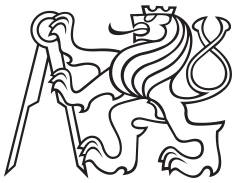
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra měření

Softwarový terminál pro sběr dat prostřednictvím sériové linky, UDP a TCP

Jiří Štefan

Vedoucí: Ing. Ondrej Tereň
Obor: Kybernetika a robotika
Leden 2021

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Ondreji Tereňovi za ochotu a pomoc při tvorbě práce, i v náročné době distanční výuky.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 1. ledna 2021

Abstrakt

Cílem této práce je navrhnout a vyvinout program fungující jako sofwarový terminál umožňující komunikaci pomocí standardu RS-232 a protokolů UDP a TCP. Nejprve je vykonán průzkum existujících řešení. Z průzkumu jsou pak formulovány požadavky na funkce programu. Poté je implementován program splňující dané požadavky.

Klíčová slova: terminál, RS-232, UDP, TCP, C++, Qt

Vedoucí: Ing. Ondrej Tereň

Abstract

Aim of this thesis is to design and develop program working as software terminal for communication using the RS-232 standard and UDP and TCP protocols. Survey of existing solutions is done first. Requirements on function of the program are then formulated from results of the survey. Program satisfying all requirements is then implemented.

Keywords: terminal, RS-232, UDP, TCP, C++, Qt

Title translation: Software terminal for data acquisition using serial port, UDP and TCP

Obsah

Obrázky

Tabulky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Štefan** Jméno: **Jiří** Osobní číslo: **474468**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Softvérový terminál pre zber dát prostredníctvom sériovej linky, UDP a TCP

Název bakalářské práce anglicky:

Software terminal for data acquisition using serial port, UDP and TCP

Pokyny pro vypracování:

Vykonajte prieskum dostupných softvérových terminálov pre sériovú linku, UDP a TCP. Porovnajte ich pozitívne aj negatívne vlastnosti so zameraním na zber dát a užívateľsky prívetivé ovládanie. Na základe prieskumu sformulujte požiadavky na softvérový terminál, ktorý by fúzoval všetky pozitívne vlastnosti skúmaných softvérových nástrojov. Vyviňte softvérový terminál podľa formulovaných požiadaviek. Na vývoj použite vývojové prostredie Qt.
Okrem požiadaviek, ktoré vyplynú z prieskumu, navrhnutý terminál musí umožňovať zadávanie správ vo formáte ASCII, hexadecimálneho čísla alebo kombináciu obidvoch možností. Užívateľ si bude môcť nastaviť preddefinované správy, ktoré bude môcť odosielať pomocou klávesových skratiek, napríklad kláves F1 až F12, prípadne kombináciou s ďalšími klávesmi. Terminál musí ďalej umožňovať ukladanie zachytených správ do súboru. Všetky nastavenia terminálu a preddefinovaných správ budú automaticky ukladané do súboru alebo registrov operačného systému a po štarte aplikácie sa automaticky načítajú. Aplikácia umožní nastaviteľné riadkovanie prijatých správ podľa prijatia znaku koniec riadka alebo podľa časovej medzery medzi jednotlivými znakmi. V prípade UDP prenosu bude možné odosielať správy na viacero rôznych IP adries a portov.
Neoddeliteľnou súčasťou práce bude overenie funkcie vyvinutej aplikácie. Test bude vykonaný pomocou vhodného HW prípravku (hotového výrobku, nebude súčasťou BP), ktorý nasimuluje vzorový dátový tok. Cieľom testu bude nájsť limit aplikácie, kedy je schopná na danej konfigurácii PC prijímať dáta bez strát.

Seznam doporučené literatury:

[1] CHROBOCZEK, Martin: Grafická uživatelská rozhraní v Qt a C++. Brno: Computer Press, 2013. ISBN 978-80-251-4124-3.
[2] THELIN, Johan: Foundations of Qt development. New York: Distributed to the book trade worldwide by Springer-Verlag New York, c2007. ISBN 1590598318.
[3] PRATA, Stephen: Mistrovství v C++. 4., aktualiz. vyd. Přeložil Boris SOKOL. Brno: Computer Press, 2013. Bestseller (Computer Press). ISBN 978-80-251-3828-1.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Ondrej Tereň, katedra měření FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **13.02.2020** Termín odevzdání bakalářské práce: **14.08.2020**

Platnost zadání bakalářské práce:
do konce zimního semestru 2021/2022

Ing. Ondrej Tereň podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Kapitola 1
Úvod

V sestavách a sítích jejichž součástí jsou dvě a více zařízení, např. mikrokontroléry, senzory a počítače, je důležitá jejich vzájemná komunikace. Často je třeba přijímat či odesílat data na zařízení připojená po sériové lince, nebo dostupná přes síťové připojení. V případě sériového přenosu se tato práce zabývá standardem RS-232, v případě síťového přenosu pak protokoly UDP a TCP.

Při navrhování a vytváření takových sestav je třeba komunikaci testovat průběžně na jednotlivých součástech pomocí specializovaných nástrojů. Až po ověření funkce samostaně má smysl prvky propojovat. Pro tyto účely existuje mnoho specializovaných i obecných nástrojů, kde každý nabízí i určité dodatečné funkce ulehčující práci. Většina veřejně dostupných a přenosných programů však nabízí pouze některé z těch užitečných a často důležitých funkcí.

Mezi ty obecnější nástroje patří takzvané terminálové programy. Jde o software, který uživateli dává možnost odesílat data ve formě znaků a příkazů a přijímat data od externích zařízení, která zpravidla zobrazuje v hlavním okně programu pro snadné zpracování.

Ačkoliv tedy existující nástroje umožňující základní přijímání a odesílání znaků, často nemají funkce jako odesílání předdefinovaných příkazů, nebo formátovaní přijatých dat způsoby které jsou užitečné pro danou situaci.

Cílem této práce je tedy navrhnout a implementovat softwarový terminál, který by obsahoval většinu důležitých funkcí pro práci s daty a jejich přijímáním a odesíláním na sériové lince a podle protokolů UDP a TCP.

Kapitola 2
Teoretický rozbor

Často je třeba propojit dvě nebo více zařízení za účelem předávání informace. Aby se zařízení mezi sebou domluvila, bylo vytvořeno mnoho standardů pro různé druhy zařízení a situací, kde každý definuje způsob přenosu dat. Tato práce využívá standardu RS-232 a protokolů UDP a TCP.

2.1 Sériový port

Sériový port nebo také sériová linka je běžně používaný název pro komunikační standard RS-232, který definuje signály a napěťové hodnoty pro komunikaci mezi zařízeními. V dnešní době se většinou používají pouze tři z definovaných signálů. Jedná se o signály Tx, Rx, GND.

Signál GND je společná zem a umožňuje každému ze zařízení pracovat se stejným referenčním napětím. Signály Tx a Rx pak slouží pro odesílání a přijímání dat. Takto je možno využít plně duplexní komunikace, což znamená že je možno na jednom z pinů data přijímat i pokud se na druhém pinu zrovna data odesílají. Také je možno použít pouze jeden z drátů Tx nebo Rx pro poloviční duplexní komunikaci, kdy přijímání a odesílání nelze provádět ve stejnou cvhíli.

Standard dále definuje způsob jakým se předávají data na pinech Tx a Rx. V klidovém stavu je na pinech logická 1. Začátek přenosu dat je značen přechodem do logické 0 na stanovený čas, tzv. start bit. Po start bitu následuje 5 až 9 datových bitů, nesoucích požadovanou informaci. Následuje paritní bit, který umožňuje zachytit chybu vzniklou při přenosu. Nakonec následují jeden až dva stop bity, které vrátí pin na hodnotu logické 1 a značí tím konec přenosu. Ihned po přijetí stop bitu je poté možné odeslat další start bit a data.

Jedná se o asynchronní přenos informace, což znamená že neexistuje drát s hodinovým signálem který by oznamoval časy při kterých se mají číst napěťové hodnoty. Namísto toho je třeba na všech zařízeních nastavit identické parametry přenosu, zejména parametr Baud Rate, který udává počet přenesených bitů za sekundu. Podle tohoto parametru je pak možno po přijetí start bitu vždy počkat konstantní čas a odečíst hodnotu nového bitu.

2.2 Protokoly rodiny TCP/IP

TCP/IP je název skupiny protokolů zprostředkujících síťovou komunikaci. Název je odvozen podle dvou nejdůležitějších protokolů v tomto modelu, protokolů TCP (Transmission Control Protocol) a IP (Internet Protocol). Dalším důležitým protokolem vztahujícím se k této práci, je protokol UDP (User Datagram Protocol).

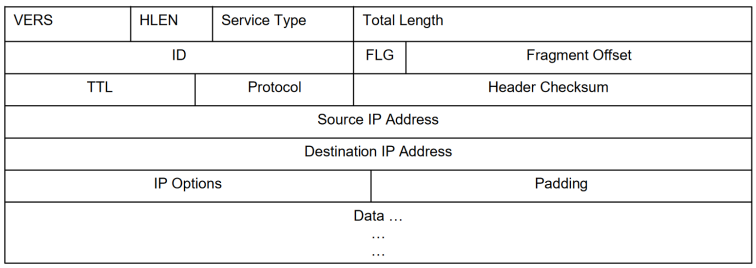
UDP a TCP jsou protokoly realizující přenos dat pomocí portu, IP je protokol realizující přenos dat na nižší úrovni, pomocí adresy. K datům odesílaným protokoly UDP a TCP se přidají informace potřebné pro zprostředkování přenosu a vytvoří takzvaný segment. K tomuto segmentu se pak připojí informace IP protokolu jako např. cílová adresa a vytvoří se takzvaný packet, který je poté odeslán.

2.2.1 IP

IP je nespojovaný protokol, který realizuje přenos dat pomocí packetů. Složení takového packetu je vidět na obrázku 2.1. Důležitou součástí IP protokolu je IP adresa. Adresa je reprezentována čtveřicí 8-bitových čísel oddělených tečkami, např. 198.162.0.66. Taková adresa je v dané síti unikátní a lze jí tak použít pro určení zdroje a cílové destinace packetu.

Na obrázku 2.1 jsou vidět části packetu Source IP Address, do které se zapisuje IP adresa zařízení ze kterého byl packet odeslán, a Destination IP Address do které se zapisuje IP adresa cílového zařízení.

Další důležité informace které jsou součástí packetu jsou například Total Length, udávající celkovou délku packetu, ID které každému packetu přidělí číslo pro snadnou identifikaci packetů při převzetí, a parametr TTL (Time To Live), který udává čas po který smí packet existovat, čímž se zamezí uvíznutí packetu v nekonečné smyčce.



Obrázek 2.1: Formát IP packetu, převzato z [7].

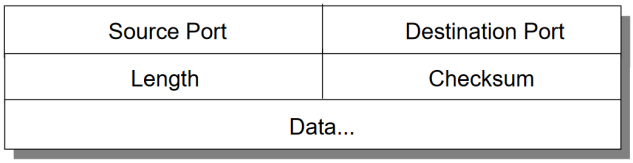
2.2.2 Port

Pokud na jedné IP adrese zařízení dochází k více nezávislým komunikacím, je třeba od sebe tyto komunikace odlišit. K tomu se používá takzvané číslo portu. Jedná se o 16-bitové číslo, které nabývá hodnot 0 až 65535. Pokud zařízení naváže např. TCP komunikaci s několik vzdálenými servery, každé

přidělí jiné číslo portu, čímž od sebe připojení odliší. Porty s čísly 0 až 1023 jsou takzvané známé porty a používají se pro často používané služby. Porty s čísly 1024 až 49151 lze registrovat pro specififcký účel. Nakonec porty s čísly 49152 až 65535 nelze registrovat a lze je volně používat pro osobní nebo dočasné účely.

2.2.3 UDP

UDP je nespolehlivý nespojovaný protokol, který realizuje přenos dat pomocí takzvaných segmentů. Že je přenos nespolehlivý v praxi znamená, že po odeslání segmentu nezaručuje jeho doručení a ani nezaručuje pořadí, ve kterém segmenty dorazí. Na obrázku 2.2 je vidět, že pro vytvoření segmentu je k datům třeba přidat informaci o zdrojovém a cílovém portu a délku segmentu. Checksum je dobrovolné pole, které lze vyplnit pro kontrolu správnosti segmentu po doručení.

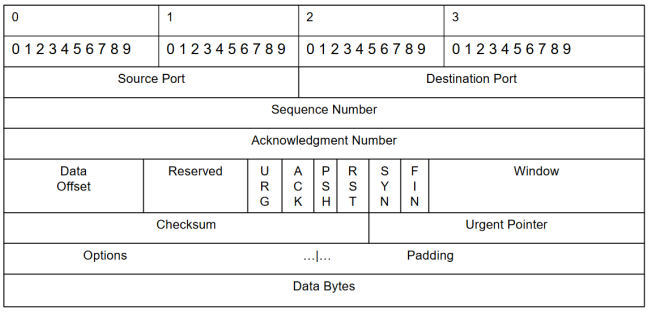


Obrázek 2.2: Formát UDP segmentu, převzato z [7].

2.2.4 TCP

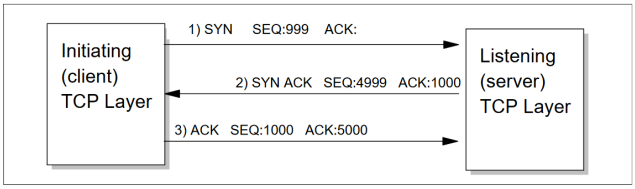
TCP je spolehlivý spojovaný protokol, který realizuje přenos dat pomocí segmentů. Zaručuje, že odeslaný packet bude přijat a pokud přijme packety v nesprávném pořadí, dokáže je přerovnat. Spolehlivého přenosu dat je dosaženo spojovanou komunikací. To znamená, že jedna strana komunikace pracuje jako server a druhá jako klient. Server je spušťen jako první a čeká dokud se nepřipojí jeden nebo více klientů.

Na obrázku 2.3 je vidět, že stejně jako u UDP začíná segment zdrojovým a cílovým portem. Dále v hlavičce segmentu se také nachází Checksum pro kontrolu správnosti doručených dat. Součástí hlavičky je dále Sequence Number, tedy číslo udávající pořadí segmentu. Acknowledgment Number pak udává další číslo sekvence kterou příjemce očekává. Důležité jsou také některé z bitových příznaků. Bit ACK udává, že hodnota Acknowledgment Number je nastavena. Bit SYN slouží pro synchronizaci čísel sekvence. Bit FIN slouží k ukončení spojení.



Obrázek 2.3: Formát TCP segmentu, převzato z [7].

K navázání spojení slouží takzvaný 3-way handshake. Klient nejdříve pošle serveru zprávu s náhodně zvoleným číslem sekvence a nastaveným příznakem SYN. Server odpoví zprávou se svým náhodně zvoleným číslem sekvence, nastavenými příznaky SYN a ACK a hodnotu Acknowledgment Number nastaví o jednu větší než tu co dostal od klienta. Klient nakonec odpoví zprávou s příznakem ACK, potvrzeným číslem sekvence a číslo serveru také potvrdí hodnotou Acknowledgment Number. Celý proces je ilustrován na obrázku 2.4.



Obrázek 2.4: Ukázka funkce 3-way handshake, převzato z [7].

Ukončení spojení se provede odesláním segmentu s nastaveným FIN bitem. Druhá strana pak odešle zbývající segmenty a odpoví vlastním segmentem s nastaveným FIN bitem. Tím se přenos ukončí.

Kapitola 3

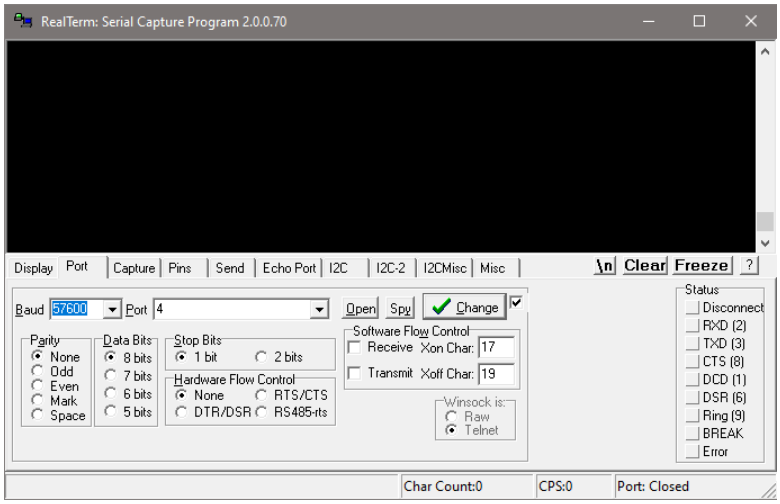
Průzkum existujících řešení

Před implementací terminálového programu je třeba identifikovat pozitivní a negativní vlastnosti již existujích programů. V této kapitole jsou shrnuty vlastnosti programů Realterm, Herkules, Packet Sender a Termite. Některé z vlastností jsou zahrnuty do projektu a implementovány, jiné časově náročnější a mimo rámec této práce jsou uvedeny jako budoucí cíle pro případnou pokračující práci.

3.1 Realterm

Program Realterm umožňuje zejména sériovou komunikaci [3]. Mimo základní nastavení sériového portu má program mnoho dalších užitečných nastavení a ovládacích prvků, jen některé z nich jsou ovšem užitečné pro základní implementaci. Tyto vlastnosti jsou:

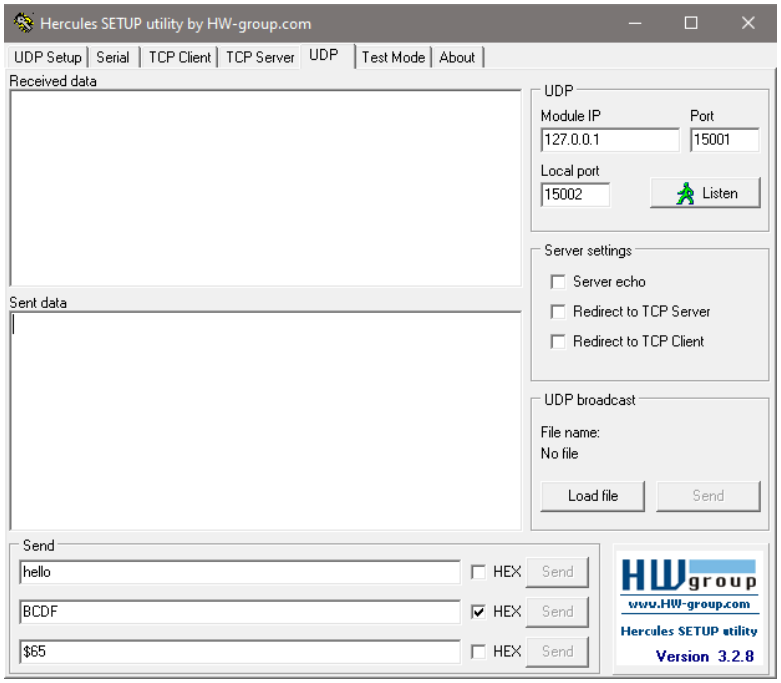
- Stavový řádek s informací o stavu sériového portu
- Možnost zachytit přijaté znaky do souboru
- Možnost odesílat uživatelem definované zprávy



Obrázek 3.1: Nastavení sériového portu programu Realterm.

3.2 Herkules

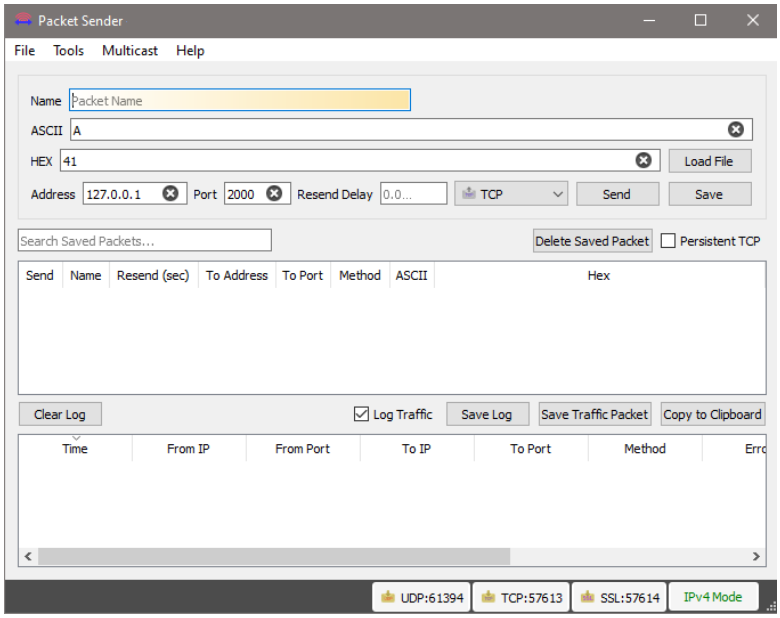
Program Herkules umožňuje komunikaci pomocí sériové linky, UDP a TCP protokolů [1]. Mimo základní implementaci těchto tří funkcí slouží zejména pro nastavení konkrétních hardwarových výrobků. Rozložení relevantních ovládacích prvků je však jednoduché a intuitivní. Program v některých případech rozděluje pole přijatých a odeslaných znaků a umožňuje přijaté znaky zobrazovat ve zvoleném formátu, například jako ASCII znak, nebo jako hexadecimální číslo. Další důležitou funkcí je práce se soubory. Přijatá data lze dle potřeby zapisovat do souboru a zároveň lze vybrat soubor který program pak dokáže odeslat. Uživatel si také může definovat několik předdefinovaných zpráv, které lze také odesílat pomocí klávesových zkratek.



Obrázek 3.2: Nastavení UDP přenosu programu Herkules.

3.3 Packet Sender

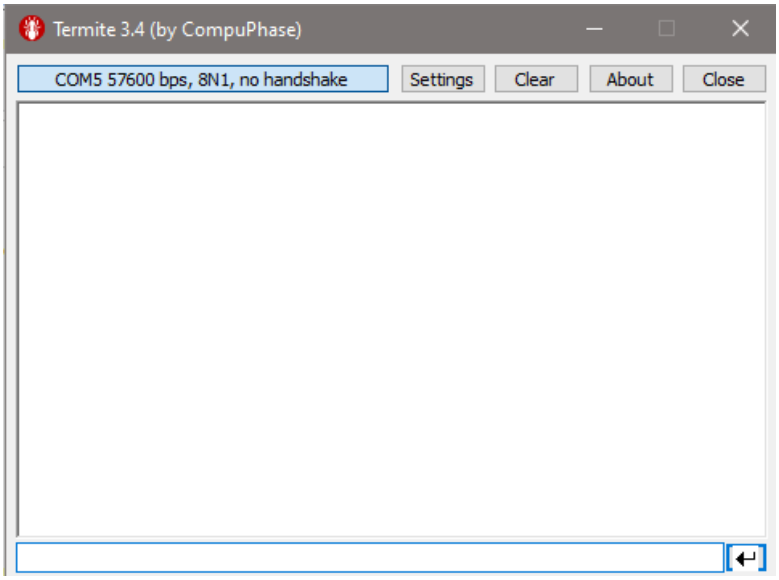
Program Packet Sender umožňuje komunikaci pomocí protokolů UDP a TCP [2]. Hlavní funkcí programu je odesílání uživatelem definovaných zpráv, kde si u každé z nich lze definovat i cílovou adresu a port. Užitečnou funkcí je neomezený počet zpráv které si uživatel může definovat. Další vlastnost programu je možnost automatických odpovědí. Uživatel si tedy může definovat zprávu na kterou program automaticky odešle odpověď, pokud ji přijme.



Obrázek 3.3: Uživatelské rozhraní programu Packet Sender.

3.4 Termite

Jedinou funkcí programu Termite je komunikace pomocí sériové linky [5]. Výhodou je minimalistické a jednoduché uživatelské rozhraní, kde je většina nastavení a ovládacích prvků schována za tlačítkem nastavení. V programu nelze předem definovat a odesílat zprávy, jediný výstup tvoří příkazová řádka. Užitečnou funkcí je pak historie odeslaných zpráv, což umožňuje automatické vyplňování.



Obrázek 3.4: Uživatelské rozhraní programu Termite.

3.5 Shrnutí

Jako předloha pro návrh uživatelského rozhraní byl zvolen program Herkules. Programy Realterm a Herkules umožňují předdefinovat zprávy, ale pouze omezený počet. Program Packet Sender pak nepodporuje průběžné ukládání do souboru.

	Realterm	Herkules	Packet Sender	Termite
Sériový port	X	X		X
UDP		X	X	
TCP		X	X	
Stavový řádek	X		X	
Ukládání do souboru	X	X	/	
Odesílání souboru	X	X	X	
Předdefinované zprávy	/	/	X	
Příkazový řádek				X
Automatická odpověď			X	

Tabulka 3.1: Tabulka vlastností zkoumaných programů, kde implementované jsou označeny symbolem X, částečně implementované symbolem /.

Výsledný software by tedy měl obsahovat tyto základní funce:

- Komunikace pomocí sériové linky a protokolů UDP a TCP
- Možnost uložit přijatá data do souboru
- Možnost odesílat uživatelem definované zprávy
- Automatické ukládání nastavení aplikace a jejich načtení po spuštění
- Možnost nastavit styl a formátování přijatých dat
- Možnost odesílat UDP packety na více adres a portů

Dle tabulky 3.1 a rozborů jednotlivých softwarů by měl program také obsahovat:

- Stavový řádek s informací o stavu programu
- Možnost odělit vstupní a výstupní data v uživatelském rozhraní
- Možnost zadat neomezený počet předdefinovaných zpráv
- Možnost průběžně ukládat přijatá data do souboru

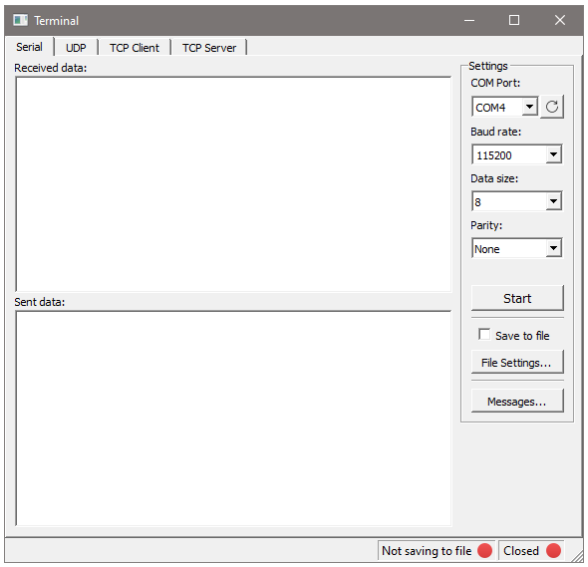
Nakonec je třeba podotknout funkcionality, které nejsou příliš běžné jako je možnost zadávání dat do příkazového řádku a nebo automatické odpovědi na přijaté zprávy. Tyto funkce má smysl implementovat, nicméně jsou nad rámec této práce.

Kapitola 4
Uživatelské rozhraní

Dle zadání byl terminálový program vytvořen ve vývojovém prostředí Qt pomocí jazyka C++ a skládá se ze čtyř hlavních oken.

4.1 Sériový terminál

V okně Serial si uživatel může nastavit parametry sériového přenosu. Přenosovou rychlost lze zvolit z předem definovaných hodnot, nebo zadat libovolnou pomocí klávesnice. Dále lze nastavit počet datových bitů a paritní bit. Tlačítkem u volby portu lze automaticky detekovat připojená zařízení. Tlačítkem Start se pak na daném portu spustí komunikace a lze odesílat a přijímat data.

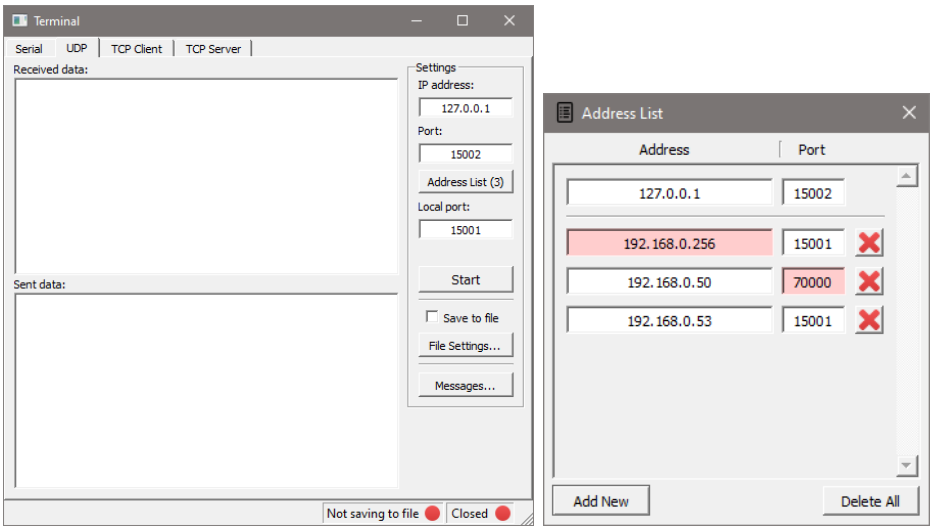


Obrázek 4.1: Okno nastavení sériového portu.

4.2 UDP terminál

V okně UDP si uživatel může nastavit cílovou adresu a port pro odesílaná data. Také si může nastavit port, na kterém budou packety přijímány. Pokud

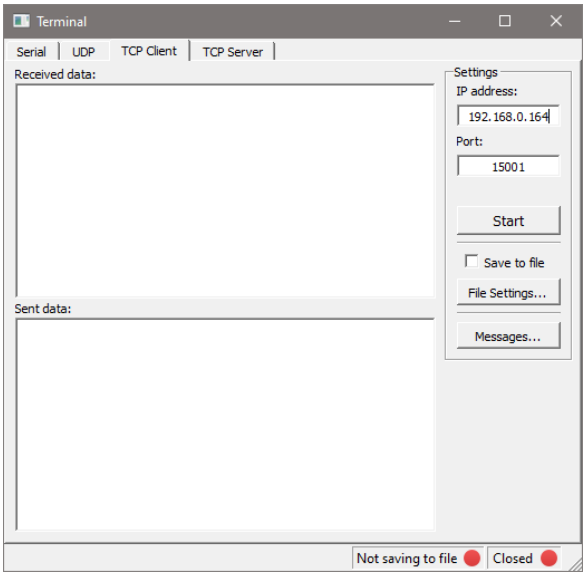
je potřeba zprávy odesílat na několik adres nebo portů najednou, lze další dodefinovat v okně pod tlačítkem Address List, které je vidět na obrázku 4.2 vpravo. Číslo uvedené na tlačítku odpovídá počtu takto definovaných adres. Pokud uživatel zadá neplatnou adresu nebo číslo portu, je na to upozorněn červeným zbarvením pole zprávy.



Obrázek 4.2: Okno nastavení UDP a dalších adres.

4.3 TCP Client terminál

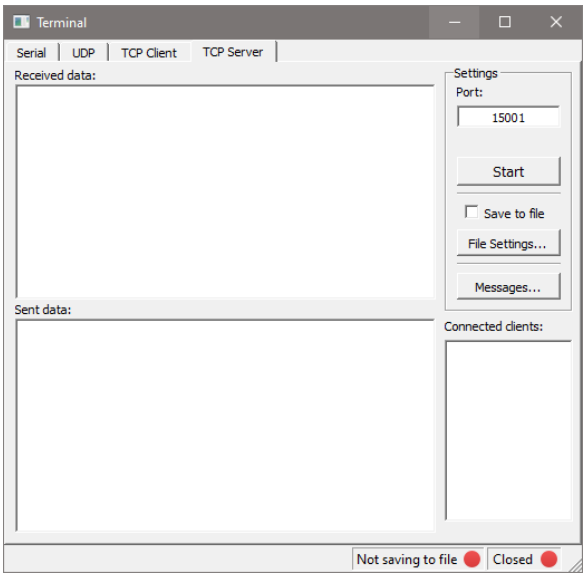
V okně TCP Client si uživatel může nastavit cílovou adresu a port pro odesílaná data. Po zmáčknutí tlačítka Start se klient pokusí připojit. Pokud se během tří sekund klient nepřipojí, pokus se přeruší.



Obrázek 4.3: Okno nastavení TCP klienta.

4.4 TCP Server terminál

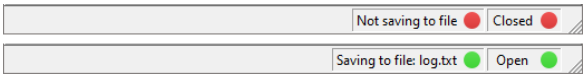
V okně TCP Server si uživatel může nastavit port, na kterém bude server čekat na příchozí připojení. Seznam IP adres připojených klientů je zobrazen v okně Connected clients. Odchozí data jsou odeslána postupně každému z klientů v uvedeném pořadí.



Obrázek 4.4: Okno nastavení TCP serveru.

4.5 Stavový řádek

Stavový řádek je viditelný v každém z hlavních oken, zobrazené informace jsou však relevantní pouze pro zvolený terminál. V právé části je uveden stav zvoleného portu. Déle nalevo je pak stav zapisování do souboru s případným názvem souboru, pokud je zapisování aktivní.



Obrázek 4.5: Funkce stavového řádku.

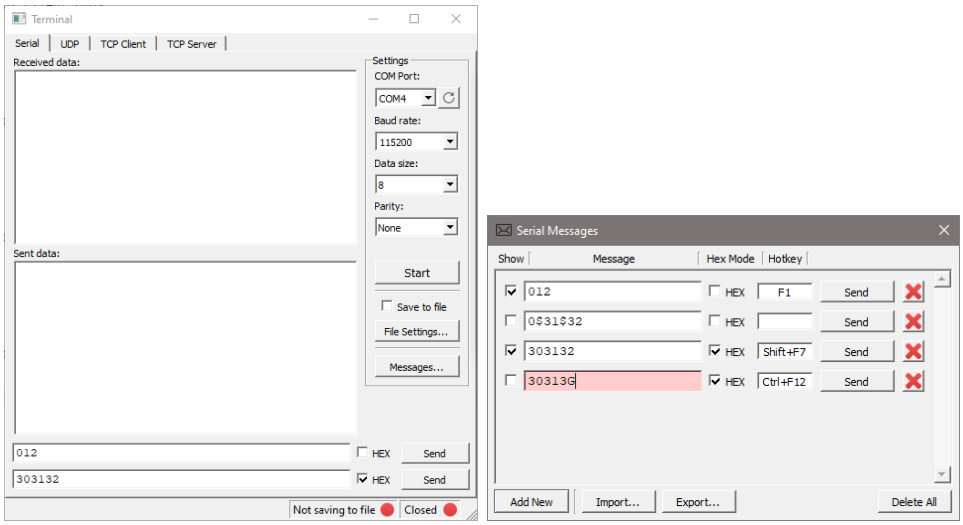
4.6 Předdefinované zprávy

Nastavení předdefinovaných zpráv je dostupné pod tlačítkem Messages z každého okna. Zprávy nejsou společné, a tak je možné pro každé okno nastavit zprávy zvlášť.

Tlačítkem Add New je možné přidat novou zprávu, která se ihned zobrazí v okně nastavení. Tlačítkem Export je možné seznam zpráv uložit i s jejich nastavením do textového souboru. Tlačítkem Import je pak možné takto vy-

tvořený textový soubor načíst. Tlačítko Delete All pak všechny předdefinované zprávy smaže.

U každé zprávy je možné nastavit několik parametrů. Směrem zleva je první parametr Show, který určuje zda se má zpráva zobrazit kromě okna nastavení i v hlavním okně programu. Následuje textové pole kam uživatel zadává zprávu ve formě znaků. Pokud je třeba odeslat hodnotu, kterou nelze napsat na klávesnici, je možné použít znak $ následovaný dvoumístnou hexadecimální hodnotou. Například pro odeslání znaku $ lze použít jeho hexadecimální ASCII hodnotu, tedy $24. Další parametr Hex přepne zadávání zprávy pouze na dvojce hexadecimálních čísel, bez nutnosti použití znaku $. Pokud uživatel v tomto módu nezadá sudý počet znaků a nebo zadá znak který není hexadecimální číslo, je na to upozorněn červeným zbarvením pole zprávy. Následuje pole pro zadání klávesové zkratky. Uživatel vybere příslušné pole kliknutím a stiskne požadovanou klávesovou zkratku. Tou pak lze odesílat zprávy z hlavního okna. Platné zkratky jsou klávesy F1 až F12 kombinované s klávesami Shift a Ctrl. Následuje tlačítko Send které zprávu odešle a tlačítko smazání zprávy.



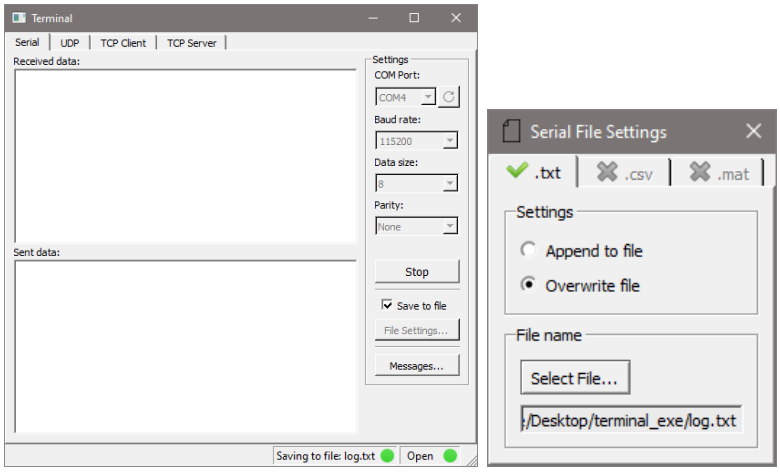
Obrázek 4.6: Ukázka zadávání předdefinovaných zpráv.

4.7 Ukládání do souboru

Nastavení ukládání do souboru je dostupné pod tlačítkem File Settings z každého okna. Nastavení není společné a tak je možné pro každé okno nastavit jiný soubor.

V době psaní tohoto textu je dostupné pouze ukládání do textového souboru. Uživatel si může zvolit název souboru, do kterého chce přijatá data ukládat tlačítkem Select File. Také lze určit, jestli se mají data do souboru přidávat, nebo jestli se má soubor přepsat.

Zapisování do souboru je možné spustit zaškrtávacím tlačítkem Save to file. Během zapisování nelze měnit nastavení.



Obrázek 4.7: Ukázka nastavení zapisování do souboru.

4.8 Ukládání nastavení

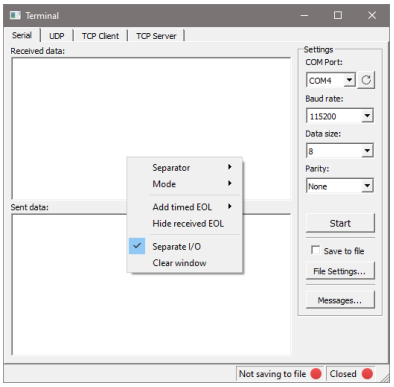
Terminálový program také ukládá uživatelské nastavení, polohy a velikosti oken. Po spuštění jsou nastavení automaticky načtena.

4.9 Formát přijatých dat

Pravým tlačítkem myši lze otevřít menu nastavení zobrazení dat.

V menu Separator je možné nastavit znak, kterým se oddělí každý příchozí znak. V menu Mode lze zvolit, jestli se má přijatý znak zobrazit jako hexadecimální číslo a nebo podle své ASCII reprezentace.

Menu Add timed EOL umožňuje nastavit hodnotu času po jejímž uplynutí se za přijaté znaky automaticky přidá znak nového řádku. Tento časovač je resetován po přijetí nového znaku. Tlačítko Hide received EOL umožňuje schovat přijaté znaky konce řádku. Tlačítko Separate I/O přepíná mezi jedním oknem pro přijatá a odeslaná data, a mezi dvojicí oken, jak je vidět na obrázku 4.8. Tlačítko Clear Window smaže přijaté a odeslané znaky.



Obrázek 4.8: Menu nastavení formátu přijatých dat.

Kapitola 5

Implementace

Program byl implementovaný v jazyce C++ pomocí vývojového prostředí QCreator a aplikačního rámce Qt. Program je sestaven dle principů objektově orientovaného programování, je tedy rozdělen na množství tříd které definují jednotlivé funkce programu [8]. Z těchto tříd jsou pak při spuštění programu vytvořeny objekty které mezi sebou komunikují. Qt nabízí spousty již implementovaných tříd, které je vhodné využít, a proto jsou od nich třídy vytvořené v rámci této práce odvozené. Třídy na kterých tato implemetace staví lze identifikovat podle názvu, začínají vždy písmenem Q [6].

V tabulce 5.1 je seznam tříd terminálového programu, které tvoří hlavní část programu.

Název	Základní třída	Hlavičkový soubor
MainWindow	QMainWindow	mainwindow.h
SerialPort	QSerialPort	serialport.h
UDPSocket	QUdpSocket	udpsocket.h
TCPClient	QTcpSocket	tcpclient.h
TCPServer	QTcpServer	tcpserver.h
StatusBar	QStatusBar	statusbar.h

Tabulka 5.1: Tabulka tříd hlavní části programu.

Důležitou částí programu je odesílání předdefinovaných zpráv. Pro tento účel byly vytvořeny třídy popsané v tabulce 5.2.

Název	Základní třída	Hlavičkový soubor
MessageDialog	QDialog	messagedialog.h
MessageData	QObject	messagedata.h
MessageWidget	QWidget	messagewidget.h
ShortcutEdit	QLineEdit	shortcutedit.h
MessageSettings	-	messagedialog.h

Tabulka 5.2: Tabulka tříd předdefinovaných zpráv.

Další sekcí kódu je implementace textových polí do kterých uživatel zadává znaky které chce odeslat a do kterých se vypisují přijaté znaky. Pro tento

účel byly vytvořeny třídy popsané v tabulce 5.3.

Název	Základní třída	Hlavičkový soubor
Console	QStackedWidget	console.h
ConsoleTextEdit	QTextEdit	savefile.h
ConsoleSettings	-	console.h

Tabulka 5.3: Tabulka tříd konzolí pro zobrazování dat.

Poslední částí je ukládání přijatých dat do souboru. Pro tento účel byly vytvořeny třídy popsané v tabulce 5.4.

Název	Základní třída	Hlavičkový soubor
SaveFileDialog	QDialog	savefiledialog.h
SaveFile	QFile	savefile.h
FileSettings	-	savefiledialog.h

Tabulka 5.4: Tabulka tříd ukládání dat.

5.1 Signály a sloty

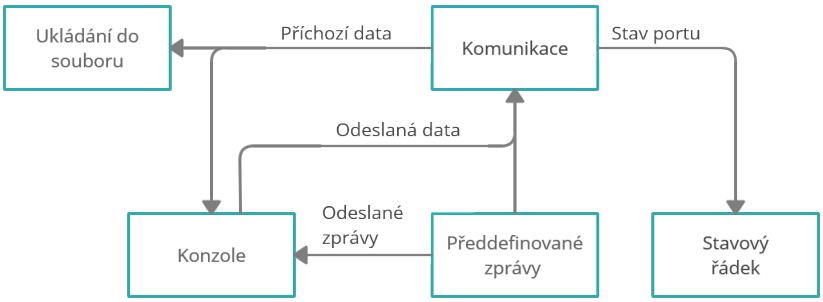
Je třeba zmínit jednu ze stěžejních funkcí Qt, mechanizmus signálů a slotů. Jednoduchým způsobem předávání dat mezi třídami je použití ukazatelů a funkcí příslušných tříd. Každá třída by pak měla v paměti uložené všechny propojené třídy, za účelem předávání informace. S větším počtem takovýchto spojení však vzniká problém čitelnosti kódu.

Alternativní metodou kterou Qt nabízí, je implementace výstupních funkcí třídy, takzvaných signálů a vstupních funkcí, takzvaných slotů. Ty je pak možné mezi třídami podle potřeby propojit tak, že když dojde v jedné třídě k zavolání signálu, zavolá se propojený slot. Tento slot může být součástí té samé, nebo i jiné třídy [9].

Důležitou výhodou je, že lze stejným způsobem volat signály z jednoho procesu propojené na sloty v procesu jiném. Není tak třeba nějak zvlášť ošetřovat vícevláknové aplikace.

5.2 Hlavní část

Hlavní třídou programu je třída MainWindow. Existuje pouze jedna její instance a obsahuje hlavní uživatelské rozhraní a ukazatele na většinu dalších objektů. Obsahuje zejména funkce na inicializaci programu, načítání a ukládání nastavení a aktualizaci grafického rozhraní.



Obrázek 5.1: Diagram toku dat mezi jednotlivými částmi programu.

5.2.1 Komunikace

Blok označený Komunikace na obrázku 5.1 představuje čtyři třídy, které jsou po inicializaci programu spuštěny každá ve vlastním procesu a obsluhují odchozí a příchozí komunikaci. Tyto třídy mají společné základní rysy a liší se pouze způsobem odesílání a přijímání dat. Data pro odeslání dostávájí od třídy Console jako uživatelský vstup z klávesnice, a od třídy MessageDialog jako odeslané zprávy. Přijatá data posílají do třídy Console pro zobrazení a do třídy SaveFile pro uložení do souboru, pokud je zapnuté.

5.2.2 Třída SerialPort

Tato třída zakládá na implementaci sériového přenosu dat třídy QSerialPort. Posílání a přijímání dat zůstává u této třídy nezměněno. Parametry jako baud rate, název portu, parita a počet datových bitů se posílají do této třídy pomocí signálů které jsou volány při jakékoliv změně jejich hodnot.

5.2.3 Třída UDPSocket

Tato třída zakládá na implementaci přenosu UDP packetů třídy QUdpSocket. Přijímání dat zůstává nezměněno. Odesílání dat probíhá dle seznamu adres a portů uloženého v této třídě. V pořadí definovaném třídou AddressDialog se odchozí data postupně odešlou na jednotlivé adresy. Seznam adres a portů se předává pomocí signálu, který je volán při změně hodnoty jakékoliv ze zadaných adres či portů.

5.2.4 Třída TCPClient

Tato třída zakládá na implementaci přenosu TCP packetů třídy QTcpSocket. Posílání a přijímání dat zůstává u této třídy nezměněno. Cílová adresa a port se předává signálem který je volán při změně jejich hodnot

5.2.5 Třída TCPServer

Tato třída zakládá na implementaci přenosu TCP packetů třídy QTcpServer. Třída si vytváří seznam připojených klientů a sleduje od kterých přicházejí data. Odesílání dat probíhá v pořadí připojení klientů. Odchozí data se postupně odešlou jednotlivým klientům. Port je při jakékoliv změně jeho hodnoty předán signálem této třídě.

5.3 Předdefinované zprávy

Blok označený Předdefinované zprávy na obrázku 5.1 představuje třídu MessageDialog. Ta je pomocí signálů a slotů přímo propojená na textovou konzoli a blok komunikace. Při stisknutí tlačítka odeslání kterékoliv zprávy, nebo při stisknutí uživatelem definované klávesové, se pomocí příslušného signálu do zmíněných tříd odešle sekvence znaků.

Třída MessageDialog obsahuje uživatelské rozhraní dialogového okna nastavení předdefinovaných zpráv a seznam uživatelem aktuálně zadaných zpráv. Této třídy existují čtyři instance, jedna pro každou metodu komunikace. Jednotlivé zprávy obsažené v této třídě jsou implementovány třídami MessageData a MessageWidget, které se starají o zpracování a zobrazování uživatelem zadaných zpráv. Nastavení zadaných zpráv se ukládá pomocí třídy MessageSettings.

5.4 Ukládání do souboru

Blok označený Ukládání do souboru na obrázku 5.1 představuje třídu SaveFile. Její slot na přijímání dat je připojen na příslušný signál přijímání dat bloku komunikace. Jakákoliv přijatá zpráva se tak zapíše do souboru. Toto spojení existuje pouze pokud uživatel spustil zapisování do souboru. V opačném případě je toho spojení rozpojeno.

Třída SaveFile je po inicializaci programu spuštěna ve vedlejším procesu a existují jí čtyři instance, jedna pro každou metodu komunikace. Poté, co uživatel spustí ukládání do souboru, dostane tato třída informace od třídy SaveFileDialog o názvu souboru a nastaveních. Poté čeká na přijetí dat příslušnou metodou komunikace a data zapisuje do nastaveného souboru. Třída SaveFileDialog obsahuje uživatelské rozhraní nastavení ukládání do souboru. Nastavení souboru a způsobu zapisování se ukládají pomocí třídy FileSettings.

5.5 Konzole

Blok označený Konzole na obrázku 5.1 představuje třídu Console. Ta je připojená podobným způsobem jako třída SaveFile na přijatá data. Je však také připojena slotem na data odeslaná pomocí předdefinované zprávy. Tyto zprávy se vypisují do okna odeslaných zpráv společně s uživatelským vstupem

z klávesnice. Pokud uživatel zadá znak z klávesnice, je pomocí signálu odeslán do bloku Komunikace.

Třída Console obsahuje tři instance třídy ConsoleTextEdit, jednu pro zobrazení přijatých i odeslaných dat současně, dvě pro zobrazení přijatých i odeslaných dat zvlášť. Mezi těmito lze přepínat pomocí kontextového menu. Všechna nastavení této třídy jsou přístupná pomocí kontextového menu. To bylo implementováno třídami QMenu a QAction. Nastavení formátování textu se ukládají pomocí třídy ConsoleSettings.

Kapitola 6

Testy spolehlivosti

Dle zadání je třeba otestovat funkčnost a spolehlivost terminálového programu. Jednou z požadovaných vlastností je, aby byl program stabilní i po delší době chodu. Další požadovanou vlastností je spolehlivost při ukládání a zobrazování dat, pokud je program přijímá bez zastavení při vysoké rychlosti, např. nad 100 Kbit/s. Pro otestování terminálového programu byl zvolen kit NUCLEO s procesorem STM32F303RE. Ten byl zvolen jednak pro svou dostupnost, ale také pro své dostačující parametry pro test programu. Hlavním parametrem byla rychlost, kterou přípravek dokáže posílat data. U tohoto procesoru to je (bez externích modifikací) 1 Mbit/s. Chování přípravku je dáno krátkým programem, který byl zhotoven v rámci práce v prostředí Keil. Přijatá data jsou ukládána do souboru a jejich korektnost je zkontrolována pomocí krátkého skriptu v programu Matlab, který byl také zhotoven v rámci práce.

Kód použitý pro dané testy se nachází v příslušných sekcích a obsahuje několik funkcí:

- Funkce GPIO_INIT nastaví vstupní a výstupní piny a zvolenou rychlost přenosu
- Funkce UART_GETCHAR čeká, dokud procesor nepřijme znak
- Funkce UART_SEND_NUM vyšle hodnotu uloženou v registru R0 jako sérii číslic
- Funkce UART_SEND_CHAR vyšle první byte uložený v registru R0 jako znak
- Funkce WAIT_MS čeká počet milisekund uložených v registru R0

6.1 Dlouhodobý test

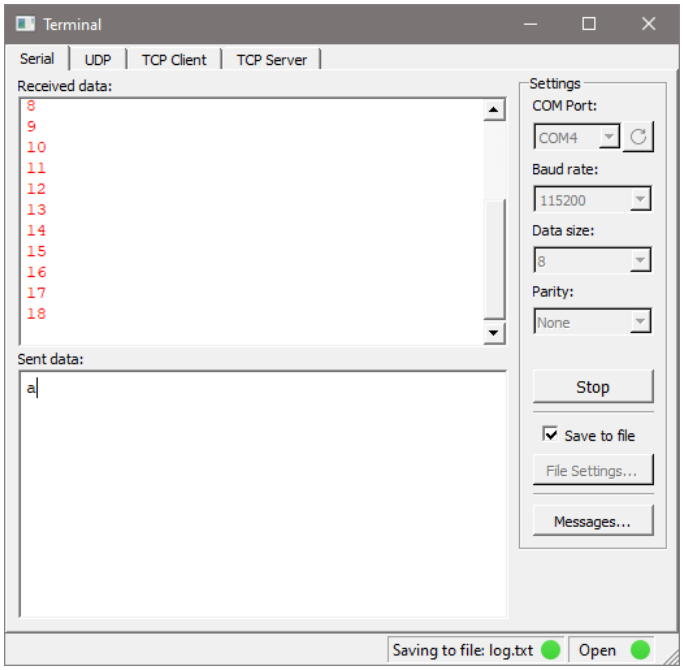
Cílem tohoto testu je ověřit, zda je program plně funkční po několika hodinách od spuštění. Samotný test spočívá v přijimání dat každých 5 sekund po dobu 9 hodin a následném ověření zda byla všechna data odeslaná za celou dobu testu přijata správně a že program lze dále používat i po odpojení přípravku po uplynutí stanovené doby. Test je spuštěn odesláním jakéhokoliv znaku do procesoru STM32F303RE pomocí sériového přenosu. Procesor poté až

do restartu odesílá každých 5 sekund sérii znaků odpovídajících číslicové reprezentaci aktuální iterace zakončenou znakem nového řádku.

Kód použitý pro posílání znaků výše popsaným způsobem pro procesor STM32F303RE:

```
MAIN
    BL GPIO_INIT
    MOV R1,#1
    BL UART_GETCHAR
MAIN_LOOP
    MOV R0,R1
    BL UART_SEND_NUM
    ADD R1,R1,#1
    MOV R0,#10
    BL UART_SEND_CHAR
    MOV R0,#5000
    BL WAIT_MS
    B MAIN_LOOP
```

Test proběhl dle očekávání. Po odpojení přípravku po přibližně 9 hodinách byla největší uložená hodnota číslo 6506. Pokud přípravek provedl 6506 iterací a každá trvala přibližně 5 sekund, lze zpětně dopočíst hodnotu času, která s malou odchylkou skutečně odpovídá 9 hodinám. Dále bylo ověřeno, že posloupnost čísel se zvyšuje vždy o stejnou hodnotu, a tedy že za celých 9 hodin nebyla ztracena žádná data. Po uplynutí 9 hodin a odpojení desky byla ověřena funkce programu odesláním a přijmutím několika znaků přes TCP přenos. Na obrázku 6.1 je vidět průběh testu.



Obrázek 6.1: Průběh dlouhodobého testu.

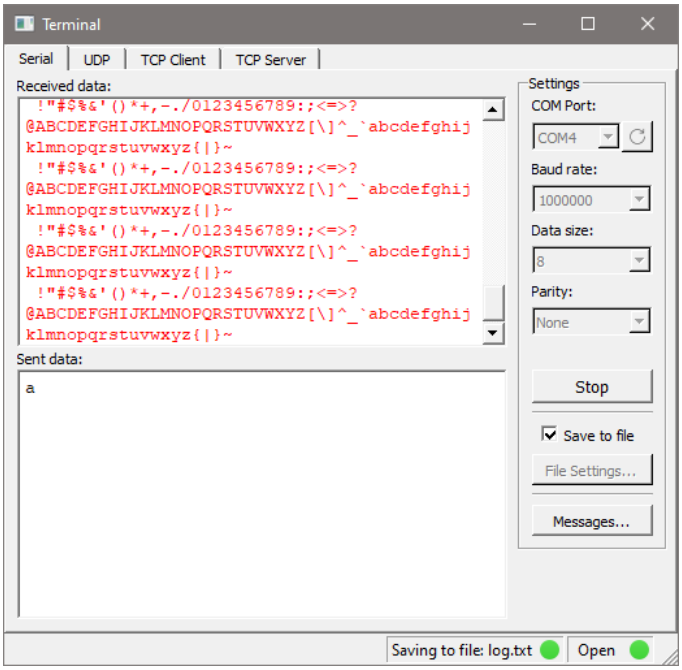
6.2 Rychlostní test

Cílem tohoto testu je ověřit, zda program dokáže přijímat nepřetržitý proud dat po sériové lince po dobu několika minut při různých datových rychlostech. Procesor odesílá data v iteracích. Jedna iterace obsahuje 256 bytů počínaje hodnotou 0, která se postupně zvyšuje o 1 až do hodnoty 255. V rámci testu provede procesor 10 000 iterací při zvolené rychlosti odesílání dat. Testované rychlosti jsou 115200 bit/s, 0.5 Mbit/s a 1 Mbit/s. V základním nastavení umožňuje procesor STM32F303RE rychlost přenosu až 0.5 Mbit/s při hodinovém cyklu 8 MHz. Tato rychlost se dá zdvojnásobit na maximální 1 Mbit/s nastavením příznaku OVER8 na 1 při nastavování UART komunikace. Tím se rychlost přenosu zdvojnásobí, za cenu poloviního počtu vzorků odebraných procesorem z komunikačních pinů [4].

Kód použitý pro posílání znaků výše popsaným způsobem pro procesor STM32F303RE:

```
MAIN
    BL GPIO_INIT
    MOV R1,#0
    MOV R2,#256
    MOV R3,#0
    MOV R4,#10000
    BL UART_GETCHAR
MAIN_LOOP
    MOV R0,R1
    BL UART_SEND_CHAR
    ADD R1,R1,#1
    CMP R1,R2
    BNE MAIN_LOOP
    MOV R1,#0
    ADD R3,R3,#1
    CMP R3,R4
    BNE MAIN_LOOP
MAIN_END
    B MAIN_END
```

Tento test byl proveden třikrát pro každou zvolenou datovou rychlost a také proběhl dle očekávání. Znakové posloupnosti ve výstupních souborech odpovídají vyslaným datům. Na obrázku 6.2 je vidět přijímání dat při rychlosti 1 Mbit/s.



Obrázek 6.2: Průběh rychlostního testu.

Kapitola 7

Závěr

Dle zadání práce byl vytvořen terminálový program, který umožňuje komunikaci pomocí sériové linky a protokolů UDP a TCP. Program obsahuje všechny funkce stanovené během průzkumu dostupných programů. Implementované funkce jsou

- Komunikace pomocí sériové linky a protokolů UDP a TCP
- Ukládání přijatých data do souboru
- Odesílání uživatelem definovaných zpráv
- Automatické ukládání nastavení aplikace a jejich načtení po spuštění
- Nastavení formátování přijatých dat
- Odesílání UDP packetů na více adres a portů
- Stavový řádek s informací o stavu programu

Ve vývoji programu je možné dále pokračovat implementací několika méně dostupných funkcí. Například přidáním příkazové řádky s historií odeslaných příkazů, nebo odesílání souborů a automatických odpovědí.

Funkce výsledného programu byla ověřena několika testy. Pomocí dostupné desky Nucleo STM32F303RE bylo zjištěno, že program dokáže spolehlivě přijímat nepřerušovaný přísun dat rychlostí až 1 Mbit/s. Pro otestování vyšších rychlostí je třeba využít rychlejšího procesoru. Také bylo ověřeno že program dokáže pracovat i delší časové intervaly, jak bylo ukázáno přijímáním dat 9 hodin. Konečná implementace tedy splnila všechny body zadání.

Literatura

[1] *Herkules*. https://www.hw-group.com/software, citováno dne: 1.1.2021.

[2] *Packet sender*. https://packetsender.com/, citováno dne: 1.1.2021.

[3] *Realterm*. https://realterm.sourceforge.io/, citováno dne: 1.1.2021.

[4] *Referenční manuál k stm32f303re*. https://www.st.com/content/ccc/resource/technical/document/reference_manual/4a/19/6e/18/9d/92/43/32/DM00043574.pdf/files/DM00043574.pdf/jcr:content/translations/en.DM00043574.pdf, citováno dne: 1.1.2021.

[5] *Termite*. https://www.compuphase.com/software_termite.htm, citováno dne: 1.1.2021.

[6] M. Chroboczek, *Grafická uživatelská rozhraní v Qt a C++*, 2013.

[7] L. Parziale, *TCP/IP Tutorial and Technical Overview*, 2006.

[8] S. Prata, *Mistrovství v C++*, 2013.

[9] J. Thelin, *Foundations of Qt development*, 2007.