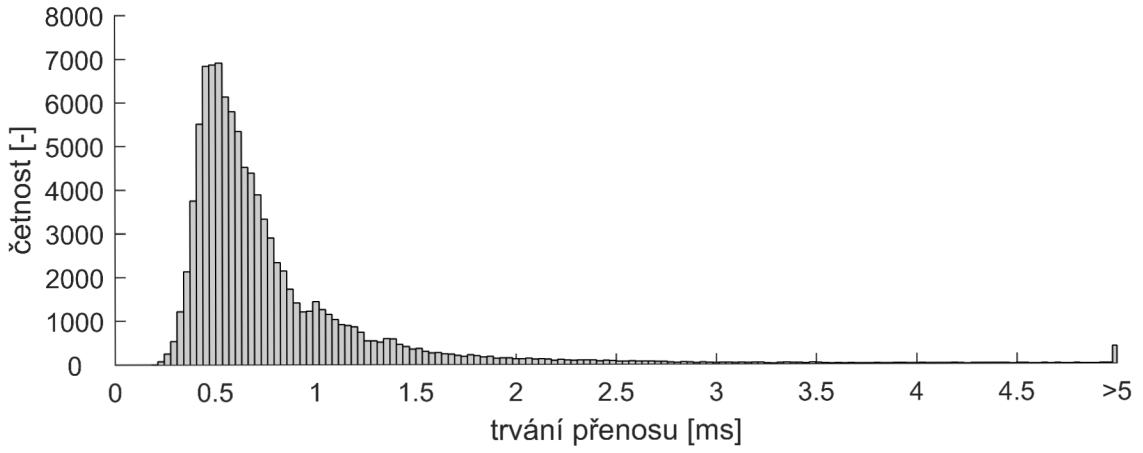


(a) časování 10 *ms*.

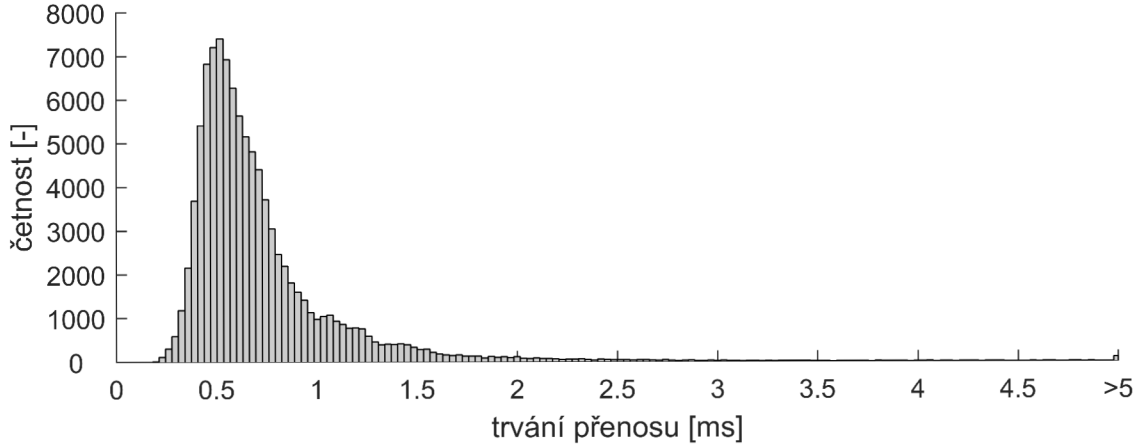


(b) časování 15 *ms*.

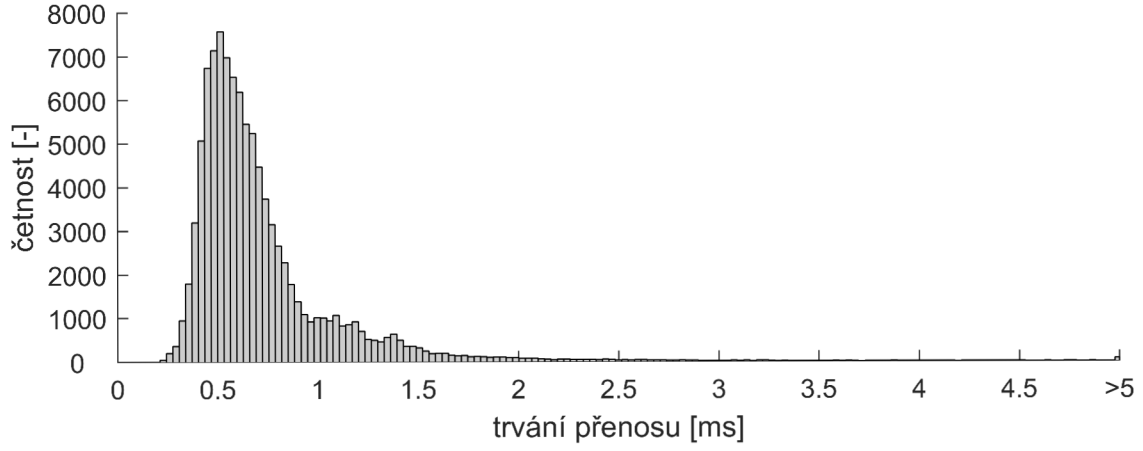
Obrázek 6.2: Histogramy z měření pro zprávy délky 1 při časování 10, 15, 16, 19 a 20 *ms*.



(c) časování 16 *ms*.



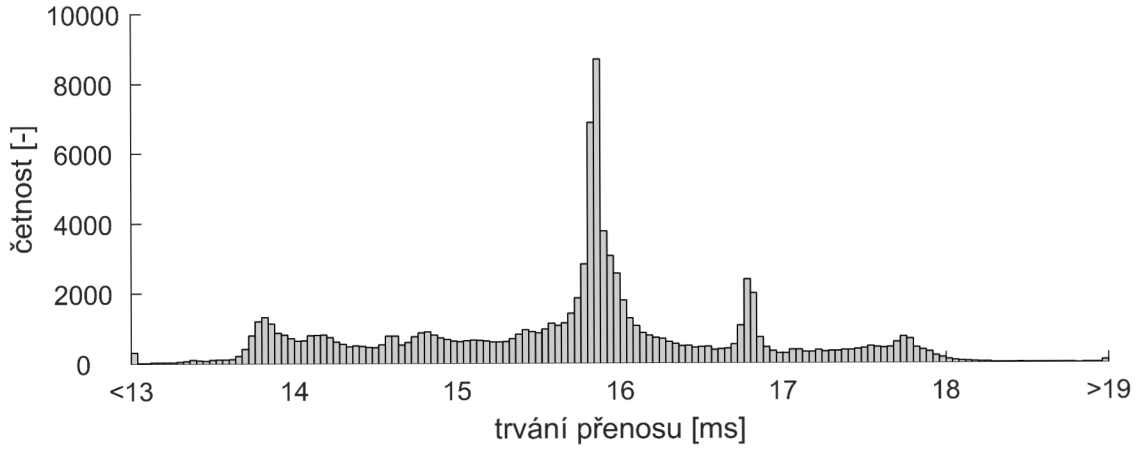
(d) časování 19 *ms*.



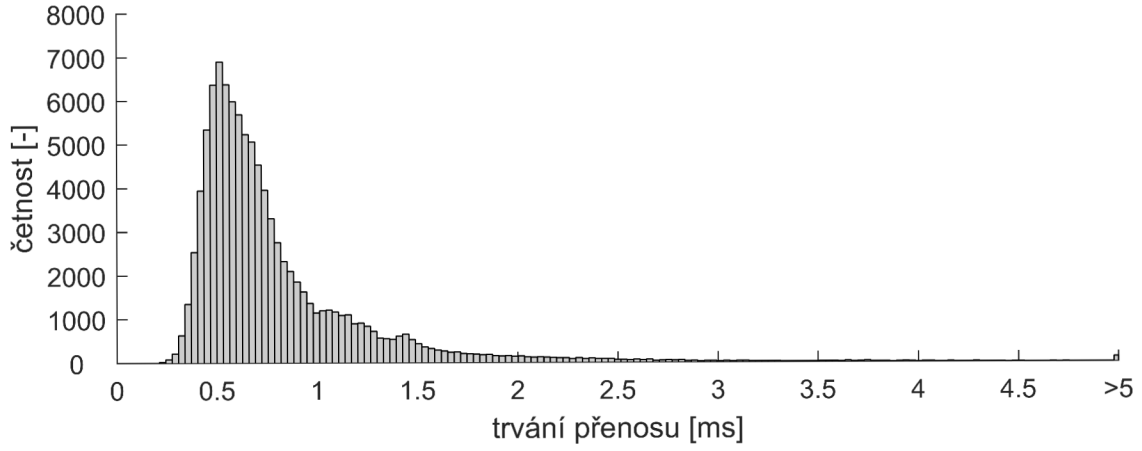
(e) časování 20 *ms*.

Obrázek 6.2: Histogramy z měření pro zprávy délky 1 při časování 10, 15, 16, 19 a 20 *ms* (pokračování).

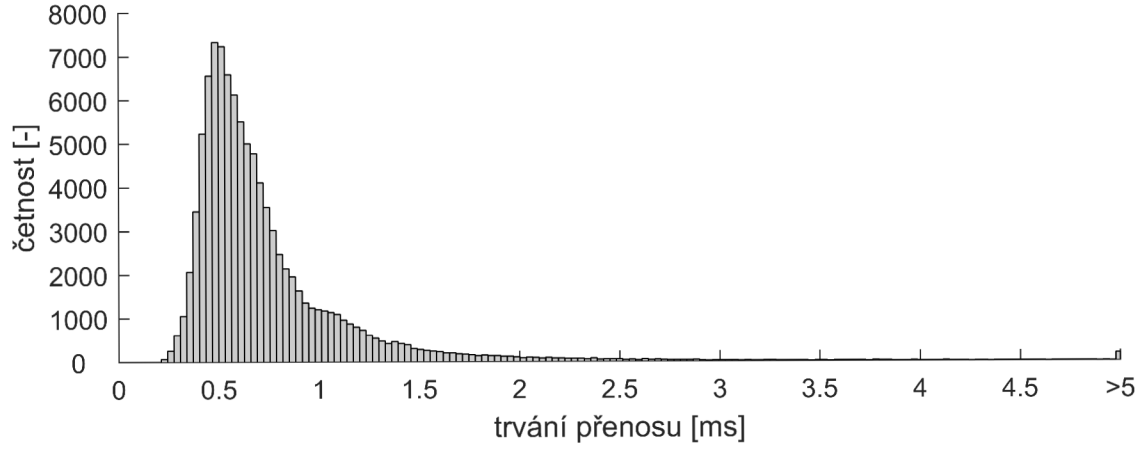
A.2 Měření pro zprávu délky 5



(a) časování 15 *ms*.

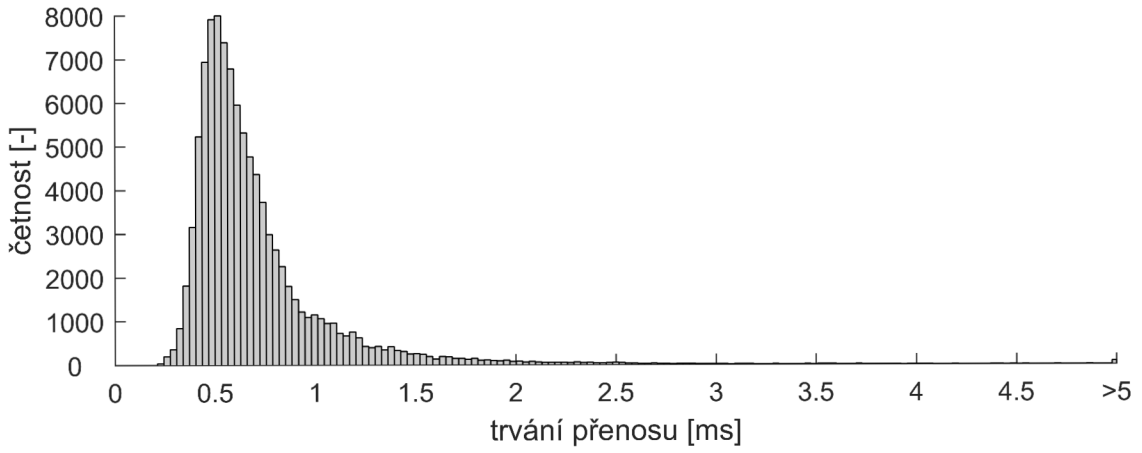


(b) časování 16 *ms*.

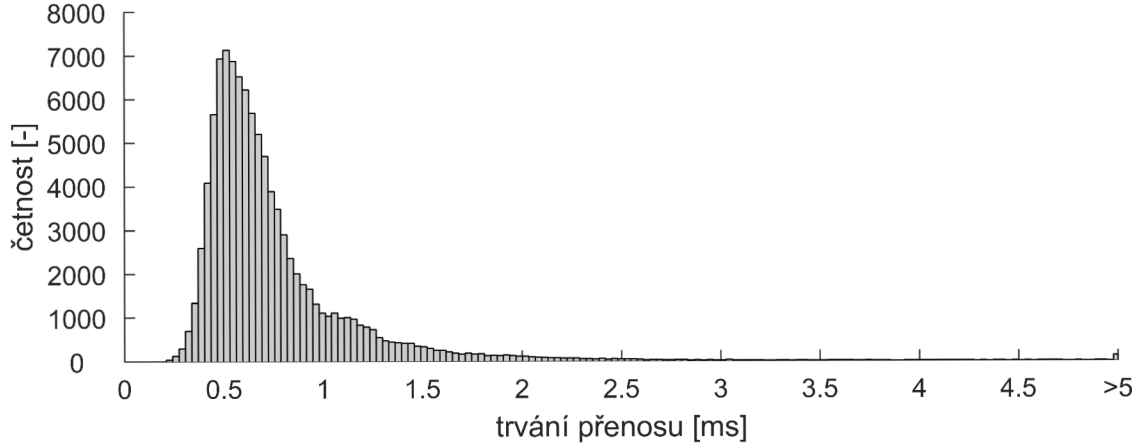


(c) časování 18 *ms*.

Obrázek 6.3: Histogramy z měření pro zprávy délky 5 při časování 15, 16, 18, 22 a 25 *ms*.



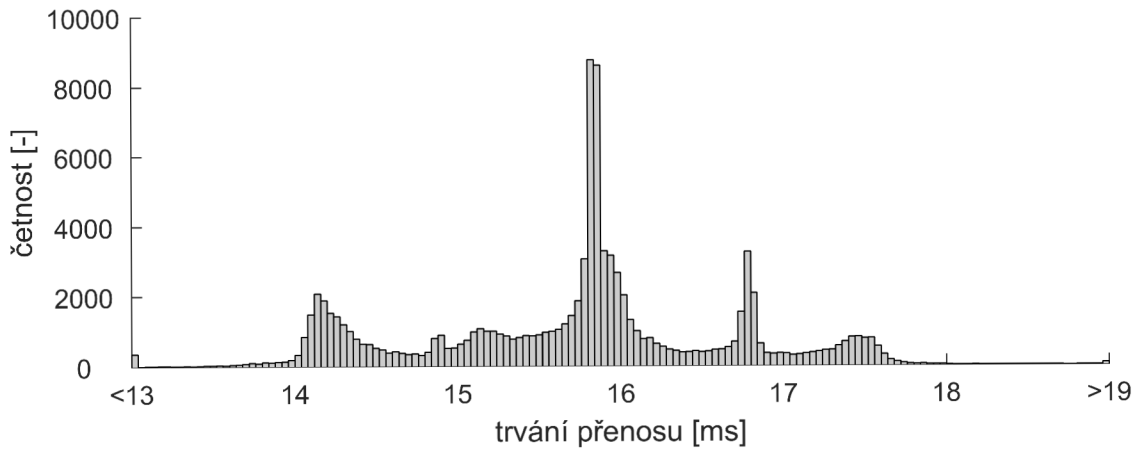
(d) časování 22 *ms*.



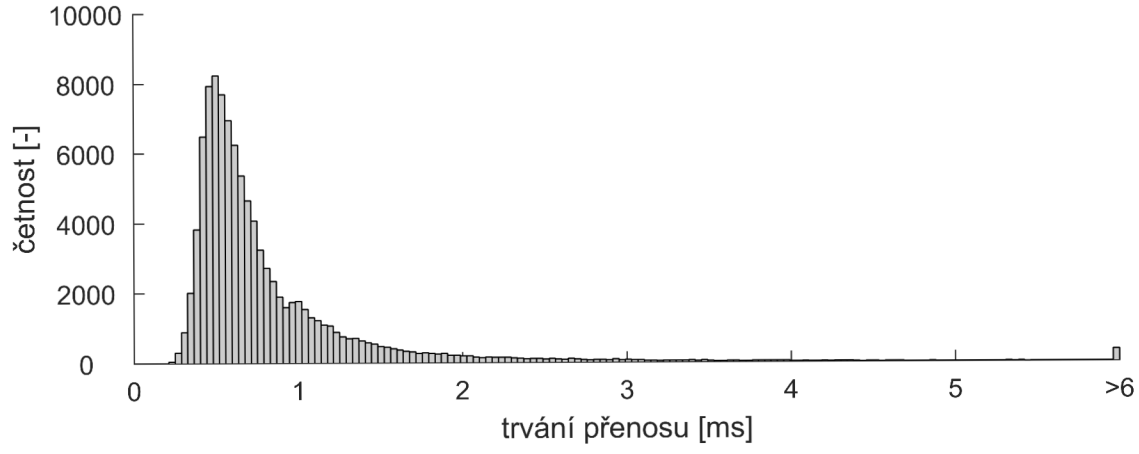
(e) časování 25 *ms*.

Obrázek 6.3: Histogramy z měření pro zprávy délky 5 při časování 15, 16, 18, 22 a 25 *ms* (pokračování).

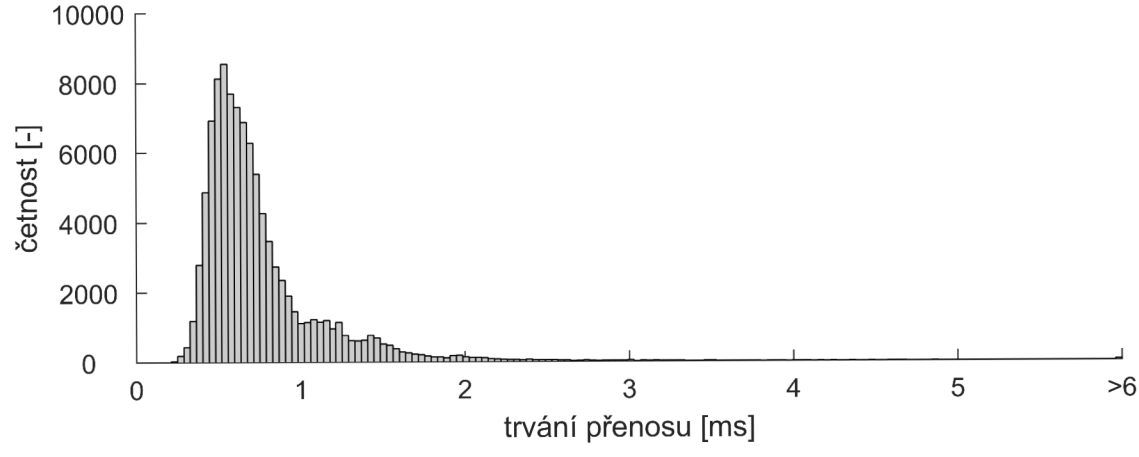
A.3 Měření pro zprávu délky 10



(a) časování 15 *ms*.

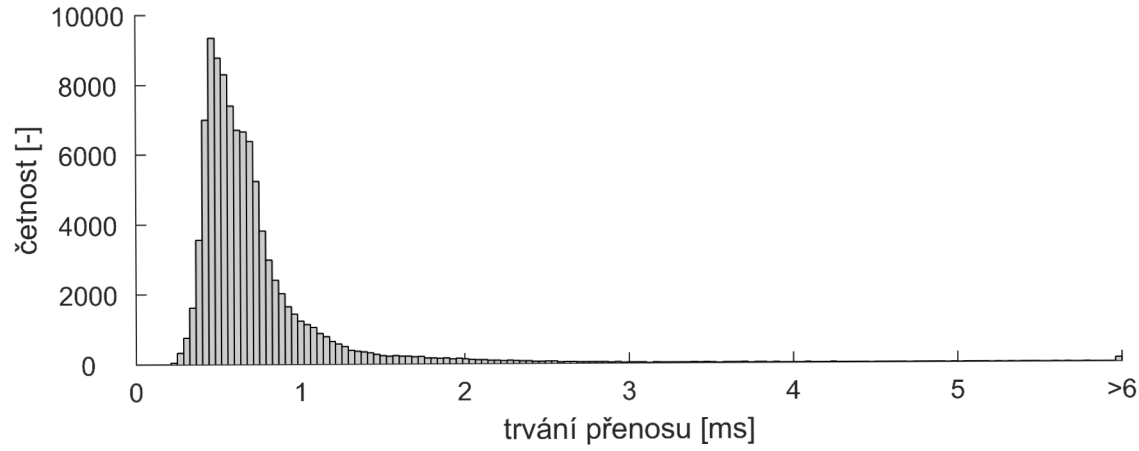


(b) časování 16 *ms*.

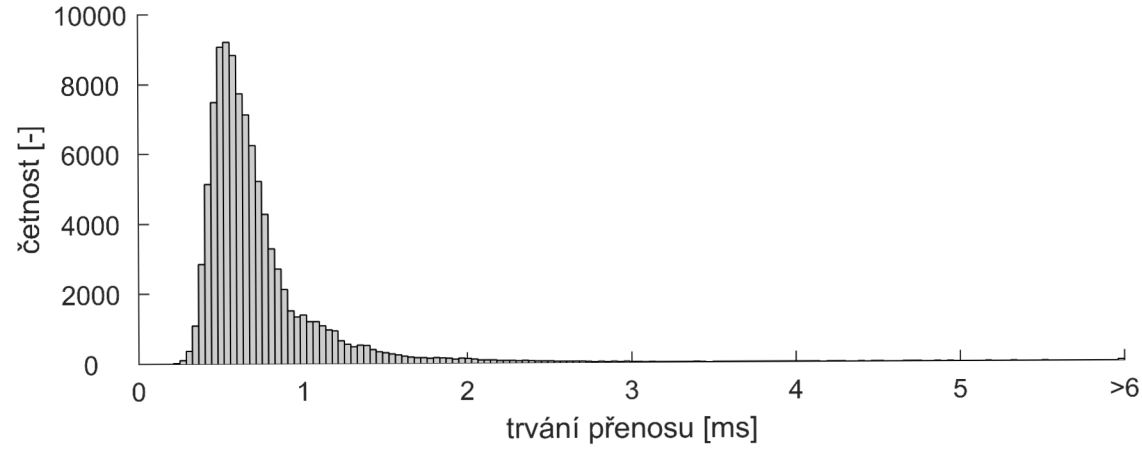


(c) časování 20 *ms*.

Obrázek 6.4: Histogramy z měření pro zprávy délky 10 při časování 15, 16, 20, 22 a 25 *ms*.



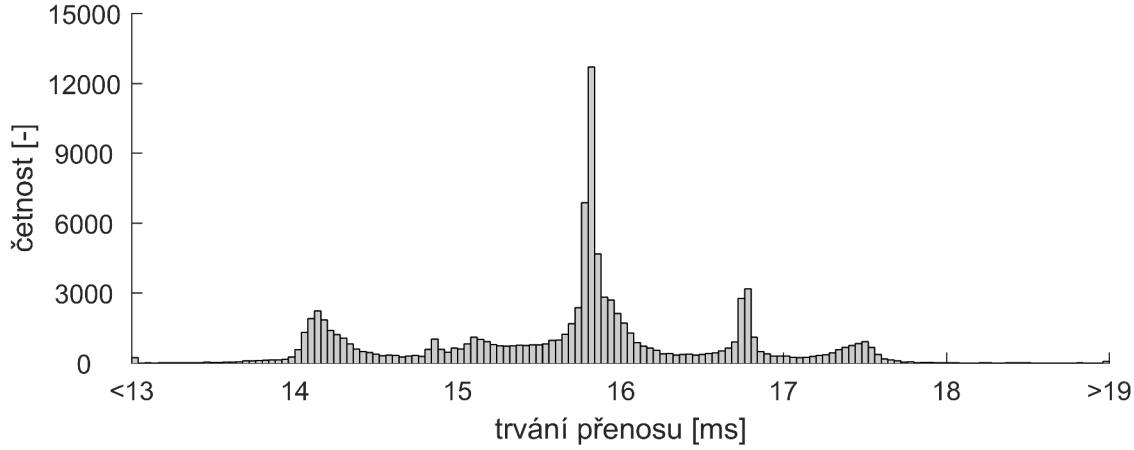
(d) časování 22 *ms*.



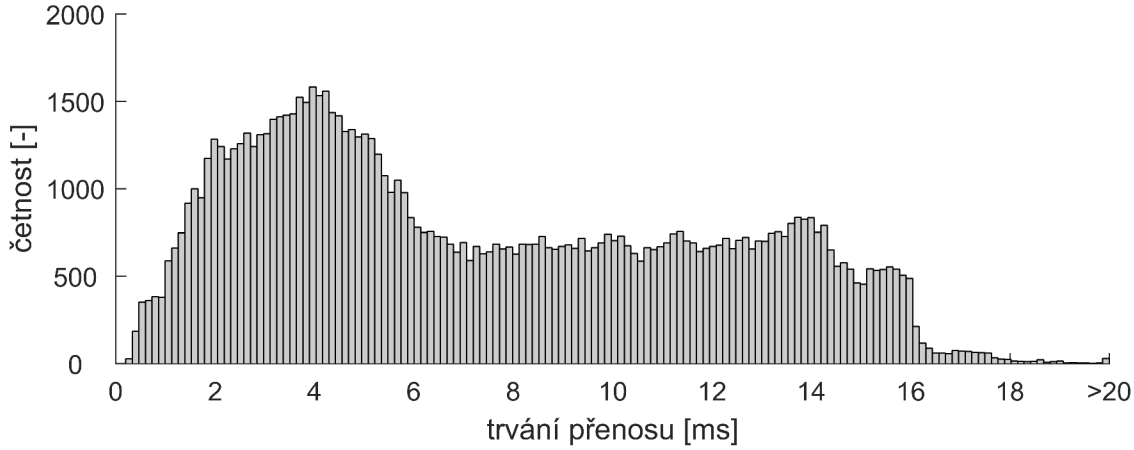
(e) časování 25 *ms*.

Obrázek 6.4: Histogramy z měření pro zprávy délky 10 při časování 15, 16, 20, 22 a 25 *ms* (pokračování).

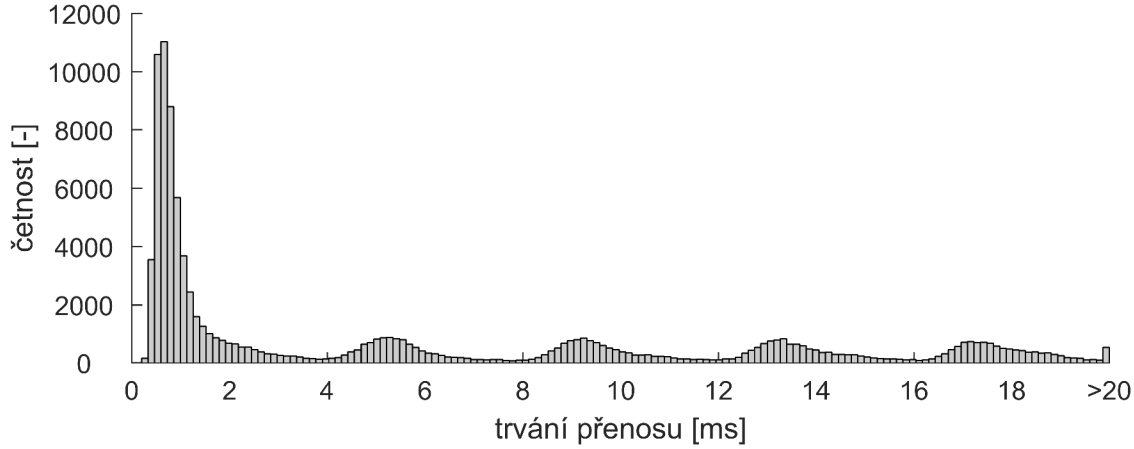
A.4 Měření pro zprávu délky 50



(a) časování 15 *ms*.

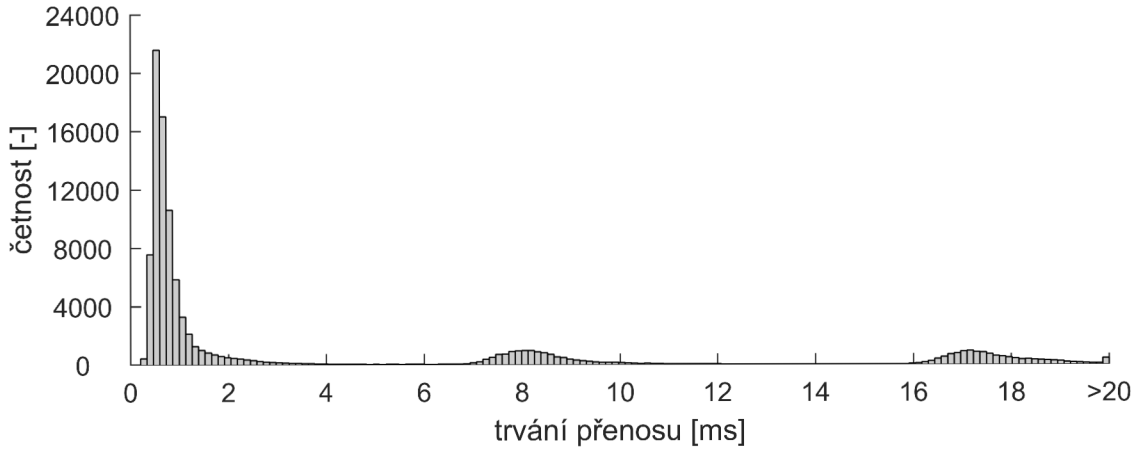


(b) časování 16 *ms*.

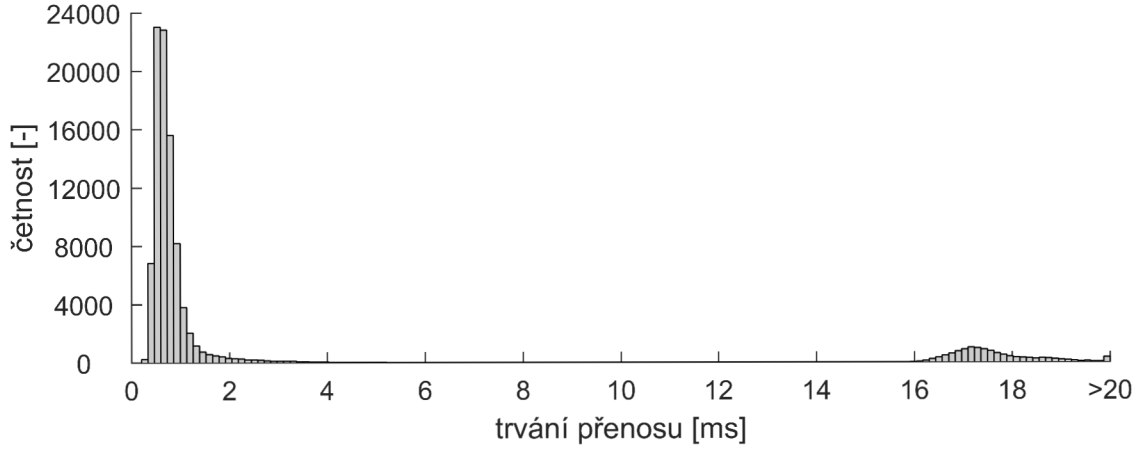


(c) časování 20 *ms*.

Obrázek 6.5: Histogramy z měření pro zprávy délky 50 při časování 15, 16, 20, 25 a 35 *ms*.



(d) časování 25 *ms*.



(e) časování 35 *ms*.

Obrázek 6.5: Histogramy z měření pro zprávy délky 50 při časování 15, 16, 20, 25 a 35 *ms* (pokračování).

B Elektronické přílohy

Tato práce obsahuje následující elektronické přílohy, obsahující jednotlivé zdrojové kódy a aplikace pro OS Android:

- 01_Komunikace.apk
- 01_Komunikace_Kod_Android_OS.zip
- 01_Komunikace_Kod_uC.apk
- 02_Regulace_polohy_motoru.apk
- 02_Regulace_polohy_motoru_Kod_Android_OS.zip
- 02_Regulace_polohy_motoru_Kod_uC.slx
- 03_Segway.apk
- 03_Segway_Kod_Android_OS.zip