FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Konfigurovatelný terminál pro rychlou komunikaci s externím zařízením s pomocí sériové linky
Student:	Juraj Kožík
Vedoucí:	Ing. Pavel Kubalík, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

1) Prozkoumejte existující řešení.
2) Pomocí metod softwarového inženýrství navrhněte vlastní řešení vyhovující níže uvedeným požadavkům.
3) Navržené řešení naprogramujte, řádně ho zdokumentujte a otestujte.
4) Požadavky:
- platforma OS MS Windows,
- výběr vhodného programovacího jazyka,
- třída pro vlastní komunikaci a definici jednoduchých protokolů,
- GUI pro testování a ukázku funkčnosti vytvořené třídy,
- možnost snadného nastavení parametrů sériové linky včetně její dostupnosti,
- posílání raw dat po sériové lince maximální dostupnou rychlostí,
- posílání zabezpečených dat po sériové lince,
- posílání souboru,
- možnost implementace jednoduchého komunikačního protokolu na základě regulárního výrazu.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 12. října 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Konfigurovatelný terminál pro rychlou komunikaci s externím zařízením s pomocí sériové linky

Katedra softwarového inženýrství
Vedoucí práce: Ing. Pavel Kubalík, Ph.D.

4. ledna 2022

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce, panu Ing. Kubalíkovi, Ph.D., za cenné rady, odborné vedení práce a za vynaložený čas. V neposlední řadě chci také poděkovat rodině a kamarádům za podporu během tvorby práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 4. ledna 2022 ……

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2022 Juraj Kožík. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.

Abstrakt

Tato práce se zabývá rychlou komunikací pomocí sériového portu. Cílem práce je analýza, návrh a implementace aplikace, které je základem třída pro jednoduchou komunikaci na sériovém portu. Zároveň umožní uživateli některá tato data šifrovat, uchovávat si přijatá data v souboru. Dále umožňuje definovat vlastní komunikaci pomocí regulárních výrazů. Řešení je vytvořeno jako aplikace s grafickým uživatelským rozhraním v jazyce C++ s frameworkem Qt.

Klíčová slova Sériový port, komunikace, baudová rychlost, RSA, AES, MS Windows, C++, Qt

Abstract

This thesis focuses on fast communication using serial port. Aim of the this thesis is the analysis, design and implemention of an aplication, which basis is a class for simple communication on serial port. At the same time, it allows the user to encrypt some of this data and logging the received data in a file. It also allows user to define your own simple communication protocol using regular expressions. The solution is created as an application with graphical user interface using C++ with the Qt framework.

Keywords Serial port, communication, baud rate, RSA, AES, MS Windows, C++, Qt

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V dnešní době jsou počítače rozšířené téměř všude. Samotné počítače ovšem často nestačí a je potřeba k nim připojit další zařízení, nebo propojit dva počítače dohromady. K tomuto slouží celá řada různých portů. Patří mezi ně i známé USB, HDMI, VGA a například sériový port, který je využit v bakalářské práci. Pomocí tohoto portu se k počítači připojovala různá zařízení jako například klávesnice, myši a modemy. Sloužil také k propojování samotných počítačů, čímž se vytvářely menší počítačové sítě.

V současné době se sériový port na osobních počítačích vyskytuje méně často. Nicméně je k nalezení v průmyslu nebo ve školách pro výuku některých předmětů v oblasti struktury počítačů. Mnoho nadšenců využívá sériový port k programování a menším projektům na přípravcích jako „Arduino“ , kterému se dá naprogramovat mnoho funkcí.

Hlavním cílem práce je navrhnout, implementovat a otestovat aplikaci, která bude posílat data sériovým portem zvolenou rychlostí. Práce se nejdříve zaměří na problematiku použití sériového portu a dále také na prozkoumání stávajících řešení a na jejich rozdíly. Poté se další kapitola zaměří na výběr vhodných technologií a návrh aplikace. Poté bude následovat popis implementace a testování vyvíjené aplikace.

Kapitola 1

Cíl práce

Cílem bakalářské práce je navrhnout a implementovat aplikaci, které základem je třída pro jednoduchou komunikaci na sériovém portu. Aplikace bude navržena za pomoci metod softwarového inženýrství a testování vzniklé aplikace pro platformu OS MS Windows. Dále je také důležité vybrat vhodné technologie pro tvorbu práce jako třeba programovací jazyk. Požadavky na bakalářskou práci jsou:

- GUI pro testování a ukázku funkčnosti vytvořené třídy
- možnost snadného nastavení parametrů sériové linky včetně její dostupnosti
- posílání raw dat po sériové lince maximální dostupnou rychlostí
- posílání zabezpečených dat po sériové lince
- posílání souboru
- možnost implementace jednoduchého komunikačního protokolu na základě regulárního výrazu

Kapitola 2

Rešerše

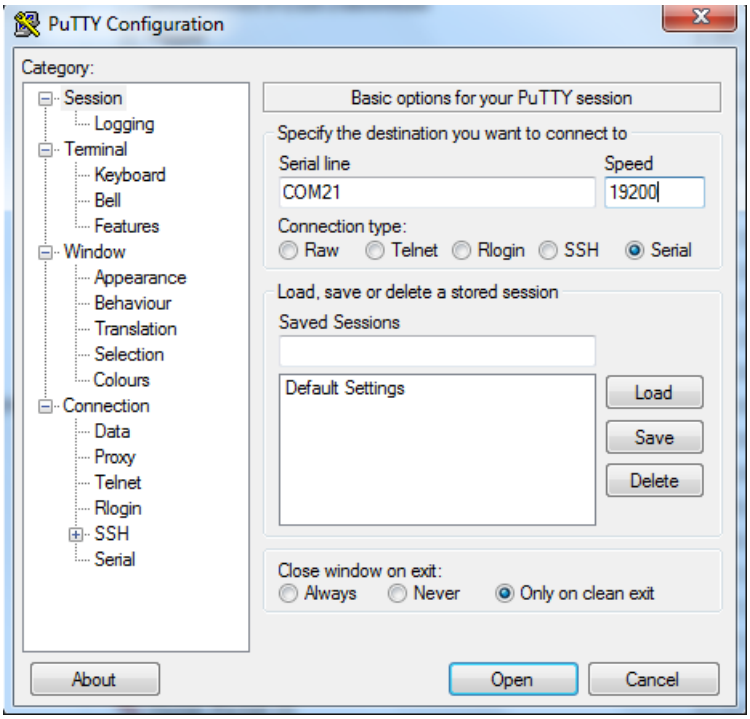
Tato kapitola se zaměřuje na současný stav, tj. jaké aplikace jsou v současné době využívány a co jednotlivá řešení nabízí. Také bude uvedena tabulka s přehledem nabízených vlastností aplikací.

2.1 Současný stav

V současné době existuje pro komunikaci na sériovém portu několik aplikací, se kterými je možné pracovat. Práce představí některé z nich. Jednou z nejznámějších aplikací je PuTTY. Dále se využívá méně známá aplikace Hercules SETUP a aplikace RealTERM. Všechny aplikace podporují komunikaci na sériovém portu a mají i další funkce. Pro internetovou komunikaci je využívána zejména aplikace PuTTY.

2.2 PuTTY

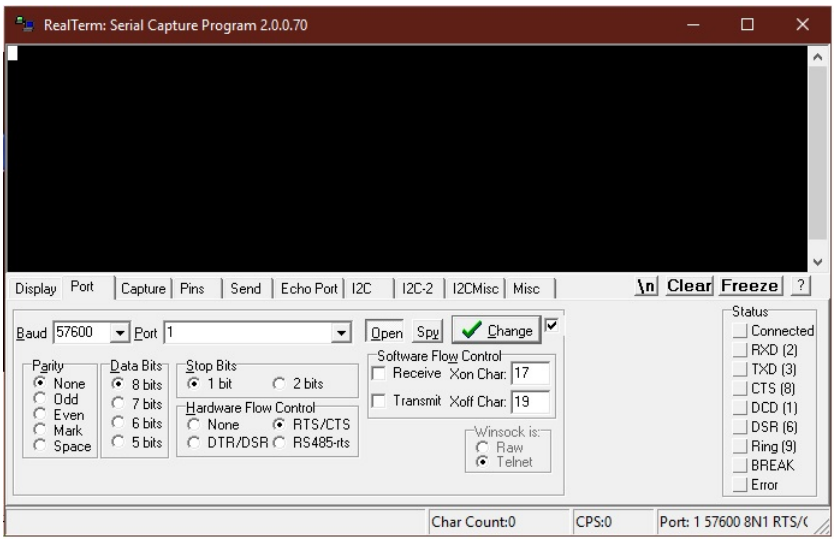
PuTTY [1] je volně dostupný terminálový klient pro platformy Windows a Linux. Nabízí množství funkcí. PuTTY je neznámější pro využití funkcí SSH a Telnet klient. Nicméně nabízí také klienta pro příkazový řádek SCP a také pro holý TCP. Jak bylo zmíněno výše, PuTTY podporuje mnoho variant pro zabezpečený vzdálený terminál, který nabízí několik možností šifer konkrétně například AES, 3DES, DES a jiné.



Obrázek 2.1: PuTTY

2.3 RealTerm

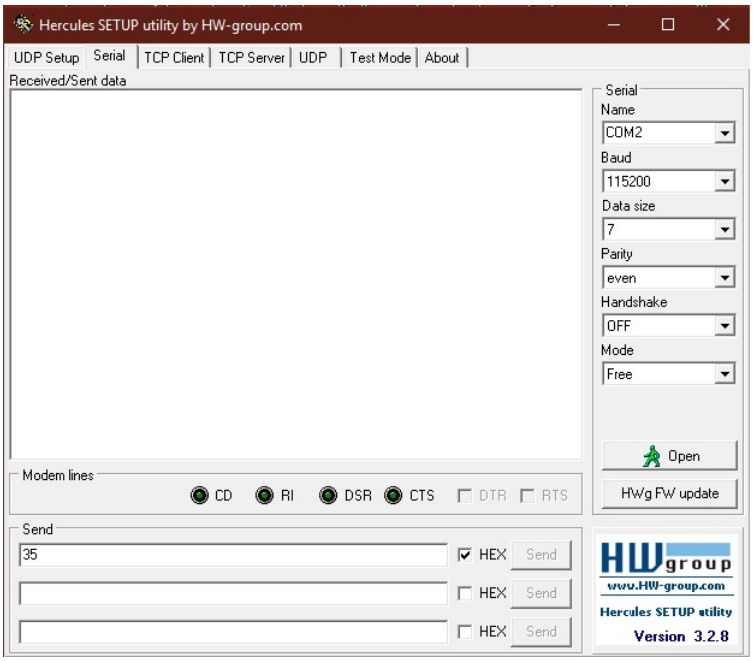
RealTerm [2] je aplikace pro Windows, která je pokračovatelem známého DOS programu COMMHEX. RealTerm se dá ovládat pomocí příkazové řádky a pomocí ovladače ActiveX. Dále ji lze ovládat pomocí myši anebo ji lze propojit s programy jako Excel [3] a Matlab [4]. Tato aplikace dokáže zobrazovat data přijímaná přes port, vysílat sekvence dat a čísel nebo také celý soubor. Dále umožňuje zachytávat přijímaná data a zapisovat je do souboru.



Obrázek 2.2: RealTerm

2.4 Hercules SETUP

Aplikace Hercules SETUP [5] je terminál pro práci na sériového portu, UDP/IP a TCP/IP a je volně dostupná. Tato aplikace spolupracuje s virtuálními sériovými porty. Umožňuje přijímat a odesílat data i opakovaně. Dalšími funkcemi aplikace je zobrazení speciálních znaků (v ASCII, HEX nebo DEC), zálohování komunikace do souboru a vytváření a odesílání testovacího souboru. Aplikace nenabízí bezpečnou komunikaci pomocí šifrování.



Obrázek 2.3: Hercules SETUP

2.5 Zhodnocení

Každá ze zmíněných aplikací se liší v použitelnosti a v nastavitelnosti. Aplikace PuTTY umožňuje nastavit pouze rychlost komunikace a výběr sériového portu, který uživatel musí znát, jelikož se zadává ručně do textového pole. Aplikace RealTerm nabízí oproti aplikaci PuTTY mnohem víc nastavení. Uživatel si může nastavit *Paritu*, *Datové bity*, *Stop bity* a další parametry. Hercules SETUP nabízí rovněž jako aplikace RealTerm nastavení rychlosti komunikace, *Paritu*, *Datové bity* a také *Stop bity*.
Vybrané aplikace se liší v použitelnosti a také ve funkcích, které nabízí. Tabulka ukazuje rozdíly mezi jednotlivými aplikacemi a také vyvíjenou aplikací.

Tabulka 2.1: Porovnání aplikací

Funkcionalita	PuTTY	RealTerm	Hercules	Vyvíjená aplikace
Seznam dostupných COM portů	Ne	Ano	Ano	Ano
Zobrazování průběhu komunikace	Ne	Ne	Ano	Ano
Posílání souborů	Ano	Ano	Ano	Ano
Ukládání příchozích zpráv do souboru	Ne	Ano	Ano	Ano
Nastavení vlastního typu komunikace(regex)	Ne	Ne	Ne	Ano
Bezpečná komunikace	Ano	Ne	Ne	Ano

Kapitola 3

Analýza

Tato kapitola se zaměřuje na výběr jednotlivých technologií a postupů pro vyvíjenou aplikaci. Také zde bude uveden zběžný přehled různých vnějších konektorů používaných v minulosti a i konektor USB (i typ C), který bude sloužit k porovnání starších a nejnovějších technologií v oblasti konektorů.
Moderní počítače využívají rovnou několik vnějších konektorů, které umožňují připojit nejrůznější periferie pro ulehčení práce s počítačem. Pro připojení monitoru k počítači sloužilo již více konektorů, například D-SUB analogový port, DVI digitální konektor, později VGA, dnes HDMI nebo Display Port. Pro klávesnici a myš to byl například PS/2 konektor, který se využíval například i k připojení trackballu. Sériový port, známy také jako RS-232, byl dělán ve dvou variantách, a to 9 pinů nebo 25 pinů, který se používal například k přípojení myši či modemu. Sériový port umožňuje posílání jednotlivých bitů postupně v sérii za sebou. Naproti tomu paralelní port umožňuje přenášet několik bitů současně, obvykle 8 bitů. Využíván byl obvykle k připojení tiskáren, scannerů a podobně. USB port je nyní nejpopulárnější port pro mnoho periferií, například klávesnice, myši, monitory (USB-C), napájení a další. Poslední tři porty budou v další části více blíže popsány.

3.1 Sériový port(RS–232C)

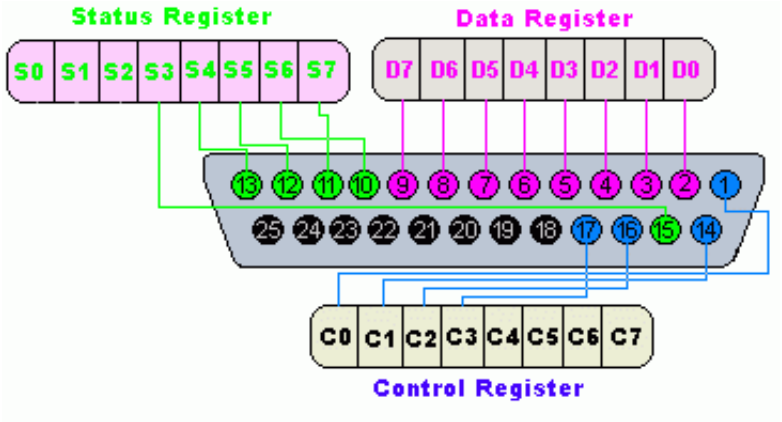
Sériový port umožňuje dvěma propojeným zařízením provádět takzvanou sériovou komunikaci, čímž se rozumí posílání jednotlivých bitů postupně v sérii za sebou. Dle [6] byl původně navržen jako připojení terminálů k jednoduchému modemu nebo přímo k blízkému serveru. Později se však využití sériového portu rozšířilo na různé další technologie, například počítačové myši, klávesnice, dotykové LCD a tiskárny. Dokonce se pomocí sériového portu propojovali počítače a budovali se jednoduché počítačové sítě. Tento typ portu se využívá dodnes jak u osobních počítačů, tak i v jiných zařízeních. U osobních počítačů však jeho využití ubývá a postupně je nahrazován jinými typy portů, nejčastěji USB portem. V průmyslu se nicméně stále používá například k programování regulátorů motorů v průmyslových strojích a jako frekvenční měnič a podobně. Sériový port se též nadále využívá k nastavování některých aktivních síťových prvků (Cisco). Mnohé servery též podporují terminály připojené k sériovým portům, například známý program PuTTY.

Obrázek 3.1: Sériový port

3.2 Paralelní port

Obrázek 3.2: Paralelní port

Paralelní port [7] na rozdíl od sériového portu posílá bity paralelně, a to tak, že je použito více vodičů. Některé vodiče jsou rezervovány pro přenos dat, a jiné pro řídící účely. Konektor paralelního portu obsahuje dvacet pět pinů ve dvou řadách, a to po třinácti a dvanácti pinech. Piny jsou rozděleny do čtyř skupin. V první skupině je osm pinů pro přenos dat, přičemž jejich počet není náhodný. Osm proto, aby šel najednou poslat celý byte. Druhá skupina obsahuje čtyři piny pro řídící vodiče, které umožňují počítači řídit veškerou komunikaci s připojeným zařízením. Třetí skupinou jsou stavové piny, které naopak používá připojené zařízení pro komunikaci s počítačem. V této skupině se nachází pět pinů. Poslední, čtvrtou skupinu tvoří piny, které jsou využity jako uzemnění. Ke každému datovému vodiči patří jeden takovýto vodič, tedy jich je také osm.
Běžně byl tento druh portu využíván k připojení například tiskáren, skenerů, starších webkamer a nebo starších externích CD-ROM a hard disků.



Obrázek 3.3: Paralelní port - detail

3.3 USB port

USB port [8] je také známý jako universální sériová sběrnice, z čehož vznikla i jeho zkratka. V současnosti je tento port nejrozšířenějším ze všech, přičemž do značné míry nahradil první dva zmíněné porty, a to jak sériový tak paralelní. Výhodou je jistě to, že zde byl vyřešen problém s napájecím napětím, a zaručenou proudovou zátěží, kde sériový port jistě zaostával. Další nezpochybnitelnou výhodou je provedení propojení jednotlivých komunikačních uzlů, kde není potřeba pro opětovné připojení a odpojení vypínat nebo zapínat připojované zařízení ani zařízení, ke kterému se periferie připojuje. Přes tento port lze připojit myši, klávesnice, monitory, chytré telefony, kamery, tiskárny, web kamery a mnoho dalších. Nicméně aby to bylo vše možné, prošel USB port znatelným vývojem.

Obrázek 3.4: USB typy

Tabulka 3.1: Porovnání verzí USB dle rychlostí

USB Verze	Rok vydání	Maximální rychlost v Mbps
USB 1.1	1998	12 Mbps
USB 2.0	2000	480 Mbps
USB 3.0	2008	5 120 Mbps
USB-C (USB 3.2 Gen 1)	2014	10 240 Mbps

3.4 Baudová rychlost

Přenosová rychlost dat se vyjadřuje jednou z následujících možností:

- bity za sekundu
- baudy

Bity za sekundu udává kolik bitů se přenese za jednu sekundu, značíme bits/s nebo bps (bits per second).
Obvykle se nesetkáváme se základní jednotkou ale spíše s jejími násobky.

Tabulka 3.2: Násobky bitů za sekundu

Název	Zkratka	počet bitů
Kilobit za sekundu	kbps	1 000 bps
Megabit za sekundu	Mbps	1 000 000 bps
Gigabit za sekundu	Gbps	1 000 000 000 bps

Baudová rychlost [9] se liší od bitů za sekundu v tom, že baudová rychlost udává počet změn signálu nebo symbolu za jednu sekundu. Symbol může být změna v napětí, frekvence nebo také fáze. Binární NRZ[1], které sériový port využívá, má dva symboly - 1 a 0. Tyto symboly představují úroveň napětí. V případě, že jsou využívány pouze dva symboly, jsou baudová rychlost a přenosová rychlost stejné. Pokud symbol představuje více bitů, pak je baudová rychlost vyšší a je nutné využít modulační techniky pro přenos dat. Například pokud je baudová rychlost 4 800 a každý symbol představuje dva bity, pak je přenosová rychlost rovna 9 600 bitů za sekundu. Většinou je symbol tvořen z počtu bitů mocniny dvou. Mějme N, které představuje počet bitů na symbol. Počet požadovaných symbolů pak je $S = 2^N$. Výpočet přenosové rychlosti by vypadal následovně:

$$R = baudrate * \log_2 S = baudrate * 3,22 * log_{10} S$$

Pokud bychom zvolili baudovou rychlost 4 800 baudů, přičemž by symbol představovaly dva bity, pak je potřeba $2^2 = 4$ symboly. Přenosová rychlost by tedy byla:

$$R = 4800 * 3,32 * \log 4 = 4800 * 2 = 9600bps.$$

[1] NRZ pochází z anglického Non Return To Zero - v překladu znamená bez návratu k nule

V následující tabulce jsou uvedeny rychlosti podporované vetšinou sériových portů.

Tabulka 3.3: Baudové rychlosti

Baudové rychlosti
110
300
600
1200
2400
4800
9600
14440
19200
28800
38400
56000
57600
115200
128000
153600
230400
256000
460800
921600

3.5 Převodník USB/RS–232

Převodník [10] je kabel zakončený na jednom konci jako USB typu A a na druhém konci jako sériový port RS–232. Po zapojení USB do počítače se tento převodník zobrazuje jako sériový port, který si uživatel může nastavit v libovolném rozmezí COM1 až COM256 pomocí správce zařízení. S převodníkem se pracuje úplně stejně jako se sériovým portem. Lze ho také použít pro různé aplikace standardně fungující na sériovém portu jako například sériové tiskárny, skenery a další. Jelikož je převodník připojen přes USB, je rychlost na sériovém portu nastavena na maximální možnou hodnotu. Tato hodnota může dosahovat až několik mega baudů, záleží na konkrétním převodníku.

Tabulka 3.4: Porovnání konektorů dle rychlostí

Konetkor	Maximální rychlost v kbps
Seriový port	115.2 kbps
Paralelní port	150 kbps
Paralelní port (EPP)	16 000 kbps
USB 1.1	12 000 kbps
USB 2.0	480 000 kbps
USB 3.0	5 120 000 kbps
USB-C (USB 3.2 Gen1)	10 240 000 kbps

3.6 Technologie

Tato kapitola se zaměří na porovnání a výběr programovacího jazyka, grafického frameworku a dalších technologií.

3.6.1 MS Windows

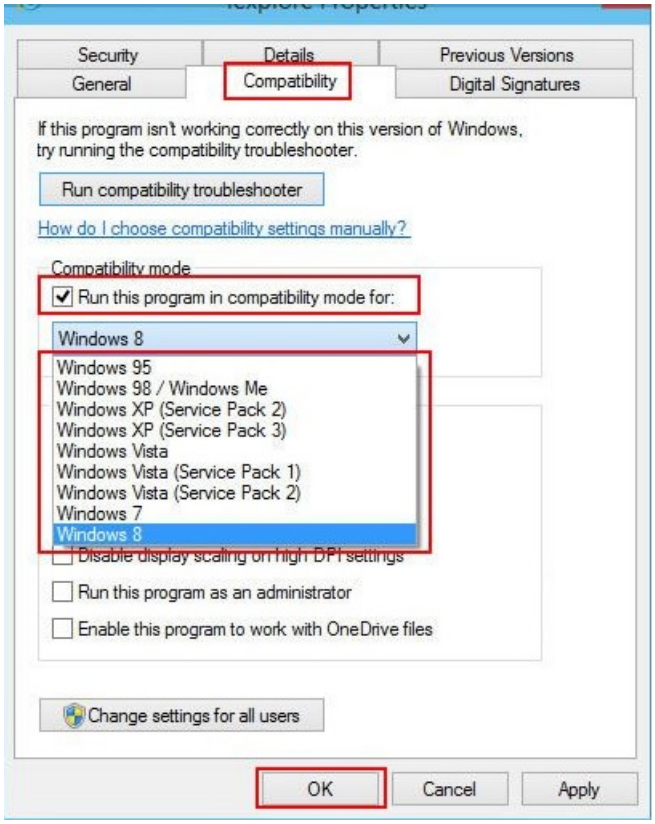
Operační systém Windows [11] vyvinula společnost Microsoft. V roce 1980 začínající společnost Microsoft vydala operační systém pod názvem Microsoft Disk Operating System (MS-DOS). Po roce od vydání se začaly prodávat první počítače s tímto systémem. Zároveň společnost začala s vývojem grafického rozhraní. V průběhu let společnost Microsoft vytvořila mnoho systémů Windows.

Důvodem použití platformy Windows je zejména jeho rozšíření a poměr na trhu. Výhodou je také možnost používat technologie předchozích verzí Windows-u pomocí módu kompatibility, viz 3.5. Aplikace je vytvářena na platformě Windows 10. Windows 11 není ještě plně funkční a vývoj aplikace by nebyl bezproblémový. Nicméně, díky funkci kompatibility od Windows bude možné aplikace pouštět i na Windows 11.

3.6.2 Programovací jazyk

Tato sekce se zaměří na výběr programovacího jazyka, který bude využit při implementaci řešení. Požadavkem na jazyk jsou zejména rychlost provádění instrukcí.

Java [12] je jeden z nejpoužívanějších jazyků. Jednou z nejvýraznějších vlastností jazyka Java je jeho téměř výhradní zaměřenost na objektově orientované programování. Do značné části byl tento jazyk odvozen od jazyka C++. Díky své přenositelnosti mezi různými platformami je populární. Není až tak rychlý jako například C++.



Obrázek 3.5: Mód kompatibility

Python [13] je dynamický interpretovaný vysokoúrovňový programovací jazyk. Často bývá zařazen mezi skriptovací jazyky. Také je objektově orientován. Jedna z největších výhod jazyka Python je velké množství knihoven pro jakoukoliv problematiku. Výkonnost Pythonu je velice nevyrovnaná. V současné době jsou počítače velice výkonné a tak se nejedná o tak výrazný problém.

C# [14] je programovací jazyk, který vyvinula společnost Microsoft. Je to objektově orientovaný vysokoúrovňový programovací jazyk. Byl vyvinut ze základů jazyků C++ a Java. Využívá se k vytváření databázových aplikací, webových aplikací, webových stránek a služeb, formulářových aplikací pro Windows a aplikace pro mobilní zařízení.

C++ [15] je nízkoúrovňový, objektově orientovaný jazyk. Jazyk C++ vychází z jazyka C. Programátor má na starost správu všech zdrojů. Vzhledem k tomu velice lehce dochází k chybám. Na druhou stranu je ale C++ velice rychlý a efektivní jazyk. Hlavní využití nachází v oblasti vývoje videoher, kde je

zapotřebí rychlost pro práci s grafikou (3D kvůli častému přepočítávání) a také v oblasti desktopových aplikací. Jelikož je jazyk C++ jedním z nejrozšířenějších programovacích jazyků této doby, lze za jeho plus považovat velkou základnu uživatelů, která může programátorovi pomoci s jakýmkoliv problémem při používání jazyka. Jazyk C++ byl vybrán zejména kvůli jeho rychlosti a efektivitě.

3.6.3 Visual Studio

Pro vývoj vytvářené aplikace bylo použito Visual Studio [16] od Microsoftu. Hlavním důvodem výběru Visual Studia byl výběr C++ jako programovacího jazyka na MS Windows. Byla využita školní licence. Výhodou je jednoduché vývojové prostředí pro vývoj konzolových aplikací, aplikací s grafickým rozhraním, pro vývoj webových služeb a také i pro aplikace spolupracující s databázemi a mnohé jiné.

Visual Studio obsahuje mnoho nástrojů pro vývoj. Nejdůležitějším nástrojem je editor kódu podporující IntelliSense a refaktorování kódu. Dále obsahuje i integrovaný debbuger, který pracuje na úrovni kódu a také na úrovni stroje. Dalšími nástroji jsou například designer formulářů pro tvorbu webových aplikací a pro tvorbu aplikací s uživatelským grafickým rozhraním, designer webu a tříd, databázových schémat a dalších. Visual Studio podporuje mnoho jazyků, mezi které patří C/C++, C#, F#, Python, Ruby, JavaScript a jiné. Největším přínosem bylo lehké vytvoření Qt projektu a jeho úpravy.

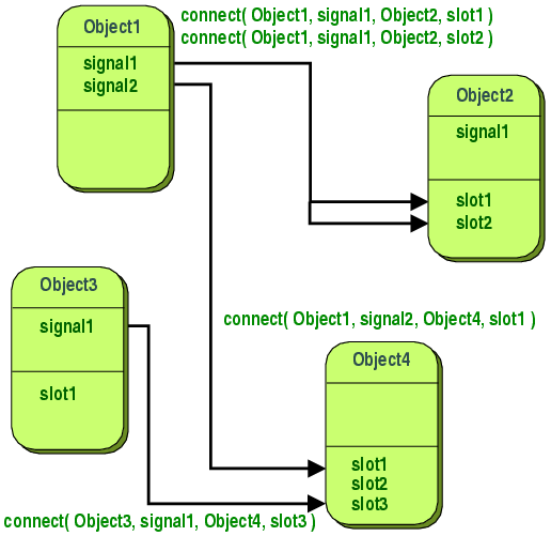
3.6.4 Grafický framework

Při výběru grafického frameworku záleží zejména na lehkém a intuitivním ovládání. Stejné důležité při výběru je jeho začlenění do Visual Studia, které bylo vybráno jako vývojové prostředí. Grafický framework Qt splňuje obě tyto požadavky. Lze ho přidat do Visual Studia pomocí instalace jako plugin.

Qt [17] je multiplatformní aplikační vývojový framework široce používaný pro vytváření aplikací s grafickým uživatelským rozhraním určeným pro různé softwarové a hardwarové platformy. Qt je knihovna programovacího jazyka C++, ale existuje i pro jiné jazyky, například Python, Ruby, C, Perl, C#, Java a další. Podporuje také SQL, zpracování XML, správu vláken, přístup k souborům a práci s grafikou a multimédii. Qt funguje na platformách Windows, Linux a iOS. Největší výhodou je lehké použití ve Visual Studiu - lze jej nainstalovat pomocí pluginu. Propojení grafického rozhraní a jádra aplikace bude zajišťovat technologie Signals/Slots od Qt.

3.6.5 Šifrovací algoritmy

K šifrování budou použity šifrovací algoritmy RSA a AES. Algoritmus RSA je použit pro šifrování kratších posílaných zpráv. Pro šifrování posílaných sou-



Obrázek 3.6: Signal/Slot

borů bude použit algoritmus AES.

Algoritmus RSA, [18] pojmenovaný po jeho tvůrcích (Ronald **R**ivest, Adi **S**hamir a Leonard **A**dleman), je asymetrická šifra, která se používá jak pro šifrování, tak i pro podepisování dokumentů. Je založena na matematickém výpočtu známém jako *Eulerova věta*. Asymetrická šifra má dva klíče. Jeden je veřejný a druhý soukromý. Veřejný klíč je dostupný všem, soukromý klíč si držitel pečlivě uschovává. Pokud chce někdo poslat zašifrovanou zprávu držiteli soukromého klíče, zašifruje zprávu pomocí veřejného klíče. Tuto zprávu lze dešifrovat pouze pomocí soukromého klíče.

Algoritmus AES [19] (**A**dvanced **E**ncryption **S**tandard) je symetrická blokova šifra. Autory jsou Joan Daemen a Vincent Rijmen. Byl také podle autorů nazýván „Rijndae“. AES jako symetrická šifra má pouze jeden klíč, který je používán u šifrování i u dešifrování. Nahradila dříve užívanou šifru DES, která kvůli délce klíče již nebyla bezpečná. AES podporuje různé délky klíčů (128 bitů, 192 bitů a 256 bitů), přičemž čímž je větší délka klíče, tím efektivněji zabraňuje útokům, hlavně brute force. U použití symetrické šifry je problematické bezpečné posílání klíče druhé straně. Z tohoto důvodu se používá v kombinaci s asymetrickou šifrou, například RSA. Šifrování pomocí AES je rychlé, a proto se hodí také na šifrování souborů.

Další důležitou funkcionalitou je definice regulárního výrazu. Regulární výraz je typ řetězce, kterým se popisuje celá množina řetězců. Tato množina se popisuje speciálním zápisem. V tabulce je uvedeno několik příkladů jakými způsoby lze popsat jistou množinu řetězců.

Regulární výrazy [20] se nejčastěji používají při práci s textem. Pomocí

Tabulka 3.5: Regulární výrazy

Výraz	Význam
z	znak z
*	výskyt znaku libovolně mnoho krát (včetně 0-krát)
?	výskyt znaku jednou nebo vůbec
[abcd]	skupina znaků, jeden ze znaků a, b, c nebo d
[^abcd]	jakýkoliv znak kromě a, b, c nebo d
[a-zA-Z]	jeden ze znaků a až z, nebo A az Z, včetně krajních znaků
[0-9]	libovolné z čísel 0 až 9, včetně
^	začátek řádku
$	konec řádku

těchto výrazů lze v textu vyhledávat a nahrazovat znaky jinými a přetvářet data do požadované podoby. Uživatel si může předdefinovat typy výrazů, které bude aplikace přijímat a posílat. Pokud se zadaný výraz neshoduje s definicí, je uživatel informován o chybě.

Kapitola 4

Návrh řešení

Tato kapitola se zaměřuje na návrh požadované aplikace. Návrh aplikace se skládá z více částí. První částí je vytvoření funkčního konceptu pomocí konzolové aplikace. Součástí této aplikace bude jádro aplikace, které bude také použito i při aplikaci s grafickým uživatelským rozhraním, kde toto rozhraní bude vytvořeno pomocí Qt platformy. Obrázek 4.1 zobrazuje schéma navrhované aplikace. Na obrázku 4.2 základní návrh tříd obsahuje kontrolní třídu, třídu pro komunikaci a třídu sériového portu. Tento návrh je pouze předběžný, může se měnit podle potřeby. U funkcí tříd nejsou uvedeny vstupní a výstupní parametry, a také jejich návratové hodnoty. U proměnných, nakolik se jedná o návrh, nejsou uvedeny typy.

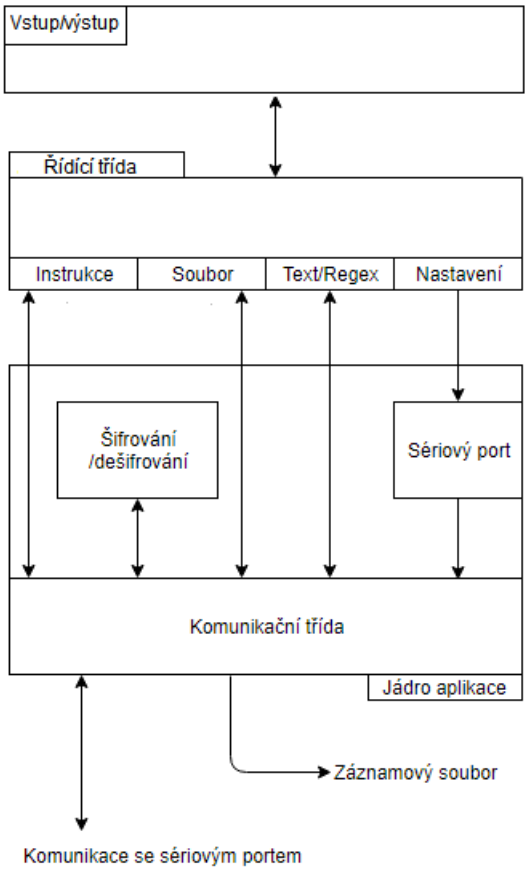
4.1 Návrh jádra aplikace

Jádro aplikace se bude starat o přípravu a zahájení komunikace na sériovém portu a o další potřebné činnosti.

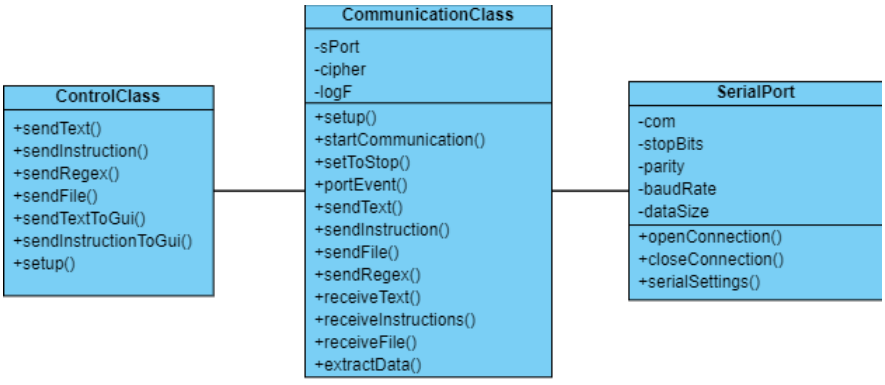
Základem bude třída představující sériový port, která bude nastavena pomocí vstupu od uživatele. O načítání nastavení pro sériový port konzolové aplikace se bude starat samostatná třída. U aplikace s grafickým prostředím tato třída nebude potřebná, načítání jednotlivých nastavení obstará grafické rozhraní, kde si uživatel vybere nastavení pomocí komponent grafického rozhraní, například combo box.

Nejdůležitější částí aplikace bude třída, která bude zajišťovat komunikaci přes sériový port. Dále bude zajišťovat předání přijatých informací uživateli pomocí grafického rozhraní nebo informace předá pro zobrazení na konzoli.

Aplikace umožňuje šifrování dat typu instrukce a souborů. Pro šifrování dat jsou zvoleny dva šifrovací algoritmy. První je algoritmus RSA, který je vhodný zejména pro kratší data, proto bude použit pro šifrování dat typu instrukce. Druhý zvolený algoritmus je AES, který je vhodný pro šifrování větších dat, například souborů.



Obrázek 4.1: Návrh aplikace



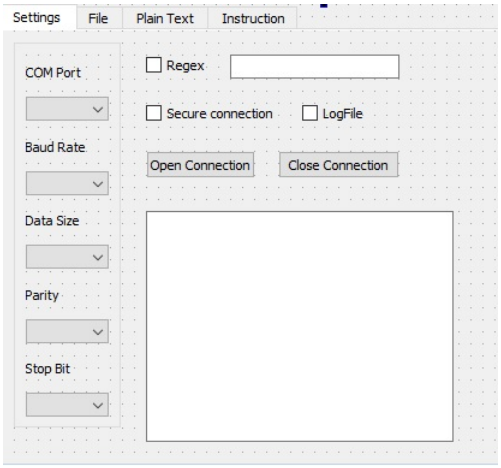
Obrázek 4.2: Základní návrh tříd

Pro logování komunikace je zde samostatná třída, která se stará o vytvoření a zapisování komunikace. Třída vytvoří logovací soubor na disku C ve složce

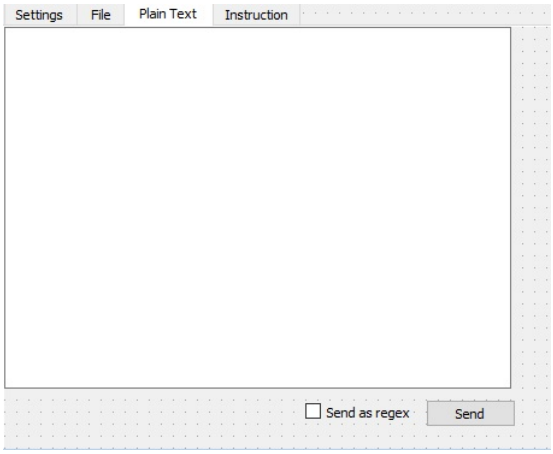
Temp.

4.2 Návrh grafického uživatelského rozhraní

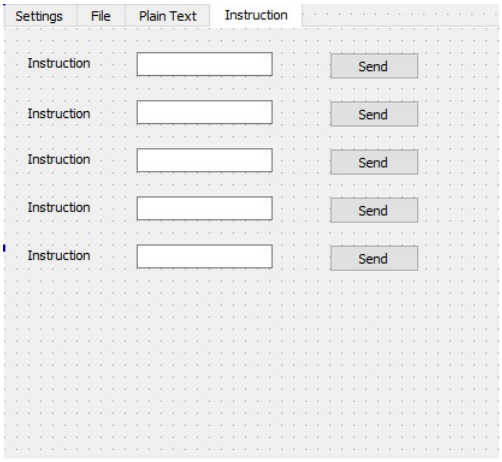
Grafická část aplikace se bude skládat z jednotlivých záložek s různými funkcemi. Po otevření aplikace bude uživateli zobrazena první záložka „Nastavení“, viz obrázek 4.3. V záložce se nastavují jednotlivé parametry pro sériový port, regulární výraz, soubor pro záznam komunikace a nastavení bezpečné komunikace. Dále záložka s nastavením slouží k otevření spojení na sériovém portu, ověření spojení a jeho ukončení. Druhá záložka „Text“ viz obrázek 4.4 slouží k posílání klasického textového řetězce nebo řetězce definovaného prostřednictvím regulárního výrazu. Záložka „Instrukce“ viz obrázek 4.5 umožňuje posílat až pět různých instrukcí opakovaně. Poslední záložka „Soubor“ viz obrázek 4.6 nabízí výběr libovolného souboru, který následně odešle.



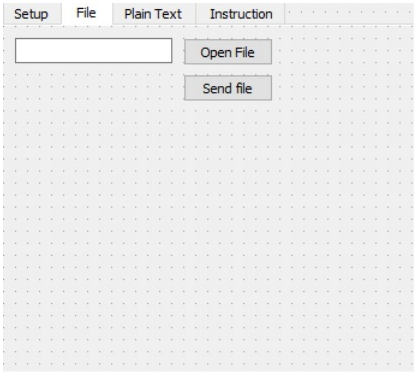
Obrázek 4.3: Záložka nastavení



Obrázek 4.4: Záložka Text



Obrázek 4.5: Záložka Instrukce



Obrázek 4.6: Záložka Soubor

Kapitola 5

Popis implementace

Tato kapitola se zaměřuje na implementaci aplikace. Budou popsány nejdůležitější části aplikace. Vše bude doplněno o ukázky kódu, pro který bude využit jazyk C++, především pro možnost zobrazení předpisu třídy obsaženého v hlavičkových souborech. Některé části budou mírně změněné oproti podobě ve zdrojovém kódu nebo mohou být popsány pseudokódem pro lepší názornost.

5.1 Inicializace aplikace a sériového portu

První částí aplikace, která byla implementována byla třída sériového portu a třída pro načítání dat od uživatele. Tato část se liší pro konzolovou a grafickou aplikaci. Konzolová aplikace načítá nastavení portu přes konzoli a vstup od uživatele a grafická aplikace pomocí jednotlivých prvků grafického rozhraní. Poté co jsou načteny jednotlivé parametry, jsou dál předány třídě představující sériový port. K výběru konkrétního COM portu slouží pomocná třída, která zkontroluje všechny dostupné COM porty počítače, i ty virtuální.

5.1.1 Dostupné sériové porty

Třída SerialPortTester se stará o zkontrolování všech dostupných sériových COM portů. Tato třída má jedinou funkci, která zajišťuje kontrolu a načtení všech dostupných COM portů. Na počítači je možné mít COM0 až COM255, tedy stačí otestovat všechny možnosti pomocí funkce *QueryDosDevice*. Pokud funkce vrátí nulovou hodnotu, tak příslušný COM port nebyl nalezen. Pokud vrátí nenulovou hodnotu, tak port je nalezen a je možné ho používat.

```
std::vector<std::string> SerialPortTester::testAllPorts()
{
   TCHAR lpTarget[5000];
   DWORD test;
   std::vector<std::string> ports;
   for (int i = 0; i < 255; ++i) {
      std::wstring str;
      str = std::to_wstring(i);
      std::wstring COM(L"COM");
      std::wstring ComNameWString = COM + str;
      LPCWSTR ComName = ComNameWString.c_str();

      test = QueryDosDevice(ComName, (LPWSTR)lpTarget, 5000);
      if (test != 0)
      {
         std::wstring ComNameWStr(ComName);
         std::string port = std::string(ComNameWStr.begin(),
             ComNameWStr.end());
         ports.push_back(port);
      }
   }
   return ports;
}
```

5.1.2 Konzolová aplikace

Načítaní dat od uživatele zajišťuje třída *ConfigurationClass*, která má dvě metody. První se stará o nastavení aplikace, tedy o nastavení šifrování a logování do souboru. Také se v této metodě nachází možnost nastavit sériový port pomocí výchozího nastavení. Pokud si uživatel chce nastavit port podle svých požadavků, stará se o to druhá třída, kde si uživatel může vybrat z jednotlivých možností příslušných parametrů.

5.1.3 Grafická aplikace

Načtení všech potřebných nastavení sériového portu je zajištěno pomocí prvků grafického rozhraní, jako je combo box. Uživatel si vybere jaké nastavení potřebuje a zahájí začátek komunikace.

5.2 Nastavení sériového portu

Po načtení všech parametrů jsou tyto parametry předány třídě sériového portu, který je pak nastaven. Následující rozhraní sériového portu přijímá načtené parametry jako parametry konstruktoru třídy sériového portu.

```
class SerialPortClass
{
public:
   SerialPortClass(int &g_baudRate, int &g_parity, int &g_stopBit,
      int g_dataSize, std::string & g_ComPort);
   ~SerialPortClass();
   bool openConnection();
   void closeConnection();
   void * Port();
private:
   std::string com;
   int stopBits;
   int parity;
   int baudRate;
   int dataSize;
};
```

Metoda *openConnection* se stará o samotné nastavení sériového portu, nastavení timeout-ů a podobně. Metoda *closeConnection* se stará o uzavření sériového portu po dokončení komunikace. Poslední třída se stará o předání ukazatele na otevřený port, a to pro třídu, která se stará o všechnu komunikaci.

5.3 Nastavení aplikace

Aplikace nabízí možnost šifrování dat typu instrukce a souborů, a také zaznamenání komunikace do logovacího souboru. Pokud je zvoleno šifrování, je nutné zvolit pořadí, ve kterém budou spolu aplikace komunikovat pro výměnu šifrovacích klíčů. Toto pořadí je zvoleno pomocí jednoduchého handshake protokolu. V prvním kroku se posílá zpráva, která představuje zahájení komunikace. Pokud druhá strana obdrží tuto zprávu, pošle zprávu, která představuje potvrzení přijetí první zprávy. Nakonec vysílač, který vyslal první zprávu zašle finální zprávu s potvrzením o přijetí druhé zprávy. Pokud aplikace poslala první a třetí zprávu, pak je v pozici posílání klíčů pro šifrování vedena jako první. V opačném případě je vedena jako druhá. Toto je příklad takzvaného 3-way handshake protokolu.
Další možnou volbou aplikace je možnost ukládaní komunikace. V tomto případě aplikace zajistí vytvoření logovacího souboru ve složce, kde se nachází aplikace a v podsložce logfiles.
Tato třída také zajišťuje ukládání přijatých souborů pomocí aplikace. Takto přijaté soubory jsou uloženy do dočasného uložiště složky „tmp“ disku C. Soubor je pojmenován podle aktuálního data ve formátu
YYYY_MM_DD.ReceivedFile.

5.4 Začátek komunikace

Zahájení komunikace je klíčová vlastnost aplikace. Samotné zahájení probíhá již v metodě, která se stará o výměnu klíčů pro šifrování. Okamžitě po této výměně je uživatel informován, že je aplikace připravena pro jeho vstup.

5.4.1 Konzolová aplikace

Konzolová aplikace po dokončení výměny klíčů vypíše na konzoli uživateli hlášku, že je připojen. Při volání této metody je také vytvořeno nové vlákno. To zajistí, aby komunikační třída mohla neustále načítat data, která přijdou po sériovém portu. Poté vlákno běží, dokud uživatel nezadá příkaz pro ukončení aplikace, nebo nepřečte ze sériového portu informaci, že druhá strana se odpojila.

5.4.2 Grafická aplikace

U aplikace s grafickým uživatelským rozhraním není potřeba vytvářet vlákna pro jednotlivé vstupy. Platforma Qt funguje na principu *Signal/Slots*, kde se při konkrétním eventu zavolá metoda označená jako signal, a tato metoda zavolá metodu, která je označená příznakem slots. Jednotlivé signály a sloty se musí propojit.
Například kód

```
connect(this, SIGNAL(sendTextToGui(QString)), ptrGUI,
     SLOT(showTextGui(QString)));
```

V uvedeném příkladu se propojuje třída pro komunikaci s grafickým uživatelským rozhraním tak, aby se na prvku rozhraní vypsal přijatý text. Metoda *sendTextToGui* je signal třídy *ControlQtClass* a slot *showTextGui* je slot třídy *QtBcApp*. Třída *ControlQtClass* kontroluje zda komunikační třída přijala data ze sériového portu. Také kontroluje zda-li při komunikaci nedošlo k chybě nebo ukončení komunikace. Pokud byla přijata data, řídící třída si je načte a předá je grafickému rozhraní pomocí vyvolání spojení *Signal-Slot*. K tomuto stačí jednoduché volání ve tvaru

```
emit sendTextToGui(QString::fromStdString(output));
```

a přijatý řetězec se jednoduše vypíše.

5.5 Komunikace a komunikační třída

Komunikační třída *CommuniactionClass* přijímá a posílá data po sériovém portu. Začátek komunikace je zavolání metody *startCommunication*. Po určení pozice pro výměnu šifrovacích klíčů, dojde k samotné výměně a nastavení šifrovacích klíčů. Poté je vše připraveno pro samotnou komunikaci.
Metoda *PortEvent* neustále kontroluje zda nejsou na sériovém portu nějaká data pro načtení. Pokud tam nějaká data jsou, přečte se úvodní zpráva, podle které se pak zavolá příslušná metoda pro načtení dat. Pro načtení dat typu text slouží metoda receiveText. Data typu instrukce přijímá metoda *receiveInstruction* a pro příjem souborů slouží metoda *receiveFile*. Obdobně fungují funkce pro posílání dat na sériový port. Metoda *sendText(data)* pošle úvodní zprávu, která je informací pro druhou stranu o příchozí zprávě a jejím typu. Jejím parametrem jsou data typu string zadána uživatelem. Pro posílání dat typu instrukce slouží metoda *sendInstuction(data)*, pro posílání souborů metoda *sendFile(file)* a pro posílání dat ve formě regulárního výrazu slouží metoda *sendRegex(data)*.
Přijatá data se ukládají a čekají na řídící třídu než si je vyzvedne. Metody *extractTextData(string)*, *extractInstData(BYTE)* a *extractErrorData(string)* vrací hodnotu typu bool pokud data byla načtena do parametru. Pokud data byla načtena, vrací hodnotu true a data se vrátí přes proměnnou v parametru. Pokud nebylo nic načteno, funkce vrátí false. Všechny tři metody fungují stejně, pouze berou data z jiného zdroje. Pro ukázku bude sloužit metoda pro předání dat typu text.

```
bool CommunicationClass::extractTextData(std::string & data)
{
   {
      std::lock_guard<std::mutex> lck(mutexQ);
      if (!myQueue.empty()) {
         data = myQueue.front();
         myQueue.pop();
      }
      else {
         return false;
      }
   }
   return true;
}
```

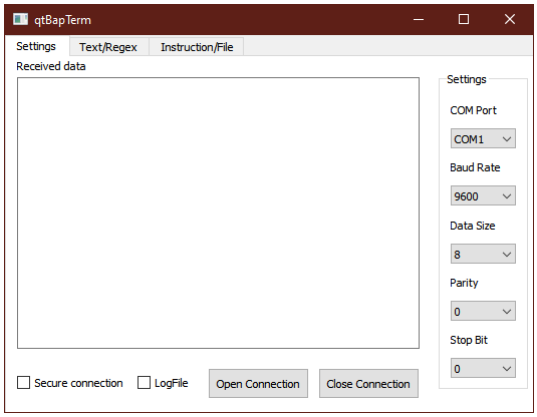
Jelikož data jsou ze sdíleného prostředku fronty, je nutné zajistit, aby nevznikla situace, kdy by se do fronty zapisovalo a zároveň z fronty i četlo. Toto zajistí prostředek *std::mutex*, který omezuje přístup ke sdíleným prostředkům.

5.6 Ukončení komunikace

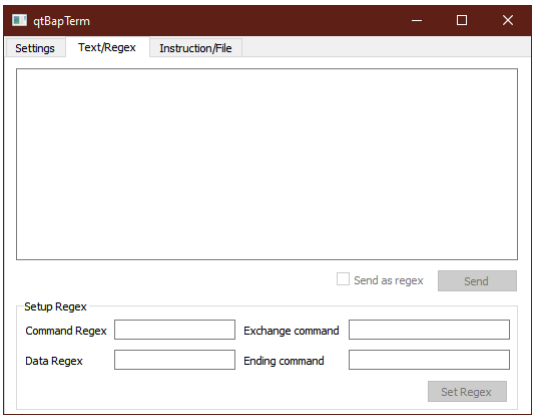
Komunikace se ukončuje vstupem od uživatele, nebo pokud druhá strana ukončí komunikaci obdobně, vstupem od uživatele. Pokud komunikaci ukončuje vstup od uživatele, zavolá se metoda *setToStop* třídy *CommunicationClass*. Tato metoda zajistí ukončení všech procesů komunikační třídy, a pokud je již navázáno spojení, pošle zprávu o ukončení spojení. Druhým způsobem ukončení komunikace je přijetí zprávy o ukončení spojení druhou stranou. V tomto případě je potřeba kontrolovat, zda komunikační třída potřebuje ukončit. K tomuto účelu slouží metoda *getReadyToStop*, která vrací hodnoty typu bool. Metoda vrátí true, pokud komunikační třída obdržela zprávu o ukončení, a false, pokud takovou zprávu doposud nedostala.

5.7 Výsledná aplikace

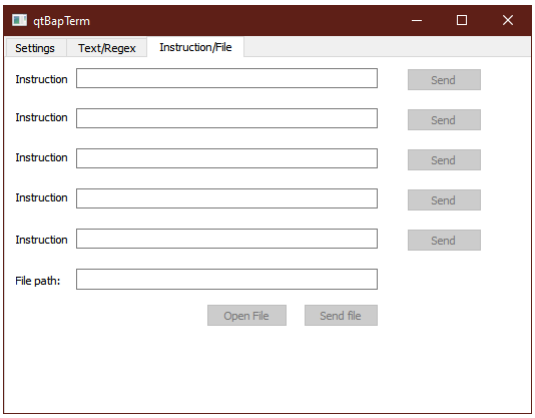
Výsledná podoba grafické aplikace má tři záložky. Jednotlivé záložky znázorňují následující obrázky.



Obrázek 5.1: Settings záložka



Obrázek 5.2: Text a regex záložka



Obrázek 5.3: File/Instruction záložka

Kapitola 6

Testování

Aplikace se nezaměřuje na náročné výpočty a práci s důležitými daty. Je proto nutné otestovat požadovanou funkcionalitu. Byly vybrány případy užití, které jsou větvené a mohou vést k chybám. U všech testovaných případů je uveden očekávaný průběh. Testování je rozděleno na dvě části:

- konzolová aplikace
- grafická aplikace

6.1 Zadání

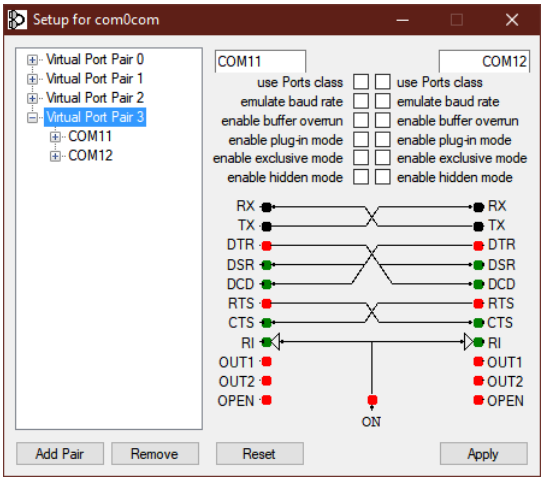
Aplikace byla otestována uživatelem v několika krocích:

- test rychlosti přenášených dat
- spuštění aplikace a nastavení sériového portu
- posílání dat
- nastavení a odeslání regulárního výrazu
- poslání souboru

Testování i kontrola po dobu vývoje byly prováděny pomocí aplikace com0com [21], viz obrázek 6.1, která umožňuje vytvořit virtuálně propojené sériové porty. Pomocí právě takto vytvořených sériových portů byla aplikace testována.

6.2 Průběh testování

Každá chyba, která byla objevena během testování, bude popsána komentářem a bude představeno její opravení.



Obrázek 6.1: Aplikace com0com

6.3 Test rychlosti

Test rychlosti se zaměřuje na rychlost posílání a přijímání dat. Pro tento test je aplikace poupravena pro měření rychlosti a následně pro zápis dat.

```
auto started = std::chrono::high_resolution_clock::now();
WriteFile(Port, buffer, FILE_MAXSIZE * 2, &bytesWritten, NULL);
auto done = std::chrono::high_resolution_clock::now();
auto result =
    std::chrono::duration_cast<std::chrono::milliseconds>(done -
    started).count();
```

Pro příjem dat je aplikace poupravena obdobně.

```
auto started = std::chrono::high_resolution_clock::now();
ReadFile(Port, dataBuffer, (FILE_MAXSIZE * 2) + 4, &bytesRead, NULL);
auto done = std::chrono::high_resolution_clock::now();
auto result =
    std::chrono::duration_cast<std::chrono::milliseconds>(done -
    started).count();
```

Začne se měření před zavoláním funkce pro zápis/čtení na sériový port. Měření se ukončí ihned po volání funkce zápis/čtení. Poté se od sebe odečtou časy začátku a konce měření a výsledkem je počet milisekund.

Jelikož aplikace byla testována pomocí virtuálně vytvořených COM portů, které byly virtuálně propojené, z měření vyplývá, že v tomto případě nezáleží

Tabulka 6.1: Zápis dat

Zapisovací rycyhlost	Počet poslaných dat	Čas
9600	204800 B	201 ms
9600	204800 B	201 ms
9600	204800 B	200 ms
460800	204800 B	201 ms
460800	204800 B	200 ms
460800	204800 B	201 ms
921600	204800 B	200 ms
921600	204800 B	200 ms
921600	204800 B	201 ms

Tabulka 6.2: Čtení dat

Čtecí rycyhlost	Počet poslaných dat	Čas
9600	204800 B	201 ms
9600	204800 B	201 ms
9600	204800 B	201 ms
460800	204800 B	201 ms
460800	204800 B	201 ms
460800	204800 B	201 ms
921600	204800 B	200 ms
921600	204800 B	201 ms
921600	204800 B	201 ms

na zvolené baudové rychlosti, protože takto vytvořené a propojené COM porty neovlivňuje zvolená baudová rychlost.

6.4 Testování konzolové aplikace

Při testování konzolové aplikace je důležité otestovat chybné vstupy a nesprávné příkazy. Testování konzolové aplikace se bude skládat z vícero kroků.

6.4.1 Spuštění a nastavení portu

Uživatel vyzkoušel nastavit pro komunikaci již obsazený sériový port. Aplikace selhala. (chyba opravena - kontrola správnosti vytvořeného portu) Aplikace vypsala chybovou hlášku `Selected port is not availible!`. Po stisknutí libovolné klávesy se aplikace ukončila.
Uživatel po zapnutí aplikace nastavil volný sériový port. Po nastavení portu a nastavení záznamového souboru a bezpečné komunikace byla aplikace připravena na příchozí a odchozí komunikaci vypsáním zprávy `Connected.`.

6.4.2 Odeslání dat

Nastavená aplikace je připravena ke komunikaci. Uživatel zadal příkaz `text Hello`. Aplikace příkaz zpracovala a odeslala zprávu. Při zadání chybného příkazu, například `ttext`, aplikace informovala uživatele o špatném příkazu `Unkown command.`.

6.4.3 Nastavení a odeslání regulárního výrazu

Uživatel zadal při nastavení regulárního výrazu nesprávný vstup. Aplikace nereagovala. (chyba opravena) Poté uživatel nastavil správně regulární výraz. Při pokusu o odeslání nesprávného regulárního výrazu byl uživatel informován výpisem v terminálovém okně.

6.4.4 Výběr a odeslání souboru

Uživatel vyzkoušel zadat název neexistujícího souboru ve tvaru `C:\Users\username\Destkop\example.txt`. Aplikace vypsala informaci o tom, že tento soubor nelze otevřít. V případě zadání správného názvu souboru byl soubor úspěšně odeslán.

6.5 Testování grafické aplikace

U testování grafické aplikace není nutné testovat vstupy, jelikož uživatel nemůže zadat parametry sériového portu podle sebe, má na výběr pouze z přípustných hodnot pro nastavení sériového portu.

6.5.1 Spuštění a nastavení portu

Aplikace nabízí uživateli pouze sériové porty, které jsou dostupné. Tedy uživatel nemůže vybrat nedostupný port. Uživatel si vybral sériový port, který už byl obsazen. Uživatel byl informován pomocí dialogového okna, že se nepodařilo připojit.
Uživatel vybral neobsazený port a úspěšně se připojil. Informace o úspěšném připojení je zobrazena v textovém okně v záložce „Nastavení“.

6.5.2 Odeslání dat

Uživatel zadal do textového pole data k odeslání. Po kliknutí na tlačítko `Send data`, aplikace data odeslala. Při zadání nesprávného formátu odesílaných instrukcí nebo regulárního výrazu byl uživatel informován pomocí dialogového okna.

6.5.3 Nastavení a odeslání regulárního výrazu

Při pokusu o nastavení nesprávného tvaru regulárního výrazu byl uživatel informován dialogovým oknem. Uživatel byl takto informován i v případě, kdy se pokusil odeslat nesprávný formát regulárního výrazu.

6.5.4 Výběr a odeslání souboru

Uživatel pro vybrání souboru kliknul na tlačítko `Open File`. Tímto kliknutím se otevřelo dialogové okno pro výběr souboru. Lze vybrat pouze soubory, které existují. Do textového pole se vyplní název souboru, který lze upravovat. Při zadání jména neexistujícího souboru aplikace upozorní na neexistující soubor pomocí dialogového okna.

6.6 Shrnutí

Uživatelské testování odhalilo chyby v aplikaci. Testování tedy má smysl. Nalezené chyby byly opraveny.

Závěr

Výsledkem této práce je třída pro komunikaci po sériovém portu. Třída byla implementována do konzolové a grafické aplikace, ve kterých byla testována její funkčnost. Aplikace umožňuje snadnou komunikaci po sériovém portu, který je zcela nastavitelný. Aplikace nabízí zabezpečení posílaných dat a také ukládání příchozích dat do souboru. Uživatel si také může nastavit speciální typ zprávy, který může posílat pomocí regulárních výrazů. Aplikace je spustitelná na libovolném počítači s operačním systémem Windows.

Sériové porty nejsou v dnešní době tolik rozšířené, proto je vhodné používat USB připojení, které se tváří jako sériový port. K tomuto slouží převodníky mezi USB a RS–232C. USB CDC (Communications Device Class) zajišťuje, že převodník připojený pomocí USB portu je označen jako sériový port. Aplikace komunikuje i s těmito porty.

V budoucnu by se vývoj aplikace měl zaměřit především na přepracování aplikace tak, aby byla multiplatformní, například pro systémy Linux a macOS, a kvůli blízké spolupráci s dalšími třídami by mohlo být vhodné vytvoření knihovny. Dále by se mohl vývoj zaměřit na přidání dalších portů, jako třeba USB, ethernet a dalších.

Bibliografie

[1] PuTTY [software]. [Cited 2021-12-05]. Dostupné z: https://www.putty.org

[2] Jedlička, T.: RealTerm – Sériový terminál [online]. Květen 2003, [Cited 2021-12-05]. Dostupné z: https://vyvoj.hw.cz/navrh-obvodu/software/realterm-seriovy-terminal.html

[3] Excel [software]. [Cited 2019-05-11]. Dostupné z: https://products.office.com/cs-cz/excel

[4] MATLAB [software]. [Cited 2021-12-11]. Dostupné z: https://www.mathworks.com/products/matlab.html

[5] Aplikace Hercules SETUP [software]. [Cited 2021-12-05]. Dostupné z: https://www.hw-group.com/cs/software/aplikace-hercules-setup

[6] Tišnovský, P.: Sériový port RS–232C [online]. Listopad 2008, [Cited 2021-12-05]. Dostupné z: https://www.root.cz/clanky/seriovy-port-rs-232c/

[7] Tišnovksý, P.: Paralelní port a rozhraní Centronics [online]. listopad 2008, [Cited 2021-12-10]. Dostupné z: https://www.root.cz/clanky/paralelni-port-a-rozhrani-centronics/

[8] Tišnovksý, P.: Universální sériová sběrnice (USB) [online]. leden 2009, [Cited 2021-12-10]. Dostupné z: https://www.root.cz/clanky/universalni-seriova-sbernice-usb/

[9] Frenzel, L.: What's The Difference Between Bit Rate And Baud Rate? [online]. Duben 2012, [Cited 2021-12-05]. Dostupné z: https://www.electronicdesign.com/communications/what-s-difference-between-bit-rate-and-baud-rate

[10] Mareš, J.: První sériově vyráběný převodník USB-RS232 (COM) v ČR - UCAB232 [online]. Prosinec 2002, [Cited 2021-12-05]. Dostupné z: https://vyvoj.hw.cz/produkty/prvni-seriove-vyrabeny-prevodnik-usb-rs232-com-v-cr-ucab232.html

[11] Všetečka, R.: Operační systém od Microsoftu má 30 let. Kam došla Windows od 1 po 10 [online]. Listopad 2015, [Cited 2021-12-06]. Dostupné z: https://www.idnes.cz/technet/software/windows-1-az-10-vyroci.A151119_093211_software_vse

[12] Oracle Java [online]. [Cited 2022-01-02]. Dostupné z: https://www.thttps://www.oracle.com/java/

[13] Python For Beginners [online]. [Cited 2022-01-02]. Dostupné z: https://www.python.org/about/gettingstarted/

[14] Dokumentace k jazyku C [online]. [Cited 2022-01-02]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/csharp/

[15] C++ Programming Language [online]. [Cited 2021-12-05]. Dostupné z: https://www.techopedia.com/definition/26184/c-programming-language

[16] Dokumentace sady Visual Studio [software]. [Cited 2021-12-01]. Dostupné z: https://docs.microsoft.com/cs-cz/visualstudio/?view=vs-2019#pivot=get-started

[17] About Qt [online]. [Cited 2021-12-05]. Dostupné z: https://wiki.qt.io/About_Qt

[18] Rouse, M.: RSA algorithm (Rivest-Shamir-Adleman) [online]. Listopad 2018, [Cited 2021-12-05]. Dostupné z: https://searchsecurity.techtarget.com/definition/RSA

[19] Rouse, M.: Advanced Encryption Standard (AES) [online]. Březen 2017, [Cited 2021-12-05]. Dostupné z: https://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard

[20] Satrapa, P.: Regulární výrazy (1) [online]. Duben 2000, [Cited 2021-12-05]. Dostupné z: https://www.root.cz/clanky/regularni-vyrazy-1/

[21] com0com [software]. [Cited 2021-12-20]. Dostupné z: http://com0com.sourceforge.net

Příloha A

Seznam použitých zkratek

GUI Graphical user interface

XML Extensible markup language

USB Universal Serial Bus

HDMI High-Definition Multi-media Interface

VGA Video Graphics Array

TCP Transmission Control Protocol

SSH Secure Shell

SCP Secure Copy

AES Advanced Encryption Standard

DES Data (Digital) Encryption Standard

3DES Triple DES

DOS Disk operating system

TCP/IP Transmission Control Protocol/Internet Protocol

UDP/IP User Datagram Protocol/Internet Protocol

ASCII American Standard Code for Information Interchange

HEX Hexadecimální soustava

DEC Desítková soustava

DVI Digital Visual Interface

LCD Liquid crystal display

OS Operating system

SQL Structured Query Language

Příloha B

Instalační a uživatelská příručka

B.1 Instalační příručka

Instalace nebo stahování dodatečných souborů je potřeba pouze u grafické aplikace.

B.1.1 Grafická aplikace

Požadavky:

- Qt Enviroment https://www.qt.io/download

Eventuelně zkopírováním `.dll` souborů ze složky „dll_ files“.

B.2 Uživatelská příručka

V uživatelské příručce je uvedeno jak se aplikace spustí a obsluhuje.

B.2.0.1 Spuštění terminálové aplikace

Nastavení portu probíhá podle instrukcí vypisovaných na terminálové obrazovce. Po nastavení sériového portu lze kdykoliv vyvolat pomocný výpis pomoci příkazu **help**. Nejdřív se zadá konkrétní sériový port, například „COM1“. Následně se postupně zadají další parametry k nastavení.

B.2.0.2 Grafická aplikace

Nastavení sériového portu se provádí pomocí jednotlivých rozbalovacích seznamů. Po výběru uživatel klikne na tlačítko „Open connection“, čímž se zahájí komunikace. Až proběhne potřebné nastavení aplikace, vypíše informaci o úspěšném připojení.

B.2.1 Posílaní dat

Tato sekce popisuje jak se posílají data v jednotlivých aplikacích.

B.2.1.1 Konzolová aplikace

V konzolové aplikaci se posílaní dat provádí pomocí příkazů:

- `text`
- `instruction`
- `regex`
- `file`

Příkaz `text Hello world!` odešle data typu text s obsahem „Hello world!“. Příkaz `instruction` se používá ve speciálním tvaru `instruction @cABC@h00ff`, kde se odešlou data typu instrukce ve tvaru ABC 0x00 0xff. Tedy **@c** se používá pro posílané znaky a **@h** pro bajty.

B.2.1.2 Grafická aplikace

Všechny typy dat se zadávají do textových polí a odesílají pomocí tlačítek. Formát posílaných instrukcí je stejný jako u konzolové aplikace.

B.2.2 Regulární výrazy

Aplikace dovoluje nastavit dva regulární výrazy. První regulární výraz představuje výraz pro řízení komunikace. Druhý regulární výraz představuje data, která se budou posílat. U řízení komunikace se nastavují dva řídící výrazy. První jako úvodní zpráva. Tento výraz může být ponechán prázdný. Druhý výraz je určen pro ukončení komunikace, a musí být vyplněn.
Jednoduchý příklad:

- řídící regulární výraz : [a-z][a-z]][a-z][a-z]

  řídící výraz jako úvodní zpráva: „“ - prázdný řetězec nebo „text“

  řídící výraz jako ukončující výraz: „quit“

- regulární výraz pro data: [A-Z][a-z][a-z][0-9]

  příklad dat: „Txt1“

Uživatel není omezen pouze na podobné typy regulárních výrazů, ale může zadat všechny typy regulárních výrazů, které podporuje *std::regex* z knihovny C++.

Příloha C

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu přiloženého média
- `exe` ..... adresář se spustitelnou formou implementace
  - `lib` ..... adresář s knihovnami pro spuštění aplikací
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce